UNIVERSIDAD DE GRANADA

REAL-TIME MONOCULAR 3D RECONSTRUCTION OF SCENARIOS USING ARTIFICIAL INTELLIGENCE TECHNIQUES

DOCTORAL DISSERTATION
presented to obtain the
DOCTOR OF PHILOSOPHY DEGREE
in the
INFORMATION AND COMMUNICATION TECHNOLOGY PROGRAM
by
Erick Patricio Herrera Granda

PhD Advisors
Juan C. Torres-Cantero & Diego H. Peluffo-Ordóñez

Granada, August 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Erick Patricio Herrera Granda
ISBN: 978-84-1195-258-3
URI: https://hdl.handle.net/10481/90846

This work was supported by the SDAS Research Group (www.sdas-group.com).

Dedicado a toda mi familia.

Agradecimientos

Llegar a este punto de mi vida ha sido una travesía sumamente larga y digna de ser vivida. Especialmente el estudio y desarrollo de mi programa doctoral ha ocurrido en medio de múltiples circunstancias, empecé con este sueño en el año 2019 con muchas expectativas y planes, con la sensación de que todo marcharía acorde a lo planificado. Sin embargo, nadie tenía previsto que lo tendría que realizar en medio de una pandemia. el COVID-19. Esto sin duda marcó mi vida y la vida de todos quienes están y quienes ya no están hoy en este mundo. Este suceso desencadenaría una serie de sucesos que no tenía contemplados, pasar por múltiples ocasiones por el desempleo, la mudanza a nuevas ciudades donde trabajar, el confinamiento, la crisis económica, alimentaria y de medicamentos; y sin duda todo esto desembocó en la enfermedad a la que tuve que enfrentarme desde enero de 2022 hasta junio de 2023.

Sin duda alguna todos estos sucesos pudieron haber doblegado hasta al más fuerte guerrero, pero con la bendición de Dios me encuentro aquí redactando estas palabras, dedicadas a mí mismo, a mi familia, a mis descendientes y a todos mis amigos, para plasmar que lo he logrado y comprometerme en seguir alcanzando todos mis sueños. En esta vida todo es posible si estamos dispuestos a hacer los sacrificios que sean necesarios, creemos en nosotros mismos, y nos rodeamos de buenas personas. Por esto, agradezco especialmente a mis tutores Juan Carlos Torres Cantero y Diego Hernán Peluffo Ordóñez, quienes más allá de ser mis tutores se han convertido en los mejores amigos que la vida ha podido colocar en mi camino, quienes no solo han estado ahí para transmitirme su invaluable conocimiento, sino que han sido el soporte que necesitaba en medio de todas las adversidades que se me han presentado durante estos últimos 4 años. De la manera más sincera, considero que, si no hubiese sido por sus palabras de apoyo, su guía, su paciencia y sabiduría, no hubiese podido alcanzar ésta, que es mi más grande meta académica. Que Dios bendiga el gran trabajo que realizan.

Además, agradezco a la vida, el haber crecido con la mejor familia del mundo, mi papá, mi mamá y mis hermanos, quienes nunca dudaron que alcanzaría mi meta, quienes siempre han invertido todo lo que han tenido en mí. A mis padres especialmente, que siempre han dejado todo de lado por la educación de sus hijos y son el más grande ejemplo y orgullo que tengo. Finalmente, agradezco a mi esposa María José, a quien conocí a inicios de este doctorado, quien ha tenido que pasar junto a mi por todas las adversidades y ha sido mi soporte y apoyo para aprender a crecer en medio de la tormenta.

Gracias, de todo corazón.

Table of Contents

Acronyms

ATE	Absolute Trajectory Error
BA	Bundle Adjustment
BoW	Bag of Words
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DF-ORB-SLAM	Dense-Indirect ORB-SLAM
DSM	Direct Sparse Mapping
DSO	Direct Sparse Odometry
DROID-SLAM	Differentiable Recurrent Optimization-Inspired Design SLAM
DynaSLAM	Dynamic SLAM
EKF	Extended Kalman Filter
FC-CRF	Fully Connected Conditional Random Field
FOV	Field of View
FPFDCNN	Feature Point and CNN Feature Description
GPS	Global Positioning System
GRU	Gated Recurrent Unit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LDSO	Loop-Closure Direct Sparse Odometry
LMedS	Least Median of Squares
LMCW	Local Map Covisibility Window
LORANSAC	Local Optimization RANSAC
LS	Least Squares
ML	Machine Learning
MonoSLAM	Monocular SLAM
MRF	Markov Random Field
NeW	Neural Window
OpenMVG	Open Multiple View Geometry
ORB	Oriented FAST and Rotated BRIEF
PBA	Photometric Bundle Adjustment
PCL	Point Cloud Library
PnP	Perspective-n-Point
PTAM	Parallel Tracking and Mapping
RANSAC	Random Sample Consensus
REMODE	Regularized Monocular Depth Estimation
RPE	Relative Pose Error
RMSE	Root Mean Square Error
SDF	Signed Distance Function
SDFCNN	Deep Semantic Fusion CNN
SFM	Structure from Motion
SIDE	Single Image Depth Estimation

SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SSIM	Structural Similarity Index Measure
SVO	Semi-Direct Visual Odometry
TSDF	Truncated Signed Distance Function
VO	Visual Odometry
VOLDOR	VO using Log-logistic Depth Residuals

Chapter I - PhD Dissertation

1. Title

Real-time monocular 3D reconstruction of scenarios using artificial intelligence techniques.

2. Abstract

This research presents a comprehensive study on monocular 3D reconstruction of environments using only RGB images as input acquired through a monocular sensor. The objectives were to develop a suitable taxonomy, review seminal algorithms, compare open-source methods, and develop a novel 3D reconstruction system using the principal classic techniques combined with artificial intelligence to improve the overall system performance. An exhaustive literature review led to a proposed taxonomy with three classifications: direct vs indirect, dense vs sparse, and classic vs machine learning. This resulted in 10 categories used to classify 42 notable monocular SLAM, SFM, and VO systems based on 11 identified criteria. Subsequently, through rigorous benchmarking, ten prominent open-source algorithms were implemented across the taxonomy to discern each method's advantages and limitations. The TUM-Mono dataset, considered the most complete benchmark comprising 50 outdoor and indoor sequences, was used for evaluation. Statistical analysis revealed that sparse-direct methods significantly outperformed others, with DSO excelling. In addition, it was evidenced that integrating machine learning modules into the SLAM pipeline significantly contributes to the system performance and the final reconstruction quality. Consequently, DSO was selected for enhancement by integrating the state-of-the-art single image depth estimation NeW-CRFs CNN module. This module introduced depth prior knowledge to refine DSO's depth initialization and tracking. Using the TUM-Mono dataset, the new DeepDSO method was benchmarked against DSO and CNN-DSO. DeepDSO surpassed the others across various metrics, including translation error, rotation error, scale error, alignment error, and RMSE. Statistical tests confirmed DeepDSO's superiority, achieving an impressive RMSE of 0.0624, which corresponds to an error reduction close to 13.35% with respect to the original DSO system. DeepDSO pushes monocular VO boundaries by strategically integrating machine learning-based depth estimation. In addition, the taxonomy and comparative analysis provide guidelines for appropriate algorithm selection and implementation. This study validates the benefits of implementing artificial intelligence within SLAM, VO and SFM systems and lays the groundwork for continued depth initialization and point-tracking optimisations.

3. Introduction

In computer vision research, reconstructing a scene from images captured by a moving camera is called "Structure from Motion" (SFM). In robotics, this challenge is recognized as "Simultaneous Localization and Mapping" (SLAM) and "Visual Odometry" (VO). Despite numerous methodologies proposed, such as multi-view devices, RGB-D monocular, and RGB monocular cameras [1], this area warrants further exploration. Multi-view devices consist of multiple cameras that triangulate images captured simultaneously from various angles. While they offer precision, they come at a higher cost and implementation complexity. On the other hand, RGB-D devices utilize active or passive sensors to measure depth, represented as a 3D point cloud, facilitating scene reconstruction [1]. However, their limited measurement range confines them to smaller scenarios [2]. Moreover, depth sensors often underperform in sunlight, making reconstructions less accurate [3]. Given these constraints, RGB monocular cameras have gained traction due to their versatility, cost-effectiveness, and adaptability to various lighting conditions [3]–[6].

Recent research in this domain has spanned classic geometric-based methodologies employing probabilistic tools [4], [5], [7] to the integration of Machine Learning (ML) approaches, given the significant strides in artificial intelligence. Many traditional methods have been adapted to this evolving paradigm, resulting in enhanced reconstruction quality, depth prediction, camera pose estimation, and tracking [3], [8]–[12].

3D reconstruction has garnered significant attention due to its diverse applications, including:

- Robotics: scene recognition, autonomous driving, and Unmanned Aerial Vehicles (UAV) piloting (Engel et al., 2014; Pizzoli et al., 2014; Tateno et al., 2017).
- Augmented Reality: scene and body reconstruction (Czarnowski et al., 2020; Mur-Artal et al., 2015; Tateno et al., 2017).
- Film Industry: facial puppetry, face replacement, speech-driven animation, and more (Zollhöfer et al., 2018).
- Medical Field: 3D reconstruction of surgical cavities and virtual endoscopy (Shallik et al., n.d.; Su et al., 2020).

The challenge of 3D reconstruction has persisted for over two decades, evolving through various techniques such as place recognition and matching. Historically, solutions were categorized into direct and indirect methods. Direct methods work directly with pixel information. An example from this category is LSD-SLAM [4], which leverages probabilistic and odometry techniques, focusing on semi-dense depth maps. Indirect methods use preprocessing techniques to reduce the amount of input information. An example from this category is ORB-SLAM [7], which utilizes ORB-type features for tracking, mapping, and loop-closing tasks. ML approaches also extend the classic pipelines, adding neural networks, like CNN-SLAM [3], that represent a fusion of traditional and modern approaches, demonstrating the potential of convolutional neural networks in 3D reconstruction.

Despite advancements, 3D reconstruction using SFM, VO or SLAM remains a formidable challenge. Integrating 3D computer vision with probabilistic mechanisms, optimization, and artificial intelligence seeks to enhance reconstruction accuracy while addressing issues related to camera pose, depth estimation, and more. A growing aspiration is to achieve real-time system operation. However, the computational demands of 3D reconstruction often necessitate high-performance hardware, prompting researchers to explore hybrid approaches like [3], [6], [13]. In addition, one primary objective for many researchers is enhancing depth map accuracy, a critical component of monocular 3D reconstruction, where state-of-the-art methods have reached accuracy levels lower than 70% [3].

In brief, this research field presents numerous challenges, including global shadow analysis, robust reconstruction during intense rotational movements, and the fusion of machine learning with optimization. Consequently, this research aims to develop a framework to integrate the best components and advancements of SLAM, VO, SFM and ML to contribute to the state of the art regarding precision, speed of execution, robustness or flexibility.

4. Objectives

4.1. General objective.

Propose a framework for real-time monocular 3D reconstruction of scenarios using artificial intelligence techniques in conjunction with classic SLAM tools.

4.2. Specific objectives

1. Identify the main techniques of classic SLAM and artificial intelligence applicable in the 3D reconstruction of scenarios and sufficient to carry out this task.

2. Implement prototypes in various configurations to identify the benefits and limitations of the various methodologies and their ability to integrate with existing ones.

3. Validate the performance of the 3D reconstruction prototypes in various scenarios under the metrics, allowing for a comparison with the state of the art.

5. Methodology

This section delineates the methodological design employed to achieve the research objectives. Figure 1 provides a graphical representation of the key research components utilized in this study.

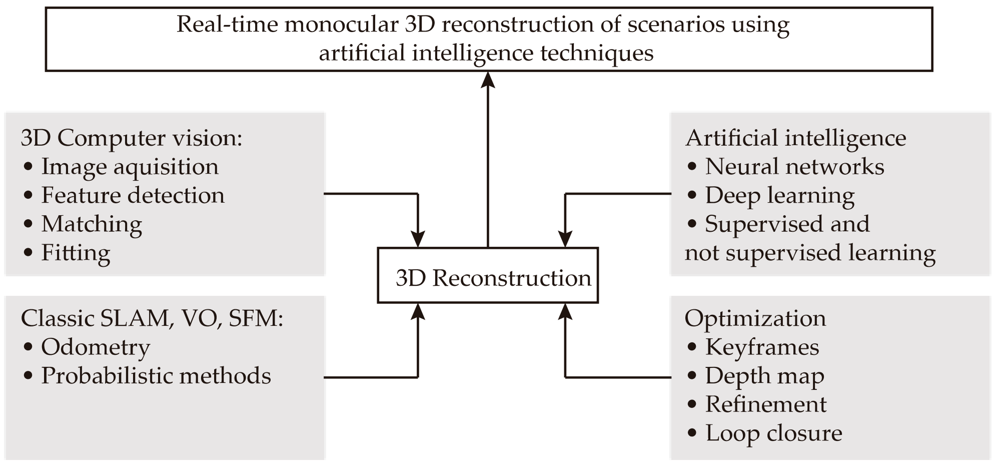


Figure 1. Graphical Illustration of the Key Research Components

The methodological design of this investigation was structured in phases, detailed as follows:

Chapter 2: Literature Review and Taxonomy

- Phase 1: Bibliographic Review - An in-depth systematic review was conducted on specialized topics such as monocular 3D reconstruction, odometry, SLAM, optimization, and artificial intelligence applied to depth map estimation, camera pose estimation, tracking, and other components of dense and sparse SLAM, SFM and VO.

- Phase 2: Technical Study and Observation - An exploration of monocular SLAM, SFM, and VO methodologies proposed by researchers to date was undertaken. This facilitated the delineation of the necessary hardware and software tools for this research. A taxonomy proposal was also presented.

Chapter 3: Comparative Analysis and Selection

- Phase 3: Replication of Previous Studies - Prior studies with sufficient information for replication were identified, serving as a comparative foundation for selecting techniques and structuring the developed framework. The chosen studies were open-source, enabling their proper implementation for comparative purposes.

- Phase 4: Comparison of Replicated Models - After replicating the foundational research under the parameters set by their authors, their performance and the performance of their constituent tools were compared. This provided a pool of potential components for integration into the subsequent prototypes.

- Phase 5: Identification of Various Configurations - Upon identifying the standout components from previous studies, the performance of various configurations in open-source methods was analyzed, considering compatibility and relevance criteria.

Chapter 4: Proposal

- Phase 6: Development of Complementary Algorithms - The amalgamation of methodologies and components from various sources necessitated algorithms that facilitate their integration. Hence, this phase saw the development of algorithms and structures that enabled the integration of methodologies and the creation of complementary codes.

- Phase 7: Preliminary Performance Verification - After prototyping the proposal, its behaviour was verified using relevant evaluation metrics regarding accuracy and error, serving as a foundation for enhancing the prototype's performance.

- Phase 8: Proposal for Scientific Contribution - Experimentation with the developed prototype delineated avenues for potential improvements in methodologies proposed in the literature. This phase involved the integration of algorithms or neural structures that enhanced prototype performance.

- Phase 9: Real-time Execution - Upon finalizing the prototype, adjustments or adaptations were made to facilitate its real-time execution by reducing computational costs and optimizing the code.

- Phase 10: Prototype Accuracy and Execution Time Validation - The developed framework was compared with leading models from the literature regarding reconstruction accuracy and depth map acquisition precision using the TUM-MONO database. This facilitated the computation of comparative metrics against the state of the art.

- Phase 11: Documentation - Throughout the research development, a documentation protocol was maintained, recording results and findings via a website, articles, and the final thesis report.

Chapter 4: Results and conclusions

Chapter II - Literature Review and Taxonomy

1. Introduction

In recent years, there has been growing interest in using computer vision and cameras to obtain key information that allows machines to interact with their surroundings. This is done by using motion cues to improve navigation, tracking, and manipulation tasks [14]. To address this need, solutions have emerged under the umbrella terms Simultaneous Localization and Mapping (SLAM) in robotics and Visual Odometry (VO) or Structure from Motion (SFM) in computer vision [2]. Many researchers, like [15] and [16], define SLAM as the ability of a mobile robot to start in an unknown environment and incrementally build a map using information acquired from camera observations. At the same time, it can estimate its trajectory using single or multiple cameras. SFM algorithms have been referred to as Monocular SLAM [17] due to their similarity in generating sparse or dense world models to obtain a geometric representation. This is done using direct or indirect techniques to generate and track the models.

Similarly, VO aims to determine the position and orientation of a robot using camera images. Many solutions can perform odometry tasks, such as wheel odometry, global positioning systems (GPS), global navigation satellite systems (GNSS), inertial navigation systems (INS), laser sensors, ultrasonic sensors and VO [18]. Wheel odometry uses low-cost encoders but is prone to drift due to wheel slippage. INS calculates position and orientation along three axes using accelerometers and gyroscopes. The position is obtained by integrating acceleration, which can cause drift from minor acceleration errors accumulating into significant position errors. GPS/GNSS obtain the position by trilateration of signals from multiple satellites, avoiding long-term error accumulation. However, GPS/GNSS can have meter-level inaccuracies and cannot be used indoors or underwater. Laser and ultrasonic sensors provide scalar distance measurements to objects using time of flight or phase shift principles. However, they can have issues with reflective materials or surface orientations.

In contrast, VO offers a low-cost, more accurate solution than GPS, INS, ultrasonic sensors, and wheel odometry. This is due to VO's low position error range from 0.1 to 2% [19]. As an odometry technique, VO balances cost, reliability, and complexity [20]. Moreover, researchers like [2] and [13] found that VO performs better by generating geometric maps of the environment for robot localization. Thus, many VO systems like [2], [13], [21]–[28] incorporate 3D reconstruction to generate maps for estimating vehicle location. A key difference from SLAM is that VO typically discards points once they leave the camera's view, whereas SLAM enables reusing previously triangulated points when they reenter the field of view during loop closures [18].

Another major difference is that SLAM concurrently builds a map while using it to estimate camera pose. In contrast, SFM focuses more on scene reconstruction, while VO focuses on estimating ego-motion (camera translation and orientation) [19]. It should be noted that SLAM systems are more complex, aiming to provide orientation, trajectory, loop closure, and other information for navigation and scene understanding. The geometric maps generated by SLAM are also often optimized. Thus, the final 3D reconstructions from SLAM tend to be superior to those from purely VO or SFM, although these techniques inspire SLAM. In all cases, camera sensors acquire visual information to reconstruct the environment map. When using a single camera, these are called Monocular SLAM [18], Monocular VO, or Monocular SFM. Despite the different names, all these techniques share the common goal of 3D scene reconstruction and camera pose estimation from imaging sensors (Aqel et al., 2016; Engel et al., 2017; Tateno et al., 2017).

In this overview, we are particularly interested in exploring techniques to generate 3D geometric environment representations (reconstruction) from a single moving RGB camera. Thus, approaches like Visual-Inertial (using INS), RGB-D (using depth sensors), omnidirectional, and stereo techniques are out of scope.

2. Related works

Within academic circles, monocular 3D reconstruction is a complex challenge, often addressed by merging various strategies and algorithms from realms like computer vision, robotics, and machine learning. To date, only select studies delve into this realm in depth. [18] crafted an exhaustive exploration of visual odometry, shedding light on its types, methodologies, obstacles, and practical uses. Their study highlighted diverse input methods and the driving forces behind investigating the VO challenge, leading to an initial classification distinguishing feature-based from appearance-based methods. Our study diverges by diving into the trifecta of SLAM, VO, and SFM. Conversely, the prior work emphasized solely the VO aspect, sidelining the monocular input angle. Fast forward to 2021, Servières et al. [29] tabled an intricate breakdown of contemporary classification and practical testing of visual and visual-inertial SLAM methods, sketching out the fundamental framework of SLAM and its progressive evolution. Their empirical analysis incorporated systems such as DSO, LSD-SLAM, and ROVIO. Yet, our research angle contrasts with that of Servières et al. Beyond spotlighting SLAM, our focus pivots towards 3D reconstruction, weaving in SLAM, VO, and SFM. Rather than merely cataloguing historical data or method descriptions, our chief aim is to formulate a precise taxonomy and elaborate on pivotal algorithms, offering readers a clear framework that aids in discerning the optimal techniques for their scholarly endeavours and projects.

Within the domain of survey articles, only a handful of works exclusively focus on the pure visual monocular methodology. Taketomi et al. [30] have elaborated on a study covering the period between 2010 and 2016 specifically tailored for the visual SLAM (vSLAM) approach. This piece highlights the vSLAM problem's chief components and employs feature-based and direct classification to encapsulate the core SLAM systems from 2016 to 2010. Feature-based methodologies like MonoSLAM [31], PTAM [32], and ORB-SLAM [33] were explored, while direct formulations included systems like DTAM [17], LSD-SLAM [4], SVO [13], and DSO [2]. Further, Taketomi et al.'s survey dissected the RGB-D techniques, spotlighting KinectFusion [34] and other notable systems. Another important survey to note is Chahine & Pradalier's (2018) piece [35], focusing on monocular SLAM algorithms suitable for outdoor scenarios, particularly those environments rich in natural elements that demand longer-range capabilities unaffected by direct sunlight, adept at handling textured visuals filled with lush greenery. Their selection highlighted algorithms up to 2018, with a concentration on systems like DSO, ORB-SLAM, and LSD-SLAM. These were analyzed and juxtaposed based on the RMSE of translational error across eight sequences curated by the authors. Their findings accentuated DSO's superiority for exterior applications, though they also highlighted the scope for refinement in reconstruction and tracking quality. Similarly, a 2018 work by Chen et al. [36] delves deep into primary visual SLAM systems, giving weight to those that left a mark before 2018. This research encapsulates the narrative of SLAM, its synergy with SFM and VO issues, SLAM categorization, pertinent challenges, and future trajectories. While several SLAM systems were acknowledged, the study notably assessed PTAM and ORB-SLAM as feature-based methodologies and platforms like DTAM, SVO, LSD-SLAM, and DSO as direct classifications. More recently, a 2020 investigation by M. He et al. [37] predominantly navigated the waters of the monocular visual odometry challenge, elucidating the VO dilemma, its structural formulation, and its parallels with emergent SLAM suggestions. The authors highlighted numerous systems, including MonoSLAM, PTAM, and RGB-D methodologies. In juxtaposition with the abovementioned studies, our efforts culminated in a detailed, coherent survey encompassing 42 algorithms. We didn't restrict ourselves to a single problem type (SLAM, SFM, or VO) or a unique classification category. Our expansive approach was underpinned by a holistic extended classification that encompassed all monocular pure visual systems that play a part in the monocular 3D reconstruction domain.

The study closest to this approach is the research penned by Macario et al. [38]. This article serves as an exhaustive survey tailored for visual SLAM algorithms. In it, the authors elucidate foundational concepts, proffer a taxonomy rooted in direct and indirect classifications, and comprehensively describe eight pure visual SLAM techniques, six visual-inertial SLAM methods, and five RGB-D SLAM strategies. Moreover, the article delves into unresolved issues and potential future directions. The discourse was meticulously articulated, with each algorithm being systematically overviewed and systematized. Our trajectory mirrored theirs in many respects. However, our exploration predominantly zoomed in on the monocular input modality. Consequently, we delved deeper into this modality, establishing and implementing a taxonomy that more aptly encapsulates each algorithm's nuances. Furthermore, we meticulously examined 42 of the most prominent available algorithms, integrating a comprehensive exploration of the latest Machine Learning (ML) paradigms that remained untouched in the abovementioned studies.

3. Contributions and outline

This study provides the first comprehensive review explicitly tailored to the monocular 3D reconstruction issue, amalgamating SLAM, VO, and SFM solutions under a singular taxonomy. In essence, the prime hallmarks of our investigation encompass:

1. A crafted taxonomy that encapsulates the entirety of contemporary literature methodologies. This taxonomy is structured upon three distinct classifications, melding them into a spectrum of ten meticulously detailed categories.
2. An exhaustive exploration of 42 emblematic monocular SLAM, VO, and SFM algorithms, comprising 18 traditional monocular strategies and 24 approaches harnessing machine learning modalities. Each dissected methodology delineates its algorithmic configuration, highlighting mathematical tenets for those methods pioneering alterations in their blueprints—particularly in areas of depth map assessment or refinement, given the 3D reconstruction emphasis of this research.
3. Eleven distinct criteria have been delineated to provide the readers with the foundational components needed to implement, select, or devise a 3D reconstruction system. Of these, nine pertain to traditional systems, whereas an additional two are solely relevant to machine learning methodologies. Details for each algorithm based on these criteria can be found in Tables 2 and 3. The specified criteria encompass: algorithm type (SLAM, VO, or SFM), tracking approach (direct vs. indirect), map density (dense vs. sparse), pixel utilization methodology, estimation technique (about the depth map estimation), global refinement, relocalization, loop closure (indicating the algorithm's incorporation of optimization, relocalization, or loop closure phases), software accessibility (open-source repositories where the algorithm is hosted), CNN framework (widely-recognized employed CNN architectures), and the principal operations for which a CNN is utilized.
4. An exploration into prevailing challenges, extant solutions, and prospective trajectories for each classification is offered, accompanied by a temporal analysis of the citation scores garnered by each category within the taxonomy. This is intended to give readers insights into the impact and reception that each classification, technique, and category has garnered in this field of study.

The structure of this manuscript is as such: Section 4 sheds light on input modalities, section 5 elucidates foundational concepts, terminologies, and the taxonomy, section 6 ventures into a review of the most representative classic techniques, section 5 addresses the methods that infuse machine learning, section 6 delves into a discussion concerning the comprehensive taxonomy, and the paper culminates with conclusions in section 7.

4. Input modalities

Creating a 3D visual depiction of an environment is a challenging task, made possible through camera sensors. Historically, cutting-edge systems relied on intricate camera arrangements and specific lighting designs, mainly tailored for indoor settings. In contemporary times, equipment choices span from high-priced multi-view and stereo configurations to more affordable single-lens sensors. Following this, a concise summary of various techniques for 3D reconstruction will be provided.

4.1. Stereo setups

In multi-view configurations, a collection of pairwise stereo cameras is organized to permit the concurrent capture of multiple perspectives of the same entity [1]. Notably, systems employing a pair of cameras are designated as stereo systems [18]. Such binocular systems integrate two distinct camera sensors, enabling the immediate calculation of depth and image scale through triangulation, given their pre-defined and recognized stereo baseline dimension. Nonetheless, compared to monocular sensors, these binocular cameras generally come at a higher cost and demand a greater calibration commitment. Furthermore, for consistent results, these sensors must capture images within identical time frames, a feat accomplished by coordinating the shutter speeds via an external triggering mechanism [39]. Notably, preserving a steady calibration baseline between two cameras often entails more diligence than its monocular counterpart. If the baseline diminishes significantly compared to the distance from the subject to the camera, stereo configurations are compromised to a monocular status [40]. Consequently, their application is predominantly confined to small or indoor applications.

4.2. Omni-directional cameras

Numerous researchers, including Ke et al. [41][25], [41]–[43], have opted to employ omnidirectional cameras owing to their expansive field of vision (FOV). Valiente García et al. [41] noted that these cameras offer an information yield surpassing conventional cameras. Furthermore, image features persist longer, facilitating the derivation of meticulously detailed 3D landscape models. Nonetheless, the deployment of omnidirectional cameras is not without challenges. They are not only costly but also necessitate significant setup efforts and lack compatibility with mobile devices. Some of these devices employ mechanical rotation for incremental scene scanning, positioning them as best suited for static settings and potentially unsuitable for dynamic environments [3]. An important challenge faced during 3D reconstruction tasks with these devices is image distortion stemming from the equirectangular representation. This distortion arises when the spherical pixels, once projected onto a plane, undergo substantial warping, potentially leading to errors in depth estimation [44].

4.3. Monocular RGB-D

RGB-D sensors are equipped with an auxiliary active or passive sensor, facilitating real-time depth determinations of the environment for each pixel in the image. Such depth data provides a solution to the depth and scale uncertainty encountered in monocular reconstructions, approximating the environmental geometry. These instruments can be categorized into active and passive classes. Unlike their stereo counterparts, passive RGB-D devices often incorporate a projector instead of an additional camera, casting a pattern onto the image to identify corresponding points. These devices predominantly operate within the infrared (IR) domain to ensure the projected pattern remains imperceptible to the human gaze. RGB-D devices employing infrared projectors are labelled as active. It's imperative to note that active RGB-D devices might yield inaccurate readings under sunlight [3], which is attributable to the sun's overpowering IR radiation. Consequently, certain RGB-D cameras amalgamate both active and passive sensors, which toggle based on sunlight exposure, though this adaptation can amplify the device's cost substantially. One pervasive drawback of these light instruments is their inability to reconstruct entities smaller than the illumination pattern [1].

The Time of Flight (ToF) methodology offers an alternative approach for active RGB-D cameras. Depth determinations are estimated by releasing a light pulse and calculating the duration required for it to contact the target. Since this interaction occurs nearly at light speed, capturing accurate measurements proves challenging. Hence, ToF RGB-D cameras often underperform when juxtaposed with other RGB-D illuminative models [1].

4.4. Monocular RBG

RGB cameras measure light intensity across three channels: red, green, and blue. Different designs for these devices exist, from CCD sensors that individually capture each channel using distinct sensors to those using Bayer pattern sensors, where colour filters are arranged interwoven ahead of a singular sensor [1]. Monocular cameras are recognized for their ability to mitigate calibration discrepancies. Due to their prevalence, affordability, ease of installation, and incorporation in many handheld devices, these cameras remain a primary focal point for researchers. The latter typically gravitate towards this input form when executing reconstruction tasks in SLAM, SFM, and VO. However, it's worth noting that employing this input introduces complexities. Monocular vision inherently grapples with scale ambiguities [45], [46], and methodologies aimed at 3D reconstruction often necessitate considerable computational capacity. Table 1 provides a concise overview of the advantages and limitations associated with various input modalities employed for 3D reconstruction.

Table 1. Input modalities used in 3D reconstruction

Type of camera	Pros	Cons
Stereo	Instant computation of depth information and image scale. Ease in obtaining depth information. Capable of providing 3D data.	Demand more intensive calibration relative to monocular devices, increasing costs. Complications in synchronizing shutter operations. Risk of degradation to a monocular status when the stereo baseline falls short compared to the distance between the camera and the object.
Omni-directional	Offer an expansive field of view, nearly 360º, leading to richer image data. Image features persist for extended durations, aiding in the derivation of refined models.	Generally pricier than alternative camera types. Lack of compatibility with mobile devices. Some models may struggle in dynamic settings. Susceptibility to distortions, particularly those stemming from equirectangular representations.
Monocular RGB-D	Facilitate real-time depth measurements in conjunction with image captures. Convenient deployment mechanisms. Harmonious integration with mobile platforms. Optimal for applications in small robotics and indoor settings.	Possibility of generating inaccurate measurements under sunlight exposure. Constraints set by the range of the active sensor and the dimensions of the projected pattern. Typically priced higher than monocular RGB sensors.
Monocular RGB	Economical choice with the most affordable pricing structures. Widespread availability and use. Straightforward deployment processes. Integrated into a majority of mobile devices. Simplified calibration methods. Not bound by sensor range limitations. Versatile applications include small robotics, both indoor and outdoor settings, and functionality under sunlight.	Present challenges with image scale ambiguity. Lacks inherent depth measurement capabilities. 3D reconstruction operations may necessitate substantial computational input.

Referring to Table 1, it's evident that while monocular RGB cameras have certain challenges, primarily stemming from their inability to directly provide depth information due to their sensorless configuration, they offer a compelling suite of advantages, particularly for applications in small robotics.

Among the myriad camera variants, monocular RGB sensors boast the most economical price point, coupled with ease of deployment and a compatibility range that spans almost all contemporary portable devices and processors, including Single Board Computers (SBC) and Field Programmable Gate Arrays (FPGA). This combination of attributes renders the monocular RGB input modality especially appealing to the academic community, and it serves as the foundational premise for the focus of this research.

5. Basics and notation

5.1. Literature review process

Engaging in 3D scene reconstruction via monocular cameras represents a daunting challenge, one that has captivated a considerable number of researchers. Surprisingly, this field remains sparsely researched. Given the ill-posed nature of this issue, existing methodologies exhibit considerable divergence, and terminology may lack consistent usage. Therefore, the primary aim of this manuscript is to introduce a coherent taxonomy and furnish foundational insights into this field. To this end, our research strategy employed specific search parameters on both Scopus and Google Scholar databases: TITLE-ABS-KEY (("SLAM" OR "VO" OR "SFM" OR "Simultaneous Landing and Mapping" OR "Visual Odometry" OR "Structure from Motion")) AND ("Monocular" OR "Visual" OR "RGB") AND NOT ("RGB-D" OR "Stereo" OR "omnidirectional" OR "Visual Inertial" OR "VI")). An exhaustive, independent review enabled the exclusion of studies centring on RGB-D, stereo, or omnidirectional camera frameworks. Ultimately, our inquiry yielded 137 studies dedicated to 3D reconstruction via monocular RGB methodologies. Utilizing both Scopus and Mendeley, we extracted the bibliometric particulars of each study, resulting in the generation of a *.ris* file. This data was subsequently analyzed by employing the VOSviewer software [47]. Implementing a bibliometric co-authorship assessment grounded in our chosen bibliometric data, an association strength methodology, a full-count approach, and a stipulation of at least two publications per author, we identified 64 authors who satisfied these criteria. The outcomes of this bibliometric evaluation are depicted in Figure 1.

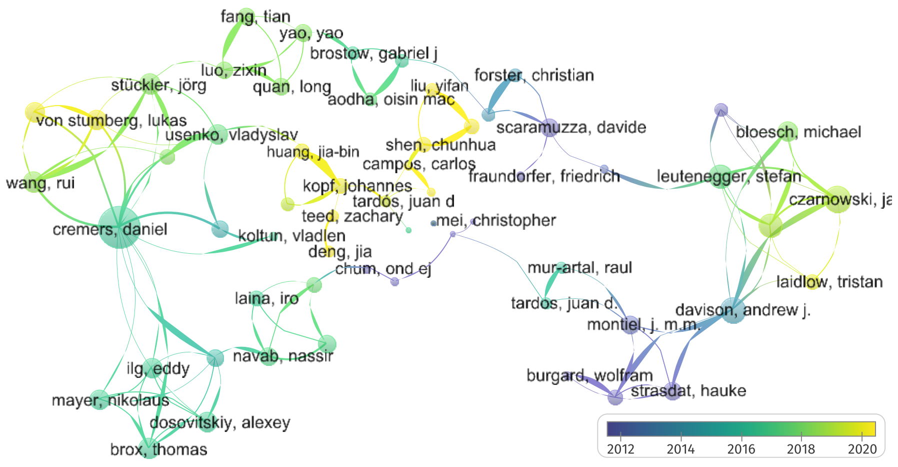


Figure 2. Results from the bibliometric co-authorship analysis, carried out using VOSviewer, employed the association strength approach, a full-count technique, and mandated at least two publications for each author.

From the bibliometric assessment, it was noticeable that certain authors exhibited prominent link strengths due to their important contributions to cutting-edge developments in the field. Notably, Cremers, D., Czarnowski, J., Davison, A., Clark, R., and Leuteneger, S. stood out with link strengths of 28, 13, 12, 11, and 11, respectively. Guided by these findings, our literature exploration started with a thorough review of works by these foremost authors to establish a foundational understanding. Subsequently, our bibliographic repository expanded by incorporating references cited within each article, both as antecedent works and comparative studies.

5.2. Notation

In this manuscript, vectors in equations and declarations are denoted by bold lowercase letters, such as ($\boldsymbol{x}$). Matrices are symbolized by bold uppercase characters, for instance ($\boldsymbol{R}$). Scalars are designated using non-bold lowercase letters, exemplified by (c). Functions and images utilize non-bold uppercase letters, like (I). Consider an image (I) composed of a certain pixel set. For each pixel, denoted as $\boldsymbol{q}$, in the image, it is posited that a depth value d exists, facilitating the projection of the related 3D coordinates $\boldsymbol{x} = (x, y, z)^T$. Consequently, camera poses are delineated as transformation matrices $\boldsymbol{T}_i \in SE(3)$, which translates a point from the global frame to the camera-centric frame. Herein, $\boldsymbol{R}$ signifies rotation matrices, whereas Π and Π^{-1} respectively stand for projection and back-projection operations. Moreover, d^* typifies inverse depth values, implying that $\boldsymbol{D}$ and $\boldsymbol{D}^*$ pertain to depth and its inverse depth map representations, respectively.

5.3. Initial words and approaches

Camera pose estimation is a primary objective in visual SLAM, SFM, and VO domains. Historically, researchers have addressed this through three distinctive methodologies: feature-based, appearance-based method, or a fusion of both feature and appearance-based strategies [18], [19], [41].

The feature-based methodology, as implemented in studies such as [6], [7], [14], [33], [48], [49], predominantly engages in the extraction of salient image features, which might include corners, lines, and curves, among other elements. Subsequent steps involve the alignment of representative features and motion estimation. The feature vectors' Euclidean distance across the two images is utilized to identify matching candidates. Consequently, when two different-pose images of an identical scene are considered, the features from the initial image are aligned with the congruent features in the latter image. This alignment facilitates the determination of a 3D position corresponding to these matched features, a process visualized in Figure 3. The assessment of motion customarily revolves around the displacement of these features. The camera's pose is inferred by identifying a geometric transformation spanning each image pair, contingent on a set of corresponding features.

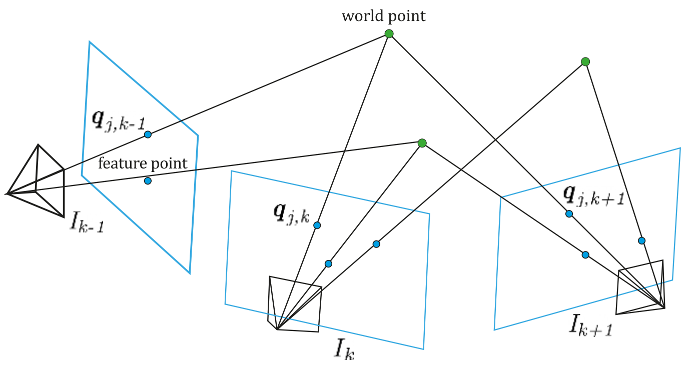


Figure 3. Feature matching and 3D triangulation from multiple views.

In the domain of visual tracking, there are distinct methodologies: the feature-based approach, which focuses on specific features within an image, and the appearance-based approach, which centres on the changes in the appearance of sequential images by leveraging pixel intensity data. One alternative that bypasses direct pixel intensity utilization is employing optical flow. This technique assesses the shift in brightness patterns between consecutive images, represented by the intensity values of neighbouring pixels. Optical flow (OF) algorithms, based on the pixel count used for deducing camera motion, are categorized as either dense or sparse. Dense strategies harness the entirety of the image data, wothout the need for feature extraction. This full utilization, however, renders them more susceptible to noise than their sparse counterparts [50].

One prevalent method within the appearance-based paradigm for estimating ego-motion is template matching. This entails selecting a "template" segment from an initial image and subsequently endeavouring to locate a matching segment in the succeeding frame. The essence of template matching lies in discerning the presence of a smaller sub-image (the template) within a larger encompassing image (the search area). This is achieved by computing similarity metrics, such as normalized cross-correlation (NCC) and the sum of square or absolute differences (SSD/SAD). The operational premise involves the algorithm progressively shifting the template across the search area, pinpointing the position with the maximal similarity metric as the template's location within the new image [18]. Following this, the pixel displacements of the template, denoted as Δu and Δv, are articulated as vectors, thereby outlining the velocity and ensuing acceleration in the flow field, a visualization provided in Figure 4. These pixel displacements can be transmuted through camera calibration parameters into tangible horizontal and vertical shifts, suitable for various applications.

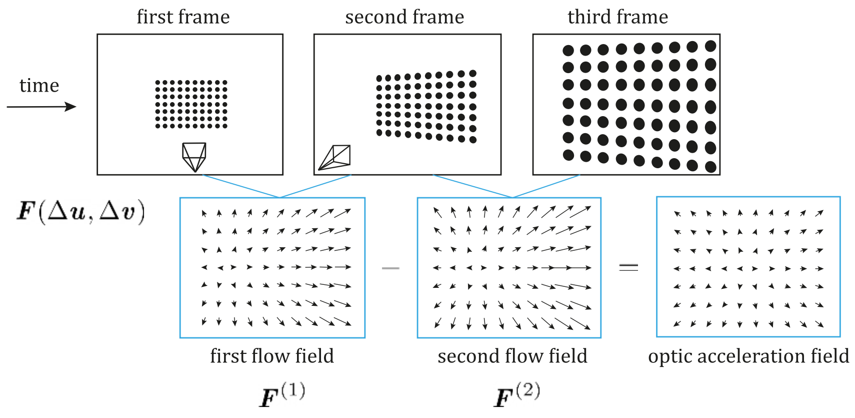


Figure 3. Optical flow and optical flow acceleration fields are generated from consecutive frames. Where F is the optical flow field, and Δu and Δv are the pixel displacements of the template.

Gonzalez et al. [51] and Nourani-Vatani and Borges [52] indicate that feature-based methodologies often encounter challenges in environments with minimal textures, such as walls or roads. This is primarily attributed to the limited availability of distinguishable features that can be effectively identified and tracked in such contexts. Conversely, appearance-based techniques are recognized for their resilience, offering superior tracking capabilities in low-textured settings (Nourani-Vatani & Borges, 2011). Nonetheless, these methods are not without their limitations; they exhibit sensitivity to changes in photometric conditions and necessitate meticulous initialization to yield optimal outcomes. To address these complexities, systems like the one presented by [43] were designed by amalgamating feature- and appearance-based strategies, offering a more comprehensive approach to visual tracking.

Given the intrinsic complexity of 3D reconstruction inherent to monocular SLAM, SFM, and VO methodologies, the prior categorization—based on feature, appearance, or a hybrid of both—cannot encompass the breadth of solutions available for this monocular ill-conditioned challenge. A more refined taxonomy can be discerned from the contributions of researchers such as [16], [17], [22], [23], [53]. Two predominant categories are evident from these works: direct versus indirect and dense versus sparse.

Thus, monocular SLAM, VO, and SFM techniques can be categorized primarily based on the volume of features they employ for 3D reconstruction tasks, leading to a bifurcation into sparse and dense methodologies. Furthermore, another differentiation emerges based on whether there's a necessity for initial preprocessing to extract relevant parameters and measurements. Indirect techniques hinge on this preprocessing phase, producing an intermediary depiction of imprecise measurements that undergo optimization before determining geometry and camera movement. Direct methods, on the other hand, directly work with pixel data. With the rapid advancements in machine learning and its notable outcomes, a novel category has been integrated into this classification, as illustrated in Figure 4. As a result, we delineate three primary classifications to create a holistic taxonomy for the monocular 3D reconstruction challenge: direct vs. indirect, dense vs. sparse, and classic vs. machine learning approaches.

Direct vs. Indirect: Direct methods bypass preprocessing stages such as feature or optical flow extraction. In contrast, indirect methods incorporate these preprocessing stages in their workflows.

Dense vs. Sparse: The term "dense" describes methodologies that utilize the whole or a significant portion of image pixel data. Conversely, "sparse" denotes techniques that leverage only a subset of the available pixel information.

Classic vs. Machine Learning: Classic techniques, often called geometric-based methods, base their operations on geometric principles, odometry, or probabilistic frameworks. These methods lack learning phases and necessitate precise tuning and calibration for optimal performance. In juxtaposition, machine learning-driven approaches have showcased their prowess in executing low-level functions (such as feature extraction, depth estimation, and pose estimation) and high-level operations (like classification and semantic segmentation). Owing to these advantages, many researchers are exploring neural networks for tasks like pose prediction, depth evaluation, feature discernment, and semantic segmentation. These are often integrated with traditional architectures to augment their precision, adaptability, resilience, scene interpretation abilities, and other salient attributes.

Figure 4 illustrates the triadic classification framework adopted in this investigation. Taking into account these categorizations and acknowledging the existence of hybrid methodologies [13], [54] that meld both direct and indirect strategies for reconstruction, we put forth the following taxonomy: Classic + Dense + Direct, Classic + Sparse + Direct, Classic + Dense + Indirect, Classic + Sparse + Indirect, Classic + Hybrid, ML + Dense + Direct, ML + Sparse + Direct, ML + Dense + Indirect, ML + Classic + Sparse + Indirect and ML + Hybrid. The classification of the currently existing methods is depicted in Figure 5.

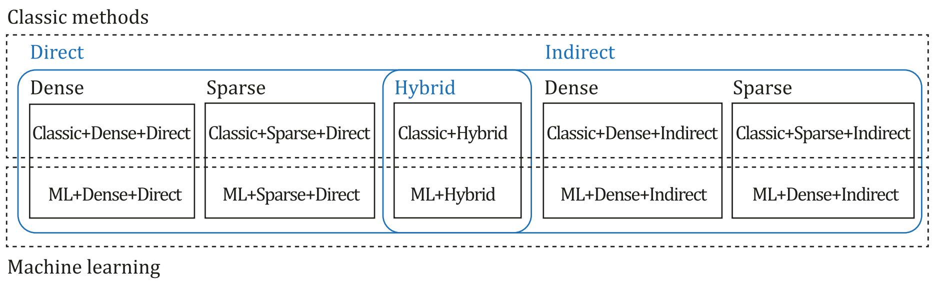


Figure 4. Proposed taxonomy for monocular 3D reconstruction methods

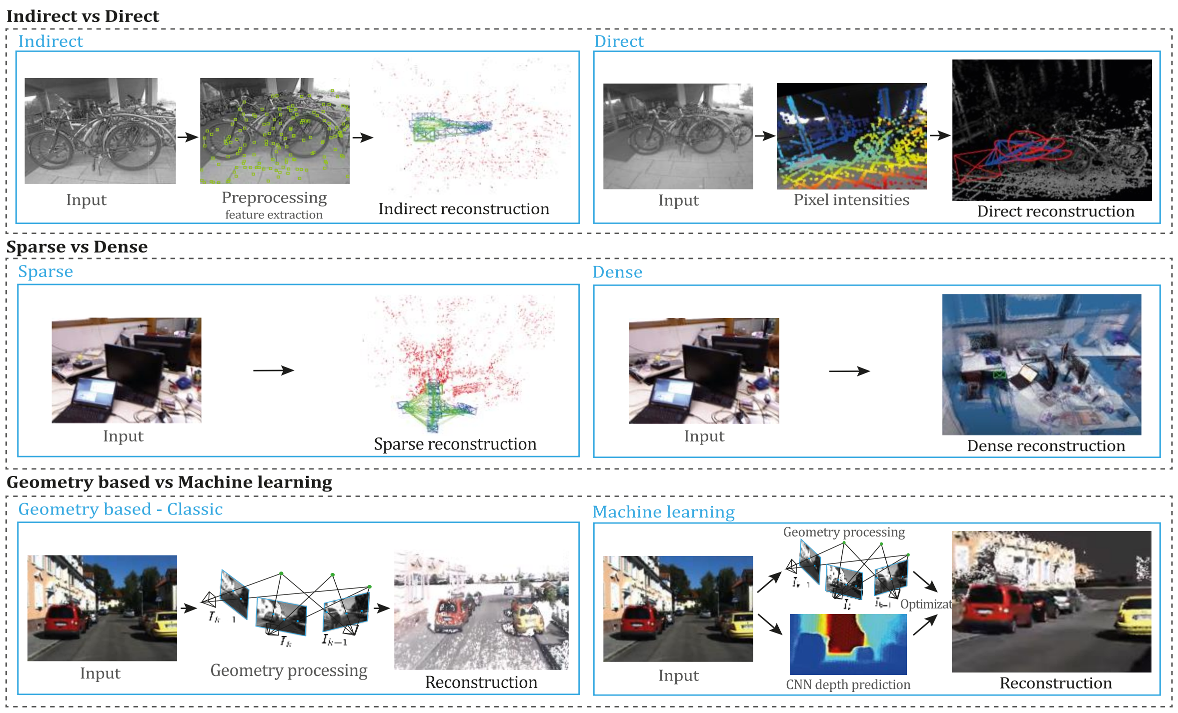


Figure 5. Classifications for monocular 3D reconstruction approaches. The top examples represent direct and indirect classification, while the middle examples belong to sparse and dense classification. The bottom examples correspond to classic and machine learning classification. Examples were obtained by implementations of ORB-SLAM2 [21], DSO [2], LDSO [10], and MonoRec [43] on the datasets TUM-Mono [44] sequence 42, TUM-RGB-D [45] sequence Freiburg-1-room and KITTI [46] sequence 7, respectively.

6. Classic methods

In this study, the initial category was focused on the traditional Monocular SLAM, SFM, and VO techniques, collectively named the geometric-based approach. These methodologies predominantly harness geometric data to facilitate 3D reconstruction and to execute their comprehensive SLAM, VO, or SFM processes. Given the inherent complexity of reconstructing scene geometry, which is characterized as an ill-posed problem, numerous proposals have emerged that amalgamate conventional geometric methods with other strategies, such as probabilistic, optimization, computer vision, and heuristic techniques. However, it is pertinent to note that this categorization deliberately omits methods incorporating machine learning, a topic slated for discussion in Section 5.

As delineated by [2], the estimation of scene geometry can be achieved through a probabilistic model that capitalizes on noisy measurements, denoted as $\boldsymbol{Y}$, sourced from images. This subsequently yields an X estimator for the 3D model and ego-motion [2]. Addressing the 3D reconstruction challenge can be approached through two primary paradigms: the indirect and the direct. Within the indirect paradigm, measurements from the camera undergo preprocessing, resulting in an intermediate representation that addresses a portion of the overarching problem. The values derived from this phase then serve as noisy inputs for the X estimator. On the other hand, the direct paradigm sees systems forgoing the preprocessing phase, opting instead to directly utilize measurements gleaned from observations as noisy inputs for the X estimator within a probabilistic framework. This bifurcation also influences the optimization strategy: while direct methods prioritize the optimization of photometric errors (differences in pixel intensity) due to their reliance on photometric measurements, indirect methods emphasize the optimization of geometric errors, stemming from the geometric values computed during preprocessing.

6.1. *Classic + Indirect methods*

As mentioned, indirect methodologies employ preprocessing stages to distil information from the input image sequence, typically manifesting as visual features, descriptors, or optical flow. Concurrently, traditional indirect systems can be designed to facilitate either sparse or dense 3D reconstructions of the environment. These nuanced subcategories will be further detailed in sections 4.2.1 and 4.2.2. Presented in Figure 6 is a chronological overview of the most prominent methods within this category, curated based on their citation frequency and pivotal roles in pioneering new implementations or comparative studies. Notably, Figure 6 underscores the ORB-SLAM system and its subsequent iterations as a cornerstone development within this category, boasting one of the most outstanding citation metrics in the context of this research.

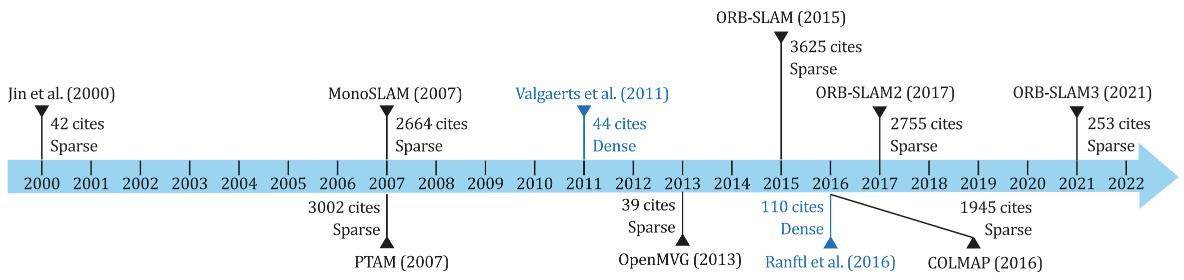


Figure 6. Monocular SLAM, VO, and SFM classic indirect systems--A Timeline.

6.1.1. *Classic + Sparse + Indirect methods*

Within this category, several predominant approaches can be identified, primarily grounded in 3D geometry estimation derived from keypoint matches, utilizing geometry error without prior geometry knowledge. The SLAM methodologies in this category leverage a specific subset of pixel data to execute their inherent processes. Consequently, the resultant output mirrors a subset of the anticipated scene reconstruction. In certain applications, such as robotics, this level of reconstruction is deemed adequate for undertakings like robot navigation or place recognition. The characterization of these methods as "indirect" stems from their preprocessing stages. During these stages, novel variables, encompassing features and their respective positions are deduced by replacing pixel data, ensuring subsequent processes utilize these newly derived values.

Jin et al. (2000). An early pivotal work of this category was presented by Jin et al. in 2000, titled "Real-Time 3-D Motion and Structure of Point Features" [14]. This research delineated a system proficient in selecting and monitoring a designated set of high-contrast point features within an image sequence. The system's capability extended to estimating a three-dimensional relative position and motion rooted in an inertial reference. This objective was realized by pinpointing an $N - tuple$ of points, designating a reference plane Y_0, and ascertaining their depth through the projection ρ rays. This approach culminated in the derivation of a discrete-time non-linear dynamic system, articulated using a translation vector function T, a rotation matrix $\boldsymbol{R}$, and its associated linear and rotational velocities V and $\widehat{\omega}$ (expressed using the hat notation). Herein, α symbolizes the pertinent accelerations. The authors posited that the noisy projection could be articulated as $Y^i(t) = \pi\left(\boldsymbol{R}(t)Y_0^i\rho^i + T(t)\right) + \boldsymbol{n}^i(t) \in \mathbb{R}^2$, aligning with the projection model's framework. Additionally, the research validated the model's minimalistic nature and affirmed the stability of the Extended Kalman Filter predicated on this model. The model's traditional state space representation was subsequently represented as:

$$\begin{cases} Y_0^i(t+1) = Y_0^i(t), \quad \text{with} \quad i \in \{4, \dots, N\}, & Y_0^i(0) = Y_0^i, \\ \rho^i(t+1) = \rho^i(t) \quad i \in \{2, \dots, N\}, & \rho^i(0) = \rho_0^i \\ T(t+1) = \exp\big(\widehat{\omega}(t)\big) T(t) + V(t), & T(0) = T_0 \\ \Omega(t+1) = Log_{SO(3)}\left(\exp\big(\widehat{\omega}(t)\big) \exp\left(\widehat{\Omega}(t)\right)\right), & \Omega(0) = \Omega_0 \\ V(t+1) = V(t) + \alpha_v(t), & V(0) = V_0 \\ \omega(t+1) = \omega(t) + \alpha_\omega(t), & \omega(0) = \omega_0 \\ Y^i(t) = \pi\left(\exp\left(\widehat{\Omega}(t)\right) Y_0^i(t) p^i(t) + T(t)\right) + \boldsymbol{n}^i(t), & \end{cases} \tag{1}$$

where $\boldsymbol{n}^i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_n)$, $Log_{SO(3)}(\boldsymbol{R})$ stands for Ω and $i = \{1 \dots N\}$. The algorithm employs the model mentioned above and undergoes phases of initialization (of the initial selection of features), transient operations (encompassing prediction, update, gain, and linearization), and regime tasks (which include initialization, prediction, and update). Features are meticulously tracked, eliminated, and refreshed within these phases following the Riccati equation. New features are integrated into the state after a designated probationary period. Figure 7 delineates the algorithm conceived by Jin et al., inspired by their publication [14].

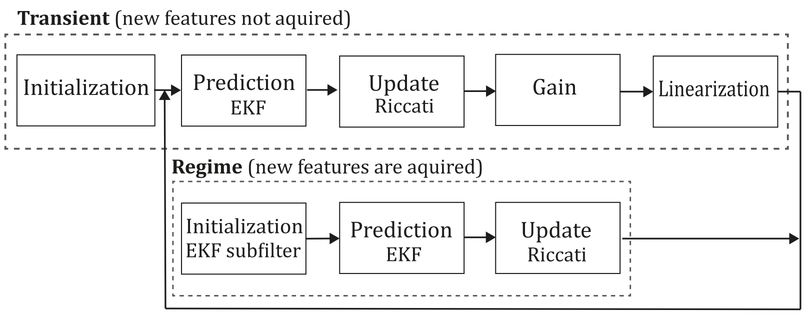

Transient (new features not aquired)
Initialization
Prediction EKF
Update Riccati
Gain
Linearization
Regime (new features are aquired)
Initialization EKF subfilter
Prediction EKF
Update Riccati

Figure 7. Jin et al., algorithm diagram. Adapted from [14].

MonoSLAM (2007). In 2007, Davison et al. [31] pioneered introducing the first monocular visual SLAM system. This innovation was constructed upon foundational works [55]–[57] with the intent of devising a "pure vision" system. This system was designed to dynamically construct persistent 3D environmental maps while concurrently estimating camera ego-motion, rectifying drift through loop closures, and operating in real-time, all by exclusively harnessing a monocular camera as its information source. This system dedicates a persistent sparse map through a probabilistic framework that utilizes a landmark set. Noteworthy advancements in this study encompass incorporating active guided measurement, superior mapping features, amalgamating a motion model for refined camera motion estimation (facilitating the capture of prior data within an image sequence), and introducing feature initialization and orientation estimation techniques. The authors established that leveraging SLAM, wherein simultaneous probabilistic camera state estimation occurs in tandem with its map, as opposed to SFM, offers advantages in terms of efficient processing. The probabilistic feature map estimation is central to the MonoSLAM methodology, which encapsulates the camera's state at any given moment and all pertinent features. The Extended Kalman Filter perpetually refreshes this map. MonoSLAM employs expansive image patches (11×11 pixels) as enduring landmarks, with detection facilitated by Shi & Tomasi operators (Shi & Tomasi, 1994). Given the inherent scale ambiguity of monocular SLAM, system initialization necessitated the provision of some scene-prior data, achieved by positioning a recognizable rectangular target before the camera. Subsequent depth estimation involved placing a semi-infinite line at each 2D position, oriented in a specific direction. Along this line, discrete depth hypotheses were sequentially estimated as the camera moved, culminating when the distribution converged to a peak. The distribution could be approximated as Gaussian when the standard deviation diminishes below a set threshold. A notable limitation of this approach is the algorithm's complexity, which escalates directly to scene size, yielding a sparse landmark map that doesn't fully capture the intricacies of the environment. Figure 8 depicts the MonoSLAM algorithm inspired by the article [31].

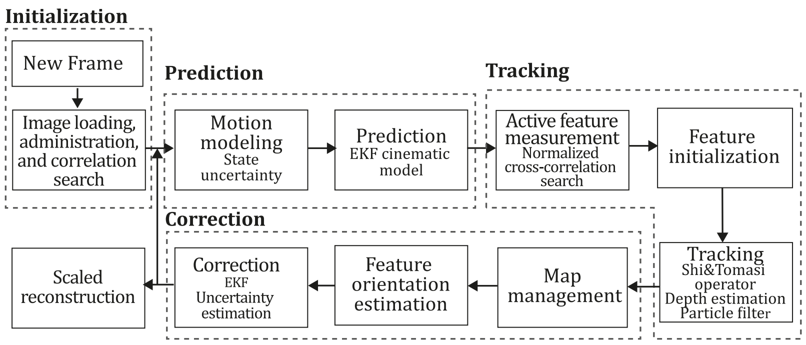


Figure 8. Diagram of MonoSLAM algorithm diagram. Adapted from [31].

PTAM (2009). In 2009, Klein & Murray [58] introduced a system known as "Parallel Tracking and Mapping" (PTAM), specifically tailored for the augmented reality domain. The primary objective of PTAM was to transform a planar surface into an interactive arena suitable for VR simulations, albeit its application was primarily confined to smaller-scale AR endeavours. PTAM distinguished itself as the first system to separate tracking and mapping tasks into two concurrent threads, capitalizing on the multi-core architecture of contemporary computers. Within this framework, the tracking thread bore the responsibility of prior pose estimation, projecting map points onto an image, coarsest-scale feature identification within the image, camera pose updates based on these matches, patch searches of reprojected points, and the estimation of the current frame's pose from the identified matches. Conversely, the mapping thread was meticulously crafted to oversee map initialization, its subsequent refinement, and expansion, drawing keyframes from the tracking thread. This also encompassed the Levenberg-Marquardt bundle adjustment to recalibrate each keyframe's pose and refine data associations, leveraging an outlier management strategy rooted in a Tukey estimator. A prominent feature of PTAM is decoupling tracking and mapping, bypassing numerous redundant frames. This liberates processing capabilities to focus on a select group of keyframes, facilitating operations with an expanded map size. This approach eschews incremental mapping in favour of a more precise batch method, exemplified by Bundle Adjustment. PTAM was conceived with an emphasis on AR integrations within video games and, when juxtaposed with EKF-SLAM [59], demonstrated a marked reduction in trajectory errors. Figure 9 visually represents the PTAM algorithm, a drawing inspired by the article [58].

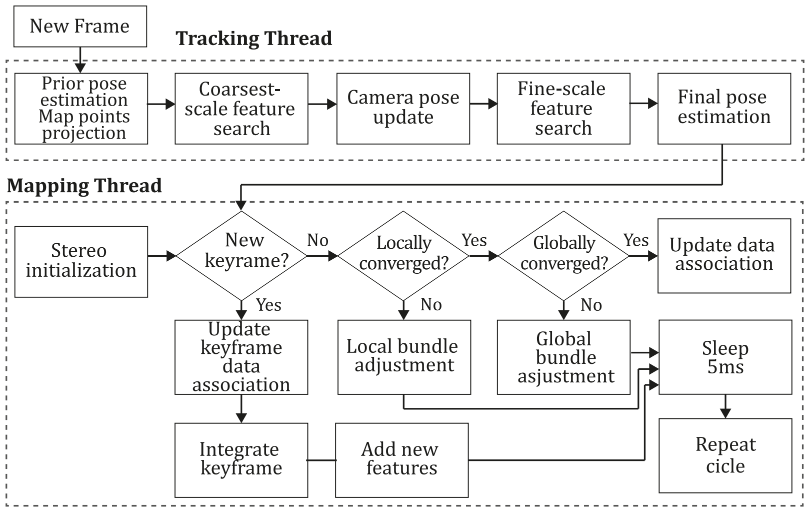


Figure 9. Diagram of PTAM algorithm. Adapted from [58].

OpenMVG (2013). The OpenMVG (Open Multiple View Geometry) Structure from Motion (SfM) system stands as an open-source initiative aimed at reconstructing 3D representations of objects, scenes, or structures from a collection of 2-dimensional images. These images can be simultaneously captured (via stereo setups) or sourced from a sequence of monocular images (incrementally). This system was introduced by Moulon et al. in 2013 [60] and has since been enhanced, building upon the contributions of various researchers. Notably, it integrates a hash-map algorithm, adopts a graph-based geometry verification mechanism, and is tailored to manage expansive datasets [60].

OpenMVG's operational pipeline encompasses camera calibration, feature extraction and matching, geometric filtering, 3D reconstruction, global optimization, and colourization. A defining advancement of OpenMVG over preceding SfM systems is its hash-map algorithm, which streamlines the feature extraction and matching process. This is achieved by efficiently cataloguing the positions and descriptors of features for swift access and comparison, thereby facilitating the management of voluminous datasets and expediting computation times. The system's graph-based geometry verification process also juxtaposes estimated camera poses and 3D reconstructions against a similarity graph. This graph, which encapsulates recurring geometric relationships, aids in the exclusion of erroneous matches and outliers, thereby enhancing the precision of the reconstruction. OpenMVG's prowess in managing vast datasets and its scalability are also commendable. The system can process up to 100,000 images in a singular execution and adeptly reconstruct intricate scenes and structures. Such scalability is attributed to its efficient memory utilization and data structure optimizations, such as employing locally homogenous patches and a sparsity-centric representation of feature matches. The reconstruction quality is contingent upon various factors, including camera calibration, image quality and quantity, system scalability, and feature extraction and matching precision. OpenMVG addresses these considerations through automated camera calibration, parallelized feature extraction and matching, robust camera pose, and 3D point estimation. Additionally, the system offers users the flexibility to refine the reconstruction process and furnishes various visualization tools for inspecting and modifying the 3D models.

The OpenMVG library is structured around a suite of modules presented on a user-friendly interface, facilitating effortless configuration. These modules span image processing, feature extraction and description, feature and image collection matching, multiple view geometry, robust estimation, structure from motion, and localization. Each module is meticulously designed to perform specific tasks, ranging from image encoding and processing to feature detection, description, and matching, as well as geometric constraint verification on matched pairs. The robust estimation module is adept at identifying and discarding corrupted or noisy image pairs. The Structure from Motion module enables OpenMVG to produce 3D reconstructions through global or incremental pipelines, even with images exhibiting minimal cross-coverage. Notably, the incremental approach is recognized for its susceptibility to drift due to its sequential nature. Bundle Adjustment is executed as a concluding step within the SFM module to refine the SFM scene by minimising reprojection error. Figure 10 offers a visual representation of the OpenMVG algorithm inspired by the article [61].

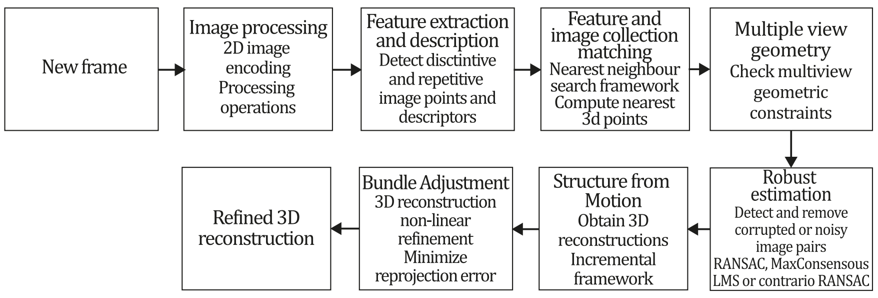

New frame
Image processing
2D image encoding
Processing operations
Feature extraction and description
Detect disctintive and repetitive image points and descriptors
Feature and image collection matching
Nearest neighbour search framework
Compute nearest 3d points
Multiple view geometry
Check multiview geometric constraints
Robust estimation
Detect and remove corrupted or noisy image pairs
RANSAC, MaxConsensous LMS or contrario RANSAC
Structure from Motion
Obtain 3D reconstructions
Incremental framework
Bundle Adjustment
3D reconstruction non-linear refinement
Minimize reprojection error
Refined 3D reconstruction

Figure 10. Diagram of OpenMVG algorithm. Adapted from the article [61].

ORB-SLAM (2015). Several years after earlier implementations, a notable monocular system named ORB-SLAM was introduced by Mur-Artal et al. [33]. This system was built based on the utilization of ORB features, characterized as multiscale FAST corners accompanied by a 256-bit descriptor. Intriguingly, this descriptor was consistently employed across various processes, including tracking, mapping, relocalization, and loop closing, primarily owing to its rapid computation and matching capabilities. The ORB-SLAM system execution starts with extracting features, a preliminary step, thereby categorizing ORB-SLAM as an indirect methodology. Within this phase, the input undergoes preprocessing to distil ORB features from prominent key-point locales. Subsequent processes are executed based on these extracted features, rendering the remaining input image data redundant. Similarly to PTAM, ORB-SLAM incorporates bundle adjustment, which is widely regarded as the gold standard for SFM due to its efficacy. Historically, the computational intensity of Bundle Adjustment rendered it impractical for real-time applications. Nonetheless, the authors postulated that its computational demands could be attenuated by concentrating on pertinent scene feature observations, specifically by selecting a subset of keyframes. This approach aimed to circumvent redundancy in keyframe selection and emphasized using keyframes characterized by pronounced parallax and abundant matches during loop closure. Consequently, ORB-SLAM employs an initial approximation for keyframe poses and point positioning during optimization. This system is designed to prioritize optimization centred on local map scrutiny while also retaining the capacity to execute global optimizations for loop closures.

Within this methodology, the Bundle Adjustment optimization is orchestrated for the 3-D coordinates $\boldsymbol{x}_{\omega,j} = \left(x_{\omega,j}, y_{\omega,j}, z_{\omega,j}\right)^T$. Concurrently, the poses of the keyframes, denoted as $\boldsymbol{T}_{i\omega}$ are refined by minimizing the reprojection error associated with the $x_{i,j}$ points. Thus, the error attributed to a point j within a keyframe i is articulated as:

$$
\begin{gathered}
e_{i,j} = x_{i,j} - \pi_i\left(\boldsymbol{T}_{i\omega}, \boldsymbol{x}_{\omega j}\right), \\
\pi_i\left(T_{i\omega}, X_{\omega,j}\right) = \begin{bmatrix} f_{i,u}\dfrac{x_{i,j}}{z_{i,j}} + c_{i,u} \\ f_{i,v}\dfrac{y_{i,j}}{z_{i,j}} + c_{i,u} \end{bmatrix}, \text{and} \\
\begin{bmatrix} x_{i,j} & y_{i,j} & z_{i,j} \end{bmatrix}^T = \boldsymbol{R}_{i\omega}\boldsymbol{x}_{\omega,j} + \boldsymbol{t}_{i\omega},
\end{gathered}
\tag{2}
$$

where π_i is the projection function, $\boldsymbol{R}_{i\omega}$ and $\boldsymbol{t}_{i\omega}$ are rotation and translation components of $\boldsymbol{T}_{i\omega}$. $\left(f_{i,u}, f_{i,v}\right)$ and $\left(c_{i,u}, c_{i,v}\right)$ are the focal length and main point associated with the i camera. The objective function targeted for minimization is articulated as: $C = \sum_{i,j} H_h(e_{i,j}^T, \Omega_{i,j}^{-1} e_{i,j})$, where H_h represents the Huber robust cost function and $\boldsymbol{\Omega}_{i,j} = \sigma_{i,j}^2 \mathrm{I}_{2\times 2}$ denotes the covariance matrix corresponding to the scale of each identified keypoint.

ORB-SLAM is an intricate system, synthesized from foundational systems such as PTAM [58], the place recognition methodology termed Bags of Words [62], the scale-aware loop closing technique [63], and the integration of co-visibility information explored in both [64], [65]. The amalgamation and enhancement of these techniques culminated in a novel system. Its notable contributions encompass: the uniform application of features for tracking, mapping, relocalization, and loop closing in real-time, exhibiting commendable resilience to alterations in viewpoint and lighting; the deployment of a local covisibility graph for tracking and mapping, ensuring map size independence and facilitating real-time operations in expansive environments; the adoption of a loop-closing method anchored in pose graph optimization; the capability to recuperate from tracking discrepancies; an automatic initialization predicated on model selection; and the strategic selection of map points and keyframes via the "survival of the fittest" approach. Mirroring PTAM, ORB-SLAM displays three concurrent threads dedicated to tracking, local mapping, and loop closing. The tracking thread is tasked with camera localization for each frame and discerns the opportune moments for keyframe insertion. The local mapping thread processes each emergent keyframe and conducts bundle adjustment to reconstruct proximate 3D elements. The loop-closing thread is responsible for identifying loops with every new keyframe and computing a similarity transformation to gauge the accumulated drift upon loop detection. Furthermore, ORB-SLAM has instituted a flexible policy for the generation and removal of keyframes, facilitating dynamic map expansion and the efficient identification and elimination of superfluous keyframes. Empirical results underscored that one of the primary advantages of indirect methodologies is their proficiency in feature matching, even across extensive baselines. As per Mur-Artal et al. [33], precision can be further augmented by incorporating points at infinity into the tracking, which encapsulates pivotal rotational data. Potential enhancements include utilising a denser map or leveraging the system as a foundational structure upon which a more precise dense map can be constructed. Figure 11 offers a visual depiction of the ORB-SLAM algorithm inspired by the article [33].

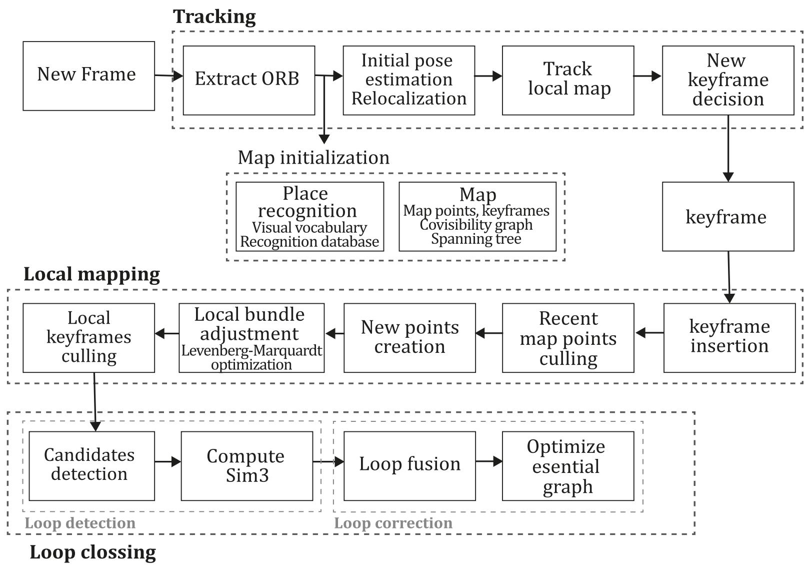


Figure 11. Diagram of ORB-SLAM algorithm. Adapted from [33].

COLMAP (2016). COLMAP, a method tailored for photogrammetry and computer vision, endeavours to reconstruct a 3D scene from an array of 2D images. Its core approach is estimating camera orientations and 3D scene points by optimizing a bundle adjustment challenge. The method capitalizes on feature-based matching to establish inter-image correspondences and subsequently refines these matches, considering geometric consistency to eliminate outliers. Notably, COLMAP introduces advancements in the optimization phase, incorporating a unique parameterization for rotation and leveraging a more resilient optimization strategy based on the Levenberg-Marquardt algorithm.

Structure from Motion (SFM) techniques conventionally unfold in two phases: correspondence search and incremental reconstruction. The former phase discerns overlaps in input images and projects identical points in overlapping images, constructing a graph of image projections for each point. In this context, COLMAP performs feature extraction to pinpoint sets of local features resilient to radiometric and geometric alterations, facilitating their recognition across multiple images. Subsequent matching identifies images capturing identical scene segments by seeking the most analogous features in each image. Given that matching is solely appearance-based, geometric verification becomes imperative to ensure that feature correspondences genuinely represent the same scene point. This verification is achieved by estimating transformations; transformations mapping many features are verified. COLMAP's second phase, incremental reconstruction, leverages the scene graph from the correspondence search to retrieve pose estimates and depict the scene structure as a point cloud. This phase starts with an initialization process, essential for averting trajectory loss complications. Image registration enables the system to incorporate new images by resolving the Perspective-n-Point (PnP) problem using previously discerned feature correspondences of triangulated points. Triangulation subsequently augments the scene's point cloud representation. Given SFM's propensity to rapidly deviate to irrecoverable states, bundle adjustment refines camera and point parameters by minimizing reprojection errors.

COLMAP's seminal contributions to the SFM process encompass enhancements in image registration, triangulation, and bundle adjustment procedures. The method introduces a robust next-best image selection technique, improving pose estimation and ensuring reliable triangulation. This approach employs a multi-resolution analysis underpinned by an S score, which elevates when more points are discernible and uniformly distributed. Furthermore, using the RANSAC approach, COLMAP introduces an innovative triangulation procedure designed for heightened resilience to outliers and the amalgamation of independent points into singular tracks. Notably, COLMAP's bundle adjustment is integrated into image registration and triangulation, facilitating local bundle adjustment for each image registration and global adjustments upon significant model expansion.

When juxtaposed with analogous methods, COLMAP exhibits superior accuracy, efficiency, and scalability performance metrics. Remarkably, it adeptly manages extensive image collections, spanning hundreds of thousands of images, without compromising reconstruction quality. Furthermore, COLMAP excels in reconstructing scene geometry, an essential attribute for virtual and augmented reality applications. Its versatility in processing diverse input data types, ranging from unordered image sets to video frames, renders it apt for many applications, spanning from 3D cultural heritage modelling to robotic vision. Additionally, COLMAP offers an exhaustive toolkit for visualizing and scrutinizing reconstruction outcomes, including point cloud visualization, texture mapping, and error evaluation. Figure 12 delineates the COLMAP algorithm inspired by the article [66].

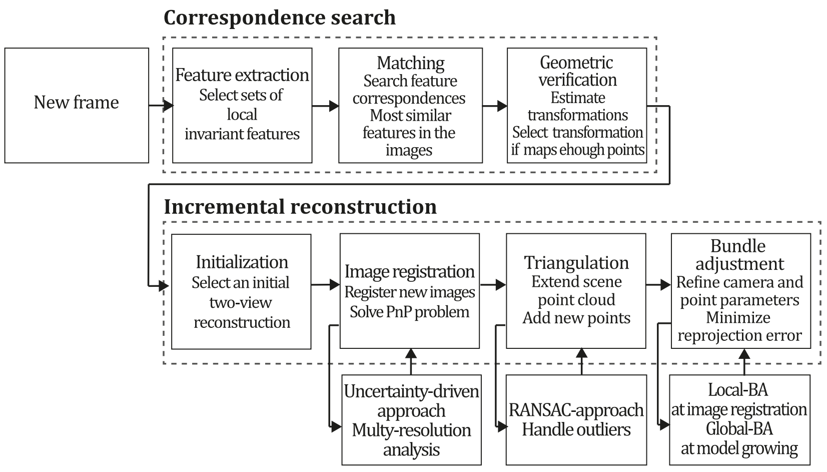

Correspondence search
New frame
Feature extraction
Select sets of local invariant features
Matching
Search feature correspondences
Most similar features in the images
Geometric verification
Estimate transformations
Select transformation if maps ehough points
Incremental reconstruction
Initialization
Select an initial two-view reconstruction
Image registration
Register new images
Solve PnP problem
Triangulation
Extend scene point cloud
Add new points
Bundle adjustment
Refine camera and point parameters
Minimize reprojection error
Uncertainty-driven approach
Multy-resolution analysis
RANSAC-approach
Handle outliers
Local-BA at image registration
Global-BA at model growing

Figure 12. Diagram of COLMAP algorithm. Adapted from [66].

ORB-SLAM2. After their initial work on ORB-SLAM, Mur-Artal and Tardós further advanced their research in developing ORB-SLAM2 [7]. This enhanced system broadened the operational scope of its antecedent, facilitating compatibility with monocular, stereo, and RGB-D sensors. While its foundation was primarily based on the original ORB-SLAM, it incorporated additional functionalities, such as stereo matching techniques for stereo cameras and stereo coordinate generation for RGB-D sensors. A significant innovation in the monocular configuration was the introduction of a fourth thread dedicated to executing a comprehensive Bundle Adjustment after the loop closure pose-graph optimization, aiming for an optimal structure and motion resolution. Given the intensive computational demands of this optimization, which encompasses all points and features, it was designated to a distinct thread. This strategic allocation ensured the system's uninterrupted ability to expand the map and identify loops concurrently. Similarly to the design of ORB-SLAM, ORB-SLAM2 integrated the DBoW2 place recognition module for relocalization and employed a co-visibility graph tailored for expansive environments. In the context of stereo and RGB-D configurations, monocular key points were retained for rotation and translation estimations. However, they were not utilized for scale information, as the respective sensor could either directly measure or triangulate it. The system adopted the Levenberg-Marquardt optimization, facilitated by the g2o module, to refine the camera pose during tracking, optimize the local window of keyframes, and refine points within the mapping thread, as well as all keyframes and points post loop closure. Consequently, the comprehensive Bundle Adjustment essentially represents a variant of the local Bundle Adjustment, wherein all map points and keyframes undergo optimization, barring the origin keyframe.

Furthermore, ORB-SLAM2 introduced a localization mode, wherein the local mapping and loop closing threads are suspended in familiar terrains, provided the environment remains relatively unchanged. This feature paves the way for enduring and efficient localization capabilities. When benchmarked against contemporary Stereo, RGB-D, and monocular systems, ORB-SLAM2 demonstrated superior performance across datasets such as EuRoC [67], TUM [68] (Sturm et al., 2012), and KITTI [69]. This underscored its adaptability across a diverse range of environments. The study further postulated the potential applicability of ORB-SLAM2 to an array of novel sensors, encompassing omnidirectional and fisheye cameras and expansive dense fusion. Figure 13 offers a visual representation of the ORB-SLAM2 algorithm inspired by the article [7].

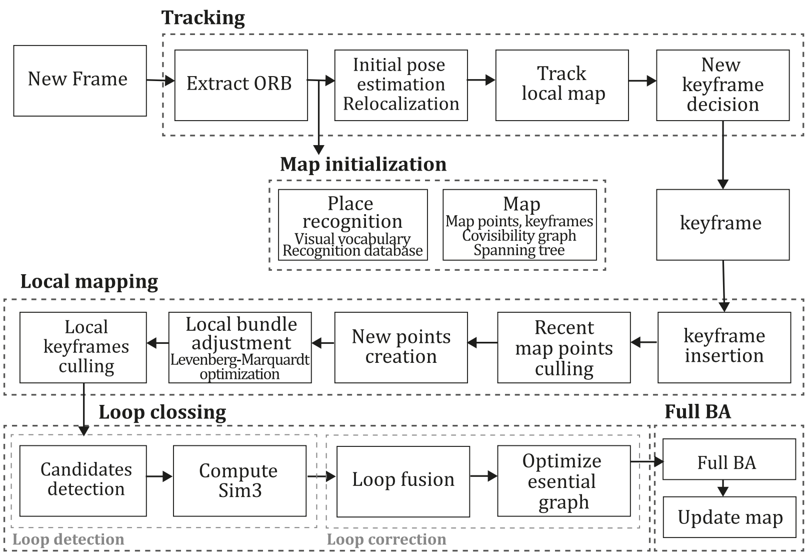


Figure 13. Diagram of ORB-SLAM2 algorithm. Adapted from [7].

ORB-SLAM3. ORB-SLAM has long been recognized as the gold standard in feature-based monocular SLAM systems, serving as the foundation for subsequent innovations such as ORB-SLAM-VI and ORB-SLAM3 [70], [71]. The most recent iteration, ORB-SLAM3, introduced by Campos et al. [71], amalgamates and refines the advancements of its forerunners. This system executes visual, visual-inertial, and multimap SLAM, compatible with monocular, stereo, and RGB-D cameras, encompassing pin-hole and fisheye configurations. A notable enhancement in ORB-SLAM3 is its proficiency in four distinct data association types: short-term, which facilitates matching of map elements from recent moments; mid-term, targeting map elements proximate to the camera with minimal accumulated drift; long-term, which identifies map elements from earlier explored areas using a place recognition technique, irrespective of accumulated drift; and multimap, which correlates and employs BA map elements from prior mapping sessions, thereby constructing a map conducive to precise localization.

Key innovations distinguishing ORB-SLAM3 from its predecessors include: a visual-inertial system for both monocular and stereo configurations grounded in Maximum-a-Posteriori estimation (MAP), enhancing ORB-VI through the initialization approach delineated in [72]; an advanced place recognition mechanism that augments recall by assessing geometrical consistency with three co-visible map keyframes, albeit with a slight computational overhead; the ORB-SLAM atlas, a multimap SLAM framework inspired by [73], which encapsulates an array of disjointed maps applicable to various map operations; and an abstract camera model, enabling compatibility with diverse camera types given the provision of their projection and un-projection Jacobian functions.

A pivotal advancement is the ATLAS system, which maintains an active map for the tracking thread to localize incoming frames, perpetually refining the Atlas with novel keyframes. Concurrently, non-active maps remain poised for integration. The tracking thread determines the pose of the current frame relative to the active map, minimizes reprojection discrepancies, and discerns which frames qualify as keyframes. The local mapping thread incorporates keyframes and points into the active map, excises superfluous points, and fine-tunes the map via bundle adjustment, considering a spectrum of adjacent keyframes. The loop and map merging thread identify shared regions within the Atlas and, upon detection, initiates loop correction, followed by a separate thread executing a comprehensive BA to refine the entire map. When benchmarked against an extensive array of monocular, stereo, monocular inertial, and stereo inertial methodologies on the EuRoC and TUM-VI datasets, utilizing RMSE (Root Mean Square Error) and ATE (Absolute Trajectory Error) metrics, ORB-SLAM3 consistently outperformed its counterparts in the majority of sequences. However, it's worth noting that ORB-SLAM3 encountered challenges in environments lacking texture, scenarios characterized by slow motion, or purely rotational applications. Figure 14 offers a visual depiction of the ORB-SLAM3 algorithm inspired by the article [71].

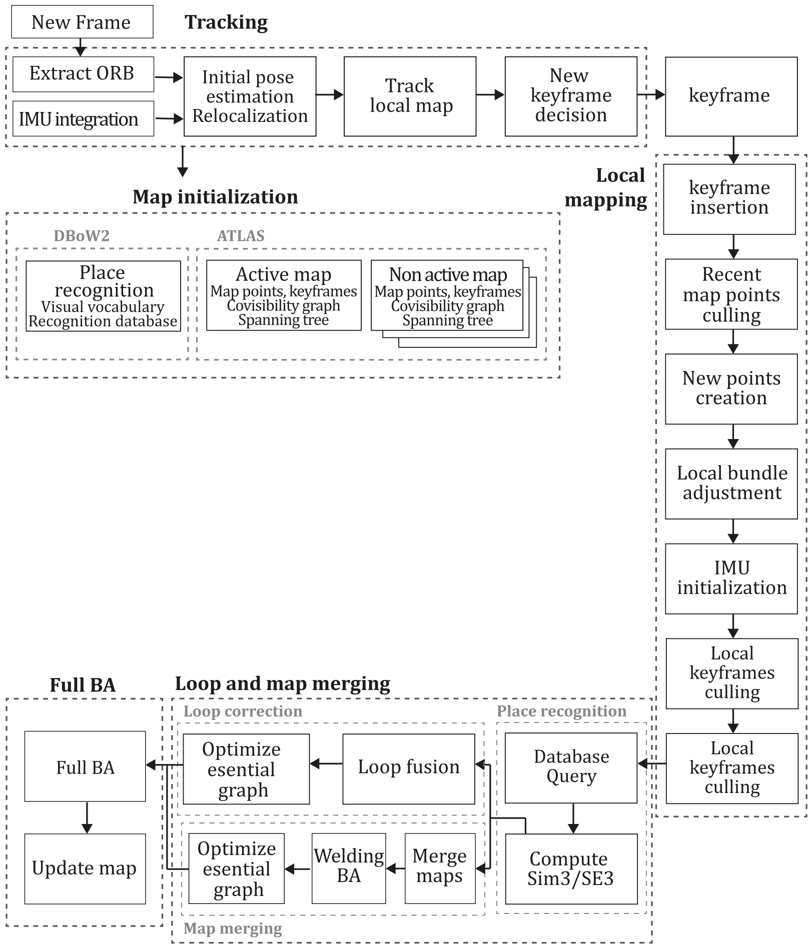


Figure 14. Diagram of ORB-SLAM3 algorithm. Adapted from [71].

6.1.2. Classic + Dense + Indirect methods

The primary objective of this category is to deduce 3D geometry either directly from a regularized optical flow field or in association with it. This involves balancing the geometric error (the divergence from the flow field) with the acquired geometric prior, typically characterized by the flow field's smoothness [2]. As previously indicated, such monocular techniques necessitate a substantial volume of input data, leveraging most pixel values to execute their inherent operations. Specifically, dense monocular strategies bypass the need to extract a specific feature subset, as they operate directly on the complete input. Consequently, while these techniques eliminate the need for discrete features, they entail significant computational overhead due to the extensive data volume being processed. Furthermore, methods within this category are typically classified as indirect, given that many rely on preliminary optical flow data acquired during an initial processing phase.

Valgaerts et al. (2011). In 2012, Valgaerts et al. [74] performed an analytical comparison between dense and sparse methodologies. Within this research, the scholars introduced a variational model for dense 3D reconstruction, aiming to deduce the fundamental matrix and optical flow by minimising a singular energy function. The study further delved into the contrasts between sparse feature-centric techniques, predominantly employed for epipolar geometry estimation, and the dense energy-driven methods frequently utilized for determining correspondences in an image sequence. The primary objective of that exploration was to underscore the potential of dense optical flow techniques in estimating epipolar geometry. This led to the proposition of a combined variational approach that concurrently estimates both the epipolar geometry and optical flow.

For comparative purposes, the researchers evaluated systems that leveraged feature-based techniques, specifically those utilizing the Scale Invariant Feature Transform (SIFT) and the Kanade-Lucas-Tomasi (KLT) feature matching algorithms. Additionally, the efficacy of the Random Sampling Consensus (RANSAC) and the Least Median of Squares (LMedS) methods in estimating the fundamental matrix was scrutinized. Extensions of RANSAC, namely LORANSAC (local optimization RANSAC) as presented by Chum et al. [75] and DEGENSAC (degenerate configurations RANSAC) as outlined by [76], were also subjected to analysis. Through a series of tests, it was discerned that dense estimation techniques, when applied to epipolar geometry, offer superior results compared to sparse methods, especially in scenarios where features lack precise localization or when a minimal set of out-of-plane correspondences is essential to address degeneracy challenges. Furthermore, the team applied their variational model across a spectrum of applications. A notable application involved automatic 3D reconstruction, which was achieved by deriving the camera projection matrices from the estimated fundamental matrix and subsequently triangulating the back-projected ray for each pixel. Alternatively, a projective transformation was employed without supplementary camera data or scene information. This innovative approach facilitated simultaneous 3D reconstruction by resolving dense epipolar geometry and two-image optical flow, associating a distinct 3D point in space with every image pixel. As a result, the researchers attained enhanced precision and robustness compared to isolated epipolar and optical flow estimations. Nonetheless, it's imperative to note that the efficacy of this method is contingent upon an image sequence that facilitates stable estimation, rendering it most suitable for rigid applications devoid of moving entities. Figure 15 provides a visual representation of the algorithm conceptualized by Valgaerts et al., as detailed in their publication [74].

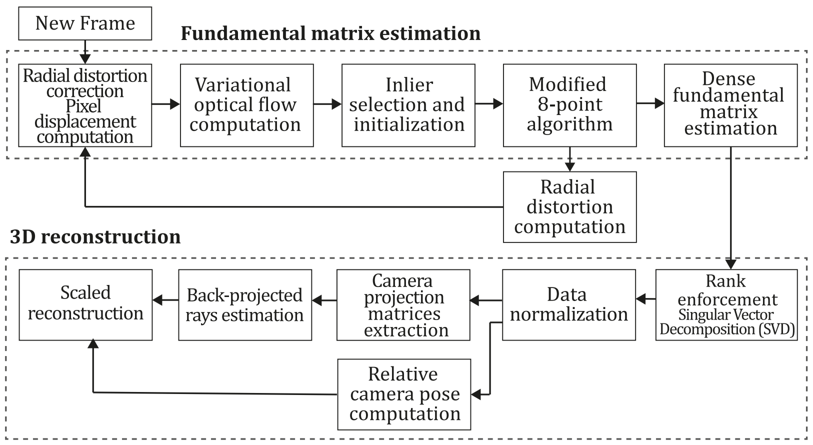

New Frame
Fundamental matrix estimation
Radial distortion correction Pixel displacement computation
Variational optical flow computation
Inlier selection and initialization
Modified 8-point algorithm
Dense fundamental matrix estimation
Radial distortion computation
3D reconstruction
Scaled reconstruction
Back-projected rays estimation
Camera projection matrices extraction
Data normalization
Rank enforcement Singular Vector Decomposition (SVD)
Relative camera pose computation

Figure 15. Diagram of Valgaerts et al., algorithm. Adapted from [74].

Ranftl et al. (2016). One of the prominent contributions in the depth estimation research field is attributed to Ranftl et al. [77]. Their work, titled "Dense Monocular Depth Estimation in Complex Dynamic Scenes", presents a system adept at deriving a comprehensive depth map for stationary and moving objects, utilizing merely two successive frames. The methodology hinges on a segmentation algorithm that segments the optical flow, resulting in an array of motion models. Each of these models possesses its distinct epipolar geometry. After this, the scene undergoes reconstruction by optimizing a convex problem. The depth estimation process in Ranftl et al.'s method unfolds in two distinct phases. The initial phase, termed motion segmentation, involves segmenting a dynamic scene into multiple moving models, each accompanied by its epipolar geometry. This is executed within an optical flow field and is conceptualized as a variational labelling challenge. The subsequent phase, termed reconstruction, entails the reassembly of the scene. This is achieved by collectively determining the scale and positioning of various components concerning the camera. This is further facilitated through object triangulation and the subsequent reconstruction of all its inherent objects.

Delving deeper into the motion segmentation phase, the dynamic scene is decomposed into a collection of independent rigid motions. Each motion is characterized by its fundamental matrix and a per-pixel designation. This intricate procedure, conceived as a joint estimation labelling challenge, mandates a dense optical flow field, denoted as $F = \left(f_x, f_y\right)$, spanning between images I_1 and I_2. This results in a soft u_l assignment where each pixel is allocated to a specific l label, representing distinct M_l motion models. Alternatively, there's provision for an additional $l + 1$ outlier label. Consequently, the formulation is articulated as:

$$\begin{gathered}(u_l^*, M_l^*) = \arg min_{u_l, F_l} \sum_{l=1}^{L+1} u_l \cdot g(F_l) + \|W_l \nabla u_l\|_{2,1}, \\ subject\ to \quad \sum_{l=1}^{L+1} u_l^i = 1, \quad u_l^i \geq 0, \\ \forall l. rank(F_l) = 2,\end{gathered} \tag{3}$$

$$g^i(M_l) = d\left(x_1^i, M_l x_2^i\right)^2 + d\left(x_2^i, M_l^T x_1^i\right)^2, \tag{4}$$

where, $g^i\left(M_l\right)$ denotes the symmetric distance to the epipolar lines for each $l \in \{1 \ldots L\}$ model, $x_1^i = \left[x^i, y^i, 1\right]^T$and $x_2^i = [x^i - f_x^i, y^i - f_y^i, 1]$ represent the homogeneous coordinates in the initial image and their corresponding coordinates in the subsequent image, respectively. The term $\|W_l \nabla u_l\|_{2,1}$ serves as the smoothness component, where ∇ is a linear operator signifying the discrete differential between x and y coordinates. The matrix W_l acts as a diagonal weighting mechanism, introduced to facilitate edge-preserving regularization. Subsequently, the energy encapsulated in the equation undergoes optimization through a modified version of the primal-dual algorithm. This adaptation incorporates entropy proximal terms, which implicitly signify simplex constraints, thereby streamlining the problem-solving process for labelling. As a result, the fundamental matrices M_l are concurrently decomposed across all L models. The soft assignments are denoted as u^i, are then employed to adjust the weighting of individual correspondence:

$$\begin{gathered}M_l^* = \arg min_{F_l} \sum_{i=1}^{M} u_l^i \left(\left(x_1^i\right)^T M_l \left(x_2^i\right)\right)^2, \\ subject\ to \quad rank(M_l) = 2,\end{gathered} \tag{5}$$

subsequent to the initial procedures, the subproblems are addressed using a reweighted adaptation of the 8-point algorithm, as delineated in [78]. To ascertain the count of dynamic models, the methodology encompasses several sequential actions: employ the 8-point algorithm to extract a preliminary set of candidates; address the energy equation by augmenting the assortment of motion candidates; introduce new models by robustly determining motion from pixels labelled as outliers; enlarge the candidate pool by segregating labels associated with unconnected regions; undertake alternating minimization iteratively until no further energy reduction is feasible. Upon the culmination of these steps, a collection of epipolar geometries denoted as F_l^*, along with membership likelihood u_l^* for each pixel, is derived. The final phase involves a sturdy reconstruction, leveraging these epipolar models in conjunction with optical flow data facilitated by a super-pixel-oriented formulation.

The efficacy of this system was rigorously evaluated using the KITTI [69] and MPI Sintel [79] datasets. The results showcased its superiority over a majority of the contemporary techniques aimed at discerning dynamic scene geometry from monocular videos. Various methodologies for optical flow computation, including Large Displacement Optical Flow (LDOF), EpicFlow, and FlowFields, were employed for these tests. A critical observation was that the system's performance is intrinsically tied to the accuracy of optical flow estimation; a failure in the latter would compromise the entire system. Additionally, the method exhibits constraints characteristic of a purely geometric approach, notably the omission of prior knowledge regarding shapes and scene dimensions. Consequently, the authors advocate for integrating Machine Learning (ML) techniques to more accurately determine the absolute scale. Figure 16 offers a visual representation of the algorithm conceptualized by Ranftl et al., as detailed in their publication [77].

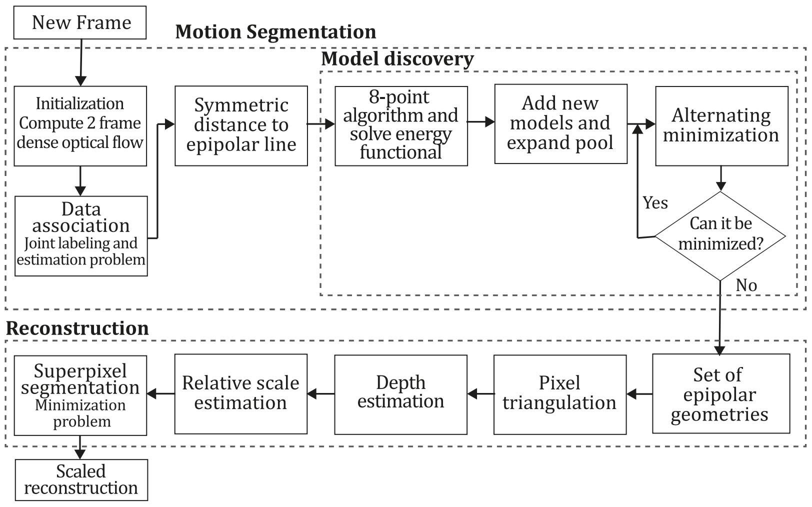

New Frame
Motion Segmentation
Model discovery
Initialization Compute 2 frame dense optical flow
Symmetric distance to epipolar line
8-point algorithm and solve energy functional
Add new models and expand pool
Alternating minimization
Data association Joint labeling and estimation problem
Yes
Can it be minimized?
No
Reconstruction
Superpixel segmentation Minimization problem
Relative scale estimation
Depth estimation
Pixel triangulation
Set of epipolar geometries
Scaled reconstruction

Figure 16. Diagram of Ranftl et al., algorithm. Adapted from [77].

6.2. Classic + Direct methods

Direct methodologies have been developed to recover the scene's geometry and the observer's movement by utilizing the raw data from pixel intensities. Unlike their indirect counterparts, these methods were built to work without initial steps such as feature extraction. This omission conserves computational resources and accelerates the process, albeit at the expense of a less dense three-dimensional reconstruction (3D reconstruction). By engaging with the intensity values of each pixel—or a substantial majority thereof—direct strategies tend to yield a 3D reconstruction of superior quality compared to indirect methods. Nonetheless, direct methods presuppose brightness constancy across various viewing angles on an object's surface—a postulate known as the brightness constancy assumption. This assumption does not hold in certain conditions, such as motion blur, dynamic objects, or non-Lambertian surfaces present, leading to potential failures in these methods.

Furthermore, direct methods can be categorized based on the density of the resulting 3D map into dense and sparse types, which will be further explored in Sections 4.2.1 and 4.2.2. A historical timeline of significant contributions to this field is depicted in Figure 17. Notably, as illustrated in Figure 17, three seminal monocular systems—DTAM [17], LSD-SLAM [4], and DSO [2]—are distinguished within this domain. It is important to acknowledge that despite DSO's recent introduction, it has garnered considerable attention from the research community, as reflected in its remarkable citation metrics.

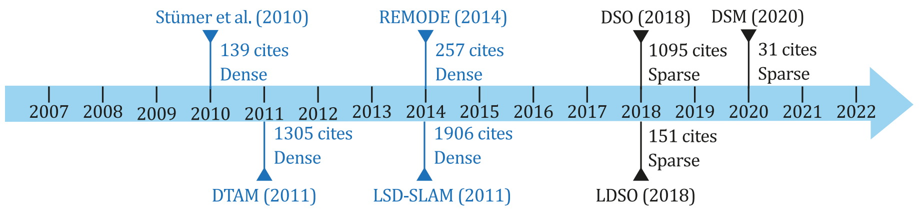


Figure 17. A timeline for the most representative monocular SLAM, VO, or SFM classic direct systems.

6.2.1 Classic + Dense + Direct methods

Approaches within this segment of simultaneous localization and mapping are categorized as dense, reflecting their utilization of the entirety, or a majority, of the input data for reconstructing the environment. Moreover, they are identified as direct techniques, owing to the absence of preliminary processing phases, allowing the immediate use of the input image to deduce the geometry of the surroundings. Typically, these methods employ photometric error assessments and geometric priors to ascertain either dense or semi-dense spatial configurations by operating directly on the information derived from pixel intensities.

Stühmer et al. (2010). Stümer, Gumhold, and Cremers pioneered the field with their early work, "Real-Time Dense Geometry from a Handheld Camera" [80], introducing a real-time variational framework for directly estimating dense depth maps from multiple images. This technique employs data terms associated with the coordinate system of a chosen viewpoint and leverages the perspective projection to transpose these coordinates onto a second camera's frame. Consequently, the authors formulated an energy function designed for deriving depth maps utilizing a series of images.

$$E(h) = \lambda \int_{Y} \sum_{i \in \mathfrak{T}(x)} |\rho_i(\boldsymbol{x}, \boldsymbol{D})| d^2\boldsymbol{x} + \int_{Y} |\nabla \boldsymbol{D}| d^2\boldsymbol{x}, \tag{6}$$

where, $\boldsymbol{D}$ represents the depth map, and $\boldsymbol{x} = (x_1, x_2, 1)^T$ denotes the homogeneous 2D coordinates. The term $d(\boldsymbol{x}, \boldsymbol{D})$ specifies the depth value corresponding to each pixel. Y_i refers to the image plane, while $\boldsymbol{T}_i$ signifies the camera pose. Here, $\mathfrak{T}(\boldsymbol{x})$ encompasses all image indices for which the perspective projection $\pi(\exp(\hat{\boldsymbol{T}}_\iota) \cdot d(\boldsymbol{x}, \boldsymbol{D}))$ falls within the image boundaries. Furthermore, $\rho_i(\boldsymbol{x}, \boldsymbol{D})$ is defined as the linearized residual data term for the image I_i.

$$\rho_i(x, \boldsymbol{D}) = I_i(\boldsymbol{x}, \boldsymbol{D}_0) + (\boldsymbol{D} - \boldsymbol{D}_0) I_i^{\boldsymbol{D}}(\boldsymbol{x}) - I_0(\boldsymbol{x}), \tag{7}$$

where $I_i^{\boldsymbol{D}}(x)$ represents the derivative $\frac{d}{d\boldsymbol{D}} I_i(\boldsymbol{x}, \boldsymbol{D})|_{\boldsymbol{D}_0}$. The described formulation integrates direct pixel information into the energy function, from which the depth map $\boldsymbol{D}$ is derived through a minimization process, characterizing it as a direct method. The complexity of this energy function is evident, as the data term is composed of the aggregate of absolute values of linear functions, which are not amenable to simplification through basic thresholding techniques. Consequently, the authors have also suggested an approach for generalized thresholding. The advantage of this dense, multi-view proposal, as compared to techniques that rely on only two images, is that multiple perspectives can contribute to the estimation of disparity in regions that may be occluded in some views but visible in others, thereby integrating information from unobstructed images. An additional benefit is enhancing the signal-to-noise ratio, which bolsters the quality of results when the input images are compromised by noise—a common issue with footage from standard webcams or handheld devices.

In essence, this method diverges from real-time pose estimation and instead calculates the depth map using the present input image in conjunction with the N keyframes closest to the current pose. The method effectively reduces noise impact by employing estimates of the camera pose from these keyframes. This approach was incorporated into the camera tracking module of an existing PTAM system [32], which can store keyframes. Each camera pose linked to a keyframe is refined iteratively, as is the depth map associated with its respective keyframe, utilizing the N nearest keyframes for refinement. Figure 18 graphically represents the Stühmer et al. algorithm inspired by their study [80].

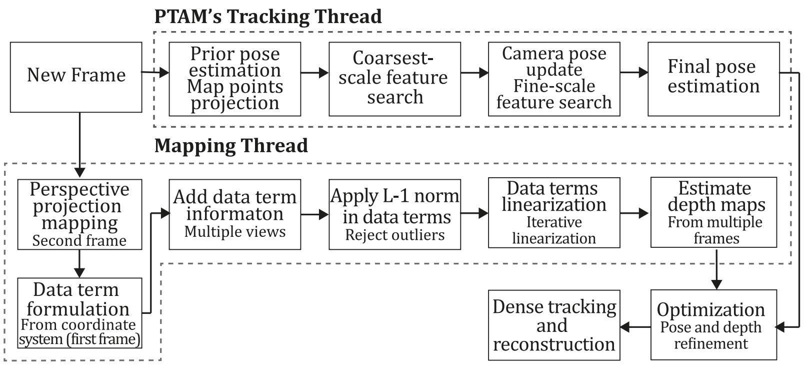

PTAM's Tracking Thread
New Frame
Prior pose estimation Map points projection
Coarsest-scale feature search
Camera pose update Fine-scale feature search
Final pose estimation
Mapping Thread
Perspective projection mapping Second frame
Add data term informaton Multiple views
Apply L-1 norm in data terms Reject outliers
Data terms linearization Iterative linearization
Estimate depth maps From multiple frames
Data term formulation From coordinate system (first frame)
Dense tracking and reconstruction
Optimization Pose and depth refinement

Figure 18. Diagram Stühmer et al. algorithm. Adapted from [80].

DTAM (2011). In 2011, Newcombe et al. introduced a dense direct methodology, characterized by its 'every-pixel' approach, designed to construct a comprehensive 3D model. This model is generated through a detailed, sub-pixel-level reconstruction that facilitates precise camera tracking. The technique involves using the dense model to align the full image with the current model, thereby enabling camera motion estimation. Subsequently, the model is augmented and refined by incorporating and honing dense depth maps derived from tracked images. In this approach, the dense model is associated with overlapping keyframes, with depth values d being back-projected from each pixel, categorizing it as a direct method. Within this framework, a keyframe r consists of an image I_r, a camera pose $T_{r\omega}$, and an associated cost C_r. For every pixel q_r, there is a corresponding cost error $C_r(q,d)$ for each depth value d. A substantial number of video frames of $m \in \mathfrak{T}(r)$ —representing a collection of frames in close proximity—are utilized to calculate the cost volume. The photometric error is then determined by projecting each point in the volume onto all overlapping images and aggregating the L_1 norms of each photometric discrepancy.

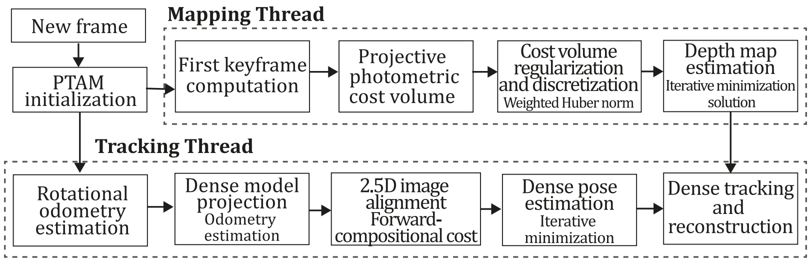
$$C_r(\boldsymbol{q},d) = \frac{1}{\mathfrak{T}(r)} \sum_{m\in\mathfrak{T}(r)} \|\rho_r(I_m,\boldsymbol{q},d)\|_1, \tag{8}$$

$$\rho_r(I_m,\boldsymbol{q},d) = I_r(\boldsymbol{q}) - I_m(\Lambda\left(\left(\boldsymbol{\zeta} T_{mr}\Lambda^{-1}(\boldsymbol{q},d)\right)\right), \tag{9}$$

where ρ_r is the photometric error for every overlapped image, $\boldsymbol{\zeta}$ is the intrinsic matrix, $\Lambda(x_c) = (x/z, y/z)^T$is de-homogenization for a 3D point $x_c = (x,y,z)^T$. The acquisition of the inverse depth map is achieved through the minimization of the energy functional E_{d^*}, which comprises a non-convex photometric error cost that serves as the data term, alongside a convex regularizer term.

$$E_{\boldsymbol{D}^*} = \int_{\Omega} \{g(\boldsymbol{q})\|\nabla d^*(\boldsymbol{q})\|_\epsilon + \lambda C\left(\boldsymbol{q}, d^*(\boldsymbol{q})\right)\} d\boldsymbol{q}, \tag{10}$$

where $g(\boldsymbol{q}) = e^{-\alpha\|\nabla \mathrm{I_r}(\boldsymbol{q})\|_2^\beta}$ assigns a weight to each pixel, where $\boldsymbol{D}^*$ signifies the inverse depth map and $\nabla d^*(\boldsymbol{q})$ is the gradient of the inverse depth map within the image domain Ω. The parameter ϵ is deliberately kept small to diminish the staircase effect. The term $\lambda = 1/(1 + 0.5d)$ gauges the quality of the data term, with α and β serving as auxiliary variables. DTAM runs over an energy minimization framework incorporating a photometric error data term and a robust spatial regularization term. It commences by establishing a projective photometric cost volume, akin to the disparity space image found in stereo matching, which is then regularized by applying a weighted Huber norm to the gradient of the inverse depth map. This model is discretized and addressed through the application of duality principles, leading to a primal-dual formulation. Here, the weighted Huber regulator is substituted with its conjugate by employing the Legendre-Fenchel transformation. This allows for the extraction of the inverse depth map by iteratively minimizing the cost volume for each pixel relative to a reference frame. Initially, DTAM employs the PTAM point feature-based method for the initial phase until the first keyframe is secured. Subsequently, the algorithm transitions to a proprietary, fully dense tracking and mapping sequence. Adding a new keyframe occurs when the number of pixels lacking visible surface data from the antecedent predicted image descends beneath a pre-defined threshold. Concisely, this method computes the camera pose in real-time by deducing motion parameters that conjure a synthetic view resembling the live video feed. Figure 19 graphically represents the DTAM algorithm inspired by the article [17].


New frame
Mapping Thread
PTAM initialization
First keyframe computation
Projective photometric cost volume
Cost volume regularization and discretization
Weighted Huber norm
Depth map estimation
Iterative minimization solution
Tracking Thread
Rotational odometry estimation
Dense model projection
Odometry estimation
2.5D image alignment
Forward-compositional cost
Dense pose estimation
Iterative minimization
Dense tracking and reconstruction
Figure 19. Diagram of DTAM algorithm. Adapted from [17].

REMODE (2014). In 2014, Pizzoli, Forster, and Scaramuzza introduced REMODE [5], "Regularized Monocular Depth Estimation." This method can compute dense depth maps through Bayesian estimation coupled with a sophisticated optimization process, effectively functioning as a depth sensor with a broad operational depth range. REMODE estimates the depth of each pixel independently using a probabilistic framework and a novel smoothing technique. This pixel-wise Bayesian depth estimation is an advancement of the methodology proposed by [81], which is further refined by incorporating a regularization step that employs a weighted Huber norm. Distinct from DTAM, REMODE leverages depth uncertainty to engage a convex formulation, thereby mitigating the impact of noise in camera localizations. In REMODE, depth determination is framed as a Bayesian estimation challenge, where triangulation is performed between a reference view and the most recently captured view. Each pixel's depth is treated as a parametric model continuously refined with each new observation, and smoothness is attained through the minimization of a regularized energy function. Moreover, REMODE adopts a probabilistic strategy in which a depth hypothesis d_k is formulated based on the observation set $\{I_k, \boldsymbol{T}_{k,\omega}\}$, by triangulating between views r and k, where $\boldsymbol{T}_{\boldsymbol{k},\boldsymbol{\omega}}$ denotes the rigid body transformation that characterizes the camera pose for each image. The depth sensor in REMODE is modeled as a distribution that merges a precise measurement, normally distributed around the true depth $\hat{d}$, with an outlier measurement that captures the depth of the targeted structure.

$$p\left(d_k|\hat{d},\rho\right) = \rho\mathcal{N}\left(d_k|\hat{d},\sigma_k^2\right) + (1-\rho)\mathcal{U}(d_k|d_{min},d_{max}), \tag{11}$$

where ρ represents the probability of a precise measurement while σ_k^2 denotes the variance associated with such a measurement. The term $p\ (\hat{d},p)$ reflects the prior, which encapsulates the pre-existing knowledge about the true depth's uncertainty and the proportion of measurements corroborating it. Subsequently, the resultant posterior is inferred as an approximation, characterized by the product of a Gaussian distribution, which accounts for the depth, and a Beta distribution, which represents the inlier ratio.

$$q\left(\hat{d},\rho|a_k,b_k,\mu_k,\tau_k^2\right) = Beta(\rho|a_k,b_k)\mathcal{N}\left(\hat{d}|\mu_k,\tau_k^2\right), \tag{12}$$

where a_k, b_k are parameters controlling Beta distribution. Then, for every $\boldsymbol{q}$ pixel μ_k and τ_k^2 are the mean depth estimation, its confidence for each observation, and the denoised depth map $F(\boldsymbol{q})$ is obtained by the following energy minimization:

$$\min_{F} \int_{\Omega} \{G(\boldsymbol{q})\|\nabla F(\boldsymbol{q})\|_{\epsilon} + \lambda\|F(\boldsymbol{q}) - D(\boldsymbol{q})\|_1\} d\boldsymbol{q}, \tag{13}$$

where $D(\boldsymbol{q})$ denotes the depth map, while $G(\boldsymbol{q})$ signifies the "G-Weighted Total Variation" weighting function, as introduced by [82]. These equations constitute the foundational methods and exemplify the integration of a probabilistic framework to derive a depth map that has been denoised, leveraging direct pixel information. REMODE employs a tracking thread that draws inspiration from the odometry system of SVO [13], which utilizes an image alignment method to ascertain the pose, relying solely on pixel intensity data. After the tracking phase, the mapping thread triangulates depth using each frame in conjunction with the reference view. Here, the depth for each pixel is conceptualized as a parametric model, the computation of which is framed as a Bayesian estimation problem. This problem incorporates a regularizer predicated on the gradient Huber norm, and the resolution is pursued iteratively through minimization. This process harnesses a primal-dual formulation and employs a gradient descent-ascent technique to converge on the solution. Figure 20 visually delineates the REMODE algorithm inspired by the article [5].

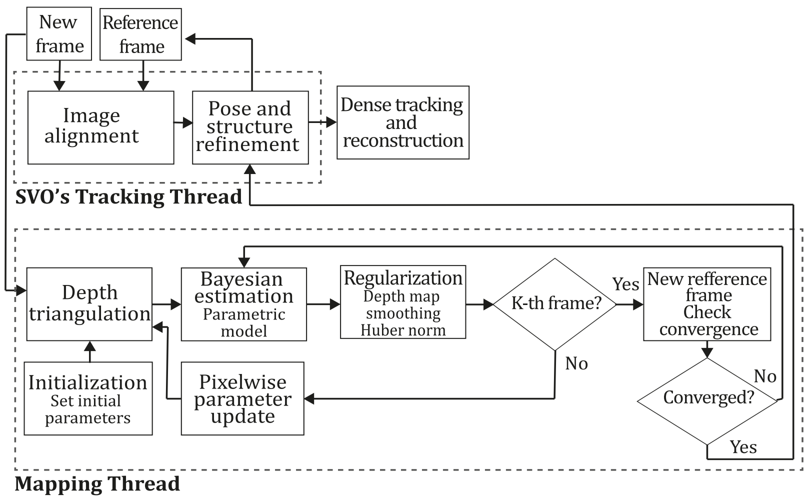


Figure 20. Diagram of REMODE algorithm. Adapted from [5].

LSD-SLAM (2014). In the study by Engel et al. [4], a real-time monocular SLAM and 3D reconstruction system was devised. This system cannot only track camera motion locally but also construct consistent, large-scale, environment-dense maps. It employs a semi-dense approach focusing on tracking depth values predominantly in areas with significant image gradients. The method is grounded in direct image alignment and a filtering-based estimation of semi-dense depth maps, building upon the earlier work of [22]. The global depth map is conceptualized as a pose graph, with keyframes serving as vertices connected by 3D similarity transforms as edges. This structure facilitates detecting and correcting scale changes and accumulated drift within the environment. LSD-SLAM incorporates an appearance-only loop detection algorithm, FAB-MAP [83], to identify candidates for substantial loop closures. It generates its features independently, without repurposing any data from the visual odometry front end. The contributions of LSD-SLAM include a direct method for aligning two keyframes within the $\xi \in sim(3)$ space, and a probabilistically consistent method for integrating the noisy uncertainty of estimated depth into the tracking process. The innovative image alignment is executed through a Gauss-Newton minimization of the photometric error.

$$E(\boldsymbol{T}) = \sum_{i} \left(I_{ref}(\boldsymbol{q}_i) - I\left(\omega\big(\boldsymbol{q}_i, \boldsymbol{D}^{*}{}_{ref}(\boldsymbol{q}_i), \boldsymbol{T}\big)\right)\right)^2, \tag{14}$$

where I denotes the images, $\boldsymbol{D}^*$ represents the per-pixel inverse depth map, $\boldsymbol{q}_i$ is a point within the image, ω is a function that performs 3D projective warping, and $\boldsymbol{T}$ encapsulates the camera pose. The comprehensive method encompasses tracking, depth map estimation, and map initialization procedures. The tracking module persistently follows new images by estimating the rigid body pose relative to the current keyframe. This relative pose is determined by minimizing the variance-normalized photometric error:

$$\min_{\boldsymbol{T} \in SE(3)} \sum_{p \in \Omega D_i} \left\| \frac{r_q^2(\boldsymbol{q}, \boldsymbol{T}_{ij})}{\sigma_{r_q}^2(\boldsymbol{q}, \boldsymbol{T}_{ij})} \right\|_\delta , \qquad (15)$$

where r_q^2 and $\sigma_{r_q}^2$ are the photometric residual and variance, respectively. Also, for adding a keyframe to the map, the closest keyframes are found, and the edges are estimated by $SE(3)$, so minimization is performed by the equation:

$$\min_{\boldsymbol{T} \in SE(3)} \sum_{\boldsymbol{q} \in \Omega \boldsymbol{D}^*_i} \left\| \frac{r_q^2(\boldsymbol{q}, \boldsymbol{T}_{ij})}{\sigma_{r_q}^2(\boldsymbol{q}, \boldsymbol{T}_{ij})} + \frac{r_d^2(\boldsymbol{q}, \boldsymbol{T}_{ij})}{\sigma_{r_d}^2(\boldsymbol{q}, \boldsymbol{T}_{ij})} \right\|_\delta . \qquad (16)$$

In summary, LSD-SLAM can generate dense depth maps by deducing the rigid body pose from camera images concerning the current keyframe, with the preceding image serving as the basis for initialization—a process known as tracking. Subsequent frames that have been tracked are then utilized to enhance or supplant the existing keyframe, thereby allowing for the refinement of depth through multiple per-pixel comparisons with small baselines, a phase referred to as depth map estimation. Ultimately, when a new keyframe supersedes the previous one as the reference for tracking and no additional refinement is conducted, it is integrated into the global map, a step known as map optimization. Figure 21 shows the LSD-SLAM algorithm inspired by the article [4].

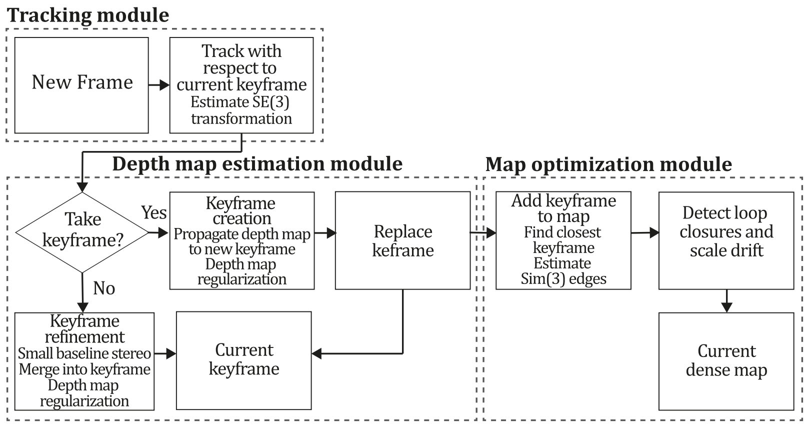


Figure 21. Diagram of LSD-SLAM algorithm. Adapted from [4].

6.2.2. Classic + Sparse + Direct methods

Formulations within this category typically optimize photometric error directly from the input frames, avoiding the need for geometric priors and preprocessing steps. A significant advantage of direct formulations is their common use of pixel-wise inverse depth, which does not necessitate individual point recognition, thereby facilitating a more refined and detailed representation of geometry. This approach also enables sampling from every pixel, capturing edges and subtle intensity variations, which enhances robustness in textured environments. Conversely, sparse methods do not require geometric priors, bypassing their constraints. Such priors, when introduced, create correlations among geometry parameters, where achieving real-time statistically consistent joint optimization is generally unfeasible. These priors may also induce bias, potentially compromising large-scale accuracy. Direct methods are not dependent on the repeatability of a set of points, allowing them to function effectively on low-texture surfaces that include contours. Many techniques in this category employ photometric bundle adjustment to reduce the photometric error of mapped point observations within a local sliding window of active keyframes. Points are sampled from pixels exhibiting significant gradients, such as edges and intensity shifts. In this realm, Visual Odometry (VO) systems often utilize sliding windows to select active keyframes that are temporally proximate, marginalizing map points that fall outside the field of view. This can be a drawback, as VO systems may not capitalize on the reobservation of map points. In contrast, sparse direct Visual Simultaneous Localization and Mapping (VSLAM) systems typically construct enduring maps of the scene, depicting a network of keyframes interconnected by observing the same region at varying times.

DSO (2017). In the early work [2], the authors introduced Direct Sparse Odometry (DSO), a sparse direct visual odometry formulation. This method synergizes the benefits of direct methods, such as the ability to reconstruct a wide array of points beyond mere corners, with the efficiency and flexibility of sparse approaches in joint optimization. The DSO algorithm is adept at tracking in low-texture environments where indirect methods often struggle. DSO executes an ongoing optimization of photometric error across a selection of recent frames, incorporating a photometrically calibrated model for image formation, drawing inspiration from [24]. Concurrently, it optimizes the full likelihood for all model parameters, including camera poses, intrinsics, extrinsics, and inverse depth values, in a process analogous to windowed sparse bundle adjustment. For successful optimization, precise initializations are crucial in the front end to address the non-convex optimization in the backend. The minimization process involves the photometric error of a point $\boldsymbol{p}$ in a reference frame I_i across a small pixel neighborhood. Empirical evidence suggests that a residual pattern spread over 8 pixels yields sufficient information for computation. The photometric error minimization is governed by the following equation:

$$E_{\boldsymbol{q}j} := \sum_{i \in \mathcal{N}_{\boldsymbol{q}}} \omega_{\boldsymbol{q}} \left\| \left(I_j[\boldsymbol{q}'] - b_j\right) - \frac{t_j e^{aj}}{t_i e^{ai}} \left(I_i[\boldsymbol{q}] - b_i\right) \right\|_{\gamma}, \tag{17}$$

where $\mathcal{N}_{\boldsymbol{q}}$ is the set of pixels, t_i and t_j are the exposure times for images I_i and I_j, respectively, and a_i, b_i, a_j and b_j are the parameters of the brightness transfer function for the poses $\boldsymbol{T}_i$, $\boldsymbol{T}_j$ of the frames involved. The gradient-dependent weighting $\omega_{\boldsymbol{q}}$ and the projected point position $\boldsymbol{q}'$, are defined as:

$$\begin{gathered} \boldsymbol{q}' = \Pi_c\left(\boldsymbol{R}\Pi_c^{-1}(\boldsymbol{q}, d_{\boldsymbol{q}}) + \boldsymbol{t}\right) \quad \text{with} \quad \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} := \boldsymbol{T}_j \boldsymbol{T}_i^{-1}, \\ \omega_{\boldsymbol{q}} := \frac{\boldsymbol{\zeta}^2}{\boldsymbol{\zeta}^2 + \|\nabla I_i(\boldsymbol{q})\|_2^2}, \end{gathered} \tag{18}$$

where $\boldsymbol{\zeta}$ denotes the camera intrinsics matrix, Π_{ζ}, Π_{ζ}^{-1} are the projection and back-projection functions, and $d_{\boldsymbol{q}}$ is the inverse depth for the projected point position. The variable i iterates over all $\mathcal{F}$ frames, $\boldsymbol{q}$ iterates over all $\mathcal{P}_i$ points of the image, and j iterates over all $obs(\boldsymbol{q})$ frames where the point is visible. Thus, the comprehensive photometric error for DSO is expressed as:

$$E_{photo} = \sum_{i \in \mathcal{F}} \sum_{\boldsymbol{q} \in \mathcal{P}_i} \sum_{j \in obs(\boldsymbol{q})} E_{\boldsymbol{q}j}. \tag{19}$$

Two modules are responsible for frame and point management manage the DSO algorithm's performance. Frame management deals with a set of active frames, tracking each new frame from the current keyframe, employing two-frame direct alignment, a multiscale image pyramid, and a constant motion model to track all points. A new keyframe is established when there is a significant change in the field of view, occlusions, disocclusions, or changes in camera exposure time. Subsequently, old keyframes are marginalized when visibility is insufficient, and the most distant keyframe is marginalized upon exceeding the maximum number of active keyframes. The point management module is tasked with selecting point candidates within the optimization window, tracking points, and activating candidate points as necessary for windowed optimization. The algorithm's efficacy was validated across three datasets, demonstrating that a larger dataset does not necessarily enhance accuracy; however, a sparse selection of points does improve accuracy and robustness. Figure 22 illustrates the DSO algorithm inspired by the publication [2].

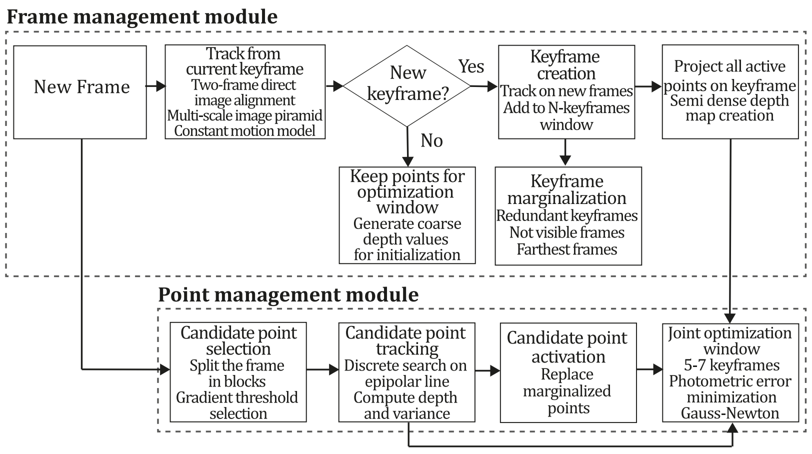

Frame management module
New Frame
Track from current keyframe
Two-frame direct image alignment
Multi-scale image piramid
Constant motion model
New keyframe?
Yes
No
Keyframe creation
Track on new frames
Add to N-keyframes window
Project all active points on keyframe
Semi dense depth map creation
Keep points for optimization window
Generate coarse depth values for initialization
Keyframe marginalization
Redundant keyframes
Not visible frames
Farthest frames
Point management module
Candidate point selection
Split the frame in blocks
Gradient threshold selection
Candidate point tracking
Discrete search on epipolar line
Compute depth and variance
Candidate point activation
Replace marginalized points
Joint optimization window
5-7 keyframes
Photometric error minimization
Gauss-Newton

Figure 22. Diagram of DSO algorithm. Adapted from [2].

LDSO (2018). In the innovative study by Gao et al. [23], the researchers expanded upon the Direct Sparse Odometry (DSO) system, creating an enhanced version known as LDSO. This extension capitalizes on DSO's ability to utilize pixels with significant intensity gradients, which ensures the repeatability of points, particularly corner features, for detecting loop closure candidates using the bag-of-words model. Depth, estimations of matched feature points are instrumental in computing $Sim(3)$ constraints, which, combined with pose-only bundle adjustment and point cloud alignment, are integrated into a co-visibility graph of relative poses derived from the DSO's sliding window optimization phase. While point selection remains a requisite in direct methods, a salient distinction between direct and indirect methods is the non-necessity of point repeatability in the former. LDSO employs corners and high-gradient pixels for tracking, with corners specifically used to generate bag-of-words (BoW) models. LDSO identifies loop closure candidates for keyframes by querying a database, excluding those within the optimization window. After this, the method endeavours to match each feature, initially estimating $SE(3)$ through RANSAC PnP, followed by the optimization of a $Sim(3)$ transformation via the Gauss-Newton method to minimize the cost function:

$$E_{loop} = \sum_{\boldsymbol{m}_i \in Q_1} \omega_1 \left\| S_{cr} \Pi^{-1}(\boldsymbol{q}_i, d^*_{\boldsymbol{q}_i}) - \Pi^{-1}(\boldsymbol{m}_i, d^*_{\boldsymbol{q}_i}) \right\|_2 + \sum_{\boldsymbol{m}_j \in Q_2} \omega_2 \left\| \Pi\left(S_{cr} \Pi^{-1}\left(\boldsymbol{q}_j, d^*_{\boldsymbol{q}_j}\right)\right) - \boldsymbol{m}_j \right\|_2, \quad (20)$$

where S_{cr} represents the loop candidate for the current keyframe, Π and Π^{-1} are the projection and back-projection functions, respectively. The weights ω_1 and ω_2 are utilized to balance the measurement units, Q_2 and Q_1 denote ORB features with and without depth information, respectively, $\boldsymbol{q}_i$ are the reconstructed features, $d^*_{\boldsymbol{q}_i}$ their inverse depths and $\boldsymbol{m}_i$ the matched features for the current frame. By amalgamating loop closure with global map optimization, LDSO successfully mitigates rotation, translation, and scale drift while preserving the tracking accuracy and robustness akin to DSO. A pivotal contribution of LDSO is the incorporation of ORB descriptors as corner trackers, which are integrated into a Bag of Words (BoW) model, significantly enhancing feature matching between keyframes. LDSO closely mirrors the original DSO framework but introduces a loop-closing module predicated on global pose graph optimization. This module operates over a sliding window of 5 to 7 DSO keyframes, allowing the system to utilize even marginalized keyframes for loop closure. This is accomplished by maintaining connections between keyframes and consistently matching ORB features across all keyframes and the current keyframe. Loop candidates are proposed by querying the keyframe database, followed by the execution of global pose graph optimization that merges the sliding window estimates with the global pose graph. Figure 23 depicts the LDSO algorithm inspired by the article [23].

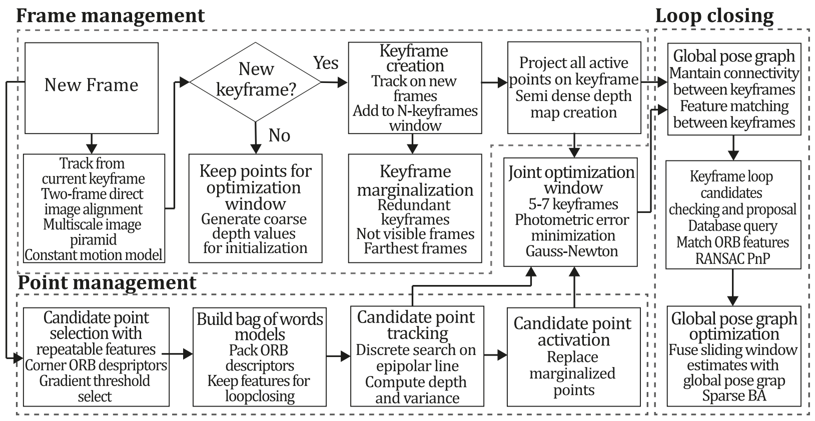

Frame management
Loop closing
New Frame
New keyframe?
Yes
No
Keyframe creation
Track on new frames
Add to N-keyframes window
Project all active points on keyframe
Semi dense depth map creation
Global pose graph
Mantain connectivity between keyframes
Feature matching between keyframes
Track from current keyframe
Two-frame direct image alignment
Multiscale image piramid
Constant motion model
Keep points for optimization window
Generate coarse depth values for initialization
Keyframe marginalization
Redundant keyframes
Not visible frames
Farthest frames
Joint optimization window
5-7 keyframes
Photometric error minimization
Gauss-Newton
Keyframe loop candidates checking and proposal
Database query
Match ORB features
RANSAC PnP
Point management
Candidate point selection with repeatable features
Corner ORB despriptors
Gradient threshold select
Build bag of words models
Pack ORB descriptors
Keep features for loopclosing
Candidate point tracking
Discrete search on epipolar line
Compute depth and variance
Candidate point activation
Replace marginalized points
Global pose graph optimization
Fuse sliding window estimates with global pose grap
Sparse BA

Figure 23. Diagram of LDSO algorithm. Adapted from [23].

DSM (2020). In the realm of visual SLAM, the work of Zubizarreta et al. [84] stands out with their introduction of direct sparse mapping (DSM), an innovative system that builds upon the foundation of Direct Sparse Odometry (DSO). As a visual SLAM variant, the DSM approach endeavours to enhance accuracy, diminish motion drift, and rectify structural inconsistencies by leveraging the information gleaned from scene reobservations. This system is distinguished as the inaugural monocular visual SLAM system capable of detecting point reobservations across entire images, thereby harnessing the comprehensive data they offer. This is in stark contrast to LDSO [23], which relies on a sparse array of feature reobservations; DSM, on the other hand, constructs a persistent map that facilitates the recycling of extant map data through a photometric approach, eschewing the need for a pose-graph or relocalization mechanisms.

DSM employs a local map co-visibility window (LMCW) criterion to identify active keyframes that observe the same region, utilizes a graduated approach for processing point reobservations with the photometric model, and adopts a robust influence function rooted in the t-distribution, coupled with a pixel-wise strategy for outlier management. This enhances the consistency of photometric bundle adjustment (PBA) in the face of outliers that may emerge as distant keyframes are activated. Like other monocular VSLAM methodologies, DSM features a tracking front end and an optimization back end. The tracking component is responsible for ascertaining the camera pose and selecting frames that could become keyframes. Concurrently, the optimization mapping thread processes each new frame to track points from active keyframes, which then serve as inputs for PBA to refine motion and structure. This thread also upholds global consistency by excising outliers, detecting occlusions, and preventing point duplication. The robust nonlinear PBA is executed using an outlier management strategy that is informed by the distribution of photometric errors, where the t-distribution has been shown to more accurately characterize dense photometric errors compared to other distributions, as evidenced by a weight function (Kerl et al., 2013b; Lange et al., 1989). Empirical evaluations have demonstrated DSM's superior performance over comparable sparse direct and indirect systems by employing a t-distribution robust influence function and utilizing four or three co-visible keyframes for LMCW across most sequences of the EuRoC dataset. Figure 24 depicts the DSM algorithm inspired by the article [84].

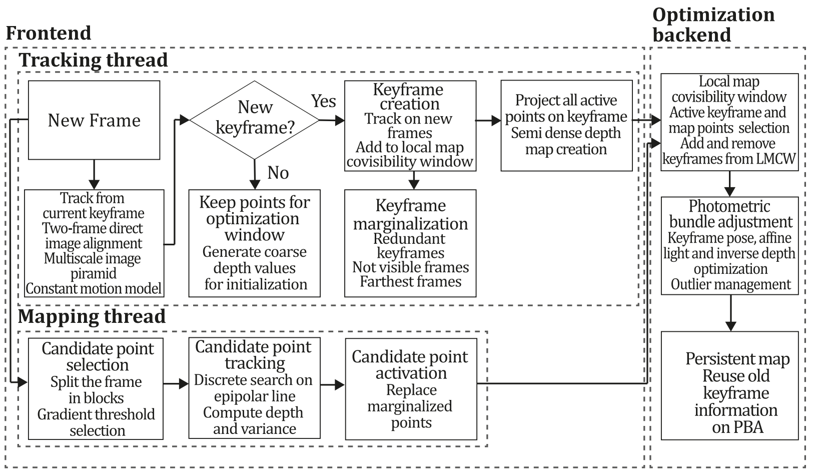

Frontend
Optimization backend
Tracking thread
New Frame
Track from current keyframe Two-frame direct image alignment Multiscale image piramid Constant motion model
New keyframe?
Yes
No
Keep points for optimization window Generate coarse depth values for initialization
Keyframe creation Track on new frames Add to local map covisibility window
Keyframe marginalization Redundant keyframes Not visible frames Farthest frames
Project all active points on keyframe Semi dense depth map creation
Local map covisibility window Active keyframe and map points selection Add and remove keyframes from LMCW
Photometric bundle adjustment Keyframe pose, affine light and inverse depth optimization Outlier management
Persistent map Reuse old keyframe information on PBA
Mapping thread
Candidate point selection Split the frame in blocks Gradient threshold selection
Candidate point tracking Discrete search on epipolar line Compute depth and variance
Candidate point activation Replace marginalized points

Figure 24. Diagram of DSM algorithm. Adapted from [84].

6.3. Classic + Hybrid methods

Within the spectrum of methodologies for monocular visual odometry, a select number of approaches can be aptly classified as a hybrid, straddling the line between direct and indirect strategies. The most prominent work in this hybrid category is the Semi-Direct Monocular Visual Odometry (SVO) introduced by Forster et al. [13], along with its subsequent variations and implementations [85], [86]. Despite the limited number of such works, they have garnered considerable attention from the research community for their adept fusion of the merits inherent in direct and indirect methods. Specifically, SVO employs an initial feature extraction only during the keyframe initialization phase; after this, the system pivots to direct motion estimation rather than relying on feature matching. This dualistic approach enables SVO to attain the precision typically associated with direct methods while benefiting from the computational efficiency characteristic of sparse indirect techniques.

SVO (2014). The publication "Fast Semi-Direct Monocular Visual Odometry" by Forster et al. [13] presents a novel visual odometry system designed to estimate ego-motion without requiring feature extraction or robust matching, instead directly utilizing pixel intensity values. Termed as semi-direct owing to its synthesis of direct and indirect methodological benefits, SVO operates on two concurrent threads: one for motion and another for mapping.

The motion thread bypasses traditional feature extraction and matching, employing direct motion estimation to establish feature correspondences. This approach minimizes the need for feature extraction, which is reserved solely for instigating new points within the map upon selecting a new keyframe. The system's preference for numerous small patches over fewer large ones enhances robustness and obviates the need for considering patch normals, echoing the findings of [87] and [88]. The motion estimation is achieved through a sparse model-based image alignment algorithm that leverages sparse point-feature data to ascertain camera motion and refine feature correspondences by minimizing photometric discrepancies. Concurrently, the mapping thread employs a Bayesian filter to manage outlier measurements and ascertain the depth at each feature location. Inserting 3D points into the map is contingent upon reducing depth filter uncertainty, thus creating an efficient tracking map of outliers and points. A notable attribute of SVO is its capability to operate at high frame rates, achieving 55 fps on embedded systems and up to 300 fps on consumer-grade laptops of that period. This performance is attributed to its probabilistic mapping method, which is resilient to outliers and its robustness in environments with redundant or sparse textures. Figure 25 depicts the SVO algorithm inspired by the article [13].

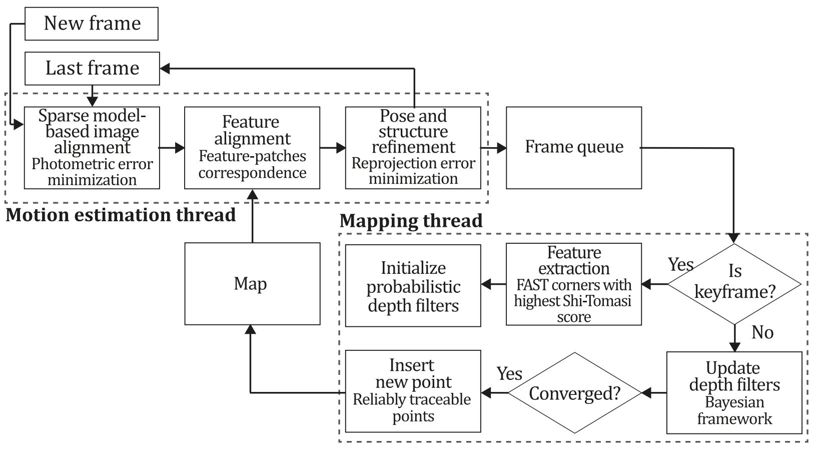

New frame
Last frame
Sparse model-based image alignment
Photometric error minimization
Feature alignment
Feature-patches correspondence
Pose and structure refinement
Reprojection error minimization
Frame queue
Motion estimation thread
Mapping thread
Map
Initialize probabilistic depth filters
Feature extraction
FAST corners with highest Shi-Tomasi score
Yes
Is keyframe?
No
Insert new point
Reliably traceable points
Yes
Converged?
Update depth filters
Bayesian framework

Figure 25. Diagram of SVO algorithm. Adapted from [13].

6.4. General comments for classic approaches

The challenge of monocular 3D reconstruction, a problem inherently fraught with complexity, can be approached, resolved, and refined from many angles. This discourse has examined a selection of the most pivotal monocular SLAM, VO, and SFM methodologies considered traditional due to their non-reliance on machine learning innovations. These methodologies have been discussed in alignment with an established classification scheme, revealing discernible commonalities within the systems grouped under each category. In the realm of direct versus indirect categorization, it is evident that various preprocessing and direct methods have been explored. Sparse indirect methods often incorporate feature extraction techniques, while dense indirect methods may utilize optical flow, and direct methods typically apply direct motion estimation. Hybrid approaches meld these techniques, offering a nuanced blend of both. This analysis has led to identifying nine distinct criteria that serve as a guide for selecting the most suitable system based on factors such as performance, dimensionality, practical application, and ease of implementation. These criteria are concisely presented in Table 2, which outlines the characteristics of each classic monocular system.

Furthermore, to substantiate the theoretical insights, we have executed available open-source implementations of these systems. Figure 26 showcases the outcomes of deploying classic monocular SLAM, VO, and SFM systems on datasets that are openly accessible to the research community.

Table 2. Input modalities used in 3D reconstruction

Method	SLAM, VO or SFM	Tracking method	Map density	Pixels used	Estimation	Global optimization	Relocalization	Loop closure	Availa-bility
Jin et al. (2000) [14]	SFM	Feature-based	Sparse	Hi.grad.	EKF	-	-	-	-
MonoSLAM (2007) [31]	SLAM	Feature-based	Sparse	Shi Tomasi	EKF	-	-	-	[89]
PTAM (2007) [58]	SLAM	Feature-based	Sparse	Hi.grad.	BA	✓	✓	-	[90]
OpenMVG (2013) [61]	SFM	Feature-based	Sparse	Hi.grad.	BA	✓	-	✓	[91]
ORB-SLAM (2015) [33]	SLAM	Feature-based	Sparse	Hi.grad.	Local BA	✓	✓	✓	[92]
COLMAP (2016) [93]	SFM	Feature-based	Sparse	PnP matches	Local and Global BA	✓	-	-	[93]
ORB-SLAM2 [7]	SLAM	Feature-based	Sparse	Hi.grad.	Local BA	✓	✓	✓	[94]
ORB-SLAM3 (2021) [71]	SLAM	Feature-based	Sparse	Hi.grad.	Local BA	✓	✓	✓	[95]

Valgaerts et al. (2011) [74]	SFM	Optical flow	Dense	8-point matches	Robust 8-point algorithm	-	-	-	-
Ranftl et al. (2016) [77]	SFM	Optical flow	Dense	FlowFields	Superpixel graph minimization	-	-	-	-
Stühmer et al. (2010) [80]	SLAM	Direct	Dense	Hi.grad.	Cost volume refinement	✓	✓	-	-
DTAM (2011) [17]	SLAM	Direct	Dense	Hi.grad.	Cost volume refinement	✓	✓	-	[96]
REMODE (2014) [5]	SLAM	Direct	Dense	Hi.grad.	Bayesian estimation	-	-	-	[97]
LSD-SLAM (2014) [4]	SLAM	Direct	Semi-Dense	Edgelets	Pose graph optimization	✓	-	✓	[98]
DSO (2017) [2]	VO	Direct	Sparse	Hi.grad.	Local BA	-	-	-	[99]
LDSO (2018) [23]	VO	Direct	Sparse	Hi.grad.	Local BA GPGO	✓	-	✓	[100]
DSM (2020) [84]	SLAM	Direct	Sparse	Hi. grad.	Photometric BA	✓	-	-	[101]
SVO (2014) [13]	VO	Hybrid	Sparse	FAST + Hi. grad.	Local BA	-	-	-	[102]

Hi.grad. is used to abbreviate a set of pixels with high-intensity gradient.
EKF is used to abbreviate the Extended Kalman Filter technique.
BA is used to abbreviate the Bundle Adjustment technique.
GPGO is used to abbreviate the Global Pose Graph Optimization technique.

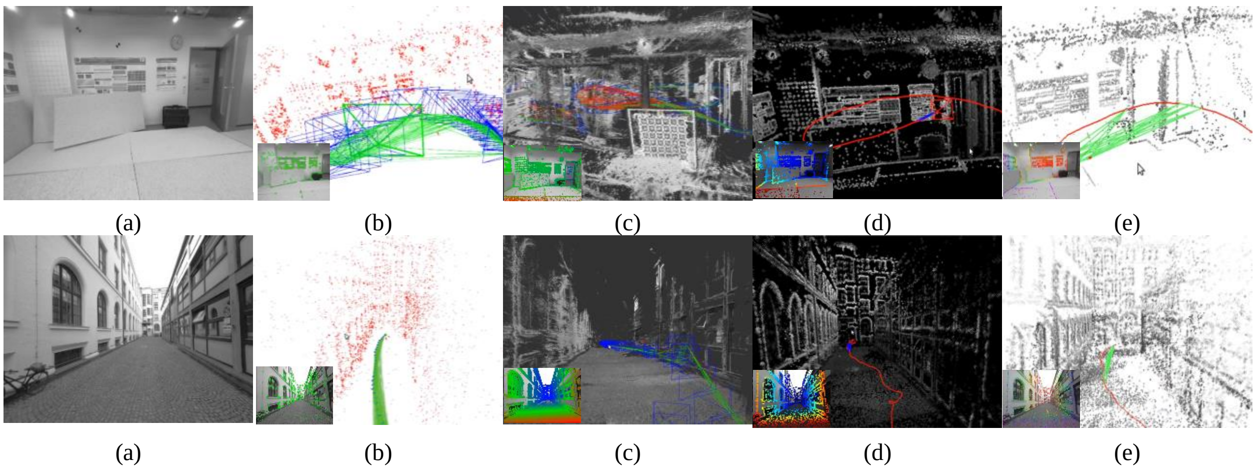


Figure 26. Examples of the results obtained by classic approach implementations. (a) represents the input image, (b) presents results obtained using the ORB-SLAM2 algorithm [7], (c) presents results obtained using the LSD-SLAM algorithm [4], (d) presents results obtained using the DSO algorithm [2], and (e) presents results obtained using DSM algorithm [84]. Top row results correspond to the indoor example sequence seq_01, and bottom row results correspond to the outdoor sequence seq_29 of the TUM-MONO dataset [103].

7. Machine Learning methods

In the wake of remarkable advancements in artificial intelligence, particularly within the domain of deep learning, the research community has ventured into enhancing traditional SLAM, SFM, and VO methodologies by incorporating deep neural networks. These networks are tasked with various functions aimed at diminishing errors in depth estimation, refining the quality of 3D reconstructions, and enhancing camera pose accuracy, among other improvements. A prominent challenge in monocular techniques is the issue of scale ambiguity. Without initial depth measurements, monocular systems may misinterpret the size of objects or surfaces due to variations in environmental contexts and camera calibrations. To address this, deep learning approaches in monocular methods have been instrumental in augmenting VO, SLAM, and SFM systems. They achieve this by leveraging learning-based depth predictions, integrating depth information into system components such as feature points and depth maps, and executing ancillary tasks like semantic segmentation and optical flow estimation.

Recent scholarly efforts have focused on supplanting manually engineered features with those derived through learning processes [104]–[107], crafting neural 3D representations [6], [8], [49], [108]–[110], and integrating learned depth predictions with conventional SLAM backends [3], [54], [111], [112]. Additionally, there has been a push towards developing SLAM or VO systems that are trained in an end-to-end fashion [8], [49], [113]–[115]. While learning-based methods may lack in explainability and often encounter difficulties when applied to novel environments or with disparate camera calibrations, they offer a compelling avenue for advancing monocular visual SLAM. This is particularly true in scenarios characterized by pure rotation, slow motion, or movements devoid of roll and pitch rotations, where traditional SLAM or VO systems might struggle with initialization or generalization.

7.1. ML + Indirect methods

Deep learning has markedly enhanced the capabilities of 3D reconstruction systems by undertaking various tasks, thereby substantially elevating the overall efficacy of these systems. These tasks span from fundamental parameter estimations crucial for the initialization of SLAM, VO, or SFM systems [28], to the substitution of entire components within a SLAM framework, such as depth or pose estimation modules [112], [113], and even endowing the system with novel functionalities like semantic segmentation [3], [116]. In the realm of indirect methods, the research encapsulated in this category integrates preprocessing stages within their algorithms, which encompass feature extraction and optical flow determination prior to pose estimation or depth prediction tasks. Consequently, the integrity of the final 3D representation is intrinsically linked to the volume and fidelity of data procured during these initial stages.

Similarly, to traditional approaches, machine learning-enhanced indirect methods can reconstruct dense and sparse environmental 3D maps. Therefore, discussions of both these categories are presented in sections 7.1.1 and 7.1.2. Figure 27 delineates a chronological overview of the most seminal SFM, VO, and SFM methodologies that indirectly employ machine learning processing image data. One of the most distinguished machine learning methods within this category is DeMoN [114], which has garnered significant acclaim within the academic community, as evidenced by its exceptional citation score, a testament to its remarkable performance.

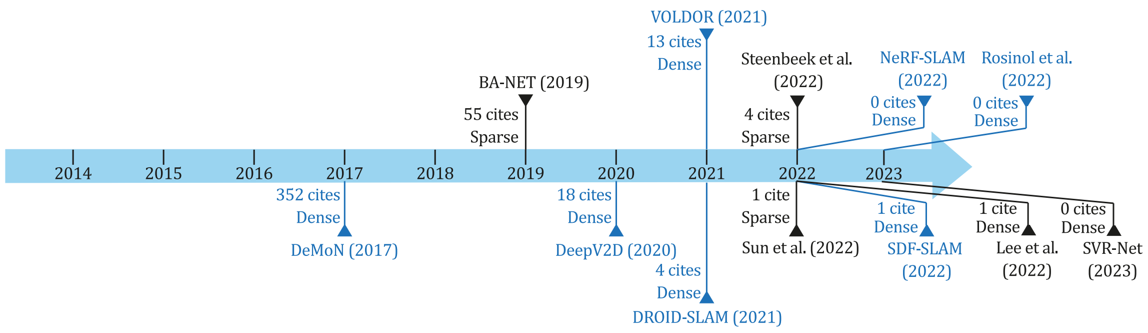


Figure 27. Most representative monocular SLAM, VO or SFM, ML + indirect systems--a Timeline.

7.1.1. ML + Sparse + Indirect methods

Analogous to traditional techniques, machine learning-based indirect methods can be subcategorized into dense and sparse types, contingent upon the density of the resultant 3D reconstructions. Sparse methods are advantageous because they necessitate processing a limited amount of data. However, this also limits the ultimate reconstruction quality, as potentially significant information might be disregarded during pixel selection or feature extraction phases. This limitation has motivated the creation of Convolutional Neural Network (CNN) architectures specifically designed to enrich the sparsity of the 3D maps [50], [117]. These architectures have shown considerable promise, enhancing the detail of the reconstructions while capitalizing on the performance merits inherent in the sparse computational framework.

DynaSLAM (2018). In the study by Bescos et al. [118], a novel algorithm was introduced to identify, segment, and inpaint dynamic elements within sequences of scene frames. This algorithm is an adaptation of the more streamlined ORB-SLAM2 [7] framework, enhanced with a convolutional neural network (CNN) for segmenting known dynamic objects and a multi-view geometry technique to detect additional dynamic entities not identifiable by the CNN. Dynamic entities such as people and bicycles, typically present in image sequences used for 3D reconstruction, are often only partially filtered out as anomalies by SLAM, SFM, and VO methods due to their foundational static environment assumption. To address this, the authors integrated the Mask R-CNN [119] for its proven efficacy in semantic segmentation tasks. This network can discern and delineate image regions likely to contain movable objects across various classes trained on the MS COCO dataset [120]. The Mask R-CNN is an extension of the ResNet C4 and FPN architectures, supplemented with a mask branch to forecast segmentation masks for each identified instance. It can be further refined with new training data to recognize additional classes. After segmenting potential dynamic objects, the algorithm employs the tracking mechanism of ORB-SLAM2 to project features, search for correspondences, minimize reprojection errors, and optimize camera pose. The authors incorporated a multi-view geometry phase to recognise that certain dynamic objects, such as a book being carried, may elude CNN detection due to their absence in the trained classes. This phase calculates the back-projections of keypoints to ascertain the parallax angle, distinguishing between movable and stationary items.

DynaSLAM leverages both geometric and machine learning techniques, mitigating initialization challenges inherent to geometric methods and the limited recognition scope of CNNs. It combines CNN-based segmentation with multi-view geometry to effectively track and segment moving objects. The integration of these approaches has been empirically shown to surpass the performance of either method used in isolation. DynaSLAM first segments out dynamic objects from scene imagery then applies cost-effective tracking based on ORB-SLAM2. It further refines the removal of dynamic object data by synthesising CNN and multi-view geometry. Subsequently, a background inpainting technique utilizing information from preceding frames fills in the excised image portions. This process enables precise tracking and mapping of the static segments of each image. Figure 28 depicts the DynaSLAM algorithm inspired by the article [118].

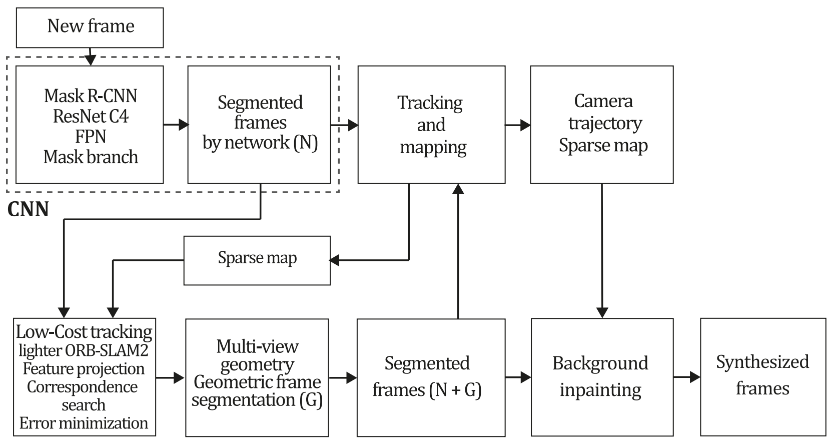


Figure 28. Diagram of DynaSLAM algorithm. Adapted from [118].

BA-NET (2019). The innovative research by Tang & Tan [49], "BA-Net" represents a significant advancement in integrating domain expertise with deep learning techniques. The network enforces multi-view geometric constraints by optimizing scene depth and camera motion through a feature-metric bundle adjustment process. The authors have ingeniously made the Levenberg-Marquardt (LM) optimization algorithm differentiable, which allows the network to learn optimal features. This is achieved by employing a multilayer perceptron (MLP) trained to predict the damping factor λ for the LM algorithm, thus facilitating the learning process. Additionally, Tang & Tan introduced a pioneering method for depth parameterization that enables the recovery of dense per-pixel depth information. Depth maps are generated by a convolutional neural network that creates a series of basis depth maps for each input image. These maps are optimized through a linear combination in a feature-metric bundle adjustment framework.

The BA-Net formulates Bundle Adjustment as a differentiable layer within the network, which minimizes the feature-metric error across aligned feature maps from multiple images. This approach allows for optimising scene geometry and camera motion using the network features as inputs. Traditional bundle adjustment methods, which rely on reprojection error, are limited by their reliance on specific feature types and are prone to outliers, necessitating robust outlier rejection techniques. To address these limitations, BA-Net employs a feature-metric bundle adjustment algorithm that estimates scene depth and camera motion parameters while minimizing the feature-metric difference of aligned pixels, as described by the equation:

$$e^f_{i,j}(\mathcal{X}) = F_i\left(\Pi(\boldsymbol{T}_i, d_j \cdot \boldsymbol{q}_j)\right) - F1(\boldsymbol{q}_j), \tag{21}$$

where $\mathbb{F} = \{F_i | i = 1 \dots N_i\}$ represents the feature pyramids for the images $\mathbb{I} = \{I_i | i = 1 \dots N_i\}$, while $\mathbb{T} = \{\boldsymbol{T}_i | i = 1 \dots N_i\}$ denotes the camera poses, $\mathcal{X} = \left[\boldsymbol{T}_1, \boldsymbol{T}_2 \dots \boldsymbol{T}_{N_i}, d_1, d_2 \dots d_{N_j}\right]^T$ is the set of optimization parameters, Π is the projection function, $d_j \in \mathbb{D} = \{d_j | j = 1 \dots N_j\}$ is the depth at pixel $\boldsymbol{q}_j$, and $d_j \cdot \boldsymbol{q}_j$ upgrades the pixel to its 3D coordinate. The BA-Layer predicts camera poses and dense depth maps during the forward pass and back-propagates the loss to the feature pyramids during training.

The architecture of BA-Net is based on the DRN-54 backbone and includes a depth map generator, a feature pyramid constructor for building multiscale feature maps, and a BA-Layer that optimizes the depth map and camera poses using a novel differentiable LM algorithm. The differentiability of the LM algorithm is crucial for solving the optimization problem, and the authors' strategy to predict the damping factor λ using an MLP network is a key innovation. In summary, BA-Net processes each new frame through the DRN-54 network, extracting feature maps to construct feature pyramids. Depth estimation is performed using a modified decoder in the DRN-54 encoder, generating multiple basis depth maps. These maps are then linearly combined to produce a final depth map, optimized alongside the camera poses by minimizing feature-metric error using a differentiable LM algorithm. Figure 29 describes the BA-Net algorithm inspired by the article [49].

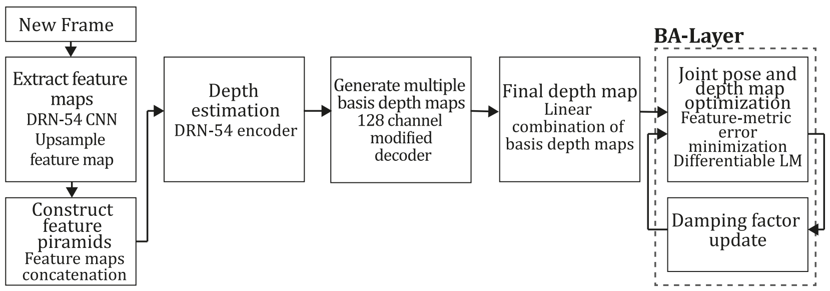

New Frame
Extract feature maps DRN-54 CNN Upsample feature map
Construct feature piramids Feature maps concatenation
Depth estimation DRN-54 encoder
Generate multiple basis depth maps 128 channel modified decoder
Final depth map Linear combination of basis depth maps
BA-Layer
Joint pose and depth map optimization Feature-metric error minimization Differentiable LM
Damping factor update

Figure 29. Diagram of BA-NET algorithm. Adapted from [49].

Steenbeek et al. (2022). In the study by Steenbeek & Nex [117], a novel implementation of the ORB-SLAM2 framework [7] is presented, which incorporates deep learning to enhance its capabilities, specifically for use in unmanned aerial vehicle (UAV) exploration under emergency scenarios. This adaptation is particularly significant as it addresses the need for lightweight and cost-effective equipment suitable for rapid 3D reconstruction in contexts where additional sensors like inertial units, ultrasound, LIDAR, RGB-D, or stereo cameras are impractical. The original ORB-SLAM2 algorithm is renowned for its geometric sparse indirect approach, delivering precise pose and trajectory estimations. However, its 3D reconstruction output is typically not dense enough for certain applications and is subject to the common scale ambiguity challenge inherent in SLAM systems. To address these limitations, the authors integrated a Convolutional Neural Network (CNN) for Single Image Depth Estimation (SIDE) [121], which has demonstrated efficiency in real-time monocular depth estimation.

The system architecture involves capturing a sequence of RGB images via a commercial drone's camera, which is then processed by the ORB-SLAM2 algorithm on a laptop to estimate the pose and generate a sparse depth map. This data serves as the CNN input, producing a scaled pose and a more detailed depth map. The outputs are subsequently combined using the OctoMap framework (Hornung et al., 2013). Minimal modifications were made to the ORB-SLAM system, with parameter adjustments based on empirical findings, such as setting six pyramid levels for feature detection, connecting frames with 25 shared features, and establishing a minimum keyframe interval of 15 to accommodate the UAV's rapid movements. The CNN SIDE network features an encoder-decoder structure, with the encoder generating a feature map that feeds into a decoder. The decoder is inspired by the design of Laina et al. (2016), employing up-projection blocks in an up-sampling strategy to enhance and refine the depth map progressively. The CNN was initially trained on the NYU Depth v2 dataset [122], which required adjustments to account for the different camera specifications between the Kinect sensor and the drone. This involved image cropping to match the field of view and resampling based on the drone camera's intrinsic parameters. A median filter was also implemented to improve the scale estimation by filtering out outliers.

Despite the advancements, the resulting map was not sufficiently dense to be classified as a dense approach, and the map quality was suboptimal compared to similar systems. The authors suggest that the limitations of current SIDE algorithms in replicating the performance of stereo methods could be a contributing factor, along with the computational constraints of the hardware used in the experiments. Future work is anticipated to involve higher computational resources to overcome these issues. Figure 30 depicts the Steenbeek et al. algorithm inspired by the article [117].

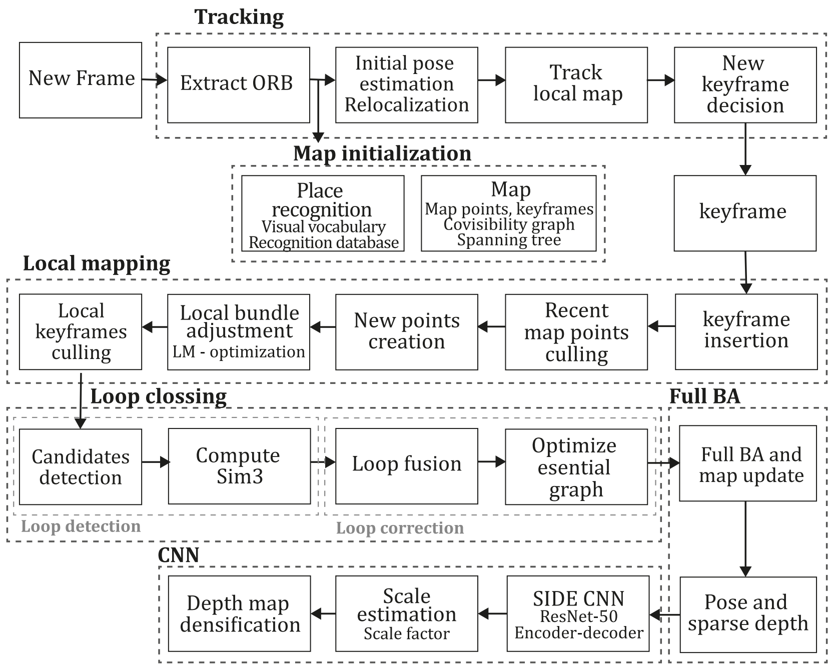


Figure 30. Diagram of Steenbeek et al., algorithm. Adapted from [117].

Sun et al. (2022). In the innovative study by Sun et al. [50], a novel framework is introduced that enhances Visual Odometry (VO) by incorporating a depth estimation module into the established ORB-

SLAM system [33]. This integration aims to expand the system's generalization ability and enrich the density of the resulting 3D reconstructions. The authors successfully integrated a neural network into ORB-SLAM, drawing on the methodology of DiverseDepth [123], and implemented it as a versatile depth module that operates in dual modes to aid both odometry and mapping tasks.

In its initial mode, the network processes a singular monocular image to predict a relative depth map. While this map lacks absolute scale, it is rich in relational depth information, which is instrumental in eliminating outliers and retaining a precise collection of points to calculate accurate camera poses. The second mode involves the network receiving an RGB image and a sparse depth map, which it uses to predict a depth map with consistent scale, subsequently utilized to achieve a more comprehensive mapping. The authors' focus on enhancing generalization led to the selection of ORB-SLAM as the foundational SLAM model, given its robust performance in diverse environments. The deep neural network employed in this research was influenced by the design proposed by Yin et al. (2019), featuring an encoder-decoder architecture adept at operating in various modes and capable of being trained to switch between them randomly. Based on the ResNetXt-50 backbone, the architecture is tailored to predict depth maps from an affine invariant perspective relative to the ground truth when provided with a single image and to forecast an accurate, scale-consistent depth map when supplemented with a sparse depth map from the SLAM system.

To enhance the network's generalization capacity, the authors utilized a heterogeneous mix of training data sourced from five distinct datasets: Taskonomy [124], DIML [125], ApolloScape [126], DiverseDepth [123], and RedWeb [127]. These datasets encompass a range of indoor and outdoor scenes captured with high to medium-precision annotation tools such as LiDAR, lasers, and stereo setups. During training, a FAST corner detector was employed to generate sparse depth points from dense depth datasets, thereby emulating the sparse depth characteristics derived from the SLAM system. The network was trained with two distinct loss functions corresponding to the two input modes. The experimental findings revealed that the modified ORB-SLAM system exhibited significant improvements in generalization, enhancing performance across both indoor and outdoor scenes, advancing depth estimation accuracy, and diminishing the absolute trajectory error as assessed on the KITTI dataset. However, it is noteworthy that while the depth maps generated were denser than those produced by the original ORB-SLAM, they may not be sufficiently dense to qualify as a dense indirect approach compared to other dense reconstruction methodologies. Figure 31 depicts the Sun et al. algorithm inspired by the article [50].

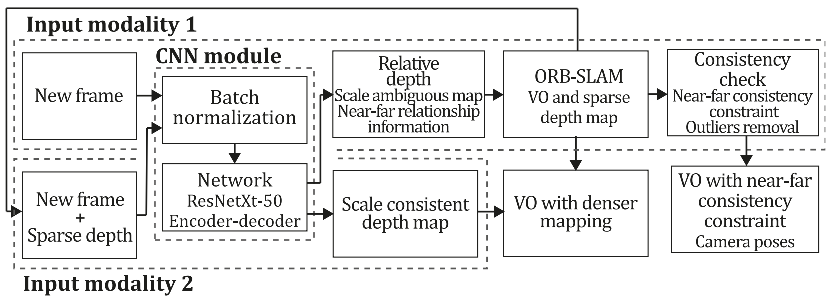


Figure 31. Diagram of Sun et al., algorithm. Adapted from [50].

Lee et al. (2023). In the cutting-edge research by Lee et al. [128], a novel contribution to the sparse indirect category of SLAM systems is presented, which incorporates deep learning to enhance the precision and robustness of SLAM methodologies for autonomous vehicles. SLAM systems are pivotal for navigation self-driving cars, providing critical capabilities for position estimation and environmental mapping through monocular vision. The authors have developed a deep neural network-augmented monocular SLAM system that synergizes semantic segmentation with 3D geometric estimation to refine the system's accuracy. This advanced system was created based on the ORB-SLAM framework, which has been augmented with a deep neural network for 3D geometric estimation and supplemented with a semantic segmentation module to elevate the fidelity of the resultant point cloud. The semantic segmentation module is adept at discerning objects of similar geometry, such as vehicles, pedestrians, and foliage, by employing a labelling strategy that enhances the precision of point cloud construction. The core innovation of the proposed monocular SLAM system lies in its utilization of semantic segmentation to achieve more accurate 3D reconstruction and environmental mapping. The deep neural network-based architecture of the system enables it to assimilate and adjust to diverse environmental and illumination conditions, thereby bolstering its robustness in real-world applications. Additionally, the system employs an innovative loss function that differentially penalizes translational and rotational errors, contributing to stabilising the estimated pose.

The methodology devised by Lee et al. encompasses three integral modules: localization, mapping, and segmentation. The localization module is tasked with keyframe selection, which, upon completion by the mapping and segmentation modules, facilitates the extraction of corner features and the estimation of camera pose for each keyframe based on the points from connected keyframes. The mapping module engages in triangulation between current and connected corner features to generate new 3D points and estimates a ground plane using only the points identified as ground by the CNN, which also aids in recovering the correct scale for camera poses. Consequently, the mapping module delivers scale-adjusted camera poses and 3D points. Concurrently, the segmentation module executes deep-learning-based semantic segmentation on each downsampled keyframe and refines the corner features on the keyframe by excluding moving objects and areas of low parallax, utilizing the ERFNet CNN as proposed by [129]. Notably, the Lee et al. approach introduces scale correction in 3D mapping and an innovative technique to mitigate factors that could lead to inaccurate mapping in each keyframe. The performance of the proposed monocular SLAM system was rigorously assessed using the KITTI benchmark dataset, which enabled the authors to establish that their system surpasses existing state-of-the-art monocular SLAM methods, including ORB-SLAM, ORB-SLAM2, and Mask-SLAM. The system demonstrated an average translational error of merely 0.19%, a significant improvement over the 0.40% error of its nearest rival. Furthermore, the semantic segmentation module's integration enhanced the accuracy of 3D reconstruction and facilitated the creation of a detailed semantic map. Figure 32 depicts the Lee et al. algorithm inspired by the article [128].

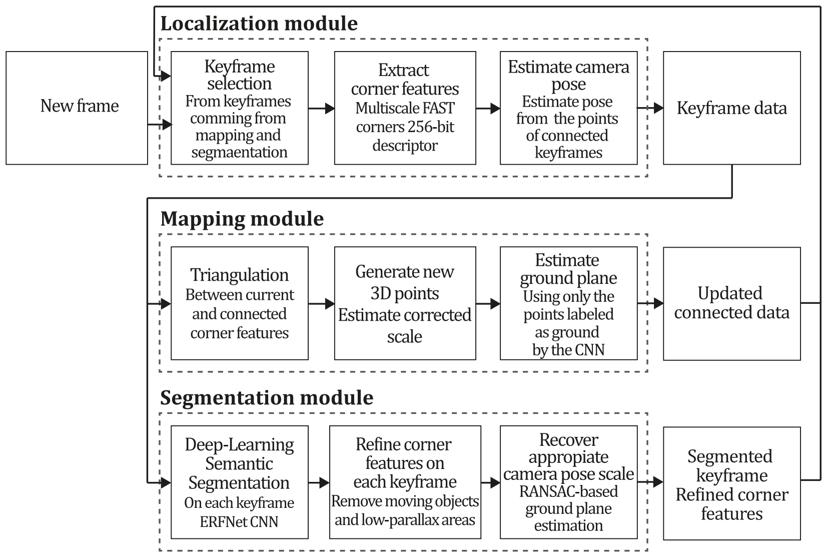

Localization module
New frame
Keyframe selection From keyframes comming from mapping and segmaentation
Extract corner features Multiscale FAST corners 256-bit descriptor
Estimate camera pose Estimate pose from the points of connected keyframes
Keyframe data
Mapping module
Triangulation Between current and connected corner features
Generate new 3D points Estimate corrected scale
Estimate ground plane Using only the points labeled as ground by the CNN
Updated connected data
Segmentation module
Deep-Learning Semantic Segmentation On each keyframe ERFNet CNN
Refine corner features on each keyframe Remove moving objects and low-parallax areas
Recover appropiate camera pose scale RANSAC-based ground plane estimation
Segmented keyframe Refined corner features

Figure 32. Diagram of Lee et al., algorithm. Adapted from [128].

SVR-Net (2023). The SVR-Net SLAM system, developed by Lang et al. [130], represents an innovative approach in the domain of simultaneous localization and mapping, designed to function with both visual and visual-plus-range sensory inputs, facilitating the generation of precise 3D mappings of uncharted environments. A salient feature of this system is the integration of a Support Vector Regression (SVR) network, which is instrumental in estimating the spatial coordinates of key points within the scene. This enhancement is pivotal in ensuring the robustness of feature tracking, even under adverse lighting conditions or when occlusions occur. Further distinguishing the SVR-Net SLAM system are several pioneering attributes. It incorporates an online learning algorithm that dynamically adjusts to evolving environmental conditions and utilizes graph optimization to refine the estimations of camera poses and the map's structure. Including loop-closure detection contributes to the map's enhanced accuracy and coherence. When juxtaposed with other SLAM systems that depend exclusively on visual or range sensors, the SVR-Net SLAM system stands out for its capacity to construct highly detailed and precise maps of intricate environments, including multi-level structures and non-planar surfaces. Moreover, the system's computational efficiency renders it suitable for real-time applications.

The SVR-Net SLAM system employs a coarse-to-fine methodology to facilitate efficient tracking and comprehensive global mapping. It encompasses two modules: one for initial data processing and another for refining the outcomes. Initially, the system processes a pair of frames to estimate the raw pose and a local map through the SVR network, representing the map as sparse voxels with TSDF values. Subsequently, this map is expanded into a global map using the data from the initial stage. In the second stage, the system undertakes voxel up-sampling and proceeds with the refinement of pose and map; after that, it merges the refined local map with the global map to enhance it. Specifically, the SVR-Net module, trained on the ScanNet(V2) dataset [131], operates as an end-to-end tracking and mapping network. It processes a pair of RGB frames alongside voxel coordinates to output a local map, relative pose, and TSDF values for the voxels. The process begins with extracting feature maps from the images, which are then converted into feature voxels for each keyframe. These feature voxels are correlated with the second frame's features to provide matching data. Following this, the system matches features based on the current pose estimation, iteratively updating the pose and map. The feature-matching outputs are then employed to refine the pose and map estimations of SVR-Net. The SLAM pipeline, underpinned by the Kinect-Fusion approach [34], leverages the local map data to augment the global map and bolster its overall consistency. Figure 33 illustrates the SVR-Net algorithm inspired by the article [130].

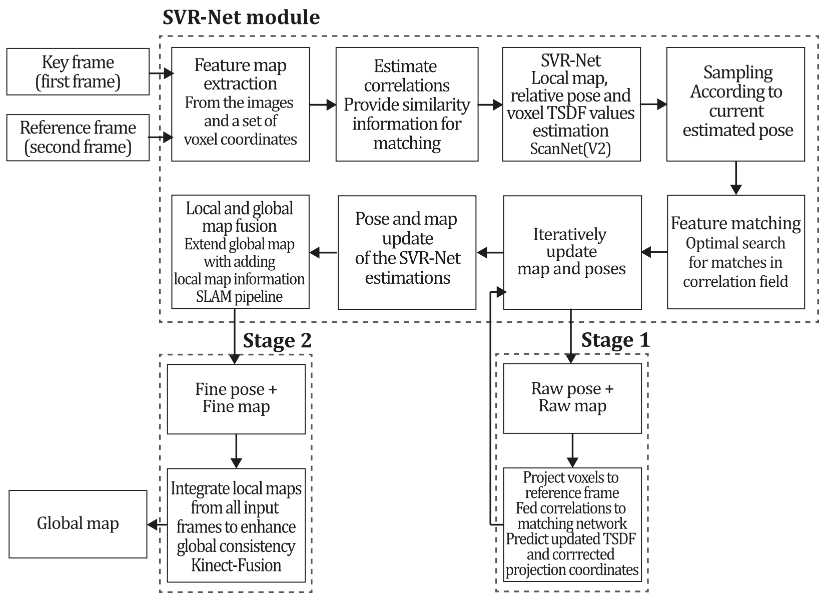

SVR-Net module
Key frame (first frame)
Reference frame (second frame)
Feature map extraction From the images and a set of voxel coordinates
Estimate correlations Provide similarity information for matching
SVR-Net Local map, relative pose and voxel TSDF values estimation ScanNet(V2)
Sampling According to current estimated pose
Feature matching Optimal search for matches in correlation field
Iteratively update map and poses
Pose and map update of the SVR-Net estimations
Local and global map fusion Extend global map with adding local map information SLAM pipeline
Stage 2
Stage 1
Fine pose + Fine map
Integrate local maps from all input frames to enhance global consistency Kinect-Fusion
Global map
Raw pose + Raw map
Project voxels to reference frame Fed correlations to matching network Predict updated TSDF and corrrected projection coordinates

Figure 33. Diagram of SVR-Net SLAM algorithm. Adapted from [130].

7.1.2. ML + Dense + Indirect methods

Indirect approaches in computer vision often employ optical flow for depth estimation tasks. A notable difficulty within the Visual Odometry (VO) paradigm is the precise quantification of errors in feature point locations, which may be adversely affected by motion blur, blockages, and changes in the camera's perspective. Direct methods, which are generally dependent on the premises of minimal movement and consistent appearance, encounter limitations in their robustness against scene variations, thus constraining their broader applicability (Min & Dunn, 2021). In recent developments, the application of machine learning to optical flow estimation has yielded impressive results. This technique, which synthesizes rigid flow associated with camera movement and a free-flowing component that accounts for the motion of objects within the scene, has achieved leading-edge performance [132], [133]. This method's accuracy, robustness, and adaptability have been established, particularly in demanding scenarios characterized by a lack of texture, motion blur, or substantial obstructions, positioning it as an exemplary solution in the field.

DeMoN (2017). In the Structure from Motion (SfM) research field, a noteworthy contribution is the DeMoN system, as delineated by Ummenhofer et al. [114]. This system pioneered integrating a Convolutional Neural Network (CNN) designed to recover depth and camera motion from sequential, unrestricted image pairs concurrently, thereby enhancing its predictive capabilities. The network extends its functionality by estimating additional parameters such as surface normals, optical flow between images, and confidence in these matches. DeMoN leverages motion parallax—a potent indicator for generalizing across novel environments—to facilitate egomotion estimation. It achieves this by alternating between optical flow prediction and estimating depth and camera motion. The system incorporates a modified version of FlowNet [134] to compute optical flow from image pairs. Specific loss functions are employed to balance the diverse outputs of the network, such as depth maps and motion vectors. The L1 loss $\mathcal{L}$ function for inverse depth values is defined as:

$$\mathcal{L}_{depth} = \sum_{i,j} \left| s d^*(i,j) - \widehat{d^*}(i,j) \right|, \tag{22}$$

where $d^* = \frac{1}{z}$ represents the inverse depth, $\widehat{d^*}$ is the ground truth inverse depth, and s is the scale factor predicted by the network. For surface normals and optical flow, the L2 norm is utilized to penalize deviations from the ground truth values $\hat{n}$ and $\widehat{w}$:

$$\mathcal{L}_{normal} = \sum_{i,j} \|\boldsymbol{n}(i,j) - \hat{\boldsymbol{n}}(i,j)\|_2, \tag{23}$$

$$\mathcal{L}_{flow} = \sum_{i,j} \|\boldsymbol{w}(i,j) - \widehat{\boldsymbol{w}}(i,j)\|_2, \tag{24}$$

then, the loss functions for motion vectors are given by:

$$\mathcal{L}_{rotation} = \|\boldsymbol{r} - \hat{\boldsymbol{r}}\|_2, \tag{25}$$

$$\mathcal{L}_{traslation} = \|\boldsymbol{t} - \hat{\boldsymbol{t}}\|_2, \tag{26}$$

where $\boldsymbol{r} = \theta \boldsymbol{v}$ denotes the minimal parameterization of rotation with angle θ and axis $\boldsymbol{v}$, and $\boldsymbol{t}$ is the translation vector. The scale-invariant loss for discrete scale-invariant gradients g is:

$$g_h[f](i,j) = \left(\frac{f(i+h,j) - f(i,j)}{|f(i+h,j)| + |f(i,j)|}, \frac{f(i,j+h) - f(i,j)}{|f(i,j+h)| + |f(i,j)|} \right)^T, \tag{27}$$

$$\mathcal{L}_{grad\ d^*} = \sum_{h \in \{1,2,4,8,16\}} \sum_{i,j} \left\| g_h[d^*](i,j) - g_h\left[\widehat{d^*}\right](i,j) \right\|_2, \tag{28}$$

this approach uses various spacings h to accommodate gradients of different scales, allowing the network to compare within a pixel's neighbourhood. The system also applies scale-invariant gradient loss to each component of optical flow, which enhances the smoothness of the estimated flow fields and sharpens the delineation of motion discontinuities. In brief, DeMoN processes a pair of images to predict a depth map from the first image and the relative pose from the second, utilizing a sequence of encoder-decoder networks that iterate over optical flow, depth map, and egomotion estimation. The system comprises three principal components: the bootstrap net, which provides initial depth and motion estimates; the iterative net, which refines these estimates, particularly focusing on discontinuities and scale; and the refinement net, which enhances the resolution of the final depth map. Figure 34 depicts the DeMoN algorithm inspired by the article [114].

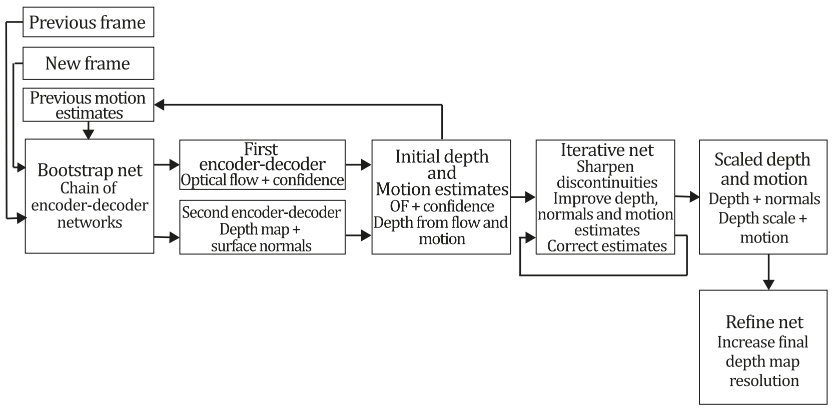

Previous frame
New frame
Previous motion estimates
Bootstrap net
Chain of encoder-decoder networks
First encoder-decoder
Optical flow + confidence
Second encoder-decoder
Depth map + surface normals
Initial depth and Motion estimates
OF + confidence
Depth from flow and motion
Iterative net
Sharpen discontinuities
Improve depth, normals and motion estimates
Correct estimates
Scaled depth and motion
Depth + normals
Depth scale + motion
Refine net
Increase final depth map resolution

Figure 34. Diagram of DeMoN algorithm. Adapted from [114].

DeepV2D (2020). The DeepV2D system, as introduced by Teed & Deng [113], represents an integrated, end-to-end trainable framework that alternates between depth and motion modules to estimate depth and camera pose. The motion module utilizes depth predictions to refine camera pose estimations, enhancing the precision of the pose determination process. Both modules are constructed upon neural networks that adhere to geometric constraints, yet they are amalgamated within a fully differentiable architecture to facilitate Structure from Motion (SfM) tasks.

The depth module of DeepV2D accepts camera motion as an input to generate depth predictions. In contrast, the motion module ingests depth information to output a motion correction factor. The depth module constructs a cost volume from learned features, enriched with data from multiple viewpoints via a pooling layer. This module encompasses a 2D feature extractor, which employs dual stacked hourglass networks to transform each frame into a dense feature representation; a cost volume backprojection module that reprojects coordinates across frames for each potential depth; and a 3D stereo matching network that conducts stereo matching across a specified array of cost volumes. The motion module outputs perturbations in error terms, which are utilized to refine the camera pose. This module's workflow includes initialization, selecting a keyframe and predicting relative motion; feature extraction, where learned features convert every frame into a feature map; an error term computation, using two frames and an hourglass network to predict residual flow between their feature maps; and an optimization layer, which employs a Gauss-Newton method to compute pose increments.

While the primary focus of DeepV2D is on depth estimation, it can evolve into a SLAM system by training the neural network to map optical flow to camera motion directly, avoiding the need for explicit optical flow supervision. This work distinguishes itself from DeMoN (Ummenhofer et al., 2017) by its motion module's capability to operate with a variable number of frames. Moreover, DeepV2D introduces a novel motion estimation architecture, termed Flow-SE3, which differentiates it from other works, such as DeMoN and DeepTAM. This architecture allows the system to apply geometric constraints on camera motion, thereby minimizing reprojection error and leveraging the benefits of end-to-end training. Figure 35 depicts the DeepV2D algorithm inspired by the article [113].

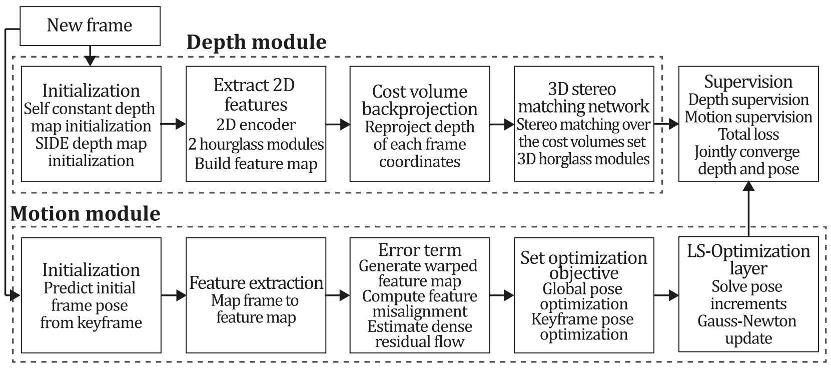

New frame
Depth module
Initialization
Self constant depth map initialization
SIDE depth map initialization
Extract 2D features
2D encoder
2 hourglass modules
Build feature map
Cost volume backprojection
Reproject depth of each frame coordinates
3D stereo matching network
Stereo matching over the cost volumes set
3D horglass modules
Supervision
Depth supervision
Motion supervision
Total loss
Jointly converge depth and pose
Motion module
Initialization
Predict initial frame pose from keyframe
Feature extraction
Map frame to feature map
Error term
Generate warped feature map
Compute feature misalignment
Estimate dense residual flow
Set optimization objective
Global pose optimization
Keyframe pose optimization
LS-Optimization layer
Solve pose increments
Gauss-Newton update

Figure 35. Diagram of DeepV2D algorithm. Adapted from [113].

VOLDOR (2021). The VO system VOLDOR, developed by Min & Dunn [135], represents a significant advancement in Visual Odometry (VO), utilizing log-logistic depth residuals for its operation. This system integrates an externally computed optical flow derived from machine learning estimators and a probabilistic model to construct a VO pipeline. This pipeline is distinctive in bypassing the need for feature extraction, RANSAC estimation, or local bundle adjustment to generate camera pose and depth maps. The observation that optical flow residuals typically adhere to a log-logistic distribution forms the cornerstone of VOLDOR's probabilistic framework, which employs a Fisk-distributed residual model. This model concurrently estimates camera motion, pixel depth, and motion-track confidence through a generalized Expectation Maximization (EM) algorithm.

VOLDOR's inference mechanism also relies on a generalized EM approach to deduce depth and rigidity, employs a maximum likelihood estimator (MLE) for initiating subsequent camera poses, utilizes a maximum inlier estimation to refine the MLE criteria, and applies a forward-backwards algorithm to deduce rigidity, reducing it to hidden Markov chains. Notably, VOLDOR is designed to be indifferent to the choice of optical flow estimator, and in this instance, PWC-net [133] was selected to determine the external dense optical flow input. PWC-net is a compact and trainable CNN that incorporates pyramidal processing, warped features, and cost volume for warping the CNN features of a subsequent image using the current image's optical flow estimate. This network is significantly more compact and user-friendly for training than FlowNet2 [132], consisting of a feature extractor, an optical flow estimator, and context networks. In this way, VOLDOR employs a Fisk residual model to infer from an external estimator's sequence of optical flows, with PWC-net being the system of choice in this context. The initial camera pose is established using epipolar geometry from the first optical flow, applying the least median-square estimator. Subsequently, depths are triangulated through two-view triangulation, utilizing optical flow and the initial camera pose. The system then updates by inferring over the Fisk residual model, where a generalized EM algorithm refines depth and rigidity. Sequential camera poses are bootstrapped using an MLE, and a maximum inlier estimation criterion is applied to mitigate biases introduced during the initialization and update phases. Coupled with the Fisk residual model and maximum inlier estimation, this CNN-based system demonstrated superior performance in an ablation study and surpassed comparable optical flow systems on the KITTY and TUM RGB-D datasets. Figure 36 depicts the VOLDOR algorithm inspired by the article [136].

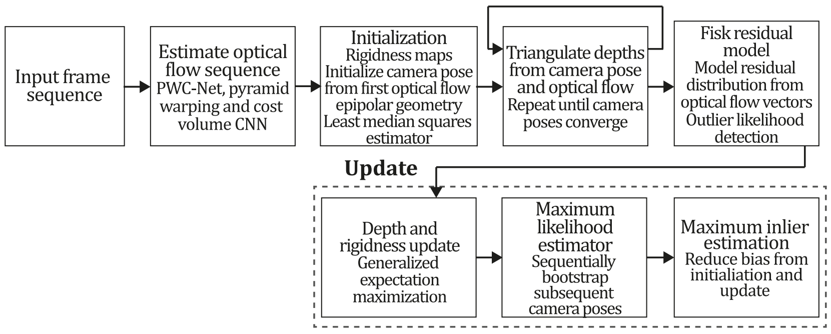

Input frame sequence
Estimate optical flow sequence
PWC-Net, pyramid warping and cost volume CNN
Initialization
Rigidness maps
Initialize camera pose from first optical flow epipolar geometry
Least median squares estimator
Triangulate depths from camera pose and optical flow
Repeat until camera poses converge
Fisk residual model
Model residual distribution from optical flow vectors
Outlier likelihood detection
Update
Depth and rigidness update
Generalized expectation maximization
Maximum likelihood estimator
Sequentially bootstrap subsequent camera poses
Maximum inlier estimation
Reduce bias from initialiation and update

Figure 36. Diagram of VOLDOR algorithm. Adapted from [136].

DROID-SLAM (2021). The DROID-SLAM system, as introduced by Teed & Deng [136], stands out as a significant advancement in the field of deep learning-based Simultaneous Localization and Mapping (SLAM). This system utilizes a Differentiable Recurrent Optimization-Inspired Design to perform iterative camera pose updates and estimate depth maps through a dense bundle adjustment layer. DROID-SLAM employs a Gated Recurrent Unit (GRU) to predict updates within the domain of dense flow fields, generating an error correction term for the dense correspondence field alongside a depth map.

DROID-SLAM can execute real-time localization and mapping by dividing its operations between frontend and backend processes. The frontend processes incoming frames, feature extraction, keyframe selection, and local bundle adjustment. Concurrently, the backend thread engages in global bundle adjustment across all historical keyframes. The system's architecture is differentiable and is built upon the RAFT [137] framework for optical flow, which stands for Recurrent All-Pairs Field Transforms. This foundation allows DROID-SLAM to handle optical flow by performing recurrent iterative updates. Unlike traditional systems that update the optical flow, DROID-SLAM updates depth maps, and camera poses through a differentiable bundle adjustment layer that computes the Gauss-Newton update, ensuring maximum consistency with the current optical flow estimate.

DROID-SLAM is versatile, supporting monocular, RGB-D, and stereo inputs, and has exhibited remarkable performance across various benchmarks, including TartanAir, EuRoC, TUM-RGB-D, and ETH3D-SLAM. It has been shown to surpass both classic and learned-based monocular systems in most tested sequences. However, the authors acknowledge that a significant limitation of this monocular system is its high computational demand, which, during experimental evaluations, necessitated up to 24GB of GPU memory to process the EuRoC, TartanAir, and ETH3D sequences. Figure 37 depicts the DROID-SLAM algorithm inspired by the article [136].

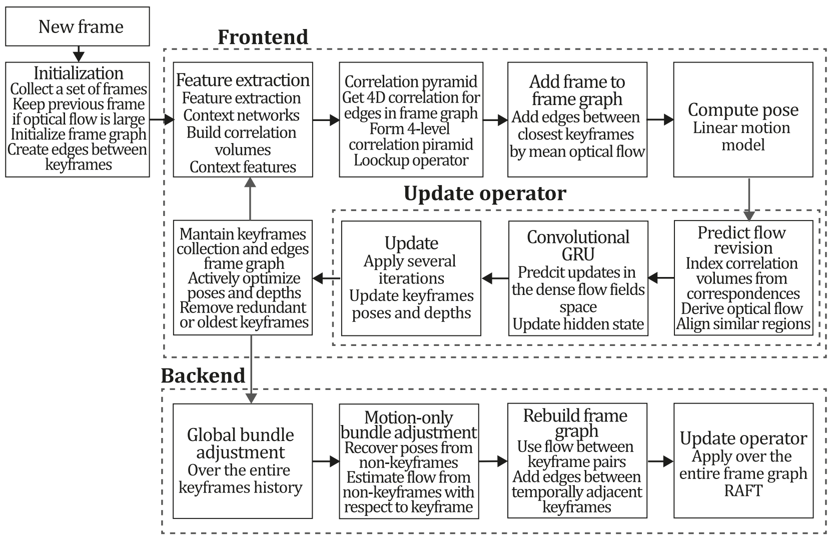

New frame
Frontend
Initialization
Collect a set of frames
Keep previous frame if optical flow is large
Initialize frame graph
Create edges between keyframes
Feature extraction
Feature extraction
Context networks
Build correlation volumes
Context features
Correlation pyramid
Get 4D correlation for edges in frame graph
Form 4-level correlation piramid
Loockup operator
Add frame to frame graph
Add edges between closest keyframes by mean optical flow
Compute pose
Linear motion model
Update operator
Mantain keyframes collection and edges frame graph
Actively optimize poses and depths
Remove redundant or oldest keyframes
Update
Apply several iterations
Update keyframes poses and depths
Convolutional GRU
Predcit updates in the dense flow fields space
Update hidden state
Predict flow revision
Index correlation volumes from correspondences
Derive optical flow
Align similar regions
Backend
Global bundle adjustment
Over the entire keyframes history
Motion-only bundle adjustment
Recover poses from non-keyframes
Estimate flow from non-keyframes with respect to keyframe
Rebuild frame graph
Use flow between keyframe pairs
Add edges between temporally adjacent keyframes
Update operator
Apply over the entire frame graph
RAFT

Figure 37. Diagram of DROID-SLAM algorithm. Adapted from [136].

SDF-SLAM (2022). The SDF-SLAM system, as presented by C. Yang et al. [116], is a monocular SLAM approach that builds upon the foundational concepts of ORB-SLAM. It introduces a novel feature extraction method and integrates a dense semantic network to estimate densely labelled depth maps, situating it within the dense category of SLAM taxonomy. The original ORB-SLAM was adept at extracting textural features such as edges and corners using a combination of SIFT, SURF, and the ORB algorithms, among others, to identify feature points in adjacent frames based on similarity. These feature points are then projected to subsequent frames using camera pose changes, with Perspective-n-

Point (PnP) algorithms minimizing the discrepancy between projected points and their actual matches. PnP further translates these feature points into 3D coordinates by utilizing the camera pose, amalgamating all features to create a depth map. However, the depth map generated by ORB-SLAM is not sufficiently dense for certain applications and lacks semantic information recognizable by machines. Aiming to enhance the accuracy of camera trajectory estimation and to recover a semantically rich three-dimensional scene map, SDF-SLAM integrates camera poses with depth and semantic data at the frame level. This integration is achieved through three primary components: a Feature Point and CNN Feature Description (FPFDCNN), which is trained to extract features from image pairs and compute vector descriptors for each feature point; a Deep Semantic Fusion CNN (SDFCNN), which is trained for concurrent semantic segmentation and depth prediction from RGB images, significantly reducing the estimated parameter count; and a monocular visual SLAM system that incorporates a deep learning framework with a data correction module to globally optimize the point cloud for consistent outputs.

SDF-SLAM employs two neural networks: the FPFDCNN extracts feature points from adjacent frames and matches them to form feature-matching pairs, followed by a minimization process to derive camera rotation and displacement matrices. These matrices and the image are inputted into the SDFCNN to recover a dense depth map and semantic segmentation. Subsequently, the data calibration module utilizes the dense map, pose, and semantic segmentation data to optimise global and local, resulting in a three-dimensional semantic map. The FPFDCNN is based on an encoder-decoder architecture comprising an input layer, encoder layers for feature map extraction, decoder layers for restoring the original image size and feature descriptors, output layers, and concatenation layers. The SDFCNN employs a unified semantic down-sampling layer to process feature map information and an up-sampling layer for feature extraction and restoration, alongside discriminative layers for feature classification, probability estimation, and depth regression estimation.

Experimental results using the TUM dataset have shown that SDF-SLAM offers marked improvements over traditional methods such as ORB-SLAM [33] and LSD-SLAM [4], as well as enhanced semantic segmentation quality compared to CNN-SLAM (Tateno et al., 2017), achieving 90% accuracy in point cloud prediction and 67% in semantic labelling. An additional benefit of SDF-SLAM is its compatibility with various classic SLAM methodologies, as demonstrated by the authors who successfully integrated the same network architectures within the DSO framework. Figure 38 illustrates the SDF-SLAM algorithm inspired by the article [116].

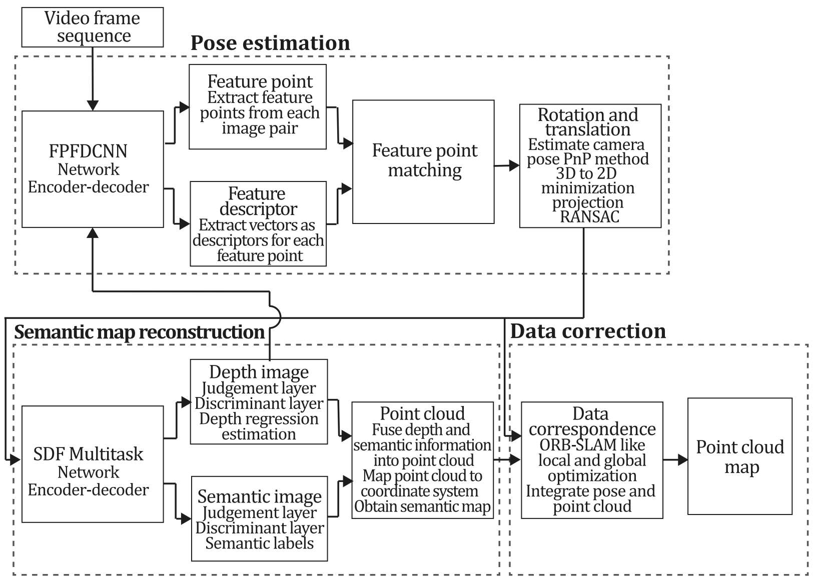

Video frame sequence
Pose estimation
FPFDCNN Network Encoder-decoder
Feature point Extract feature points from each image pair
Feature descriptor Extract vectors as descriptors for each feature point
Feature point matching
Rotation and translation Estimate camera pose PnP method 3D to 2D minimization projection RANSAC
Semantic map reconstruction
SDF Multitask Network Encoder-decoder
Depth image Judgement layer Discriminant layer Depth regression estimation
Semantic image Judgement layer Discriminant layer Semantic labels
Point cloud Fuse depth and semantic information into point cloud Map point cloud to coordinate system Obtain semantic map
Data correction
Data correspondence ORB-SLAM like local and global optimization Integrate pose and point cloud
Point cloud map

Figure 38. Diagram of SDF-SLAM algorithm. Adapted from [116].

NERF-SLAM (2022). The study by Rosinol et al. introduces NeRF-SLAM [138], a monocular technique for reconstructing indoor scenes that leverages normal priors to elevate the precision and intricacy of the reconstructions. This method synergizes the benefits of Neural Radiance Fields (NeRF) with the tracking capabilities of the DROID-SLAM system (Teed & Deng, 2021) to achieve real-time, accurate, and detailed reconstructions of indoor environments. NeRF-SLAM is engineered to surmount common challenges many monocular systems face, such as difficulties with dynamic settings, occlusions, and scalability issues. The NeRF-SLAM framework employs an implicit neural representation to model the scene's geometry and appearance, with the integration of normal priors to refine the detail and accuracy of the reconstruction. It is composed of three main components: a frontend that deduces camera poses and produces sparse 3D point clouds; a backend that amalgamates these sparse point clouds to form the final implicit scene representation; and a normal prediction module that is trained to infer surface normals from the implicit representation.

The tracking component is based on DROID-SLAM, which procures dense depth maps and poses from a sliding window of eight keyframes. DROID-SLAM computes the optical flow for each frame pair, drawing inspiration from the RAFT methodology, which utilizes a Convolutional GRU to calculate the flow and weight by correlating two frames and an estimation of the current optical flow. Subsequently, DROID-SLAM addresses the dense bundle adjustment (BA) challenge by representing the 3D geometry as a parameterized collection of inverse depth maps, facilitating an efficient resolution of the BA problem through a linear least squares method. Marginal covariances for the depth maps and poses are then computed, which, in conjunction with the depth, poses, and input RGB images, are used to optimize the parameters of the radiance field and refine the camera poses. NeRF-SLAM operates with a tracking thread that persistently minimizes the BA reprojection error across an active window of keyframes while the mapping thread optimizes the keyframes sourced from the tracking thread. The tracking thread generates a new keyframe whenever the mean optical flow between the preceding and current frames exceeds a predetermined threshold.

Through comprehensive testing on various datasets and comparison with leading-edge methods, NeRF-SLAM has demonstrated superior performance in reconstruction accuracy, detail, and resilience to dynamic scene elements and occlusions. This positions NeRF-SLAM as a highly promising method for indoor scene reconstruction, merging the strengths of neural networks with the principles of SLAM. Figure 39 illustrates the NeRF-SLAM algorithm inspired by the article [138].

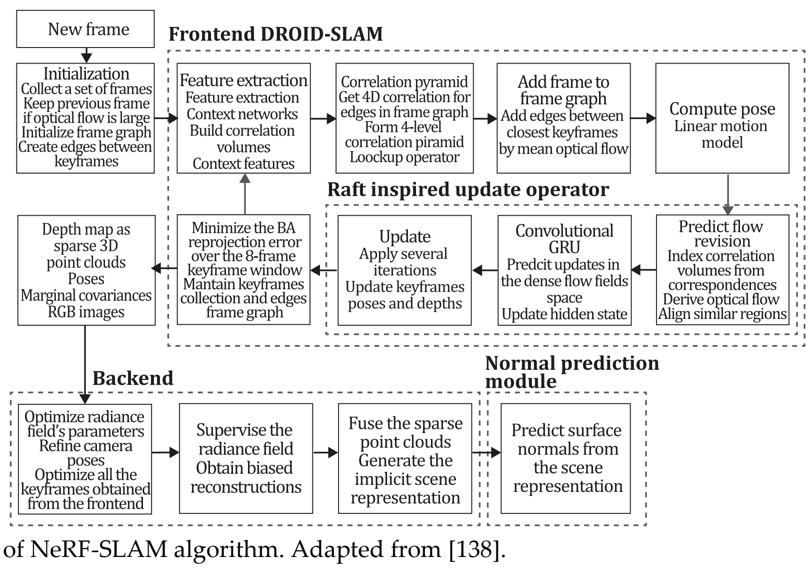

New frame
Frontend DROID-SLAM
Initialization
Collect a set of frames
Keep previous frame if optical flow is large
Initialize frame graph
Create edges between keyframes
Feature extraction
Feature extraction
Context networks
Build correlation volumes
Context features
Correlation pyramid
Get 4D correlation for edges in frame graph
Form 4-level correlation piramid
Loockup operator
Add frame to frame graph
Add edges between closest keyframes by mean optical flow
Compute pose
Linear motion model
Raft inspired update operator
Depth map as sparse 3D point clouds
Poses
Marginal covariances
RGB images
Minimize the BA reprojection error over the 8-frame keyframe window
Mantain keyframes collection and edges frame graph
Update
Apply several iterations
Update keyframes poses and depths
Convolutional GRU
Predcit updates in the dense flow fields space
Update hidden state
Predict flow revision
Index correlation volumes from correspondences
Derive optical flow
Align similar regions
Backend
Normal prediction module
Optimize radiance field's parameters
Refine camera poses
Optimize all the keyframes obtained from the frontend
Supervise the radiance field
Obtain biased reconstructions
Fuse the sparse point clouds
Generate the implicit scene representation
Predict surface normals from the scene representation

Figure 39. Diagram of NeRF-SLAM algorithm. Adapted from [138].

Rosinol et al. (2023). The innovative probabilistic volumetric fusion approach introduced by Rosinol et al. [139] represents a significant advancement in the domain of dense mapping and 3D environmental reconstruction from RGB-D or monocular RGB data. This method has demonstrated superior accuracy, robustness, and processing speed performance when benchmarked against many contemporary state-of-the-art techniques. This method enhances the conventional volumetric fusion process by integrating a probabilistic framework that employs Gaussian Process Regression (GPR). This integration allows for more robust and precise reconstruction of 3D environments from RGB data by factoring in the uncertainty inherent to depth measurements—a critical consideration for making reliable decisions in uncertain environments. A key distinction of this method is its capacity to process numerous frames in real-time, which is a considerable improvement over existing approaches. The method merges dense, noisy depth maps, applying probabilistic uncertainty estimates into a cohesive volumetric representation. Utilizing the Droid-SLAM frontend, the system retrieves pose estimates and dense depth maps, further refined to yield dense uncertainty maps.

In the context of the Droid-SLAM framework, a set of inverse depths for each keyframe is calculated to address the bundle adjustment (BA) challenge. The uncertainties of these inverse depths are derived from the information matrix of the BA problem, leveraging marginal covariances for the per-pixel depth variables. This process facilitates the recovery of sparse depth maps that are subsequently upscaled using the Raft upsampling operator. Depth variances are also computed, taking into account nonlinear uncertainty propagation. Subsequently, an uncertainty-aware volumetric mapping technique is applied, contrasting with Droid-SLAM's ad-hoc depth filter. This technique utilizes the estimated uncertainties from each depth map within a probabilistic volumetric fusion model, offering a robust and mathematically rigorous alternative for reconstructing scene geometry. The final 3D mesh is extracted from the volumetric representation, selectively meshing only those voxels that exhibit uncertainty below a predetermined maximum threshold.

The system's efficacy was rigorously assessed using various datasets, encompassing both synthetic and real-world scenarios. It was juxtaposed with several leading-edge methods, including fusion-based approaches like KinectFusion and ElasticFusion and deep-learning-based methods such as MVSNet and NeRF. The results underscored its robustness and adaptability across different contexts. The precision and quality of the proposed method render it highly applicable to fields such as augmented/virtual reality and computer graphics, potentially revolutionizing virtual gaming environments and film production. Moreover, its exceptional real-time capabilities make it well-suited for robotic applications, particularly in autonomous navigation within uncharted terrains. Figure 40 depicts the Rosinol et al. algorithm inspired by the article [139].

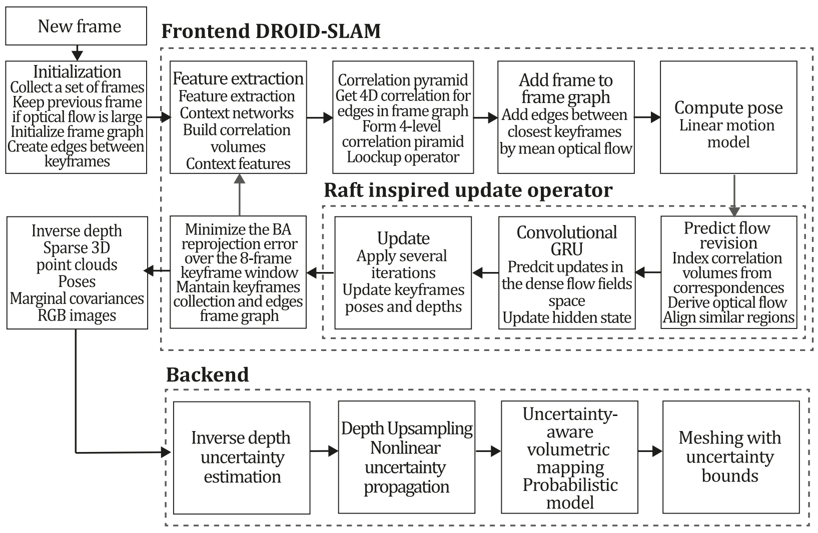

New frame
Frontend DROID-SLAM
Initialization
Collect a set of frames
Keep previous frame if optical flow is large
Initialize frame graph
Create edges between keyframes
Feature extraction
Feature extraction
Context networks
Build correlation volumes
Context features
Correlation pyramid
Get 4D correlation for edges in frame graph
Form 4-level correlation piramid
Loockup operator
Add frame to frame graph
Add edges between closest keyframes by mean optical flow
Compute pose
Linear motion model
Raft inspired update operator
Inverse depth
Sparse 3D point clouds
Poses
Marginal covariances
RGB images
Minimize the BA reprojection error over the 8-frame keyframe window
Mantain keyframes collection and edges frame graph
Update
Apply several iterations
Update keyframes poses and depths
Convolutional GRU
Predcit updates in the dense flow fields space
Update hidden state
Predict flow revision
Index correlation volumes from correspondences
Derive optical flow
Align similar regions
Backend
Inverse depth uncertainty estimation
Depth Upsampling
Nonlinear uncertainty propagation
Uncertainty-aware volumetric mapping
Probabilistic model
Meshing with uncertainty bounds

Figure 40. Diagram of Rosinol et al. algorithm. Adapted from [139].

7.2. ML + Direct methods

In the contemporary landscape of artificial intelligence, machine learning has emerged as an interesting alternative to aid monocular 3D reconstruction. This surge in interest is largely attributable to machine learning's capacity to address the inherent constraints of traditional monocular systems, such as scale indeterminacy, motion distortion, non-textured surfaces, and repetitive patterns. Classic direct systems face formidable hurdles, including reliance on precise initialization, the assumption of consistent brightness, robustness under dim lighting conditions, and adaptability to novel environments. Over the last ten years, the scientific community has made notable strides in overcoming these obstacles by integrating neural network frameworks into established SLAM, VO, and SFM methodologies, enhancing their efficacy. In numerous instances, these modernized versions have surpassed their traditional counterparts in performance.

The categorization of ML-enhanced Direct methods, akin to other classifications, bifurcates into dense and sparse, based on the density of the resultant 3D reconstructions. These variants are detailed in sections 7.2.1 and 7.2.2. Figure 41 depicts a timeline for the emergence of seminal ML-enhanced Direct methods within the previous decade. Among these, CNN-SLAM and CodeSLAM stand out as particularly influential, having made substantial contributions to the avant-garde of the field and garnering impressive citation scores. These approaches have charted new courses for inquiry, with CNN-SLAM integrating semantic segmentation and CodeSLAM employing encoder-decoder structures, respectively.

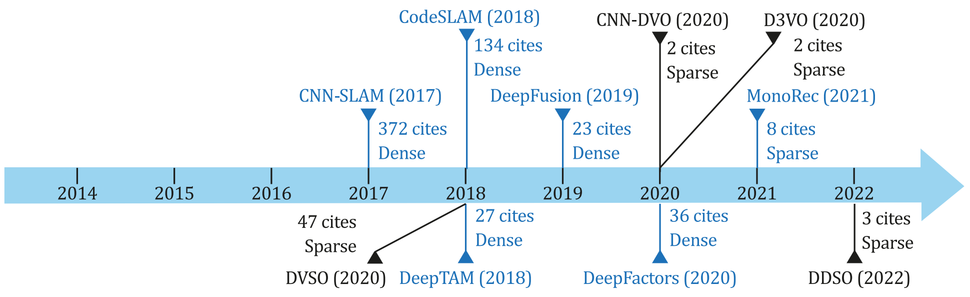


Figure 41. Timeline for the most pivotal SLAM, VO or SFM, ML + direct systems.

7.2.1. ML + Dense + Direct methods

Analogous to traditional SLAM techniques, machine learning-based dense formulations are categorized into dense and sparse types, contingent on the sparsity level of the ultimate reconstruction. Researchers in this field have suggested enhancements to pose and depth estimation and the estimation of additional parameters, such as scale factors and initialization terms. Moreover, as indicated in the preceding section, there is potential for neural networks to be trained to enhance the map outputs from conventional SLAM systems. Consequently, these traditional sparse methods have been expanded to include their machine learning-enhanced dense counterparts. As illustrated in Figure 34, most direct machine learning methodologies are classified under the dense category, reflecting the substantial improvements machine learning has contributed to the density of 3D reconstructions.

CNN-SLAM (2017). The innovative research by Tateno et al. [3] represents a significant leap in monocular SLAM technology by integrating deep convolutional depth prediction with a dense direct system. Through example-based learning, this integration utilizes CNN-generated depth maps to address the challenge of absolute scale estimation in monocular SLAM, not presenting limitations related to traditional assumptions and geometric constraints. The CNN-generated depth map serves as an initial estimate for dense reconstruction, which is subsequently refined using the LSD-SLAM direct method. Unlike most monocular SLAM systems limited by scene absolute scale ambiguity, the CNN-generated depth map introduces absolute scale data, enhancing the accuracy of pose estimation, trajectory, and scene reconstruction. Additionally, while conventional monocular systems struggle with camera rotations due to the absence of a stereo baseline, the CNN approach estimates each frame independently, thus avoiding this issue.

The system builds upon the depth prediction methodology of Laina et al. [10], incorporating the CNN-generated depth map as prior information for the SLAM system with each new keyframe. The network architecture uses a ResNet-50 base pre-trained on ImageNet to gauge environmental scale. Subsequent network layers, replacing the final pooling and fully connected layers, consist of residual up-sampling blocks, followed by the application of dropout and a final convolutional layer that outputs the depth map. The fusion of CNN depth prediction with direct SLAM is accomplished through an uncertainty map, $\mathcal{U}_{k_i}$, which quantifies the elementwise disparity between the depth map of the current keyframe k_i and the nearest keyframe k_j:

$$\mathcal{U}_{k_i}(\boldsymbol{u}) = \left(\boldsymbol{D}_{k_i}(\boldsymbol{u}) - \boldsymbol{D}_{k_j}\left(\Pi\left(\boldsymbol{\zeta}\boldsymbol{T}_{k_j}^{k_i}\mathcal{V}_{k_i}(\boldsymbol{u})\right)\right)\right)^2, \tag{29}$$

where $\boldsymbol{\zeta}$ denotes the camera intrinsics matrix, $\mathcal{V}_{ki}(\boldsymbol{u}) = \boldsymbol{\zeta}^{-1}\dot{\boldsymbol{u}}\boldsymbol{D}_{ki}(\boldsymbol{u})$ represents a 3D vertex map element derived from the current keyframe's depth map, $\boldsymbol{u}$ is a generic depth map element with $\dot{\boldsymbol{u}}$ being its homogeneous representation, $v = \Pi(\boldsymbol{\zeta}\boldsymbol{T}_{kj}^{ki}\mathcal{V}_{ki}(\boldsymbol{u}))$, and $\tilde{\mathcal{U}}_{k_j}$ is the uncertainty associated with the CNN estimation. The depth and uncertainty maps of a frame are combined with those of the closest keyframe to enhance the precision of every initialized keyframe, with the uncertainty of the nearest keyframe defined as:

$$\tilde{\mathcal{U}}_{k_j}(\boldsymbol{v}) = \frac{\boldsymbol{D}_{k_j}(\boldsymbol{v})}{\boldsymbol{D}_{k_i}(\boldsymbol{u})}\mathcal{U}_{k_j}(\boldsymbol{v}) + \sigma_p^2, \tag{30}$$

Subsequently, the two maps are merged using weighted formulas:

$$\boldsymbol{D}_{k_i}(\boldsymbol{u}) = \frac{\tilde{\mathcal{U}}_{k_j}(\boldsymbol{v}) \cdot \boldsymbol{D}_{k_i}(\boldsymbol{u}) + \mathcal{U}_{k_i}(\boldsymbol{u}) \cdot \boldsymbol{D}_{k_j}(\boldsymbol{v})}{\mathcal{U}_{k_i}(\boldsymbol{u}) + \tilde{\mathcal{U}}_{k_j}(\boldsymbol{v})}, \tag{31}$$

$$\mathcal{U}_{k_i}(\boldsymbol{u}) = \frac{\tilde{\mathcal{U}}_{k_j}(\boldsymbol{v}) \cdot \mathcal{U}_{k_i}(\boldsymbol{u})}{\mathcal{U}_{k_i}(\boldsymbol{u}) + \tilde{\mathcal{U}}_{k_j}(\boldsymbol{v})}. \tag{32}$$

In CNN-SLAM, the authors also executed semantic segmentation based on the premise that the same network can undertake high-dimensional regression tasks. This positions CNN-SLAM as a pioneering example of simultaneous semantic segmentation and 3D reconstruction, paving the way for research where multiple 3D regression tasks can be conducted alongside depth prediction. In essence, CNN-SLAM gathers keyframes, refines their pose via pose graph optimization, and concurrently estimates camera pose by determining the transformation to the closest keyframe. Depth maps predicted by CNN are generated for keyframes at high frame rates, and an uncertainty map is created to gauge the confidence of each pixel-wise prediction. Concurrently, a second convolutional network predicts semantic segmentation for each frame. The relative pose is then optimized through a pose graph on keyframes. Figure 42 depicts the CNN-SLAM algorithm inspired by the article [3].

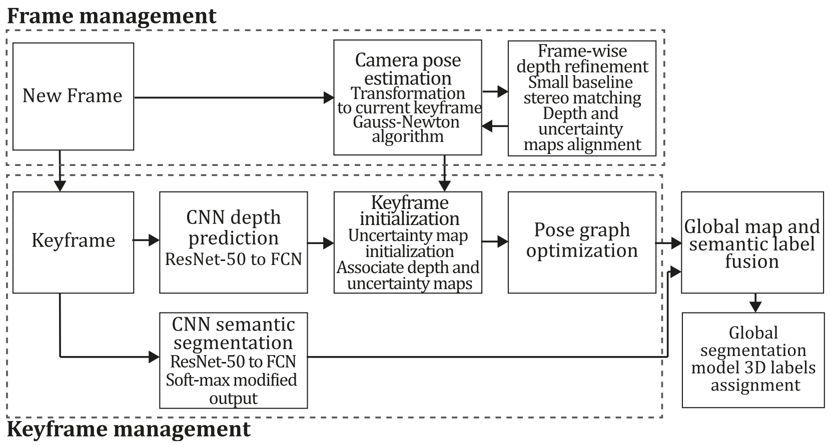

Frame management
New Frame
Camera pose estimation Transformation to current keyframe Gauss-Newton algorithm
Frame-wise depth refinement Small baseline stereo matching Depth and uncertainty maps alignment
Keyframe
CNN depth prediction ResNet-50 to FCN
Keyframe initialization Uncertainty map initialization Associate depth and uncertainty maps
Pose graph optimization
Global map and semantic label fusion
CNN semantic segmentation ResNet-50 to FCN Soft-max modified output
Global segmentation model 3D labels assignment
Keyframe management

Figure 42. Diagram of CNN-SLAM algorithm. Adapted from [3].

DeepTAM (2018). The DeepTAM system, as delineated by Zhou et al. [140], is an evolution of the DTAM framework [17], re-envisioned through the lens of deep learning with dual CNN architectures tailored for tracking and mapping operations. This system's advancements include a tracking network fine-tuned for frame-to-keyframe incremental tracking, a novel approach to camera pose estimation employing multiple hypotheses, a mapping network that fuses depth estimation with image priors, and a depth refinement process that synergizes CNN architectures with a narrow band technique.

In DeepTAM, the tracking network's role is to align the current image with a keyframe, encompassing depth and colour information, to deduce the camera pose. This is achieved by establishing 2D to 3D correspondences between the image and the keyframe. The authors implemented an encoder-decoder architecture that learns the six degrees of freedom (DOF) pose estimation relative to a keyframe, incorporating optical flow to train the network in recognizing the dynamics between image pairs. The decoder in the tracking network concurrently predicts optical flow and generates pose hypotheses. A coarse-to-fine strategy was employed to accommodate tracking of both large and small camera movements, training three separate tracking networks to propose poses at varying resolutions, thus enabling incremental pose estimation with each network specializing in a different resolution scale. The mapping network of DeepTAM draws inspiration from the plane sweep stereo concept, aggregating multi-image data into a cost volume from which a depth map is extracted via a CNN that leverages image-based priors and the collated depth data. The authors introduced a network that iteratively refines the depth estimate to enhance depth prediction by utilizing the cost volume within a narrow band proximal to the geometry inferred from previous frames. The mapping architecture of DeepTAM is bifurcated into fixed- and narrow-band modules that take the input image and cost volume to yield an initial depth estimate and interpolation factor and the latter iteratively constructing a learned cost volume, culminating in a refined depth map produced by a differentiable soft $argmin$ operation followed by a secondary encoder-decoder that refines the depth map further using the keyframe image.

To mitigate the propensity for machine learning models to overfit, authors meticulously designed its architecture and learning configurations to prevent the network from adopting simplistic shortcuts that could impair generalization. They also incorporated data augmentation strategies during training and utilized datasets such as SUN3D [141] and SUNCG [142] to enhance the network's proficiency in 6 DOF motion tracking. As evidenced through empirical evaluations, DeepTAM's robust tracking capabilities have demonstrated superior performance compared to its forerunner, CNN-SLAM. Moreover, its capacity to refine depth maps by processing multiple concurrent images reduces drift. Figure 43 depicts the DeepTAM algorithm inspired by the article [141].

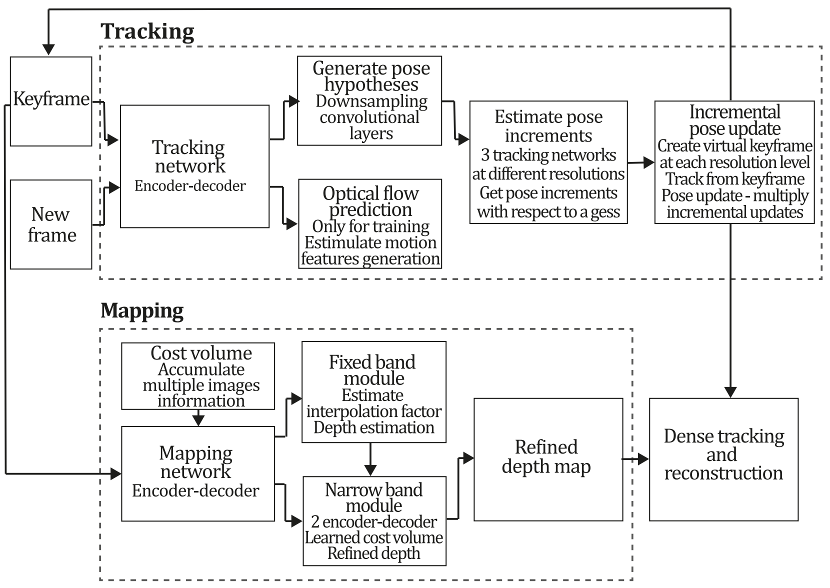

Tracking
Keyframe
New frame
Tracking network Encoder-decoder
Generate pose hypotheses Downsampling convolutional layers
Optical flow prediction Only for training Estimulate motion features generation
Estimate pose increments 3 tracking networks at different resolutions Get pose increments with respect to a gess
Incremental pose update Create virtual keyframe at each resolution level Track from keyframe Pose update - multiply incremental updates
Mapping
Cost volume Accumulate multiple images information
Mapping network Encoder-decoder
Fixed band module Estimate interpolation factor Depth estimation
Narrow band module 2 encoder-decoder Learned cost volume Refined depth
Refined depth map
Dense tracking and reconstruction

Figure 43. Diagram of DeepTAM algorithm. Adapted from [141].

DeepFusion (2019). In the work of Laidlow et al. [9], the DeepFusion system was introduced, designed to generate dense, scaled depth maps and poses in real-time from a monocular SLAM system. This system employs predicted depth gradients as constraints to maintain global consistency in the reconstruction while leveraging learned uncertainties to integrate different data modalities. DeeFusion is conceptualized by merging the sparse outputs of the monocular system ORB-SLAM2 [7] with the depth and gradient predictions from a CNN within a probabilistic framework. This framework utilizes uncertainties derived from the predicted per-pixel mean and variance and combines them with geometric constraints.

The network in DeepFusion is activated once per keyframe, allowing for the continuous optimization of the depth map for each frame with fresh geometric constraints. The system ascertains shape and absolute scale through a cost function incorporating per-pixel losses based on CNN depth predictions and depth estimates, further constrained by the network's depth gradient predictions. The network's architecture is based on the U-Net model [143], adapted to predict log-depth and modified to include three additional decoders for estimating log-depth uncertainties, log-depth gradients, and log-depth gradient uncertainties. The choice of log-depth over depth or inverse depth is due to its scale-invariant characteristics. An associated uncertainty is integrated into the system to merge the CNN outputs with the estimates from the monocular SLAM system for each pixel in the log-depth and gradient images. This integration is achieved by training the network to predict the mean and variance using the SceneNet RGB-D dataset and employing a maximum likelihood cost function:

$$\mathcal{L}_{NN}(\boldsymbol{\omega}) = \sum_{i} \frac{\left(y_i - \mu_{\boldsymbol{\omega},i}(\boldsymbol{x})\right)}{\sigma_{\boldsymbol{\omega},i}(\boldsymbol{x})^2} + \log\left(\sigma_{\boldsymbol{\omega},i}(\boldsymbol{x})^2\right), \tag{33}$$

where $\boldsymbol{\omega}$ represents the network weights, $\boldsymbol{x}$ is the set of input pixels, y_i is the ground truth for each pixel, and $\mu_{\boldsymbol{\omega},i}(\boldsymbol{x})$, $\sigma_{\boldsymbol{\omega},i}(\boldsymbol{x})^2$ are the predicted mean and variance, respectively. Monocular depth estimation follows the method of Engel et al. [22], searching for depth values along the epipolar line for texture-rich pixels in the keyframe by minimizing the sum of squared differences at five equidistant points. As a direct dense system, DeepFusion estimates depth by minimizing the photometric error:

$$E_i = \Im_i\left(\Pi\left(\boldsymbol{\zeta}\boldsymbol{T}_{WC_1}^{-1}\boldsymbol{T}_{WC_0}\Pi^{-1}\left(\boldsymbol{x}_i, d_{semi,i}\right)\right)\right) - \Im_0(\boldsymbol{x}_i), \tag{34}$$

where $\boldsymbol{\zeta}$ contains the camera intrinsics, $\boldsymbol{T}_{WC_0}$ and $\boldsymbol{T}_{WC_1}^{-1}$ are the poses for the keyframe and reference frame, respectively. $\Im_*(\cdot)$ returns intensity values for each pixel, Π is the projection and homogenization function, and $\Pi^{-1}(\boldsymbol{x}_i, d_{semi,i})$ is the back-projection function that yields a 3D point for each pixel with depth $d_{semi,i}$. The optimal depth is determined through interpolation between two steps. The uncertainty for each semi-dense measurement is approximated by $\sigma_i^2 = (J^T J)^{-1}$, where J is the Jacobian of the error function. These depth estimates and uncertainties are then converted to log space to align with the CNN outputs.

In brief, the system computes the pose using ORB-SLAM2 for each new RGB image. Subsequently, the Engel et al. [22] algorithm updates semi-dense depth estimates for each keyframe. If a new keyframe is selected, the CNN predicts log-depth, log-depth gradients, and associated uncertainties. If the current frame is not a keyframe, it contributes to optimization, where a set of log-depth values and scale correction factors are minimized within the Opt optimization framework [144]. The final output is a fusion of depth and pose, resolved using an associated uncertainty formulation. Figure 44 describes the DeepFusion algorithm inspired by the article [9].

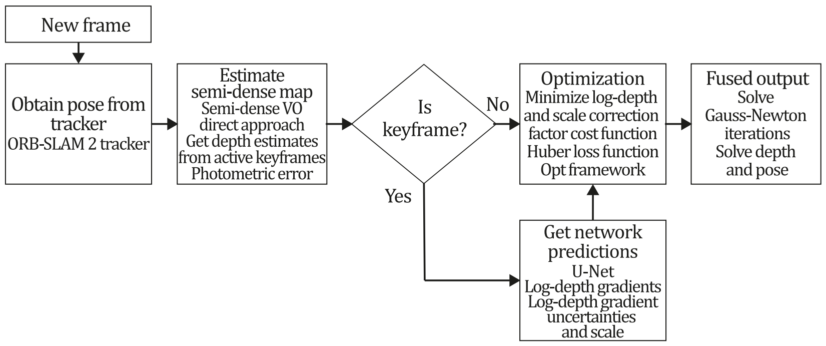


Figure 44. Diagram of DeepFusion algorithm. Adapted from [9].

CodeSLAM (2018). The work presented in Bloesch et al. [8], is an innovative system designed to capture a dense yet compact representation of scene geometry. This system generates a 'code' from a single image combined with an intensity image, where the code is a concise set of parameters that encapsulates the scene's geometry without losing detail. The underlying concept is that encoding the full depth of an image is redundant since much of the necessary information is already present in the image intensities. Therefore, the code is crafted to preserve only the information that cannot be inferred from these intensities.

In this system, the depth map is expressed as a function of the image I and an undetermined code c, which is resolved using a neural network $D = D(I, c)$. The CNN architecture employed here merges the monocular depth estimation framework of Zhou et al. [145] with the encoder network of Kingma & Welling [146], fine-tuned to produce a minimal code size while maximizing accuracy through training to minimise reconstruction error. The process begins with a U-Net taking the intensity image to extract a high-dimensional image representation. The features from the intensity image are then encoded and decoded to derive the image's depth map. The encoding and decoding involve a standard down-sampling architecture with strided convolutions, leading to a variational component at the autoencoder's bottleneck. This component comprises two fully connected layers that sample the code from a Gaussian distribution, incorporating a regularization cost. The code is then decoded through an upsampling architecture that employs bilinear interpolation, integrating the processed image intensities from preceding layers. Additionally, the network calculates the mean and depth uncertainty across four pyramid levels. CodeSLAM's inference mechanism operates within an N-frame Structure from Motion framework, where codes and poses are refined using loss functions that account for photometric and geometric discrepancies. Residuals and Jacobians are computed to facilitate the application of a damped

Gauss-Newton algorithm, which determines the optimal code and pose for each frame. Tracking is accomplished by estimating the pose of the current keyframe relative to the pre-existing keyframe map.

The SLAM methodology underpinning CodeSLAM is inspired by the PTAM concept [90], which alternates between tracking and mapping. The system initiates by processing two frames to optimize their relative poses and codes jointly. After several iterations, a keyframe is introduced to conduct a global optimization once a baseline is established. The geometry encoding strategy of CodeSLAM, which facilitates the concurrent optimization of geometry and motion, has demonstrated enhanced performance on the EuRoC dataset compared to its predecessors. This advancement has endowed the system with increased robustness, particularly in handling rapid and purely rotational movements. Figure 45 depicts the CodeSLAM algorithm inspired by the article [8].

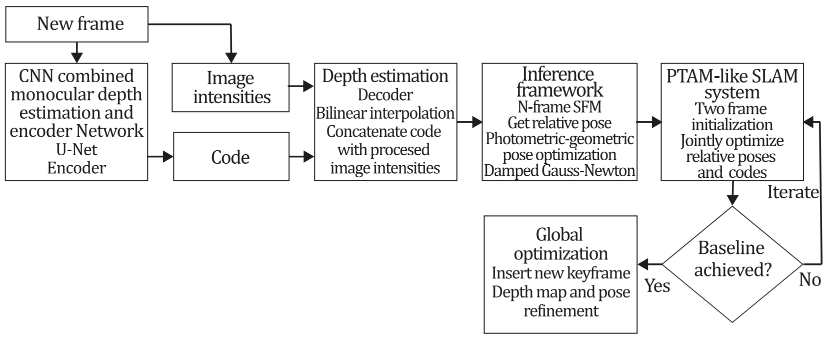

New frame
CNN combined monocular depth estimation and encoder Network U-Net Encoder
Image intensities
Code
Depth estimation Decoder Bilinear interpolation Concatenate code with procesed image intensities
Inference framework N-frame SFM Get relative pose Photometric-geometric pose optimization Damped Gauss-Newton
PTAM-like SLAM system Two frame initialization Jointly optimize relative poses and codes
Iterate
Baseline achieved?
Yes
No
Global optimization Insert new keyframe Depth map and pose refinement

Figure 45. Diagram of CodeSLAM algorithm. Adapted from [8].

DeepFactors (2020). DeepFactors, as introduced by Czarnowski et al. [6], represents a probabilistic dense SLAM system that integrates an efficiently learned depth map representation. This innovative system integrates the foundational principles of classical SLAM with geometry priors derived from data within a probabilistic factor-graph framework. The system is characterized by three distinct types of errors, or factors, which are instrumental in estimating the camera trajectory and the geometry of the scene: the photometric factor, which is the disparity in image intensities compared to a target image; the reprojection factor, which measures the variance between observed and predicted matched landmark locations; and the sparse geometric factor, which assesses the differences in scene geometry by comparing depth maps of a frame to its target image.

The ablation studies conducted by Czarnowski et al. considered the photometric error as a foundational system component, given its direct method nature. These studies concluded that the reprojection error is beneficial for avoiding local minima and enhancing the convergence rate. Meanwhile, the geometric error introduces a prior understanding of the world, which is particularly useful in anchoring depth maps to create a unified reconstruction in areas devoid of photometric information. DeepFactors advances the concepts initially presented in CodeSLAM [8], albeit with a modified mapping backend that redefines the problem as a multiview bundle adjustment. This redefinition includes adopting a three-factor error formulation and enhancements in keyframing, map maintenance, and tracking. Unlike DeepTAM [140], DeepFactors emphasises network generalisation less, opting for a more robust optimization process that corrects suboptimal predictions from the network. Here, neural networks are primarily employed to derive an image-conditioned manifold, the basis for the optimization process.

The authors leveraged a General-Purpose GPU (GPGPU) to facilitate real-time performance and restructured the CodeSLAM network architecture. They implemented a U-Net to distil features from input images, reducing image size through successive convolution steps. A Variational Auto-Encoder (VAE) is then utilized to learn a compact, optimizable depth representation, which the decoder translates back into depth values. The features intricately condition the encoder and decoder through concatenation processes. An additional output from the feature network is an uncertainty parameter, which is incorporated into the negative log of a Laplacian likelihood loss, serving as a supervisory metric for the reconstructed depth. Concurrently, the predicted depth map is regulated with an L1 loss. Figure 46 exemplifies the DeepFactors algorithm inspired by the article [6].

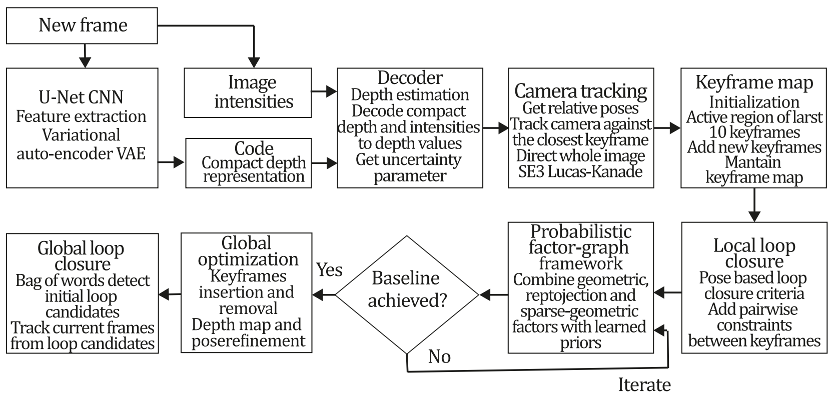


Figure 46. Diagram of DeepFactors algorithm. Adapted from [6].

7.2.2. ML + Sparse + Direct methods

In the realm of direct methods, akin to traditional techniques, a bifurcation exists based on the density of the resultant 3D map from the reconstruction process, leading to a classification into dense and sparse categories. Integrating machine learning into these methodologies yields two classifications: ML + Dense + Direct and ML + Sparse + Direct. ML + Sparse + Direct is frequently adopted due to the notable successes achieved by sparse direct classic methods such as DSO [2]. This section is dedicated to showcasing the most effective machine learning enhancements applied to the DSO system, including advanced versions where neural network integration has been employed to augment map quality, tracking accuracy, and generalization capacity and to address common failure scenarios such as violations of brightness constancy, motion blur, and repetitive textures.

Several initiatives outlined in this section have successfully attained denser reconstructions [147], [148], yielding dense or semi-dense output maps by integrating machine learning techniques. Nonetheless, these methods are categorized under ML + Sparse + Direct because they fundamentally operate on a sparse selection of points.

DVSO (2018). The pioneering "Deep Virtual Stereo Odometry" (DVSO) system, introduced by Yang et al. [111], builds upon the Direct Sparse Odometry (DSO) framework [2] and incorporates deep learning concepts alongside the stereo extension of DSO (Stereo DSO) [149]. DVSO ingeniously simulates a stereo odometry system utilizing solely monocular input. The system's efficacy is bolstered by a semi-supervised neural network that predicts depth maps from single images, initializing sparse depths for the DSO algorithm at a uniform depth scale. Additionally, it enhances odometry by integrating a novel virtual stereo term that aligns the depth estimated in DSO's windowed bundle adjustment with the depth predictions from the network.

The semi-supervised neural network employed in DVSO is tailored to predict refined disparity estimates, incorporating three pivotal components to elevate performance: a self-supervised learning paradigm anchored in photo-consistency and image reconstruction loss, a supervised learning strategy utilizing Stereo DSO depth predictions as a proxy for ground truth, and a dual-stage refinement process for network predictions deploying a stacked encoder-decoder network architecture. This network termed StackNet, layers two distinct networks, SimpleNet and ResidualNet, drawing inspiration from DispNet [150] with an encoder-decoder design. SimpleNet, utilizing a ResNet-50 backbone with skip connections, projects feature maps back to their original resolution, producing disparity maps across various scales. Conversely, ResidualNet, modelled after FlowNet 2.0 [132], refines the disparity maps generated by SimpleNet, processing this data in a stereo format that includes a reconstructed right image (achieved by warping the input image to a rectified stereo view), a synthesized left image, and the associated reconstruction error.

The learning process is meticulously directed by a loss function comprising five linearly combined terms: self-supervised loss for evaluating the reconstructed image quality; supervised loss for gauging the predicted disparity map's deviation from the sparse pixel disparities estimated by DSO; left-right disparity consistency loss to ensure symmetry between the left and right disparity images; disparity smoothness regularization to encourage local smoothness in the predicted disparity map; and occlusion regularization to refine the system's handling of background depths and abrupt transitions at occlusions. Through this finely calibrated approach, the authors achieved superior performance over traditional monocular VO systems and even stereo VO systems despite relying exclusively on monocular input. Figure 47 depicts the DVSO algorithm inspired by the article [111].

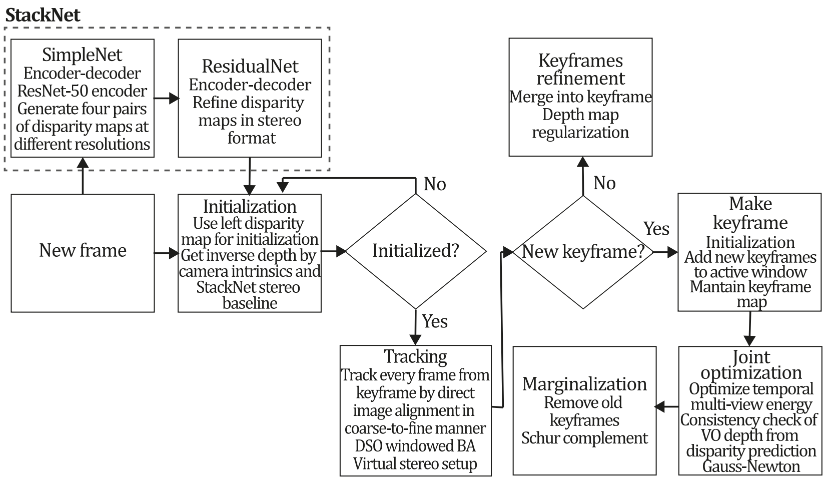

StackNet
SimpleNet
Encoder-decoder
ResNet-50 encoder
Generate four pairs of disparity maps at different resolutions
ResidualNet
Encoder-decoder
Refine disparity maps in stereo format
New frame
Initialization
Use left disparity map for initialization
Get inverse depth by camera intrinsics and StackNet stereo baseline
Initialized?
No
Yes
Tracking
Track every frame from keyframe by direct image alignment in coarse-to-fine manner
DSO windowed BA
Virtual stereo setup
New keyframe?
No
Yes
Keyframes refinement
Merge into keyframe
Depth map regularization
Make keyframe
Initialization
Add new keyframes to active window
Mantain keyframe map
Joint optimization
Optimize temporal multi-view energy
Consistency check of VO depth from disparity prediction
Gauss-Newton
Marginalization
Remove old keyframes
Schur complement

Figure 47. Diagram of DVSO algorithm. Adapted from [111].

CNN-DVO (2020). In the realm of artificial intelligence and computer vision, [21] have pioneered a novel approach with their CNN-DVO framework, which integrates deep learning with sparse direct visual odometry. This integration is achieved by leveraging deep convolutional neural network (CNN) depths as initial priors, which are then utilized across various stages, including initialization, local bundle adjustment, and loop closure. The CNN-DVO stands out as the inaugural framework to holistically integrate depth prediction with the initialization, tracking, and marginalization components, thereby enabling the simultaneous optimization of all model parameters, such as inverse depth, camera pose, and the affine model. The researchers also demonstrated the advantages of incorporating depth priors, notably in stabilizing the initialization scale, mitigating scale drift during tracking, and restoring 3D measure correspondence in loop closures while preserving consistent scale [21].

Furthermore, the study introduces an enhanced strategy for point selection, targeting regions with low and high gradients to prevent clustering. This strategy enhances the adaptability of the tracking thread, particularly in scenarios with large stereo baselines and extreme motion, thus bolstering the robustness of pose estimation. The optimization process is bifurcated into two distinct phases: initially resolving the pose using a coarse prior, then refining the depth based on the pose values obtained in the preliminary stage. Drawing inspiration from Gao et al. (2018), the methodology delineates the SLAM challenge into coarse tracking and map refinement. The authors also implemented a keyframe selection mechanism predicated on marginalization outcomes and a pose graph, employing a bag-of-words (BoW) database of ORB features uniformly extracted from the image space, which aids in ensuring the reproducibility of mapped points in subsequent frames, thereby facilitating effective loop closure. The method further incorporates a dynamic point sampling strategy within multiscale image sections. After selecting tracking points and applying the Direct Sparse Odometry (DSO) point selection criteria, the system calculates gradient histograms in defined kernel regions, selectively discarding points in sparse, low-gradient areas. The resulting set of tracking points is then optimized using a sliding window approach. The Monodepth2 model [151] serves as the CNN depth predictor, providing essential data for estimating the camera pose for the upcoming frame and refining the pose estimation. Subsequently, a Gauss-Newton optimization refines the reprojection locations and recovers the inverse depth along the epipolar line. The equation defines the variance of each observation:

$$\sigma_d^2 = \lambda\left(\frac{d'}{6}\right)^2 + (1-\lambda)\sigma_{d,obs}^2, \tag{35}$$

where d^* denotes the inverse depth, d' the predicted inverse depth, λ weight (empirically set to 0.57), σ_d the standard deviation, and $\sigma_{d,obs}$ the observation variance. This empirical approach allows for the convergence of noisy depth predictions. Depth prior is then incorporated as a residual to penalize deviations in inverse depth between keyframes, ensuring proper scaling of estimated transformations:

$$r_{id} = \left(I_j\left[p'\left(\boldsymbol{T}_i, \boldsymbol{T}_j, d', \boldsymbol{\zeta}\right)\right] - b_j\right) - \frac{t_j e^{a_j}}{t_i e^{a_i}}\left(I_i(p) - b_i\right), \tag{36}$$

where $\boldsymbol{T}_i, \boldsymbol{T}_j$ are the camera poses, $\boldsymbol{\zeta}$ is the camera intrinsics matrix, a_i, a_j, b_i, b_j correspond to affine bright coefficients, I_i and I_j are a pair of keyframes, p' in I_j are projected points from p in I_i. This optimization process is akin to that used in DSO, with BRIEF descriptors extracted from keyframes and maintained within the same BoW database for loop detection queries. RANSAC PnP initializes transformations, followed by the optimization of $Sim(3)$ transformations using 3D-2D geometric constraints. The depth filter initializes inverse depth estimation for each selected point, with error variance formulated as:

$$\sigma_{d^*}^2 = \alpha^2\left(\boldsymbol{\zeta}_d + \boldsymbol{\zeta}_d \frac{J_d \Sigma J_d + J_d' \Sigma J_d'}{J_d \Sigma J_d} + \sigma_d^2\right), \tag{37}$$

where the inverse depth $d^* = d(I_0, I_1, d', \xi, \Pi)$ is a function of depth prior and geometrical projection input, d' is the inverse depth prior, ξ is the relative transformation matrix, Π is the projection function, c_d is the normalization constant (empirically set to 0.2), J_d is the Jacobian of d, $J_d' = [dx, -dy]^T$ is the conjugated Jacobian of d, $\Sigma = \left[I_x, I_y\right]^T\left[I_x . I_y\right]$ is the input error covariance, and α is the proportionality constant defined in [22]. The inverse depth prior is then initialized, and a Gauss-Newton optimization refines the upper bound of inverse depth for each map point candidate through the inverse depth residual r_{id}:

$$r_{id} = \sum_{i \in \mathcal{P}_i} \| I_1[\Pi(\boldsymbol{q}_i, d, \xi) - e^a I_0(\boldsymbol{q}_i) + b] \|_\gamma, \tag{38}$$

where γ is the Huber norm, and $\mathcal{P}_i$ is the residual pattern. Depth prior effectively narrows the search region, rendering the method more amenable to challenges posed by large baseline stereo configurations. Depth propagation for tracking updates the inverse depth using a formulation akin to the Kalman filter, where a noisy observation $\mathcal{N}(d', \sigma_{d'}^2)$ is fused with the geometrical propagation prior $\mathcal{N}(d_1, \sigma_{d_1}^2)$::

$$d_1'(d_1, d') = \mathcal{N}\left(\frac{\sigma_{d_1}^2 d' + \sigma_{d'}^2 d_1}{\sigma_{d_1}^2 + \sigma_{d'}^2}, \frac{\sigma_{d_1}^2 \sigma_{d'}^2}{\sigma_{d_1}^2 + \sigma_{d'}^2}\right).$$

In summary, CNN-DVO is predicated on the principles of direct SLAM, extracting a pose hessian prior for each new frame, initializing depth through dynamic point sampling, estimating coarse pose via a sliding window approach, and refining depth with CNN predictions. This continuous propagation of depth into each tracking frame is refined by local bundle adjustment. The tracking thread then performs marginalization to resolve the camera pose and eliminate outlier map points, refining the depth map and pose graph using a BoW database and ORB features for loop closure. Ultimately, $Sim(3)$ transformations are optimized through 3D-2D geometric constraints. Figure 48 describes the CNN-DVO algorithm inspired by the article [21].

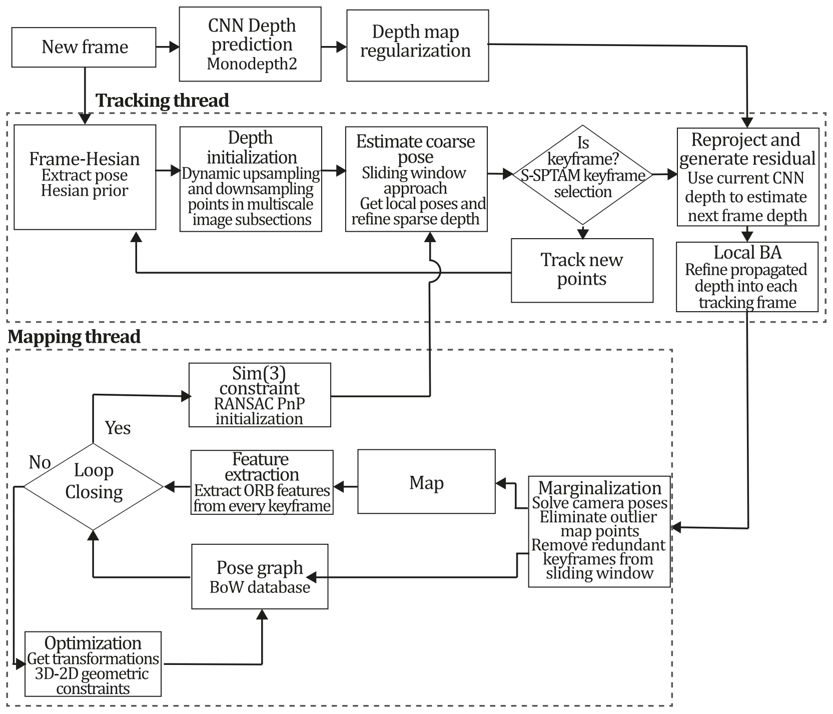

New frame
CNN Depth prediction Monodepth2
Depth map regularization
Tracking thread
Frame-Hesian Extract pose Hesian prior
Depth initialization Dynamic upsampling and downsampling points in multiscale image subsections
Estimate coarse pose Sliding window approach Get local poses and refine sparse depth
Is keyframe? S-SPTAM keyframe selection
Reproject and generate residual Use current CNN depth to estimate next frame depth
Track new points
Local BA Refine propagated depth into each tracking frame
Mapping thread
Sim(3) constraint RANSAC PnP initialization
Yes
No
Loop Closing
Feature extraction Extract ORB features from every keyframe
Map
Marginalization Solve camera poses Eliminate outlier map points Remove redundant keyframes from sliding window
Pose graph BoW database
Optimization Get transformations 3D-2D geometric constraints

Figure 48. Diagram of CNN-DVO algorithm. Adapted from [21].

D3VO (2020). In the realm of computer vision and SLAM (Simultaneous Localization and Mapping), the D3VO system, as introduced by Yang et al. [112], represents an evolution of the DSO framework [2]. This system integrates deep learning with Visual Odometry (VO) to enhance monocular depth prediction, camera pose estimation, and the assessment of photometric uncertainty. D3VO distinguishes itself from DVSO by eschewing semi-supervised methods that depend on depth data from StereoDSO [149], instead opting for training exclusively with stereo video footage without requiring of any external depth guidance signals.

The methodology involves the incorporation of DepthNet and PoseNet, which are convolutional neural networks that emulate a U-Net-like structure, building on the foundation provided by MonoDepth2 [152]. These networks are designed to predict not only depth and pose but also parameters for brightness transformation and photometric uncertainty, thus extending the system's proficiency in handling variations in illumination, reflective surfaces, detailed textures, motion blur, and dynamic objects that could otherwise violate the brightness constancy assumption. They are assigned a reduced weight to mitigate the impact of pixels likely to infringe upon this assumption. The self-supervised training network and the VO system converge on similar photometric goals, prompting the authors to suggest substituting the empirical weighting function based on photometric residuals in DSO with learned weights. The D3VO's neural network is adept at estimating depth, pose uncertainty, and affine brightness transformation parameters, facilitating the alignment of illumination across training image sets in a self-regulated manner. Consequently, the network can predict photometric uncertainty for each pixel based on the range of potential brightness values, seamlessly integrating this predicted data into both the tracking front-end and the photometric bundle adjustment in the backend. Addressing the inaccuracies introduced by the failure of the brightness constancy assumption, the authors employ the concept of heteroscedastic aleatoric uncertainty for neural networks [153] to forecast a probability distribution for each pixel characterized by its mean and variance. This predictive uncertainty allows the CNN to modify the weighting of residuals based on the input data, thereby enhancing the model's resilience to noisy data. Contrary to traditional SLAM systems that initiate depth values arbitrarily, D3VO utilizes DepthNet's predicted depth values, which are informed by scale prior knowledge. Moreover, it replaces the constant velocity model with PoseNet's predicted poses to construct a nonlinear factor graph. This graph facilitates tracking new frames from the current keyframe through direct image alignment. D3VO leverages these predicted poses for initializations in both the tracking front-end and the optimization backend, while also incorporating depth information as a stabilizing factor in the energy function for photometric bundle adjustment.

Empirical evidence suggests that D3VO's amalgamation of techniques surpasses its predecessor and delivers trajectory estimation results on par with the most advanced visual-inertial odometry methods despite solely relying on monocular data. Figure 49 depicts the D3VO algorithm inspired by the article [112].

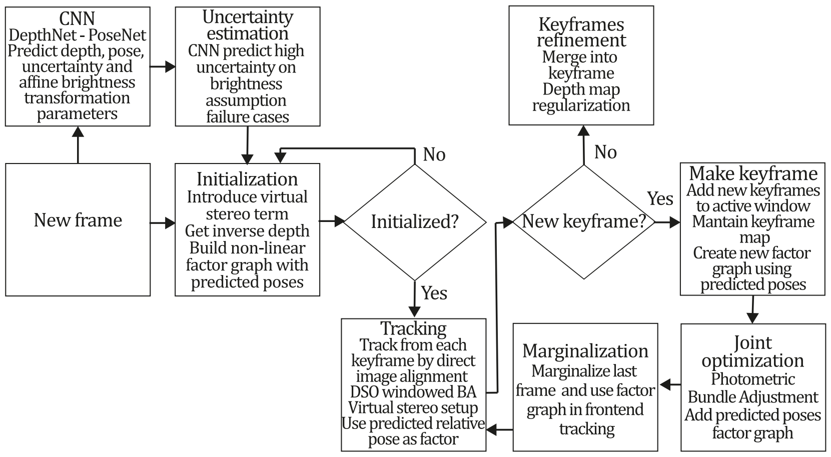

CNN
DepthNet - PoseNet
Predict depth, pose, uncertainty and affine brightness transformation parameters
Uncertainty estimation
CNN predict high uncertainty on brightness assumption failure cases
Keyframes refinement
Merge into keyframe
Depth map regularization
No
No
New frame
Initialization
Introduce virtual stereo term
Get inverse depth
Build non-linear factor graph with predicted poses
Initialized?
New keyframe?
Yes
Make keyframe
Add new keyframes to active window
Mantain keyframe map
Create new factor graph using predicted poses
Yes
Tracking
Track from each keyframe by direct image alignment
DSO windowed BA
Virtual stereo setup
Use predicted relative pose as factor
Marginalization
Marginalize last frame and use factor graph in frontend tracking
Joint optimization
Photometric Bundle Adjustment
Add predicted poses factor graph

Figure 49. Diagram of D3VO algorithm. Adapted from [112].

MonoRec (2021). Building upon the foundational concepts of Direct Sparse Odometry (DSO) [2], Wimbauer et al. introduced MonoRec [147], a system that transcends traditional VO and SLAM paradigms. Designed primarily as a 3D reconstruction system, MonoRec adeptly handles static and dynamic environments by harnessing the capabilities of two specialized neural networks: a MaskModule and a DepthModule. These networks work in tandem to broaden the scope of monocular Structure from Motion (SFM) applications. Depth prediction techniques that integrate deep learning have been bifurcated into two streams: multiview stereo (MVS) [105], [112], [154] and monocular depth prediction [112], [152], [155]. While both approaches aim to deduce depth information, their applications differ. MVS methods leverage multiple viewpoints to infer depth, operating under the premise that the environment remains static. Consequently, these methods struggle with accuracy in scenes with moving objects. On the other hand, monocular depth prediction techniques, which derive depth from single images, excel with moving subjects but are heavily dependent on the object's appearance relative to the camera's positioning.

MonoRec ingeniously amalgamates these two methodologies to harness their respective strengths. The MaskModule utilizes a cost volume tensor, constructed using the Structural Similarity Index Measure (SSIM) from multiple views, to pinpoint and de-emphasize moving pixels by down-weighting their representation in the cost volume. This module employs a U-Net-like architecture with skip connections, drawing on pre-trained ResNet-18 features to predict a probabilistic mask that discerns moving objects through inconsistent geometric data across cost volumes. The DepthModule, based on a U-Net architecture, predicts an inverse depth map by processing the complete cost volume alongside an image. The integration of the mask ensures that inaccuracies caused by moving objects are filtered out from the depth predictions. MonoRec's training regimen is a multistage process that eschews the need for LiDAR ground truth data, instead adopting a semi-supervised loss formulation. Initially, each module is trained independently: the DepthModule is refined using a semi-supervised loss that combines a self-supervised photometric component with an edge-aware smoothness constraint (Godard et al., 2019), while a binary cross-entropy loss between the predicted mask and the actual ground truth guides the MaskModule's training. Subsequent refinement stages for the MaskModule introduce a supervised loss to enhance stability and precision in mask prediction, preventing overfitting by adjusting the cost volume structure. The final stage of DepthModule refinement enables the system to accurately predict depths for moving objects through additional stereo processing, using the depth map as a prior.

Empirical evaluations of MonoRec on datasets such as KITTI, TUM-Mono, and Oxford Robot-Car have showcased its proficiency in generating semi-dense point clouds of scenes. Ablation studies further underscore the critical role of the MaskModule and DepthModule refinements in augmenting the detection and mapping of moving objects. Figure 50 describes the MonoRec algorithm inspired by the article [147].

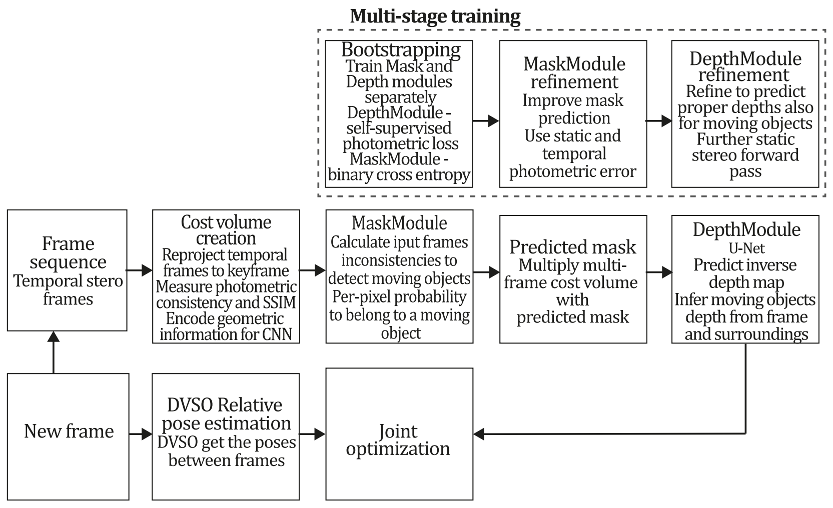

Multi-stage training
Bootstrapping
Train Mask and Depth modules separately
DepthModule - self-supervised photometric loss
MaskModule - binary cross entropy
MaskModule refinement
Improve mask prediction
Use static and temporal photometric error
DepthModule refinement
Refine to predict proper depths also for moving objects
Further static stereo forward pass
Frame sequence
Temporal stero frames
Cost volume creation
Reproject temporal frames to keyframe
Measure photometric consistency and SSIM
Encode geometric information for CNN
MaskModule
Calculate iput frames inconsistencies to detect moving objects
Per-pixel probability to belong to a moving object
Predicted mask
Multiply multi-frame cost volume with predicted mask
DepthModule
U-Net
Predict inverse depth map
Infer moving objects depth from frame and surroundings
New frame
DVSO Relative pose estimation
DVSO get the poses between frames
Joint optimization

Figure 50. Diagram of MonoRec algorithm. Adapted from [147].

DDSO (2022). Zhao et al. [28] have advanced the field of monocular SLAM with their development of DDSO, an innovative system that builds upon the DSO visual odometry framework [2]. This system integrates unsupervised deep neural networks to refine the precision and robustness of the DSO system. As a direct method, DSO calculates the camera pose using photometric data, thus obviating the requirement for feature descriptor computation. Despite its efficiency, direct methods are susceptible to photometric variances between frames and are contingent on precise initialization, which can be problematic in complex environments. The accuracy of pose estimation in DSO relies heavily on the image alignment algorithm, which determines inter-frame poses by optimizing an initial pose based on a constant motion model. This model, however, presumes that the motion between frames remains unchanged, an assumption that falters in the presence of rapid movements, motion blur, or repetitive textures. Initially, DSO employs a unit matrix for lack of prior camera pose information, an empirical choice that DDSO seeks to improve.

DDSO enhances the DSO framework by employing a deep-learning-based pose estimation for initialization and tracking, thus enriching the constant motion model with inter-frame pose data. This enhancement is facilitated by TrajNet, a CNN trained unsupervisedly with four geometric constraints, with the study's significant contribution being the novel pose-to-trajectory constraint. TrajNet operates in conjunction with DepthNet, a depth estimation network where the geometric constraints between the outputs of each network pair serve as a training guide. The supervisory signal for TrajNet comprises the view reconstruction constraint, which accounts for reconstruction errors from consecutive frame pairs using the same depth map; the smoothness constraint, encouraging the detailed representation of geometry; the depth alignment constraint, ensuring scale consistency across adjacent depth maps; and the innovative pose-to-trajectory constraint, which enhances the network's trajectory generation by maintaining scale consistency across three consecutive poses, considering inter-frame poses. In the DSO system, the Gauss-Newton algorithm is employed to optimize the total photometric error across a sliding window of keyframes, framing the process as a nonlinear optimization problem that requires an initial transformation, typically a unit matrix, which is then iteratively refined. Additionally, DSO's protocol for reinitialization during tracking loss involves multiple motion models and small rotations, a complex and computationally intensive process.

DDSO's TrajNet offers initial transformations and models for failure scenarios, simplifying and improving the original DSO's reinitialization process. Empirical evaluations have demonstrated that the integration of TrajNet into DDSO significantly surpasses the performance of the original DSO system, achieving robust and precise trajectories and depth maps without the need for intricate calibration. Figure 51 presents the DDSO algorithm inspired by the article [28].

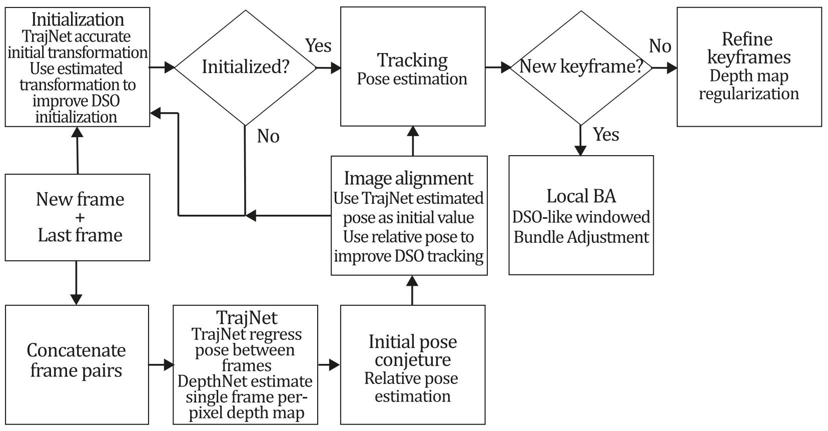

Initialization
TrajNet accurate initial transformation
Use estimated transformation to improve DSO initialization
Initialized?
Yes
No
Tracking
Pose estimation
New keyframe?
No
Yes
Refine keyframes
Depth map regularization
New frame + Last frame
Image alignment
Use TrajNet estimated pose as initial value
Use relative pose to improve DSO tracking
Local BA
DSO-like windowed Bundle Adjustment
Concatenate frame pairs
TrajNet
TrajNet regress pose between frames
DepthNet estimate single frame per-pixel depth map
Initial pose conjeture
Relative pose estimation

Figure 51. Diagram of MonoRec algorithm. Adapted from [28].

7.3. ML + Hybrid methods

Within the spectrum of methodologies for SLAM (Simultaneous Localization and Mapping), there exists a hybrid classification that adeptly merges the direct and feature-based (indirect) approaches. The Semi-Direct Visual Odometry (SVO) system, developed by [13], exemplifies this category by synergizing the direct acquisition of pixel-level data with feature-based techniques to facilitate camera tracking and the generation of sparse depth maps. In the ensuing years, the SVO framework has undergone enhancements, notably the incorporation of a Convolutional Neural Network (CNN) to bolster its generalization capabilities and improve initialization, a topic that will be further explored in the subsequent section. Although the body of literature in this hybrid category is not extensive, it has garnered a favourable reception within the scientific community, particularly for its applications in robotics. This acknowledgment is reflected in the substantial citation metrics these works have achieved, indicating their influence and utility in the field.

CNN-SVO (2019). As mentioned, the implementation of depth filters in the SVO mapping thread and the proposed use of direct pixel matching in the semi-direct framework of SVO have enabled the system to achieve efficient camera motion estimation at high frame rates. However, this proposal still has shortcomings, especially regarding the high-depth uncertainty in the map point initialization process. This problem is addressed in the study of [54], where authors incorporated a CNN to overcome this depth uncertainty limitation. The system was built entirely over the SVO framework [13], adding depth prior knowledge obtained by the single-image CNN MonoDepth of [152] for reducing the uncertainty in identifying feature correspondences, which was built over the Resnet50 backbone using a variant of its encoder-decoder architecture. In SVO's original proposal, the system was divided into mapping and tracking threads. Depth values for each feature are obtained by finding feature correspondence over the epipolar line recovering depth by triangulation. The mapping thread runs the initialization of new map points with high depth uncertainty, updating this depth uncertainty using depth filters created to approximate the mean and variance of current depth values to separate inliers from outliers; hence, a depth filter converges when a point depth uncertainty is small. However, in the original SVO, depth uncertainty tends to be large, leading to two problems: erroneous feature correspondence on the epipolar line and many depth estimations far from converging to their true depth. Consequently, the CNN depth prediction is used to better estimate the mean and variance used in each depth filter, allowing for faster and more accurate convergence.

As shown in Figure 46, MonoDepth is added as a depth estimation module in the mapping thread providing strong depth priors in the map points initialization process to initialize the depth filters. Whereas the original SVO proposal initialized depth-filters as average depth measurements of current image $\mu_n = 1/d_{avg}$ and variance were set as a function of the minimum depth of the image $\sigma_n^2 = 1/(6d_{min})^2$; CNN-SVO replaced these simple values with more precise information of the depth estimation coming from the CNN for each filter position as $\mu_n = 1/d_{CNN}$, $\sigma_n^2 = 1/(6d_{CNN})^2$. Experimental results demonstrated that adding prior CNN depth information improved the system's performance for overexposed and underexposed images thanks to its illumination invariance properties, facilitating feature correspondence between views and overcoming key- illumination issues of original SVO. Figure 52 introduces the CNN-SVO algorithm inspired by the article [54].

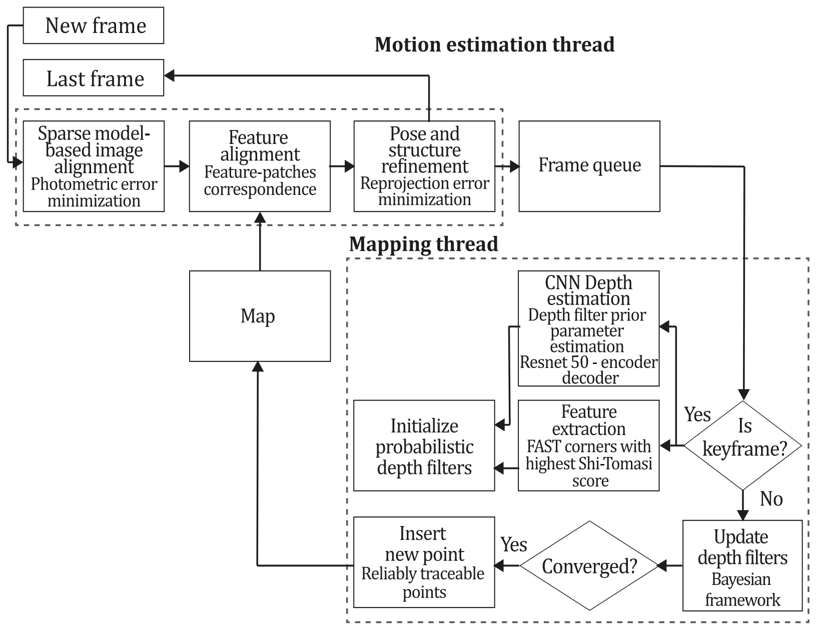

New frame
Last frame
Motion estimation thread
Sparse model-based image alignment
Photometric error minimization
Feature alignment
Feature-patches correspondence
Pose and structure refinement
Reprojection error minimization
Frame queue
Mapping thread
Map
CNN Depth estimation
Depth filter prior parameter estimation
Resnet 50 - encoder decoder
Initialize probabilistic depth filters
Feature extraction
FAST corners with highest Shi-Tomasi score
Yes
Is keyframe?
No
Insert new point
Reliably traceable points
Yes
Converged?
Update depth filters
Bayesian framework

Figure 52. Diagram of CNN-SVO algorithm. Adapted from [54].

7.4. General comments for ML approaches

As delineated in preceding discussions, machine learning paradigms, particularly convolutional neural networks (CNNs), have been extensively incorporated into various classical frameworks to address identified limitations and failure modes within each category of taxonomic classification. It is noteworthy, as highlighted in scholarly contributions [135], [156], that a segment of the research community maintains that traditional geometric strategies continue surpassing machine learning techniques regarding reconstruction fidelity and tracking precision. Nonetheless, as scrutinized in this review, proponents of machine learning have endeavoured to rectify these shortcomings by developing and applying novel or more advanced CNN architectures, alongside the construction of robust datasets aimed at refining the training process. Furthermore, as evidenced in the systems evaluated herein, CNNs extend beyond the scope of scene depth recovery or camera pose estimation. They have been effectively deployed for tasks such as densifying depth maps, estimating initialization parameters, conducting preprocessing operations like feature extraction or optical flow estimation, and executing ancillary functions such as semantic segmentation.

From various analytical angles, CNNs have been shown to significantly contribute to SLAM, VO, and SFM domains. Another frequently discussed challenge in machine learning endeavours is the propensity for overfitting [157], which has been mitigated by the continuous enhancement of datasets [68], [123]–[127], [158], thereby enabling more robust training and the integration of sophisticated semi-supervised [147] and self-supervised methods [48], [104], [121], [151].

Issues concerning the generalization of machine learning methods [50] have been addressed by utilizing datasets captured under diverse conditions, including indoor [68], outdoor [103], autonomous driving [69], and Micro Aerial Vehicle (MAV) [67] flight sequences. Consequently, machine learning techniques have explored many avenues to resolve the challenges of 3D reconstruction, making significant strides in this field of research. Moreover, the selection criteria for SLAM, VO, or SFM systems employing machine learning mirror those of classical approaches, with additional considerations specific to machine learning, such as the integrated CNN architectures and the principal estimation tasks that necessitated using CNNs for each system. Table 3 in the article encapsulates the 11 criteria assessed across various machine learning systems.

In practice, several machine learning methods with open-source code availability have been implemented, selected based on their accessibility and their reliance on monocular RGB input as the sole data source. It is pertinent to mention that methods such as those by [3], [6], [8], [9], [21], [28], [49], [112], [114], [135], [140], [147] have been made publicly accessible. However, their published open-source versions did not encompass their monocular RGB pipelines and were contingent on supplementary information such as externally estimated optical flow or depth priors, and hence were not featured in the examples depicted in Figure 53. Figure 53 illustrates the outcomes of implementing ML-based monocular SLAM, VO, and SFM systems on publicly accessible datasets.

Table 3. Summary of the most representative ml monocular SLAM, VO, and SFM systems

Method	SLAM, VO or SFM	Tracking method	Map density	Pixels used	Estimation	CNN architecture	CNN's main estimation tasks	Global optimization	Relocalization	Loop closure	Availability
DynaSLAM (2018) [118]	SLAM	Feature-based	Sparse	Hi.grad.	Local BA	Mask R-CNN	Instance segmentation	✓	✓	✓	[159]
BA-Net (2019) [49]	SFM	Feature-based	Sparse	Hi.grad.	BA	DRN-54	Depth Damping factor	-	-	-	[160]
Steenbeek et al. (2022) [117]	SLAM	Feature-based	Sparse	Hi.grad.	BA	ResNet-50 Enc.dec.	Scale Depth map densify	✓	✓	✓	[161]
Sun et al. (2022) [50]	SLAM	Feature-based	Sparse	Hi.grad.	BA	ResNetXt-50 Enc.dec.	Scale Relative depth Depth	✓	✓	✓	-
Lee et al. (2022) [128]	SLAM	Feature-based	Sparse	Hi.grad.	BA	Enc. dec.	Scale Semantic segmentation Feature refinement	✓	✓	✓	-
SVR-Net (2023) [130]	SLAM	Feature-based	Sparse	Learned features	Optimal match recurrent network	ScanNet	Local map Relative pose TSDF values	✓	-	-	-

DeMoN (2017) [114]	SFM	Optical flow	Dense	SIFT key-points matching Flow-Fields	8-point algorithm RANSAC	Chain Enc.dec.	Optical flow Depth Pose Surface normals	-	-	-	[162]
DeepV2D (2020) [113]	SLAM	Optical flow Feature-based	Dense	Learned features	3D Stereo matching over cost volumes	Residual Flow Hourglass Enc.dec.	Depth Pose 3D stereo matching	✓	-	-	[163]
VOLDOR (2020) [135]	VO	Optical flow residuals	Dense	Learned features	Generalized Expectation-Maximization	PWC-Net	Optical flow	-	-	-	[156]
DROID-SLAM (2021) [136]	SLAM	Optical flow	Dense	Learned features Between keyframes edges	BA	Residual blocks	Feature extraction Optical flow Estate estimation	✓	-	✓	[164]
SDF-SLAM (2022) [116]	SLAM	Feature-based	Dense	Learned features and descriptors	BA	Enc.dec.	Feature and descriptor extraction Semantic segmentation	✓	✓	✓	-
NeRF-SLAM (2022) [138]	SLAM	Optical-flow	Dense	Learned features Between keyframes edges	BA Radiance field optimization	Residual blocks Neural Radiance Fields	Feature extraction Optical flow Estate estimation	✓	-	✓	[165]
Rosinol et al. (2023) [139]	SLAM	Optical-flow	Dense	Learned features Between keyframes edges	BA Probabilistic volumetric fusion	Residual blocks	Feature extraction Optical flow Estate estimation	✓	-	✓	-
CNN-SLAM (2017) [3]	SLAM	Direct	Semi-dense	Hi.grad.	Pose Graph optimization	ResNet-50 FCN	Depth Semantic segmentation	✓	-	✓	[166]
DeepTAM (2018) [140]	SLAM	Direct Optical flow	Dense	Hi. grad.	Cost volume refinement	Enc.dec.	Pose hypotheses Optical flow Depth Depth refinement	✓	✓	-	[167]
DeepFusion (2019) [9]	SLAM	Direct	Semi-dense	Hi. grad.	Opt framework	U-Net	Log-depth gradients and uncertainties Scale	✓	-	-	-
CodeSLAM (2018) [8]	SLAM	Direct	Dense	Hi. grad.	BA	U-Net Variational Enc.dec.	Code Compact depth	✓	-	-	[168]
DeepFactors (2020) [6]	SLAM	Direct	Dense	Hi. grad.	Multiview BA	U-Net Variational Enc.dec.	Code Compact depth Uncertainty	✓	✓	✓	[169]
DVSO (2018) [111]	VO	Direct	Sparse	Hi. grad.	BA	ResNet-50	Disparity maps	-	-	-	[170]

CNN-DVO (2020) [21]	SLAM	Direct	Sparse	Hi. grad. Dynamic upsampling and downsampling	BA	Encoder-decoder U-Net Encoder-decoder	Depth	✓	-	✓	[171]
D3VO (2020) [112]	VO	Direct	Sparse	Hi.grad.	BA	U-Net Encoder-decoder	Depth, Pose, Uncertainty	-	-	-	-
MonoRec (2021) [147]	SFM	Direct	Sparse	Hi. grad. Mask filter	BA	U-Net ResNet-18 features Encoder-decoder	Depth Mask Moving objects	-	-	-	[172]
DDSO (2022) [28]	VO	Direct	Sparse	Hi. grad.	BA	ResNet-50 Encoder-decoder	Depth Pose Transformations	-	-	-	-
CNN-SVO (2019) [54]	VO	Hybrid	Sparse	FAST + Hi.grad.	Local BA	ResNet-50 Encoder-Decoder	Depth	-	-	-	[173]

Hi.grad. is used to abbreviate a set of pixels with a high-intensity gradient.
Enc.dec. is used to abbreviate the Encoder-decoder CNN architecture.
EKF is used to abbreviate the Extended Kalman Filter technique.
BA is used to abbreviate the Bundle Adjustment technique.
1Unofficial implementation of the CNN-SLAM method. There is not an official implementation of this method yet.
2Unofficial implementation of the Code-SLAM method. There is not an official implementation of this method yet.
3Unofficial implementation of the DVSO method. There is not an official implementation of this method yet.

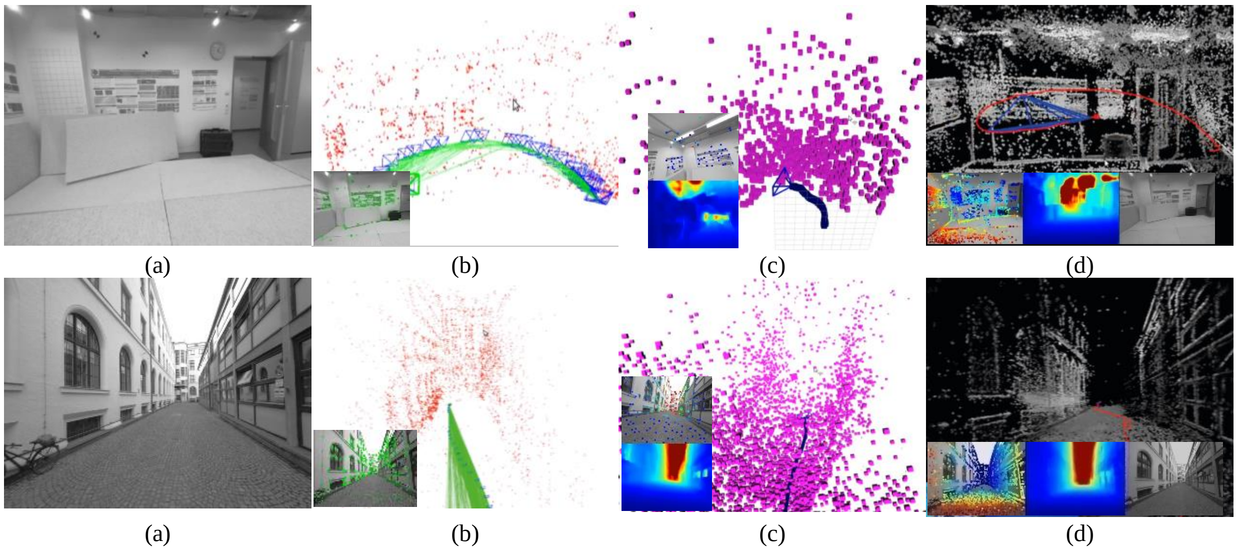


Figure 53. Examples of results obtained by ML approach implementations. (a) represents the input image, (b) presents results obtained using the DynaSLAM algorithm (Bescos, 2019), (c) presents results obtained using the CNN-SVO algorithm [54], and (d) presents results obtained using the CNN-DSO algorithm [174]. Top row results correspond to the indoor example sequence seq_01, and bottom row results correspond to the outdoor sequence seq_29 of the TUM-MONO dataset [103].

8. Discussion

The domains of SLAM (Simultaneous Localization and Mapping), VO (Visual Odometry), and SFM (Structure from Motion) have been subjects of intensive research over the past thirty years, with significant advancements in the last two owing to computational power enhancements that have facilitated their core operations. Yet, it is imperative to acknowledge the existence of unresolved challenges that have spurred the development of the systems previously delineated. These challenges encompass the fortification of existing systems, the optimization of computational resources, the integration of scene understanding, the tracking and reconstruction of moving objects, the enhancement of map density, the improvement of generalization in novel environments, and the attainment of per-pixel dense scaled reconstruction, among others.

Efforts to reinforce current methodologies have been thoroughly investigated by researchers such as [7], [17], [23], [71], [84], who have focused on refining original systems through both classical techniques and the integration of neural networks into traditional frameworks [3], [49], [54], [112], [116], [117], [140]. The optimization of computational resources has been particularly pertinent for embedded systems with limited processing capabilities, such as UAVs or drones, necessitating specialized strategies for feature, point, and frame selection to operate with minimal data [13], [117].

The incorporation of scene understanding has been approached in classical methods [7], [71] by adding modules for scene comprehension or through machine learning to classify point clouds in scenes recognizable by the trained network [3], [116]. The detection and depth estimation of moving objects has been explored using artificial intelligence to generate masks that assess the likelihood of each pixel belonging to a moving object, thus aiding in object detection, exclusion, and prediction [147], [175].

Enhancing map density has been tackled using classical dense techniques reliant on optical flow [74], [80] or pixel intensities [4], [5], [17], as well as neural networks designed for this specific end [28], [117]. To boost generalization capabilities, researchers like [28] have advocated for using diverse datasets during training, while others [147] have implemented multistage training approaches, and some have applied machine learning regularization and data augmentation techniques.

Furthermore, the pursuit of scale recovery from monocular imagery has been advanced through the use of neural networks tailored for this purpose [9], [50], [117] or by incorporating scale estimation into CNN inference models. Despite the plethora of solutions proposed to address the various failure modes and open issues in monocular 3D reconstruction, there remains a vast array of potential strategies and combinations yet to be explored in future research, along with many more beyond the purview of this overview, which pertains to forthcoming technological and theoretical developments. To provide readers with an understanding of the research trajectory, this paper collates citation metrics from the Scopus database and aligns them with the taxonomy introduced herein, offering insights into the evolution of these research fields.

8.1. Classic vs. Machine Learning

Starting with the foundational categorization of classical and machine learning (ML) techniques, Figure 54 delineates the trajectory of citation scores for each group spanning from 2005 to 2022. It is pertinent to note that the citation data from 2000 to 2005, marking the inception of the earliest system under review [14], have been omitted due to their negligible citation frequency. Additionally, the analysis excludes data for years not yet completed, hence the metrics for 2023 were not incorporated into this analysis at the time of this article's submission.

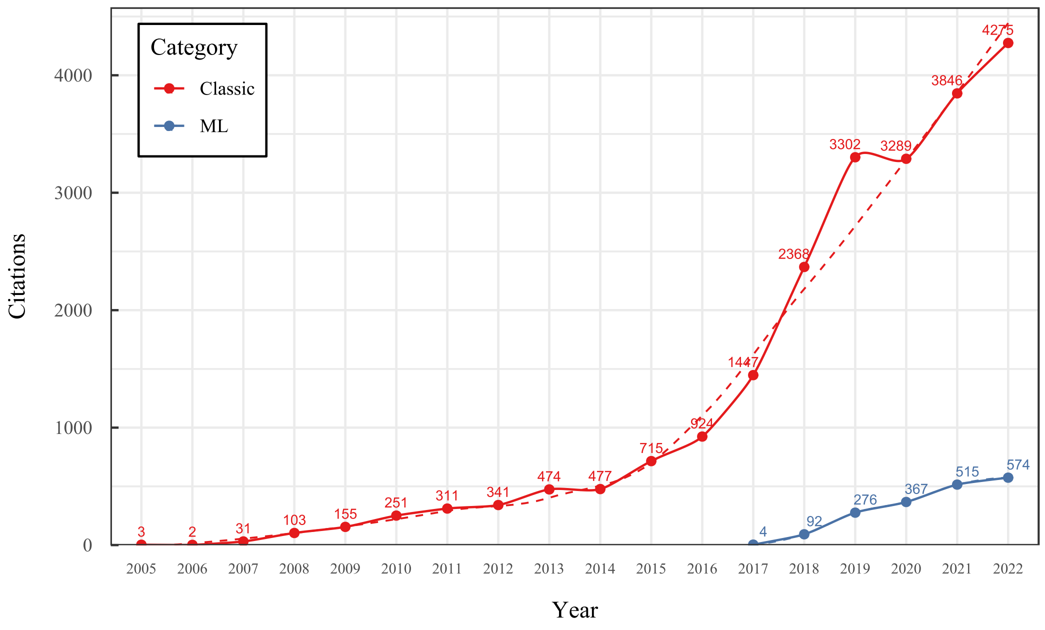


Figure 54. Citation score progression for classical and ML methodologies since 2005. The dashed lines indicate the trend lines for each category, established with a 99% confidence interval using the Loess smoothing technique.

The data presented in Figure 54 reveal that over the preceding 18 years, classical methods have consistently garnered higher citation scores. This can be largely attributed to the longer period these methods have been accessible, often as open-source implementations. It is observable that ML techniques began to captivate the interest of the academic community around 2017, with their citation scores demonstrating a steady ascent after that. The projection is that this upward trend will persist, leading to the anticipation that citation scores for ML methods will soon approximate, if not parallel, the remarkable figures attained by classical approaches.

8.2. Direct vs. Indirect

Transitioning to the indirect vs. direct classification, it is instructive to highlight prevalent challenges within both domains. Indirect techniques typically depend on geometric Bundle Adjustment (BA) predicated on reprojection error. Nonetheless, as indicated by [2] and [49], this approach is not without its limitations, such as its reliance on specific feature types (e.g., corners, blobs, or line segments) and the propensity for feature matching across frames to introduce outliers. Direct methods, as discussed by [2], [33], [49], exhibit sensitivity to initialization due to photometric variations that exacerbate non-convexity. These methods are also more vulnerable to camera exposure adjustments and white balance shifts, leading to a heightened susceptibility to outliers, including motion blur or moving objects.

In response to these challenges, innovative formulations have been developed that amalgamate the advantages of both indirect and direct approaches. For instance, certain indirect methods, such as those by [49], have integrated direct pixel information to enrich the depth maps while maintaining an indirect backend. Conversely, direct methods exemplified by [84] have incorporated feature extraction techniques to bolster tracking performance. These advancements have been formally encapsulated by semi-direct hybrid approaches [49], [84], which distinguish themselves by their capacity to fully merge both modalities within their optimization backend. The categorizations were analyzed using data from the Scopus database, facilitating a citation progression study over time. Figure 55 depicts the citation trajectory for direct versus indirect methods from 2005 to 2022.

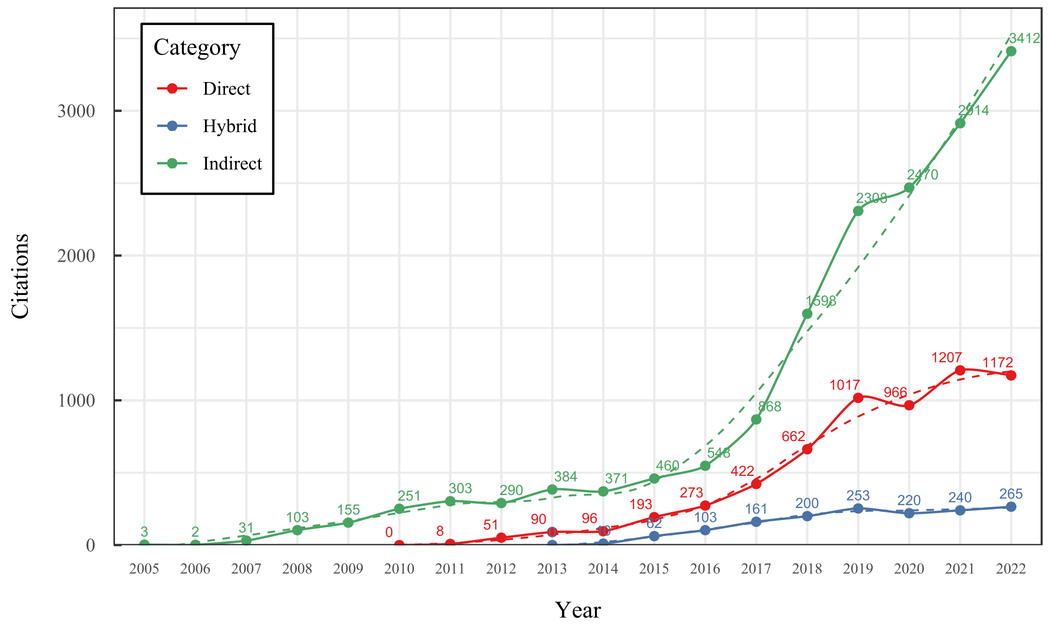


Figure 55. Citation score trends for direct, indirect, and hybrid methods since 2005. The dashed lines denote the trend lines for each category, established with a 99% confidence interval and employing the Loess smoothing method.

As illustrated in Figure 55, indirect methods have garnered the most citations among the three categories, with direct methods following closely and hybrid approaches in third place. This pattern can be ascribed to the extensive period during which indirect methods have been under scrutiny, being the pioneering proposals in this field of research. In contrast, direct methods emerged in 2010, with hybrid techniques entering the discourse in 2014. Notably, within this study's scope, 16 indirect, 17 direct, and only two hybrid methods were examined. Despite the limited number of contributions in the hybrid category, it is remarkable that they have achieved a significant citation count, underscoring the impact of hybrid approaches in the field.

8.3. Dense vs. Sparse

Advancing to the dense versus sparse classification within the third facet of the taxonomy under discussion, this classification is predicated on the density of points that constitute the ultimate 3D model. As delineated in scholarly contributions [5], [6], [13], [17], [22], [54], [74], [77], [85], [117], [176], the choice between dense and sparse approaches is intrinsically linked to the intended use-case of the 3D reconstruction system. A dense methodology is preferred for applications where a high-density map is essential, such as for navigation purposes, exploration tasks, or augmented reality. Conversely, for scenarios where a high-definition 3D model is not critical, rapid movement is necessary, or the application must function on devices with limited computational power [13], [50], [117], a sparse framework is deemed more appropriate.

Following the analytical procedure applied to the other categories, the database was queried using the dense versus sparse classification as a categorical variable. The outcomes of this analysis are encapsulated in Figure 56.

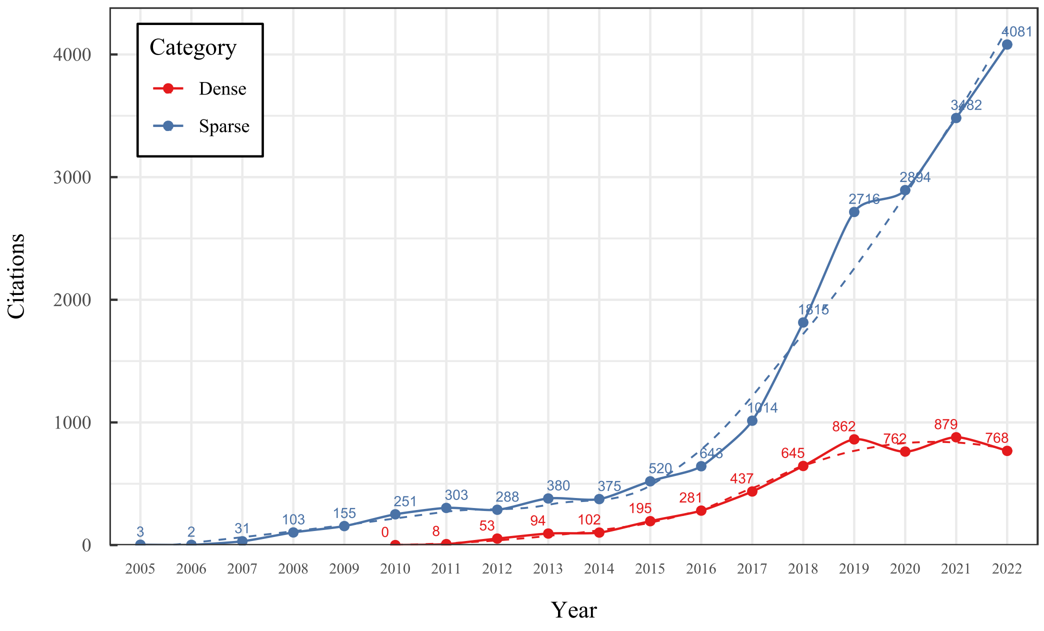


Figure 56. Evolution of citation scores for dense and sparse methodologies since 2005. The dashed lines indicate the trend lines for each category, calculated with a 99% confidence interval and utilizing the Loess smoothing method.

8.4. Complete taxonomy

To ensure a thorough and holistic examination, we analyzed the dataset, employing the full spectrum of our taxonomy as a categorical variable. This taxonomy, delineated in Section 3.3, comprises ten distinct levels, each representing a unique combination of the three primary classifications: Classic + Dense + Direct, Classic + Sparse + Direct, Classic + Dense + Indirect, Classic + Sparse + Indirect, Classic + Hybrid, ML + Dense + Direct, ML + Sparse + Direct, ML + Dense + Indirect, ML + Classic + Sparse + Indirect, and ML + Hybrid. Figure 57 delineates the trajectory of citation scores across these categories over time.

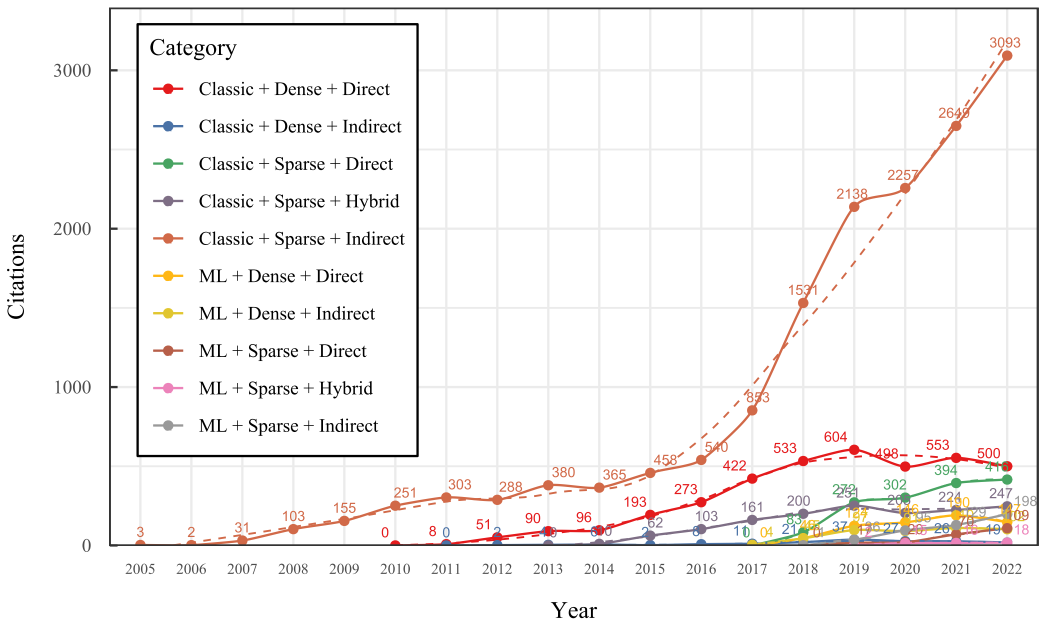


Figure 57. Citation score evolution for taxonomy proposed categories over time. Dashed lines represent the trend line for each classification obtained, setting a 99% confidence interval and the Loess method.

In Figure 57, the trend lines, demarcated by dashed lines and calculated with a 99% confidence interval using the Loess smoothing technique, reveal that the 'Classic Sparse Indirect' category has consistently garnered the highest citation scores. This enduring interest from the academic community can be largely ascribed to the early introduction and subsequent influence of seminal methods such as MonoSlam, PTAM, and ORB-SLAM, whose open-source availability has facilitated widespread adoption and comparative analysis in subsequent research. Following closely, the 'Classic Dense Direct' category has also achieved notable citation prominence, likely due to its proficiency in addressing the critical challenge faced by sparse indirect systems: generating detailed depth maps suitable for a broad array of applications. The 'Classic Sparse Direct' category, a more recent entrant from 2018, has quickly risen to prominence, a testament to the compelling advancements made by systems evolving from DSO, which have adeptly mitigated noise issues inherent in direct pixel information extraction while maintaining a manageable computational footprint. The citation trends observed across the various taxonomic categories indicate the significant contributions made by these pioneering works, which have propelled the field forward. Figure 58 catalogues the systems under review, arrayed by their year of release and their respective citation scores, with the legend on the right indicates the taxonomy classification from the most to the least cited.

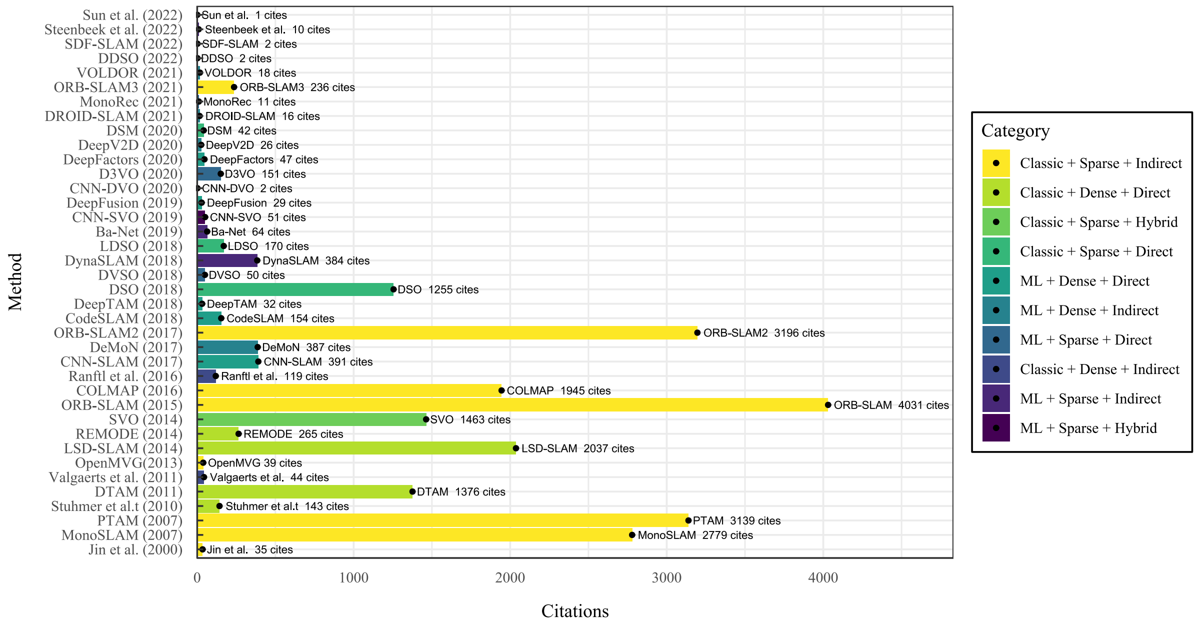


Figure 58. Reviewed methods citation score. The right legend depicts a color for each taxonomy classification ordered from the highest cited category to the lowest. The methods are ordered on the vertical axis according to their year of appearance.

9. Intermediate conclusions

The challenge of 3D scene reconstruction is a complex, underdetermined issue that can be approached through various methods and technology. In the current study, we have provided a comprehensive review of a particularly compelling method for 3D reconstruction: the visual reconstruction of environments utilizing a singular monocular RGB camera as the sole data source. This discussion has spanned various input modalities and their respective advantages and limitations, focusing on three principal techniques: SLAM, VO, and SFM. We have established a taxonomy that encapsulates the most prevalent system configurations reported in scholarly literature, delineating three primary classifications that yield ten potential combinations within an expanded taxonomy framework. Furthermore, this work has included an exhaustive review of 42 quintessential monocular systems. To assist in selecting and deploying these systems, we have compiled nine decisive criteria for each classical system, which serve as pivotal factors in implementing a 3D reconstruction system. These criteria encompass the type of algorithm, tracking method, map density, pixels utilized, estimation method, global optimization, relocalization, loop closure, and system availability. For ML-based methods, we have identified eleven criteria, mirroring those of classical approaches but incorporating two additional considerations pertinent to the network: the architecture of the CNN and the primary estimation tasks of the CNN.

The extensive information collated herein is intended to aid researchers in judiciously choosing an algorithm or taxonomy category that aligns optimally with their project objectives. Moreover, we have delineated each classification's principal strengths and weaknesses within the taxonomy. An analysis of the evolution of each classification and category within the taxonomy over the past 18 years has been conducted, based on citation metrics, to infer the impact and recognition each has garnered within the field of research. Looking ahead, our future endeavours will leverage the proposed taxonomy to conduct comparative analyses of the most significant open-source algorithms within each category, to discern their respective merits and drawbacks, and to select the most apt methodology for the monocular 3D reconstruction of indoor environments.

Chapter III – Comparative Analysis and Selection

Adapted from the novel published article: A Comparison of Monocular Visual SLAM and Visual Odometry Methods Applied to 3D Reconstruction.

Authors: Erick P. Herrera-Granda, Juan C. Torres-Cantero, Andrés Rosales, and Diego H. Peluffo-Ordóñez. Appl. Sci. 2023, 13(15), 8837.

Status: Published

Impact Factor (JCR 2022): 2.7

Quartile: Q2

DOI: http://dx.doi.org/10.3390/app13158837

1. Introduction

Monocular 3D reconstruction is a complex problem that can be solved from multiple perspectives (commonly requiring combining geometric, probabilistic, and even machine learning techniques), due to the large amount of information to be processed and the scale ambiguity problems that pure monocular sensors imply [6], [177]. This problem has been studied in the past three decades to obtain 3D representations of an environment using a sequence of images as the unique source of information for an algorithm. Previously, multiple researchers have explored the possibility of addressing this problem by using diverse hardware like radars, lasers, GPS, INS, cameras, and any possible combination thereof. Regarding the camera alternative, it can be combined with active or passive infrared sensors as RGB-D input modalities. It can also be structured as an array of cameras registering the same objects from multiple angles to allow triangulation. Monocular RGB sensors can also be used alone to register a frame sequence from which the algorithm can process a scene from multiple views [1], [18]. This last option is known as monocular RBG or monocular pure visual input modality, used in monocular Simultaneous Landing and Mapping (SLAM), Visual Odometry (VO), or Structure from Motion (SFM) to obtain 3D reconstructions of environments and estimate the ego-motion of an agent from such representations. In recent years, the pure monocular input modality has attracted the research community's attention due to the sensors' low price and availability in most handheld devices—smartphones, tablets, and laptops. Thus, monocular SLAM, VO, and SFM systems are not limited as other sensors are (like lasers or radars) to work in a limited range and have demonstrated the ability to recover precise trajectories and 3D reconstructions indoors and outdoors.

Simultaneous Localization and Mapping is the process where a robot constructs a map of its surroundings while concurrently figuring out where it is located within that map. It involves determining the positions of landmarks and objects near the robot and its position, commonly utilizing sensors and geometric and Bayesian techniques. Visual Odometry is the process of incrementally estimating the robot's ego-motion (location and orientation) by analyzing the changes between the sequential camera images from the robot, estimating the robot's local trajectory rather than obtaining a comprehensive map. VO is commonly utilized as a front-end in many visual SLAM systems. Structure from Motion refers to a reduction in the 3D structure from 2D image sequences that show a scene from different perspectives. It recovers the 3D location of points matched across multiple images and the camera pose for each image. SFM does not require knowing the camera's motion in advance and is utilized in SLAM for initializing new 3D points [178].

As mentioned before, SLAM, VO, and SFM are three disciplines that can be used to achieve the 3D reconstruction goal. SLAM is a discipline that appeared in the robotics field motivated by the objective of estimating the environment map from where the trajectory of a robot can be calculated, which can be used for autonomous navigation, driving, and flying, among other things. In the computer vision field, multiple systems have been created to address similar problems: SFM and VO. Structure from Motion specializes in recovering an environment geometry, while Visual Odometry focuses on calculating the trajectory and pose of a moving camera. However, it has been demonstrated that instead of solving each problem separately, the best results have always been obtained by solving and optimizing both problems simultaneously [18], [38], [103], [158]. That is why it is common to find VO methods that include SFM modules to improve their performance and SFM methods that use VO to improve estimation or optimization tasks. For such reasons, in this study, we aim to identify the best monocular RGB methods for 3D reconstruction; so, we included methods from these three disciplines suitable for recovering 3D environment reconstructions.

As a complex problem, pure visual monocular 3D reconstruction has been addressed from multiple perspectives combining various techniques that can be classified following different approaches. One early classification is described in the study of [18] defining the feature-based and appearance-based categories; nevertheless, this approach is unsuitable for covering all the SLAM, VO, and SFM techniques available nowadays in the state of the art. A better approach to classify monocular RGB 3D reconstruction systems is the taxonomy described in [179], considering three classifications covering dense, sparse, direct, indirect, classic, and machine learning-based proposals. Moreover, the authors listed 42 methods classified following the proposed extended taxonomy. After a careful reading and analysis of the 42 listed methods, we could identify that many of the existing methods were not adequately evaluated on large datasets [10–13] or not tested under different motion patterns and illumination changes [54], [147], [174] and not tested for indoors/outdoors [16–18]; or the results were not obtained on the same metrics [49], [158], [166] hindering comparison and selection. In addition, most of the methods performed comparisons against the currently available methods from the state of the art, providing results in tables summarizing the average mean or median of the algorithm execution on a specific scene, but they did not provide an inferential statistical analysis of the results; thus, the reported differences or improvements cannot be considered significant. Moreover, given the fact that before the extended taxonomy described in [179], there were only general classifications like direct vs. indirect and sparse vs. dense methods [2], [38], [158] or the feature-based and appearance-based classification reported in [18], none of the studies compared their results following a taxonomy that might allow identifying better the advantages and limitations of direct, indirect, dense, and sparse methods.

To address the mentioned issues, in this study, we performed a comparison of ten publicly available SLAM and VO methods following a taxonomy, where the main contributions are:

- A comparison of 10 SLAM and VO methods, following the main classification described in the taxonomy (sparse-indirect, dense-indirect, dense-direct, and sparse-direct), to identify the advantages and limitations of each method of those classifications.
- A comparison of three machine learning-based methods against their classic geometric versions to identify whether there are significant improvements in adding neuronal networks to classic approaches.
- An inferential statistical analysis describing the procedure to identify significant differences based on the most suitable metrics for testing monocular RGB methods.

We also provide video samples of each algorithm's execution as supplementary material in the GitHub repository: "https://github.com/erickherreraresearch/MonocularPureVisualSLAMComparison accessed on 16 June 2023", along with all the .txt result files of each algorithm run for reproducibility.

1.1. Related Works

Following the classification described in [2], there are four main classifications for the methods that can be used to recover a scene's 3D geometry using monocular image sequences as the unique source of information: sparse-indirect, dense-indirect, dense-direct, and sparse-direct.

1.1.1. Sparse-Indirect Methods

Sparse-indirect methods implement preprocessing steps recovering sparse reconstructions. MonoSLAM, PTAM, ORB-SLAM, and OpenMVG are the most prominent works in this classification. MonoSLAM [31] was one of the first real-time monocular SLAM systems. Its key contributions included using large image patches as features, "active" feature matching based on uncertainty, and initializing by tracking known targets. However, MonoSLAM was limited to small workspaces and lacked loop-closing abilities. PTAM [32] introduced the concept of parallel tracking and mapping threads, with the map optimized via bundle adjustment over carefully selected keyframes. This configuration achieved excellent AR tracking in small spaces, but the PTAM lacked loop closing, and the relocalization was view-dependent. ORB-SLAM [33] significantly expanded PTAM's capabilities using ORB features for tracking, mapping, and loop closing via DBoW2 place recognition. The covisibility graphs enabled local mapping, while the pose graphs distributed loop closures globally. ORB-SLAM also introduced flexible keyframe insertion/deletion policies to improve mapping during exploration while reducing redundancy. This versatility enabled state-of-the-art performance across indoor, outdoor, handheld, and robotics datasets. OpenMVG is a C++ library that provides an interface to multiple view geometry algorithms for building complete 3D reconstruction pipelines from images implementing incremental and global SfM approaches. The OpenMVG SfM pipeline stores camera poses, landmarks, and observations, providing smooth data flow between OpenMVG modules. Overall, the OpenMVG enables flexible experimentation and the development of new techniques used for multiple implementations since 2016; however, it only allows recovering widely sparse reconstructions, which are unsuitable for many applications.

1.1.2. Dense-Indirect Methods

Dense-indirect techniques incorporate preprocessing stages and recover dense depth maps. Some important prior works that defined this category were Valgaerts et al. and Ranftl et al. Valgaerts et al. [74] proposed a novel two-step method for estimating the fundamental matrix from a dense optical flow. Their key contribution was demonstrating that accurate epipolar geometry robust estimation was possible using dense correspondence fields computed by variational optical flow methods. They introduced a joint variational model that recovered the optical flow and epipolar geometry within a single energy functional, thus improving the results. However, their method was limited by its sensitivity to large displacements and occlusions. Ranftl et al. [77] presented an approach to estimate dense depth maps for complex dynamic scenes from monocular video, built on the use of dense optical flow. The key concept is a motion segmentation stage that decomposes the scene into independent rigid motions, each with its epipolar geometry enabling moving objects' reconstruction. Its method was optimized to work with object scales and geometry to assemble a globally consistent 3D model determined up to scale. A key difference from Valgaerts et al. was the explicit handling of multiple independently moving objects and the recovery of dense depth for fully dynamic scenes. However, Ranftl et al.'s approach still relied on approximate scene rigidity and the connectivity of objects to the environment. Valgaerts et al. introduced a dense optical flow for fundamental matrix estimation, while Ranftl et al. extended dense the geometric reconstruction to complex dynamic scenes. Both moved from sparse features to dense correspondence fields; in contrast, Ranftl et al. focused on depth estimation and scene assembly.

1.1.3. Dense-Direct Methods

Dense-direct techniques work directly with pixel information and can recover dense depth maps. Some of the main contributions in this field are the Stühmer et al., DTAM, REMODE, and LSD-SLAM systems. Stühmer et al. [80] proposed one of the first real-time dense monocular SLAM systems. They introduced a variational framework to estimate the dense depth maps from multiple images using robust penalizers for both the data term and the regularizer. The key contributions were integrating multiple images for noise robustness and an efficient primal-dual optimization scheme. However, their method was limited to local dense tracking and mapping without global map optimization. The DTAM system proposed by Newcombe et al. [26] enabled real-time dense tracking and global mapping using a single handheld camera. They introduced the concept of dense model-based camera tracking by aligning live images to the textured 3D surface models synthesized from the estimated dense depth maps. The depth maps were computed by filtering over the small-baseline stereo comparisons from video. A key difference from Stühmer et al. was maintaining a global map with pose graph optimization. The REMODE system of Pizzoli et al. [5] also performed per-pixel Bayesian depth estimation but introduced a convex optimization-based smoothing step using the estimated uncertainty to enforce the spatial regularity. They demonstrated probabilistic updating, allowing online refinement and error detection. A key contribution was the derivation of a measurement uncertainty model. However, REMODE was limited to local mapping without global optimization. The LSD-SLAM of Engel et al. [4], integrated many of these concepts into the first direct monocular SLAM system capable of performing consistent global semi-dense reconstruction. The key novelties were the direct alignment on the $Sim(3)$ handling scale drift and the incorporation of depth uncertainty into tracking. LSD-SLAM reached an outstanding outdoor performance by enabling large-scale accurate monocular dense reconstruction in real time. In summary, early works, like Stühmer et al. and DTAM, introduced key concepts like multiple image integration, probabilistic depth estimation, and variational optimization, while later methods, like LSD-SLAM, were built on these concepts to enable globally consistent mapping and reconstruction, with fully direct approaches finally demonstrating accurate monocular dense SLAM at scale.

1.1.4. Sparse-Direct Methods

Sparse-direct techniques work directly on pixel information but do not use all the pixels, producing sparser maps using fewer computational resources. The main contributions from this classification are the DSO, LDSO, and DSM. Direct Sparse Odometry (DSO) was introduced by Engel et al. [2] as the first direct-sparse VO technique. The DSO operates directly on image intensities, optimizing the photometric error instead of the geometric reprojection error. It represents the geometry using inverse depth parametrization and jointly optimizes all the model parameters in real time using a sliding keyframe window. The DSO demonstrated superior accuracy and robustness compared to indirect methods by utilizing edges and intensity variations in featureless areas. However, as a pure visual odometry technique, the DSO suffers from drift over long trajectories as it marginalizes old points and keyframes. Gao et al. presented the LDSO [23], extending the DSO to a more robust VO system by adding loop closure detection and pose graph optimization. The LDSO adapts the DSO's point selection to favor repeatable corner features and computes the ORB descriptors detecting the loop closures using DBoW2. It then estimates the $Sim(3)$ constraints by minimizing the 2D and 3D errors fusing them with the covisibility graph from DSO's sliding window optimization in a pose graph. While reducing the accumulated drift, the LDSO still lacks a persistent map ignoring the existing information after loop closures. Zubizarreta et al. introduced Direct Sparse Mapping (DSM) [84], the first direct sparse monocular SLAM system with a persistent map enabling point reobservations. The DSM selects active keyframes based on temporal and covisibility constraints using the Local Map Covisibility Window applying a coarse-to-fine optimization scheme and a robust cost function based on the t-distribution to handle challenges in converging when incorporating distant keyframes. The DSM demonstrated increased accuracy in trajectory and mapping on EuRoC compared to the DSO, LDSO, and ORB-SLAM. The ability to reuse existing map points resulted in more consistent maps without duplicates. In brief, the DSO pioneered direct-sparse SLAM and achieved superior odometry compared to the indirect methods. The LDSO extended it to full SLAM by adding loop closure detection and correction to reduce drift, while the DSM took a further step creating the first direct technique with a persistent map, enabling beneficial point reobservations through key innovations in window selection, optimization, and robustification.

1.1.5. Machine-Learning-Based Approaches

Recently, a new category emerged, adding machine learning modules to the SLAM, VO, and SFM pipelines. Some of the most prominent approaches are DynaSLAM, SVR-Net, VOLDOR, DROID-SLAM, SDF-SLAM, CNN-SLAM, CodeSLAM, DeepFactors, MonoRec, and CNN-SVO. CNN-SLAM [3] was one of the first systems to incorporate CNN-predicted depth maps into monocular SLAM, overcoming the scale ambiguity issues. It also performed joint semantic segmentation and 3D reconstruction, pioneering multitask learning. DynaSLAM [118] was one of the first attempts to detect and remove dynamic objects from the mapping process using a CNN for segmentation and a multiview geometry approach enabling more robust tracking and mapping in dynamic environments. CodeSLAM [8] incorporated an encoder–decoder CNN for scene geometry into a compact latent code conditioned on image intensities retaining only nonredundant information for joint geometry and motion optimization. The CNN-SVO [54] incorporated CNN depth predictions to initialize the depth filters in SVO, reducing uncertainty and improving mapping. DeepFactors [6] was built over the basis of CodeSLAM to formulate dense monocular SLAM as a factor graph optimization combining the learned depth priors, the reprojection error, and the photometric error for robust performance. VOLDOR [135] integrated a CNN into its visual odometry pipeline using log-logistic depth residuals and probabilistic inference, eliminating the need for feature extraction or RANSAC, enabling real-time performance. The DROID-SLAM [136] integrated a recurrent neural network to iteratively update camera poses and estimate depth maps through differentiable bundle adjustment. MonoRec [147] addressed the alternative to incorporate mask prediction and depth prediction modules to enable high-quality monocular reconstruction in dynamic scenes. SDF-SLAM [116] combined classic sparse feature extraction with a CNN for dense depth prediction and semantic segmentation enabling semantic 3D reconstruction while retaining real-time performance. SVR-Net [130] integrated a Support Vector Regression network to estimate 3D keypoint locations, enabling robust tracking in challenging conditions using online learning and graph optimization for map refinement. In summary, machine-learning-based methods progressively incorporated deep learning into sparse indirect SLAM systems to improve the robustness and handle the dynamics, achieving dense reconstruction enabling end-to-end learning. The key innovations included using CNNs for segmentation, depth prediction, semantic segmentation, compact scene encoding, and uncertainty modeling.

1.1.6. Comparisons

Regarding comparison studies, an early work that accurately compared monocular visual odometry systems was the study of [148], comparing the state-of-the-art methods of that time, DSO, ORB-SLAM, and SVO, on the TUM-Mono benchmark. The authors found that the DSO system, even being a visual odometry system, outperformed the SLAM method and the popular SVO. In that study, the authors also tested the photometric calibration, the motion bias, and the rolling shutter effect, with the available information provided in the TUM-Mono dataset, finding that the photometric calibration improved the performance of the direct methods considerably, and the motion bias effect was more prominent in the indirect method. In contrast, we compared ten methods following a taxonomy, where the three methods tested in [148] were addressed, exploring the same photometric calibration, motion bias, and rolling shutter effects by applying the TUM-Mono dataset. Then in 2020, Mingachev et al. published two comparisons [16], [180] testing first the DSO, LDSO, and ORB-SLAM2 and then the ROS-based methods, DSO, LDSO, DynaSLAM, and ORB-SLAM2, on the TUM-Mono and EuRoC benchmarks, where the authors verified the performance of the algorithms implementing an open-source code in their hardware to determine whether there were improvements in the LDSO and DynaSLAM—updates of the original DSO and ORB-SLAM2. The authors found that the updates achieved slight error reductions over their predecessors on both benchmarks, reported as medians of 10 executions of each algorithm in each sequence.

Comparing those studies, we tested ten methods following a taxonomy to test whether the newer versions improved their previous performance and to identify the advantages and disadvantages in the entire taxonomy. We also provided a complete inferential statistical analysis of each method's performance, not only their median values. In addition, we included machine-learning-based versions of the classic methods in our comparison. One of the most recent related works was the study of [29], which explored the state-of-the-art classification and tested visual and visual–inertial algorithms in the ERoC benchmark. In that work, the authors briefly overviewed the existing methods and reviewed the classic classification of direct, feature-based, and RGB-D methods, adding DSO, ORB-SLAM2, and Vins-Mono methods to their comparison. In contrast, this comparison is focused only on monocular RGB methods; so, we followed an appropriate taxonomy for monocular RGB SLAM and VO systems. In addition, we used the TUM-Mono benchmark and its metrics, which is a broader and more complete benchmark.

2. Materials and Methods

For this study, we used a taxonomy, algorithms, benchmarks, and metrics suitable for the monocular SLAM and VO problems discussed in the following sections.

2.1. Taxonomy

The prior work [6] described a taxonomy based on three classifications in the literature: direct vs. indirect, dense vs. sparse, and classic vs. machine learning.

- **Direct vs. indirect.** Indirect methods refer to those algorithms that implement preprocessing steps, like feature extraction or optical flow estimation, before their pose and map estimation processes; so, the amount of information that moves into the following steps is considerably reduced, requiring less computational power but also reducing the density of the final 3D reconstruction [2]. Indirect methods typically perform their optimization steps by minimizing the reprojection error due to the feature type of information that the preprocessing step outputs [29]. On the other hand, direct methods work directly on the pixel intensity information without requiring preprocessing steps, implying that the algorithm has more information for estimation tasks allowing one to obtain denser reconstructions of the scene, requiring more computational power [29]. In addition, direct methods typically perform their optimization steps based on the photometric error due to the direct pixel availability information.
- **Dense vs. sparse**. Dense vs. sparse classification refers to the amount of information recovered in the final map as a 3D reconstruction [2]. Denser reconstructions have more definition and continuity in the reconstructed objects and surfaces. In contrast, sparser reconstructions are typically represented as largely separated point clouds, where the edges and corners are commonly the only objects that can be recognized clearly [38].
- **Classic vs. machine learning.** Classic methods have been proposed in the last three decades, typically basing their formulation on geometric, optimization, and probabilistic techniques without machine learning. However, in recent years, due to the impressive advances in artificial intelligence, especially in Convolutional Neural Networks (CNN), many techniques have been applied to improve the SLAM or VO estimation tasks [54], [116], [118], [172]. The methods based on classic formulations enhanced with machine learning are called Machine-learning-based approaches (ML).

Combining these three classifications in all their possible configurations [179] establishes the taxonomy: Classic + Dense + Direct, Classic + Sparse + Direct, Classic + Dense + Indirect, Classic + Sparse + Indirect, Classic + Hybrid, ML + Dense + Direct, ML + Sparse + Direct, ML + Dense + Indirect, ML + Classic + Sparse + Indirect, and ML + Hybrid. It must be mentioned that the hybrid category was added for those methods that efficiently combine the direct and indirect principles to estimate ego-motion and scene geometry, like SVO [13] and CNN-SVO [54]. This taxonomy was completely detailed and depicted in Chapter 1.

2.2. Selected Algorithms

In this comparative analysis, we aimed to determine the taxonomy classifications, limitations, and advantages by exploring as many taxonomy categories as possible. For this purpose, we selected and implemented five methods of geometric-based classification. Furthermore, we included three machine-learning versions of the selected classic approaches to test the hypothesis of whether or not the addition of a CNN to classic approaches significantly improved the geometric-based methods' performance. In a previous work [179], many machine learning approaches were listed available as open-source code [156], [159], [170]–[173], [161]–[164], [166]–[169]. However, during their implementation, we found that many implementations were available for multiple input modalities like RBD-D or INS. However, the provided code was not available for monocular RGB as the unique input source of information, or they required external and not included modules for their implementation, e.g., [156], [168], [169], [172]; thus, we could not include those methods for this comparison. Finally, we added two additional sparse-direct implementations built over the DSO [2] system, given the impressive 3D reconstruction results that this method demonstrated during evaluations.

In this way, the algorithms selected to perform this comparative study were:

1. **ORB-SLAM2.** As a sparse-indirect representative, we selected ORB-SLAM2 [7], widely known as the gold standard of this category, as most of the currently available sparse-indirect methods are proposals inspired by this algorithm. The original ORB-SLAM [33] extracts ORB features as preprocessing of multiscale FAST corners with a 256-bit descriptor giving that algorithm information to perform a Bundle Adjustment for optimization and work in three threads for tracking, local mapping, and loop closure. In addition, the ORB-SLAM2 incorporates a fourth thread to perform full Bundle Adjustment after loop closure redefining the original method and obtaining the scene optimal geometric representation. The ORB-SLAM2 is publicly available as an open source code in [94]; it may be implemented in its C++ version or ROS version, with minimum additional requirements, Pangolin, OpenCV (tested for 2.4.3 version), Eigen 3 (tested for 3.1.0 version), DBoW2, and g2o, which are included in the repository.
2. **DF-ORB-SLAM**. Classic dense-indirect methods available in the literature, like [74], [77], are not available as open-source code for implementation; so, they could not be considered for this evaluation. Instead, a well-known classic dense-direct version of ORB-SLAM2 exists, called DF-ORB-SLAM [181], with its code publicly available on GitHub. The DF-ORB-SLAM algorithm was built based on the ORB-SLAM2 algorithm, allowing the addition of depth map retrieval capabilities and incorporating optical flow to track the detected points; thus, this algorithm uses a large amount of information obtained through the input using most of the pixel values for optical flow estimation. Once the optical flow is estimated, the ORB-SLAM2 performs feature extraction on the optical flow domain executing the rest of its pipeline. The DF-ORB-SLAM is publicly available in [181], implemented in Ubuntu 18.04 in its ROS version using its official *build_ros.sh* script.
3. **LSD-SLAM.** The LSD-SLAM [4] is one of the most popular methods of the dense-direct category, since it has been the basis and inspiration for a lot of the methods currently available [2], [3], [25]. The LSD-SLAM not only locally tracks the camera's movement but also allows the construction of dense maps through a semi-dense geometric representation tracking the depth values only in high-gradient areas. The method has direct image alignment mechanisms and estimation based on the semi-dense depth map filtering technique [22]. The global depth map is rendered as a pose graph comprising keyframes represented as vertices that present feature 3D similarity transformations as edges, adding environment scaling ability and allowing the accumulated drift to be detected and corrected. Furthermore, the LSD-SLAM uses an appearance-based loop detection algorithm called FAB-MAP [83], introducing prominent loop closure candidates that extract their features without reusing any additional visual odometry information. The LSD-SLAM is publicly available in [98] and was implemented in Ubuntu 18.04 in its ROS version.
4. **DSO**. The DSO [2] is widely known as the direct methods' gold standard due to the impressive reconstruction and odometry results that it has achieved, inspiring other implementations and new proposals. The DSO works directly on the pixel intensity information but applies a point selection strategy to reduce the amount of information to be processed efficiently, continuously optimizing the photometric error applied to the last N-frames while optimizing the complete likelihood for the parameters involved in the model, including poses, intrinsics, extrinsics, and inverse depths, executing a windowed sparse bundle adjustment. The DSO is publicly available for implementation in [99]; its code runs entirely in C++, using minor requirements like Suitesparse, Eigen3, and Pangolin.

5. **SVO.** We selected the most commonly known method, SVO [85], for the hybrid classification. The SVO efficiently combines the advantages of the direct and indirect approaches by using the feature correspondences obtained on the direct motion estimation for tracking and mapping. This procedure considerably reduces the number of required features and is only executed when a new keyframe is selected to insert new points in the map. First, camera motion is estimated by a sparse model-based image alignment algorithm, where sparse point features are used to estimate the feature correspondences. Next, this information is used to minimize the photometric error. Then the reprojected points, pose, and structure are refined by minimizing the reprojection error. The SVO is publicly available in [102] for testing and implementation running on C++ or ROS. Modern operating systems might find issues during implementation; so, Ubuntu 16.04 and ROS kinetic were used.
6. **LDSO.** As an additional sparse direct system, the LDSO [23] was selected as an update of the DSO algorithm that includes loop-closure capabilities. The LDSO enables the DSO framework to detect the loop closure by ensuring point repeatability using corner features to detect loop candidates. For this purpose, the depth estimates for point features allow the algorithm to compute the $Sim(3)$ constraints, to be combined with the pose-only bundle adjustment and point cloud alignment and fused with the relative pose DSO covisibility graph, sliding the window optimization stage. This way, the LDSO adds the loop closure to the DSO system, including a loop closure module based on a global pose graph optimization working over the last five to seven keyframes' sliding window. The LDSO was made publicly available in [100], and for this comparison, it was implemented in Ubuntu 18.04 along with OpenCV 2.4.3, Sophus, DBoW3, and g2o.
7. **DSM**. Another sparse-direct method we were interested in testing was the DSM [84], another update made to the DSO to create a complete SLAM system. The DSM aimed to include scene reobservation information to enhance the precision and reduce the drift and inconsistencies. In contrast to the LDSO, which considers a sparse set of reobservations, the DSM builds a persistent map allowing the algorithm to reuse existing information by a photometric formulation. The DSM uses local map covisibility window criteria to detect the active keyframes reobserving the same region, a coarse-to-fine strategy to process that point reobservation information and a robust nonlinear photometric bundle adjustment technique based on the photometric error for outlier detection. The DSM open-source code is publicly available in [101], which was implemented for comparisons on Ubuntu 18.04 with Eigen (v3.1.0), OpenCV (v2.4.3), and Ceres solver, which were provided in the official repository.
8. **DynaSLAM**. The Dyna-SLAM algorithm [118] is a lighter version of ORB-SLAM2 exceeded by adding the detection, segmentation, and inpainting of dynamic information on scenes' machine learning capabilities. In addition, the Mask R-CNN of [119] was integrated with the classic sparse-indirect method to detect and segment regions of each image that potentially belonged to movable objects. The authors also incorporated a multiview geometry approach calculating backprojections to define the key point parallax angles to detect additional information the CNN cannot recognize. The authors reported that this combination contributed to overcoming the ORB-SLAM2 initialization issues; so, it works in dynamic environments. The DynaSLAM is publicly available in [159], and it was implemented in Ubuntu 16.04 with ROS Kinetic, Cuda 9, Tensorflow 1.4.0, and Keras 2.0.8.
9. **CNN-DSO**. In the literature, DSO neuronal methods like D3VO [112], MonoRec [147], and DDSO [28] can be found. Nevertheless, they are not publicly available, or in the case of MonoRec, its monocular VO pipeline is not available for testing; so, the CNN-DSO was selected for this comparison, which is publicly available in [174]. This method includes a CNN depth prediction module enabling the DSO system to execute its estimation modules using additional depth prior information obtained by the network. The CNN used for this study was the MonoDepth system of [152], a single image depth estimation network that outputs a depth value for each pixel position by chains of feature maps processing. The network was built over the ResNet backbone using a variant of its encoder–decoder architecture. The

CNN-DSO requires building TensorFlow (v1.6.0) from source and MonoDepth from its official repository [182], and it was implemented in Ubuntu 18.04, with Eigen (v3.1.0) and OpenCV (v2.4.3).

10. **CNN-SVO.** In the study of [54], an extension of the hybrid method SVO was proposed by fusing the same Single Image Depth Estimation (SIDE) CNN MonoDepth module used in the CNN-DSO with the original geometric-based hybrid method. In this case, MonoDepth was included to add preliminary depth information to the SVO pipeline, minimizing the uncertainty in the feature correspondence identification steps; then, the system is initialized, obtaining high uncertainty maps. Then, the SIDE CNN creates filters to approximate the current values' mean and variance, considerably reducing the amount of information separating inliers/outliers in the depth map. The CNN-SVO is publicly available in [173] and was implemented in Ubuntu 16.04 to allow the SVO modules to work with ROS Kinetic.

2.3. Benchmarks

Today, the scientific community has considerably promoted the development of datasets, including existing open-source datasets even for evaluating complex hardware setups like visual–inertial systems (i.e., YTU [183], WHUVID [184], and VOID [185]). In this way, there are several datasets and benchmarks available in the literature for evaluating RGB SLAM, SFM, and VO systems, like [67], [69], [186], [79], [103], [123]–[127], [158]. Nevertheless, only a few are suitable for pure monocular RGB systems due to the nature of image acquisition, the type of camera calibration or camera models used, and the format of the provided ground truth. Similarly, it is safe to say that among the reviewed available datasets, the following can be applied for monocular algorithms comparison:

- The **KITTI** dataset in [69] contains 21 video sequences of a driving car, where the movement parameters are limited to forward driving. The available images have pre-rectification treatments, and the dataset provides a ground truth obtained through an assembly of GPS and INS.
- The **EUROC-MAV** dataset in [67] contains 11 inertial stereoscopic sequences of a quadcopter flying in different indoor environments providing groundtruth values of all frames and calibration parameters.
- The **TUM-Mono** dataset in [103] presents 50 sequences of indoor/outdoor environments obtained using monocular RBG cameras on monochrome uEye UI-3241LE-M-GL cameras equipped with Lensagon BM2420 (with 148° × 122° field of view) and Lensagon BM4018S118 (with 98° × 79° field of view) sensors. This benchmark includes the photometric calibration parameters, the ground truth, the timestamps for the execution of each image sequence, and the calibration file for the vignetting effect in each sequence, comprising more than 190,000 frames and more than 100 min of video.
- The **ICL-NUIM** benchmark in [158] has eight sequences in conjunction with its ray-tracing of two environments, providing the groundtruth values of each sequence and camera intrinsics; so, no photometric calibration is required. This dataset presents degenerative and purely rotational motion sequences, which are considered demanding for monocular algorithms.

As can be noticed, the most complete and largest dataset of the above is the TUM-Mono, which is why this dataset was applied in this comparison study. It also has the advantage of being the only dataset that was obtained purely depending on a monocular RGB setup, without depending on any additional sensor or source of information as mentioned in [16], [103], [180], making it ideal for comparing visual-only SLAM and VO systems. In addition, this benchmark provides the most complete set of metrics that can be explored to efficiently compare the selected algorithms in multiple dimensions—discussed in the following section.

2.4. Metrics

As SLAM, SFM, and VO are ill-posed problems that can be addressed from multiple perspectives and a wide variety of techniques, comparing the final obtained 3D reconstruction is not the best alternative for monocular RGB methods because of the different sparsity, scale, and type of output that each method brings, due to the difficulty of accruing accurate groundtruth maps [68]. At the same time, trajectories can be acquired using INS, GPS, LASER, RADAR, LIDAR, and Kinect systems, among others, with acceptable accuracy. In this way, as discussed in [103], the best way of comparing SLAM and VO algorithms of diverse nature (see Figures 1 and 2) is by comparing the output trajectory in each algorithm, because even if the method is focused on reconstruction only, it has been demonstrated that solving both problems of landing and mapping simultaneously brings the best reconstruction results [18], as the quality of the final reconstruction tightly depends in the quality of the ego-motion estimation. Hence, the metrics we used for this comparison are entirely based on ego-motion estimation, which can be effectively compared for all SLAM and VO algorithms. Among the different metrics for ego-motion available in the literature, we found that the metrics present in most of the methods listed in [179] were: the absolute trajectory RMSE (ATE), the relative pose RMSE (RPE), the cumulated trajectory, rotation, and scale errors, the alignment error, and the alignment RMSE.

The ATE and RPE are local pose accuracy metrics proposed by [68], commonly applied along with the EUROC dataset. The relative pose error is a metric for the accuracy of an estimated trajectory over a defined time interval Δ. In this way, this metric corresponds to the drift of the estimated trajectory. For a sequence of estimated poses $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_n \in SE(3)$ and a ground truth trajectory $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n \in SE(3)$, the relative pose error for each timestamp i is:

$$\boldsymbol{E}_i := (\boldsymbol{Q}_i^{-1}\boldsymbol{Q}_{i+\Delta})^{-1}(\boldsymbol{P}_i^{-1}\boldsymbol{P}_{i+\Delta}). \quad (39)$$

So, for a sequence of n poses, an $m = n - \Delta$ number of relative poses is obtained. Then the root mean square error (RMSE) for such errors over all the timestamps of the sequence can be calculated as:

$$RMSE(\boldsymbol{E}_{i:n}, \Delta) := \sqrt{\frac{1}{m}\sum_{i=1}^{m} \|trans(\boldsymbol{E}_i)\|^2}, \quad (40)$$

where $trans(\boldsymbol{E}_i)$ corresponds to the translational component of the relative pose error $\boldsymbol{E}_i$. Many VO or SLAM systems can be evaluated for a timestep interval $\Delta = 1$, but some methods work on frames or keyframes windows (like [2], [23], [101]); thus, different Δ values might be appropriate for testing. So, for SLAM systems' evaluation, it can also be useful to obtain the RMSE for all the possible time intervals:

$$RMSE(\boldsymbol{E}_{i:n}) = \frac{1}{n}\sum_{\Delta=1}^{n} RMSE(\boldsymbol{E}_{i:n}, \Delta). \quad (41)$$

The ATE was proposed to evaluate the estimated trajectory's global consistency. The ATE estimation was achieved by comparing the absolute distances between the estimated trajectory and the ground truth directly. So, the trajectories are first aligned using Horn's method [187] to find the rigid body transformation $\boldsymbol{S}$ to map the estimated trajectory $\boldsymbol{P}_{1:n}$ into the ground truth trajectory $\boldsymbol{Q}_{1:n}$; hence, the absolute trajectory error for each timestamp i can be calculated as:

$$\boldsymbol{F}_i := \boldsymbol{Q}_i^{-1}\boldsymbol{S}\boldsymbol{P}_i. \quad (42)$$

In the same way as RPE, the RMSE for all the timestamps of the translational components is calculated as follows:

$$RMSE(\boldsymbol{F}_{i:n}) := \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|trans(\boldsymbol{F}_i)\|^2}. \quad (43)$$

Thus, for the [68] benchmark, the RPE combines the translational and rotational errors elegantly, while the ATE considers only the translational error component. In contrast, for the TUM-Mono benchmark [103], the authors proposed to benefit from large loop sequences. This way, instead of using the complete exploring motion pose information, the TUM-Mono benchmark was built to register the ground truth of each sequence's first and last 10–20 s, using the LSD-SLAM [4] method to track only those segments. In this way, the authors used the accumulated drift for all their metrics, and they demonstrated that the error registered by each evaluated run was not originated in the SLAM method drift used to register the ground truth; instead, it came from the accumulated drift by the algorithm through the entire trajectory. So, any SLAM or VO system can be used to register the start and end segments' ground truth. Consequently, the TUM-Mono benchmark aligns the estimated trajectory with the start and end ground truth segments and measures their differences. Let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbb{R}^3$ be the estimated tracked positions for the 1 to n frames and $S \subset [1; n]$ and $E \subset [1; n]$ be the frame indices corresponding to the start and end segments for the groundtruth positions $\widehat{\boldsymbol{p}} \in \mathbb{R}^3$. Then, by aligning the estimated trajectory with the groundtruth start and end segments, the two relative transformations can be calculated as:

$$\boldsymbol{T}_s^{gt} := \underset{\boldsymbol{T} \in Sim(3)}{\operatorname{argmin}} \sum_{i \in S} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2 \tag{44}$$

$$\boldsymbol{T}_e^{gt} := \underset{\boldsymbol{T} \in Sim(3)}{\operatorname{argmin}} \sum\nolimits_{i \in E} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2. \tag{45}$$

By these transformations, the accumulated drift can be calculated as:

$$\boldsymbol{T}_{drift} := \boldsymbol{T}_e^{gt} \left(\boldsymbol{T}_e^{gt}\right)^{-1}. \tag{46}$$

The translation, rotation, and scale error components can be extracted as, respectively:

$$e_t := \left\| translation(\boldsymbol{T}_{drift}) \right\|$$

$$e_r := rotation(\boldsymbol{T}_{drift})$$

$$e_r := rotation(\boldsymbol{T}_{drift}).$$

As a result, the authors established the alignment error, which is a metric that equally takes into account the errors produced by the translational, rotational, and scale effects:

$$e_{align} := \sqrt{\tfrac{1}{n} \sum\nolimits_{i=1}^{n} \left\| \boldsymbol{T}_s^{gt} \boldsymbol{p}_i - \boldsymbol{T}_e^{gt} \boldsymbol{p}_i \right\|_2^2}. \tag{47}$$

This metric can be computed individually for the start and end segment, but when it is estimated by combining both intervals, it is equivalent to the translational RMSE when aligned to the ground truth. Thus, it can also be formulated as follows:

$$e_{rmse} := \sqrt{\min_{\boldsymbol{T} \in Sim(3)} \tfrac{1}{|S \cup E|} \sum\nolimits_{i \in S \cup E} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2}. \tag{48}$$

As can be observed, in contrast to the APE and ATE, which only include two metrics explaining the rotation and translation effects, the TUM-Mono benchmark analyzes the SLAM or VO performance method in a more detailed way, providing six metrics explaining the amount of the accumulated translation, rotation, and scale errors, as well as determining the performance of the algorithm in the start and end segment to better identify the initialization and accumulated drift errors, finally calculating the translational RMSE to visualize the global effects of the combined metrics on the whole evaluated sequence. For these reasons, we selected the TUM-Mono and its official metrics to execute a complete comparison.

3. Results

As mentioned above, the algorithms were selected based on their open-source availability and independence from any additional input information source other than a monocular RGB frame sequence. Following the primary taxonomy described in [2], [179], we selected the classic sparse-indirect system ORB-SLAM2 [7], the classic dense-indirect system DF-ORB-SLAM [181], the classic dense-direct system

LDS-SLAM [4], and the classic sparse-direct method DSO [2]. Then, we added to this study the currently available ML implementations of the ORB-SLAM2 [7], DSO [2], and SVO [13], which are the DynaSLAM [118], CNN-DSO [174], and CNN-SVO [54]. Additionally, the direct proposals derived from the DSO system were included due to the impressive reconstruction results that this method showed during experimental evaluation; thus, the LDSO [23] and DSM [84] systems representing the addition of loop closure and SLAM capabilities for the DSO system were selected. Figure 59 and 60 present some examples of the evaluated algorithms' execution on the outdoor and indoor sequences of the TUM-Mono dataset.

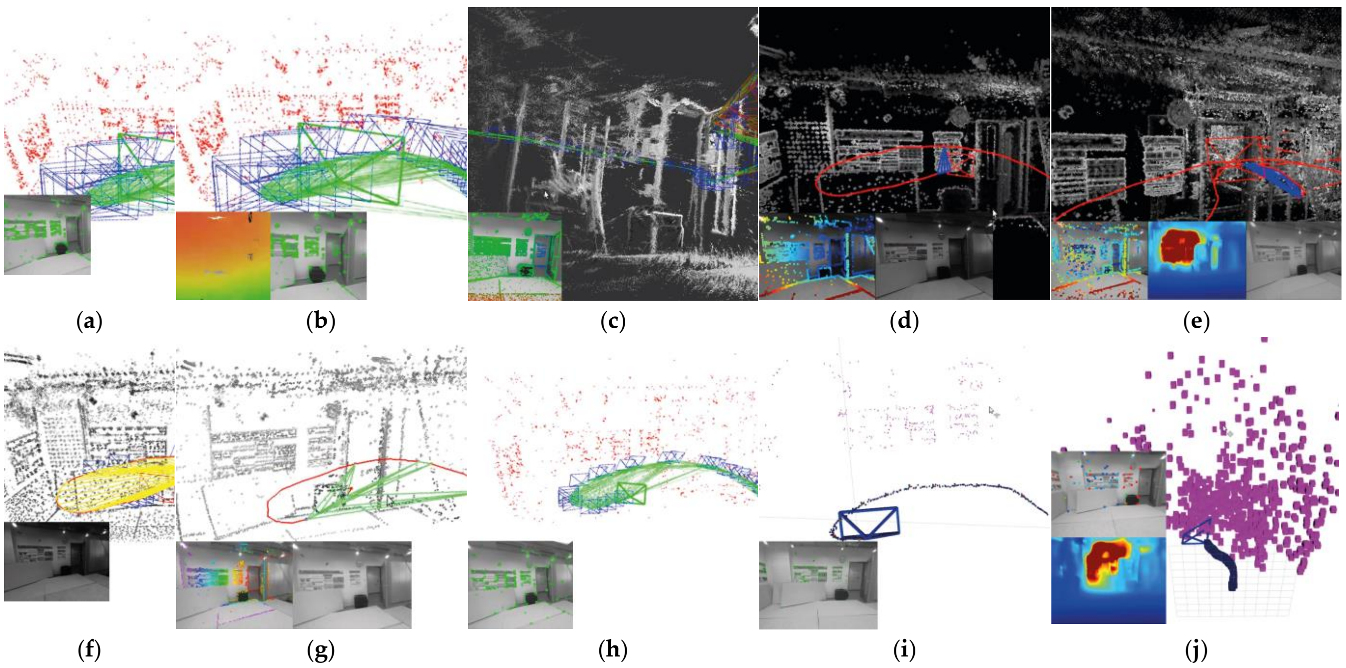


Figure 59. Examples of each indoor algorithm sequence execution *seq-01*—TUM-Mono dataset. The implemented methods were: (**a**) ORB-SLAM2, (**b**) DF-ORB-SLAM, (**c**) LSD-SLAM, (**d**) DSO, (**e**) CNN-DSO, (**f**) LDSO, (**g**) DSM, (**h**) DynaSLAM, (**i**) SVO ,and (**j**) CNN-SVO.

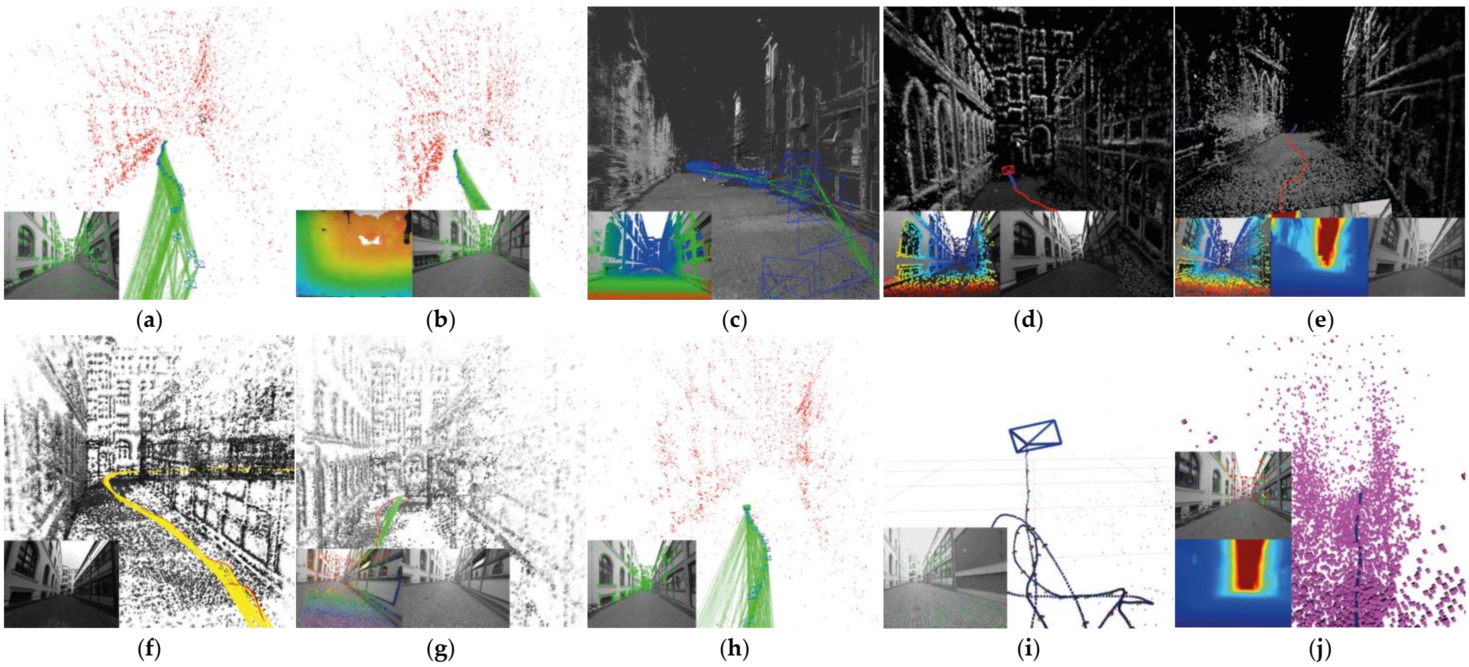


Figure 60. Algorithm executions for the outdoor sequence *seq-02* of the TUM-Mono dataset. The implemented methods were: (**a**) ORB-SLAM2, (**b**) DF-ORB-SLAM, (**c**) LSD-SLAM, (**d**) DSO, (**e**) CNN-DSO, (**f**) LDSO, (**g**) DSM, (**h**) DynaSLAM, (**i**) SVO and (**j**) CNN-SVO.

3.1. Hardware Setup

All the methods were evaluated on the same hardware platform with the same available computational and power resources using Ubuntu 18.04 and 16.04 operating systems, depending on each algorithm's software requirements. For this evaluation, we selected readily available and cheap hardware components to assemble a desktop based on the AMD Ryzen™ 7 3800X processor and the GPU NVIDIA GEFORCE GTX 1080 Ti. The technical specifications are summarized in Table 4.

Table 4. Specifications of the hardware used during experimentation.

Component	Specifications
CPU	AMD Ryzen™ 7 3800X, eight cores, 16 threads, 3.9–4.5 GHz.
GPU	NVIDIA GEFORCE GTX 1080 Ti. Pascal architecture, 1582 MHz, 3584 CUDA cores, 11 GB GDDR5X.
RAM	16 GB, DDR 4, 3200 MHz
ROM	SSD NVME M.2 Western Digital 7300 MB/s
Power consumption	750 W [1]

[1] The hardware did not reach the max power consumption. The avg. load was close to 600 W during the experiments.

Additionally, we provide some CPU performance metrics obtained by averaging the metrics of ten executions of each algorithm over the *sequence*_01 of the TUM-Mono dataset. All these CPU usage metrics were obtained using the hardware detailed in Table 1, using the official codes provided in their repositories and the same dependencies' versions listed by each author. The metrics selected to provide an idea of the computational expenses required by each algorithm were: the overall CPU usage (multicore), the current CPU usage, the amount of GPU memory required by the algorithm, the amount of RAM used while running the algorithm, the time that the algorithm required to process the *sequence*_01 of the TUM-Mono dataset, and the number of frames per second that the algorithm processed. The computational expenses associated with each approach to reconstruct the *sequence*_01 scene are presented in Table 5.

Table 5. The average CPU usage performance metrics for the ten executions of each algorithm on *sequence*_01 of the TUM-Mono dataset.

Method	Overall CPU Usage, Multicore	Current CPU Usage	GPU Usage	Memory Usage	Time (s)	FPS
ORB-SLAM2	1.8472%	16.2374%	1.2376%	9.2147%	128.4571	37
DF-ORB-SLAM	1.9254%	17.4235%	1.7861%	12.4572%	133.1217	36
LSD-SLAM	2.4578%	18.4521%	1.6423%	10.3457%	138.4172	34
DSO	1.2604%	14.6818%	1.8971%	9.3892%	91.2564	52
SVO	**1.1286%**	**10.4589%**	**1.7852%**	**8.5316%**	**87.5241**	**55**
LDSO	1.6909%	15.4717%	3.1588%	14.2962%	99.4758	48
DSM	1.8346%	31.9216%	2.7203%	24.1591%	315.4982	15
DynaSLAM	1.9247%	21.4576%	15.3467%	20.3879%	118.3245	40
CNN-DSO	4.0647%	30.9091%	27.5346%	24.6742%	161.2389	30
CNN-SVO	3.2579%	27.8461%	24.4732%	23.5476%	134.7583	35

As can be noticed in Table 5, the SVO was the fastest algorithm that required fewer computational resources to be implemented, closely followed by the DSO and ORB-SLAM2. However, as described in the following sections, the SVO presented strong trajectory loss issues and poor 3D reconstruction quality; so, it might not be considered the best alternative. In addition, it can be noticed that adding ML techniques to geometric-based approaches implied a considerable increase in the CPU, GPU, and memory use.

3.2. Comparative Analysis

As mentioned in Section 2.3, we used the TUM-Mono dataset and benchmark because it has a complete set of metrics; all its sequences were gathered using only monocular cameras, and it presents the largest collection of 50 sequences and scenarios that comprise multiple outdoor and indoor examples. In addition, it must be considered that these sequences were gathered using a pure monocular handheld camera and were recorded by a walking person; so, the results presented in this section might not be generalized to considerably different applications like autonomous driving, flying drones, and medical exploration applications, among others.

As addressed in the related works [7], [103], we followed the authors' suggestions of running each sequence of a benchmark five times to create cumulative-error plots and account for the nondeterministic nature of each system [180]. Nevertheless, authors like [23], [118] performed their experimental comparisons by running each sequence ten times in forward and backward reproduction directions to better capture the probabilistic behavior of the algorithms against multiple variations like illumination and dynamic objects. In this way, we applied this extended approach, given the large variety of algorithms we tested. In total, we performed ten runs of each of the 50 sequences in forward and backward modalities, gathering a total of 1000 runs for each method; so, for the ten evaluated algorithms, we created a database of 10,000 trajectory files saved in .txt format that were processed using the MATLAB scripts provided in the official repository of the TUM-Mono benchmark [103].

The TUM-Mono benchmark scripts require a specific structure, where each algorithm must output a .txt file containing all the camera poses registered during each sequence algorithm execution, where each $\boldsymbol{P}_i$ pose must be in the quaternion format represented in Equation (1). However, the SVO, CNN-SVO, and DSM algorithms output rotation and translation matrixes instead of quaternions, which are incompatible with the TUM-Mono format. In this way, we had to modify the codes of these methods to introduce the correct output format, following the proposal of Sarabandi and Thomas [188], by applying Equations (11)–(15) to convert the translation and rotation outputs of the SVO, CNN-SVO, and DSM to quaternions.

$$\boldsymbol{P}_i = (t_i \ x_i \ y_i \ z_i \ q_{x_i} \ q_{y_i} \ q_{z_i} \ q_{w_i}) \tag{49}$$

Given the rotation matrix:

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix},$$

$$q_x = \begin{cases} \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}}, & if\ r_{11} + r_{22} + r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{32} - r_{23})^2 + (r_{13} - r_{31})^2 + (r_{21} - r_{12})^2}{3 - r_{11} - r_{22} - r_{33}}}, & otherwise \end{cases} \tag{50}$$

$$q_y = \begin{cases} \frac{1}{2}\sqrt{1 + r_{11} - r_{22} - r_{33}}, & if\ r_{11} - r_{22} - r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{32} - r_{23})^2 + (r_{12} + r_{21})^2 + (r_{31} + r_{13})^2}{3 - r_{11} + r_{22} + r_{33}}}, & otherwise \end{cases} \tag{51}$$

$$q_z = \begin{cases} \frac{1}{2}\sqrt{1 - r_{11} + r_{22} - r_{33}}, & if - r_{11} + r_{22} - r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{13} - r_{31})^2 + (r_{12} + r_{21})^2 + (r_{23} + r_{32})^2}{3 + r_{11} - r_{22} + r_{33}}}, & otherwise \end{cases} \tag{52}$$

$$q_z = \begin{cases} \frac{1}{2}\sqrt{1 - r_{11} - r_{22} + r_{33}}, & if - r_{11} - r_{22} + r_{33} > \eta \\ \frac{1}{2}\sqrt{\frac{(r_{21}-r_{12})^2+(r_{31}+r_{13})^2+(r_{32}+r_{23})^2}{3+r_{11}+r_{22}-r_{33}}}, & otherwise \end{cases}. \quad (53)$$

As reported in [188], the best results that outperformed Shepperd's rotation to the quaternion method were achieved for $\eta = 0$; so, we set this value to build the trajectory files for those methods that did not match the evaluation format. In addition, the ORB-SLAM2, DF-ORB-SLAM, DynaSLAM, SVO, CNN-SVO, and DSM required a different calibration camera model than that the provided by the benchmark that includes full photometric data considering the geometric intrinsic calibration, photometric calibration, and the nonparametric vignette calibration, while the rest of the methods used an ATAN camera model based on the FOV distortion model of [189] provided in the official PTAM repository [90]. We used the ROS calibration package [190] to estimate three radial and two tangential distortion coefficients $\boldsymbol{d}_{coeff} = (k_1\ k_2\ p_1\ p_2\ k_3)$, following the formulation of Equations (6) and (7). The results were also tested and compared with the OpenCV Camera calibration package [191].

For each undistorted pixel at (x_u, y_u) coordinates, its position in the distorted image is (x_d, y_d):

$$\begin{aligned} x_u &= x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_u &= y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \quad (54)$$

$$\begin{aligned} x_u &= x_d + [2p_1 x_d y_d + p_2(r^2 + 2x_d^2)] \\ y_u &= y_d + [p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d], \end{aligned} \quad (55)$$

where r is the distorted radius $r_d = \sqrt{x_d^2 + y_d^2}$. As suggested in [103], to make a fair comparison based on the accumulated drift over the aligned start and end sequences, we disabled the loop closure for the SLAM methods ORB-SLAM2, DF-ORB-SLAM, DynaSLAM, LDSO, and DSM. Figure 61 presents each algorithm's cumulative error plots for the translational, rotational, and scale errors. These graphs depict the number of runs for each error type below a certain x-value. Hence, the methods close to the top left corner were better because they reached a determined error value after more executions.

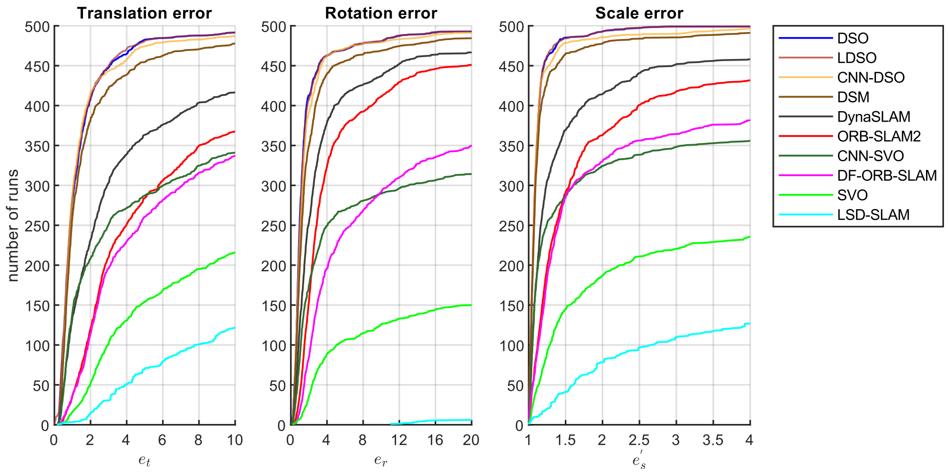


Figure 61. Translation e_t, rotation e_r, and scale e_s' accumulated errors for the ten evaluated algorithms.

As can be seen in Figure 61, the sparse-direct methods (DSO, LDSO, and DSM) achieved the best overall performance, followed by the sparse-indirect method (ORB-SLAM2), the dense-indirect method (DF-ORB-SLAM), and the hybrid method (SVO); the dense-direct method (LDS-SLAM) showed the worst performance. The ORB-SLAM2 and SVO CNN versions showed an important improvement over their classic versions. At the same time, the CNN-DSO did not outperform the DSO in accumulated translation, rotation, and scale metrics but remained close to the performance of the DSO. Finally, it must be mentioned that the large error observed in the LSD-SLAM, SVO, and CNN-SVO methods can be attributed to the severe initialization and relocalization problems that the algorithms presented during the evaluations in the TUM-Mono dataset.

As mentioned, the alignment error considers the translation, rotation, and scale errors equally. Therefore, it is equivalent to the translational RMSE when aligned to the start and end segments (the first and last 10–20 s of each sequence), for which the ground truth is available. The cumulated alignment error for each algorithm is presented in Figure 62.

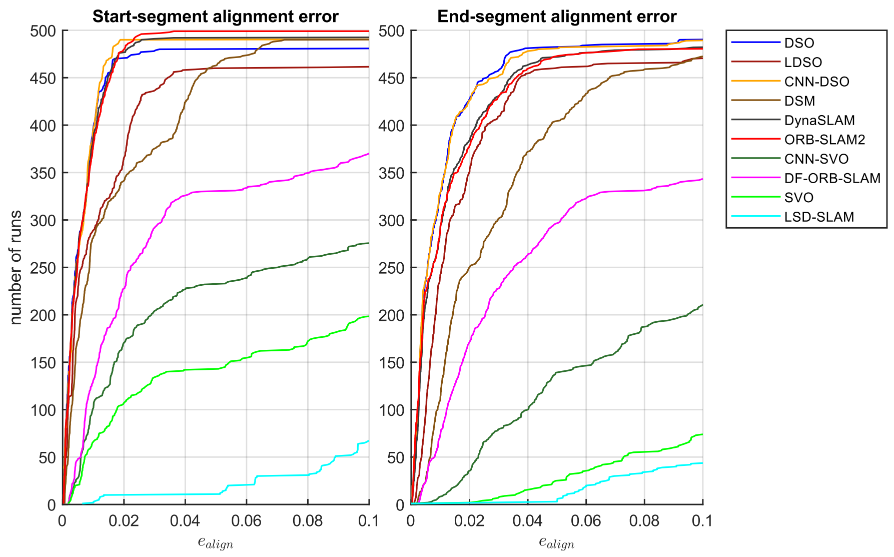


Figure 62. Start and end segment alignment error, corresponding to the RMSE of the alignment error when compared with the start and end segment's ground truth.

Figure 62 shows that the ORB-SLAM and DynaSLAM performed slightly better than the sparse direct methods for start-segment alignment errors. However, on the end segment, the cumulative drift effect was lower on the sparse-direct methods ratifying the results observed in Figure 4. In addition, it can be noticed that the CNN-DSO performed better than the DSO, suggesting that integrating the Single Image Depth Estimation (SIDE) CNN improved the DSO bootstrapping by adding the prior depth information, whereas the end-segment performance of both algorithms was similar. Moreover, the addition of the Mask R-CNN in DynaSLAM was used to remove scenes' moving objects information and did not represent a clear improvement in algorithm performance in the start segment, but, as shown in Figure 5, the benefits of adding the CNN can be observed over the end segment by the reduction in the accumulated drift. Additionally, for the hybrid approaches, the addition of the MonoDepth CNN made a paramount contribution in helping to overcome SVO loss of trajectory issues. Similar to the results in Figure 1, the overall alignment error results suggest that the sparse-direct methods performed better, followed by the sparse-indirect, dense-indirect, hybrid, and finally the dense-direct, reaching a threshold error in around 50 runs.

As suggested by [103], we examined the dataset motion bias for each algorithm by running each method ten times forwards and ten times backward in such modalities and combining both to visualize how much each algorithm is affected. This situation allowed us to consider the importance of evaluating the SLAM and VO methods in large datasets, covering as many environments and motion patterns as possible. The dataset motion bias for each method is presented in Figure 63.

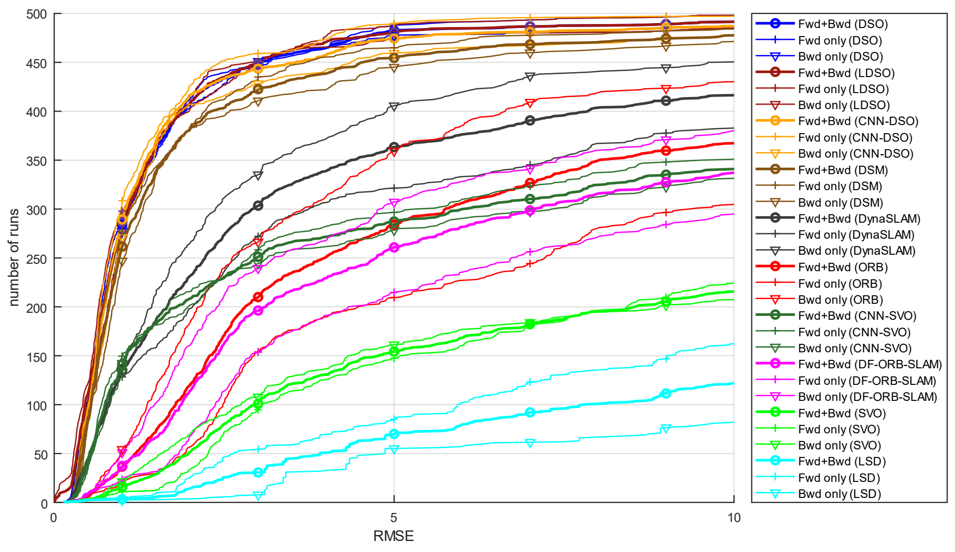


Figure 63. The dataset motion bias for each method was evaluated by running all sequences forwards and backward, as well as their combination (bold).

In Figure 63, it can be noticed that the DSO, LDSO, and SVO were not seriously affected by motion bias. In contrast, different motion patterns considerably affected the ORB-SLAM2, DynaSLAM, and DF-ORB-SLAM. This can be observed in the performance differences when running forwards versus backward. This behavior provides a reference for the consistency and robustness of each algorithm for using them in different environments or applications. It can be observed that the CNN-DSO on forward-only modality outperformed its classic version, but it suffered from a larger motion bias effect affecting its overall performance; while DynaSLAM and CNN-SVO outperformed their classic versions and presented less motion bias effect, representing an additional robustness improvement over their classic versions.

Figure 64 shows the color-coded alignment error for each of the 50 TUM-Mono sequences for each run forward and backward to observe which specific sequences were challenging for each algorithm.

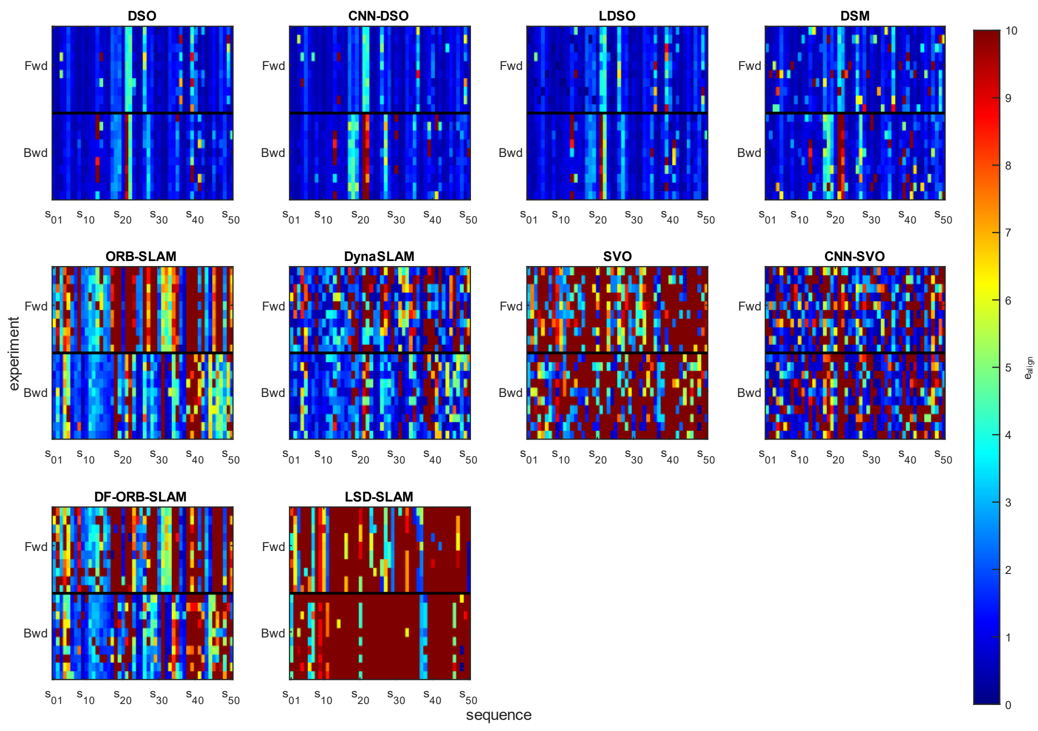
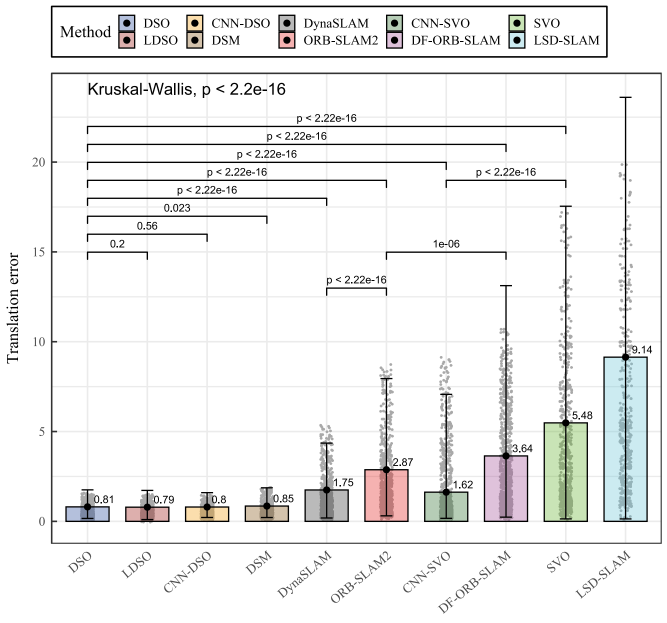


Figure 64. The color-coded alignment error e_{align} for each algorithm in the TUM-Mono dataset.

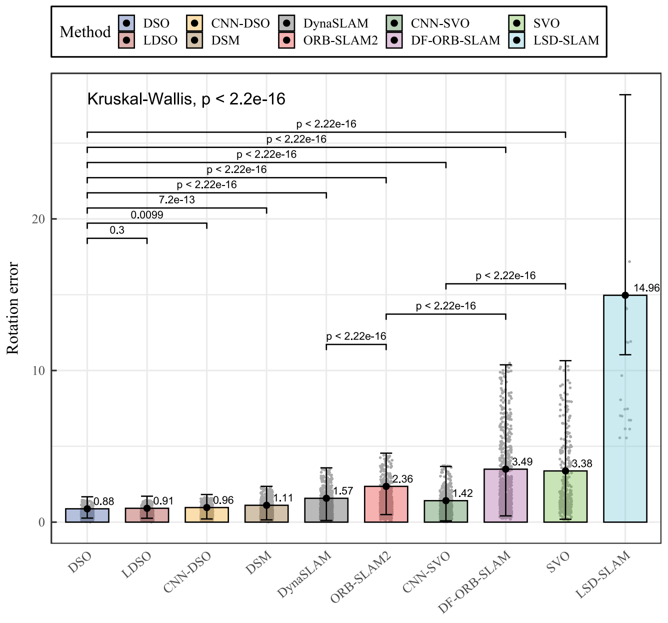
The first row of Figure 64 presents the sparse-direct methods, DSO, CNN-DSO, LDSO, and DSM, demonstrating an outstanding performance compared to the rest of the evaluated methods belonging to different taxonomy classifications placing the sparse-direct methods as the best alternative for the Visual Odometry, SLAM, and 3D reconstruction tasks. It can be noticed that the CNN-DSO performed worse than the original DSO algorithm in sequences 13 and 22 but outperformed the DSO in sequence 39. The LDSO performance was close to the DSO, but it presented a better trajectory in some forward sequences and overcame the DSO in sequence 21. The DSM performed similarly to the rest of the sparse-direct approaches but occasionally presented trajectory loss issues affecting the overall performance. Furthermore, the DynaSLAM considerably outperformed the ORB-SLAM2, especially in challenging sequences like 18, 19, 21, 22, 23, 27, 28, 38, 39, and 40, among others, where the ORB-SLAM commonly failed. However, it occasionally presented trajectory loss and initialization issues. The ORB-SLAM2 optical flow implementation performed slightly worse on forwards and considerably worse on backward, especially in scenes 21, 22, 38, 39, 40, 46, 48, and 50. In contrast, the CNN SVO version considerably reduced the RMSE in most sequences compared to the SVO but still constantly failed in the outdoor sequences 21 and 22, showing random initialization and trajectory loss issues. As reported in [2], the SVO and LSD-SLAM methods had the worst results over the whole dataset, which was why Engel et al. [2] did not include these methods in their study. However, we think it is vital to report such results and the errors attributed to these algorithms' commonly known initialization and trajectory loss errors over the sequences of the TUM-Mono dataset.

The results processed on the TUM-Mono benchmark for the cumulative translation error e_t, rotation error e_r, scale error e'_s, start-segment alignment error e^s_{align}, end-segment alignment error e^e_{align}, and the translational RMSE e_{RMSE} were gathered in a database defining the method as the categoric variable. The statistical results were processed using R programing language. First, we removed the blank observations for the executions, where each algorithm became lost or could not initialize; so, the Mahalanobis distances were [192] as a multivariate data cleaning technique to detect and remove the outlier observations. A 22.4577 cut score based on the χ^2 distribution for a 99.999% interval was set up detecting 344 outlier observations ending with a database of 8860 observations.

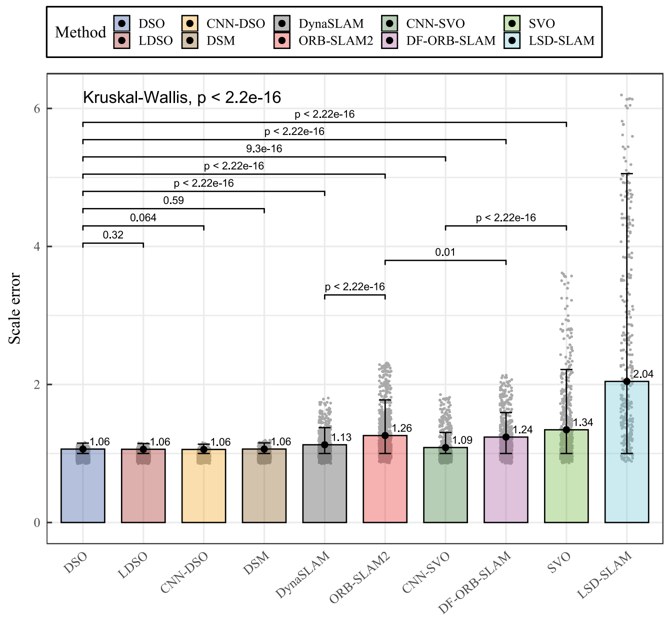
Then, each dependent variable's normality and homogeneity assumptions were verified to select the appropriate statistical test for comparisons. For example, for the translation error, the p-values of 2.2×10^{-16} for the DSO, LDSO, CNN-DSO, DSM, DynaSLAM, ORB-SLAM2, DF-ORB-SLAM, CNN-SVO, SVO, and LSD-SLAM methods were obtained in the Lilliefors (Kolmogorov–Smirnov) normality test; so, the sample did not reach the normality assumption. We applied Levene's test obtaining a p-value of 2.2×10^{-16} for the homogeneity assumption; so, the sample did not meet the homogeneity assumption. The rest of the dependent variables had similar assumptions verification results; thus, it was concluded that the sample was not parametric. Hence, the Kruskal–Wallis test was selected as the general test, with the Wilcoxon signed rank as a pairwise post hoc test. Figure 65 and Table 6 present the results obtained by applying the differences tests.



(**a**)



(**b**)



(**c**)

Figure 65. Bar plots, box-plot error bars, and Kruskal–Wallis comparisons for the medians of the cumulative errors collected after 1000 runs of each algorithm: the (a) translation error, the (b) rotation error, and the (c) scale error.

Table 6. Medians and Kruskal–Wallis comparisons for each algorithm's translation, rotation, and scale errors.

Method	Translation Error	Rotation Error	Scale Error
Kruskal–Wallis general test	$\chi^2 = 3582.9$ $p_{value} = 2.2e-16$	$\chi^2 = 2278.4$ $p_{value} = 2.2e-16$	$\chi^2 = 2419.1$ $p_{value} = 2.2e-16$
DSO	0.8064585 ^{a}	0.8800369 ^{b}	1.064086 ^{ab}
LDSO	0.7892125 ^{a}	0.9135608 ^{ab}	1.061302 ^{ab}
CNN-DSO	0.7980411 ^{a}	0.9618528 ^{a}	1.058849 ^{a}
DSM	0.8519143 ^{b}	1.1117710 ^{c}	1.064615 ^{b}
DynaSLAM	1.7473504 ^{c}	1.5730542 ^{d}	1.126499 ^{c}
ORB-SLAM2	2.8738313 ^{d}	2.3585843 ^{e}	1.260155 ^{d}
CNN-SVO	1.6248001 ^{c}	1.4159545 ^{d}	1.086399 ^{e}

DF-ORB-SLAM	3.6423921 [e]	3.4940400 [f]	1.238232 [f]
SVO	5.4819407 [f]	3.3772024 [f]	1.343603 [g]
LSD-SLAM	9.1403348 [g]	14.9621188 [g]	2.044298 [h]

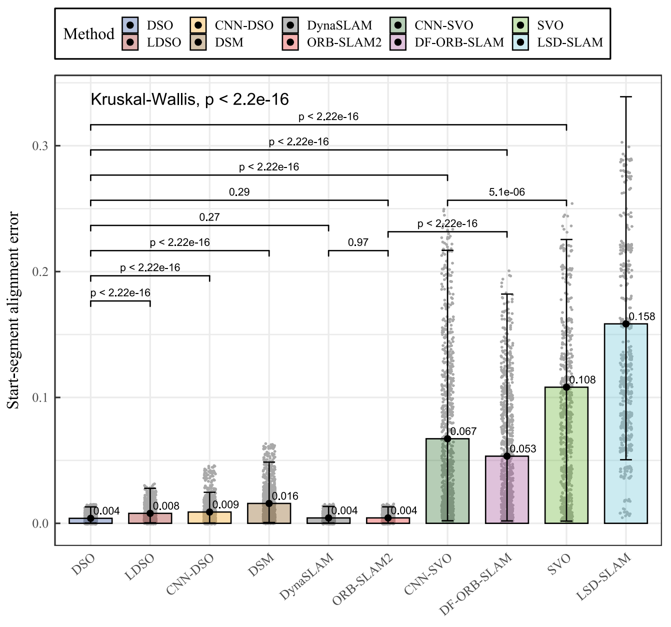
Means with different letters in the same column differ significantly according to the Kruskal–Wallis test and pair-wise Wilcoxon signed rank test for $p_{value} \leq 0.05$.

As presented in Figure 65 and Table 6, the sample identified significant differences between the implemented algorithms. By observing the translation error, it can be noticed that the DSO, LDSO, and CNN-DSO methods achieved the most significant performance of the ten evaluated algorithms; despite the DSO performing at 2.18% and 1.05% worse than the DSO and CNN-DSO in this metric, the difference was not significant among them. The DSO, LDSO, and CNN-DSO achieved significantly lower errors than the dense-direct method DSM. The feature-based methods performed significantly worse than the sparse-direct methods, where the DynaSLAM achieved a significantly better performance than the ORB-SLAM2 and DF-ORB-SLAM, reaching a 39.19% and 52.02% translation error reduction, respectively. The CNN-SVO performed slightly worse than the DynaSLAM, but the difference was not significant, while it significantly outperformed its classic version achieving a 47.57% translation error reduction. The LSD-SLAM performed substantially worse in terms of the translation error metric among the ten algorithms.

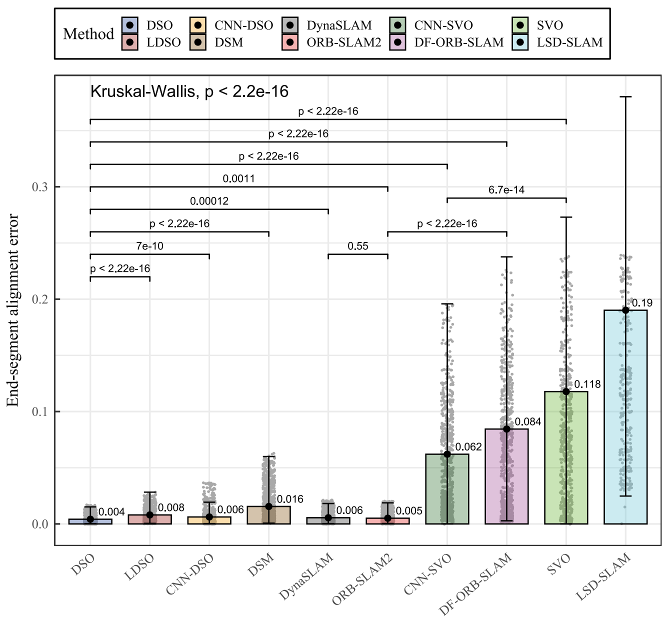
Regarding the rotation error, the DSO and LDSO achieved significantly better results than the rest of the algorithms. Although the DSO showed an average rotation error reduction close to 3.66%, the difference was not significant. The DSO performed significantly better than its neuronal version in the accumulated rotation error metric. The LDSO performed around 5.02% better than the CNN-DSO, but the difference was not significant. The DSM performed significantly worse than the rest of the sparse-direct methods. The feature-based methods performed worse than the sparse-direct methods in the rotation error metric, where the DynaSLAM achieved a downright performance than the ORB-SLAM2 and DF-ORB-SLAM, showing an average error reduction close to 33.30% and 54.97%, respectively. The CNN-SVO performed better than the DF-ORB-SLAM, SVO, and LSD-SLAM in terms of the rotation error metric, significantly outperforming its classic SVO version showing a 58.07% average reduction in the rotation error. The LSD-SLAM performed significantly worse than the other methods in the rotation error metric.

For the scale error metric, the sparse-direct methods, DSO, LDSO, and CNN-DSO, performed significantly better, where the CNN-DSO showed the best performance by an average 0.49% and 0.23% reduction compared to the DSO and LDSO, but the difference was not significant. The DSM performed significantly worse than the CNN-DSO. The feature-based methods performed significantly worse than the sparse-direct methods on the scale error metric, where the DynaSLAM achieved the significantly best performance and an average reduction of 10.60% and 9.02% compared to the ORB-SLAM2 and DF-ORB-SLAM. The CNN-SVO performed significantly better than the feature-based methods, SVO and LSD-SLAM, exhibiting a 19.14% average error reduction compared to its classic version, SVO. Again, the LSD-SLAM performed worst in the scale error metric.

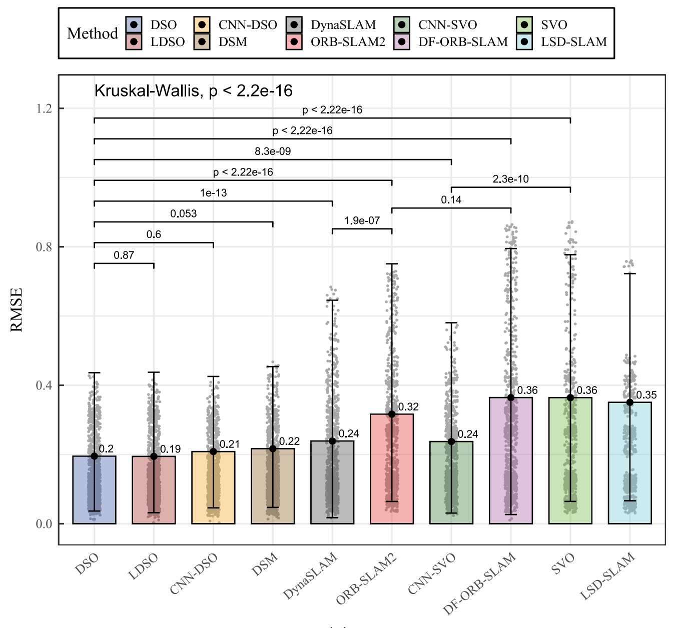
Similarly, the Kruskal–Wallis test was applied as general test, with the Wilcoxon signed rank test, for statistical comparison among the ten methods for the start- and end-segment alignment errors and the overall RMSE. Figure 66 and Table 7 present the results obtained by applying the differences tests.



(**a**)



(**b**)



(**c**)

Figure 66. Bar plots, box-plot error bars, and Kruskal–Wallis comparisons for the medians of the cumulative errors collected after 1000 runs—(**a**) only the start-segment alignment error, (**b**) only the end-segment alignment error, (**c**) the RMSE for the combined effect on the start and end segments.

Table 7. Medians and Kruskal–Wallis comparisons for the translation errors of each algorithm.

Method	Start-Segment Alignment Error	End-Segment Alignment Error	RMSE
Kruskal–Wallis general test	$\chi^2 = 4575.7$ $p_{value} = 2.2e - 16$	$\chi^2 = 3718$ $p_{value} = 2.2e - 16$	$\chi^2 = 530.78$ $p_{value} = 2.2e - 16$
DSO	0.003974759 [a]	0.004184367 [a]	0.1950799 [ab]
LDSO	0.007925665 [b]	0.008009198 [b]	0.1944492 [a]
CNN-DSO	0.008987173 [b]	0.006199582 [c]	0.2083872 [ab]
DSM	0.015794222 [c]	0.015537213 [d]	0.2167750 [b]
DynaSLAM	0.004286919 [a]	0.005516179 [e]	0.2389837 [cd]
ORB-SLAM2	0.004311949 [a]	0.005102672 [e]	0.3165024 [e]
CNN-SVO	0.067201999 [d]	0.062036008 [f]	0.2373532 [c]
DF-ORB-SLAM	0.053360456 [e]	0.084420570 [g]	0.3643844 [e]
SVO	0.108150349 [f]	0.117753996 [h]	0.3642558 [e]
LSD-SLAM	0.158469383 [g]	0.190127787 [i]	0.3507099 [d]

Means with different letters in the same column differ significantly according to the Kruskal–Wallis test and pairwise Wilcoxon signed rank test for $p_{value} \leq 0.05$.

Figure 66 and Table 7 show many significant differences between the ten compared methods on the alignment error and RMSE metrics. Regarding the start-segment alignment error, the DSO, DynaSLAM, and ORB-SLAM methods outperformed the rest of the algorithms. Despite the DSO slightly reducing the average start-segment alignment error by around 7.28% and 7.81% compared to the DynaSLAM and ORB-SLAM2, the differences were not significant. The rest of the sparse-direct methods, LDSO, CNN-DSO, and DSM, performed significantly worse than the DSO by an average of 49.84%, 55.77%, and 74.83%, respectively, in the start-segment alignment error metric. For the feature-based methods, the DynaSLAM and DSO performed significantly better than the DF-ORB-SLAM, while the DF-ORB-SLAM achieved an error significantly lower than the CNN-SVO, SVO, and LSD-SLAM. When comparing the CNN-SVO with its predecessor SVO, the difference was significant, where the neural version reduced the start-segment alignment error by an average of close to 37.86%. The LSD-SLAM achieved the worst start-segment alignment error of the ten methods, significantly.

By observing the end-segment alignment error, it was found that the sparse-direct methods significantly outperformed the rest of the compared methods. The DSO significantly outperformed all the evaluated methods, including the rest of the sparse-direct methods, the LDSO, CNN-DSO, and DSM, reducing the average alignment error by around 47.85%, 32.50%, and 73.06%, respectively. In the sparse-indirect category, the DynaSLAM and ORB-SLAM2 performed significantly better than the DF-ORB-SLAM, but even though the ORB-SLAM2 reduced the average end-segment error by approximately 7.49%, the difference was not significant. The CNN-SVO end-segment alignment error was significantly lower than the error of the SVO, reducing this metric by approximately 47.31%. The LDSO performed significantly worse than the rest of the methods.

For the RMSE metric, the sparse-direct methods performed significantly better than the rest, where the LDSO achieved RMSE values around 0.32% and 6.68% lower than the DSO and CNN-DSO; the differences were not significant. The LDSO performed significantly better than the DSM, with an average RMSE around 10% lower. In the feature-based classification, the DynaSLAM performed significantly better than the ORB-SLAM2 and the DF-ORB-SLAM, reducing the RMSE metric by approximately 24.49% and 34.41%, respectively. For the hybrid methods, the CNN-SVO performed significantly better than the SVO, reducing the RMSE by around 34.83%. As with the rest of the metrics, the LSD-SLAM performed significantly worse than the other methods in terms of the RMSE metric.

Finally, in Figure 67, we present the sample trajectories obtained by the three overall best methods evaluated in this comparison study. To exemplify the behavior of the algorithms in different environments, we selected the sequence *seq-02* of the TUM-Mono dataset as an example for indoors and the sequence *seq-29* as an example for outdoors. In addition, we provide video samples of the execution of each algorithm as supplementary material in the GitHub repository: https://github.com/erickherrerare-search/MonocularPureVisualSLAMComparison, along with all the .txt result files of each algorithm run for reproducibility.

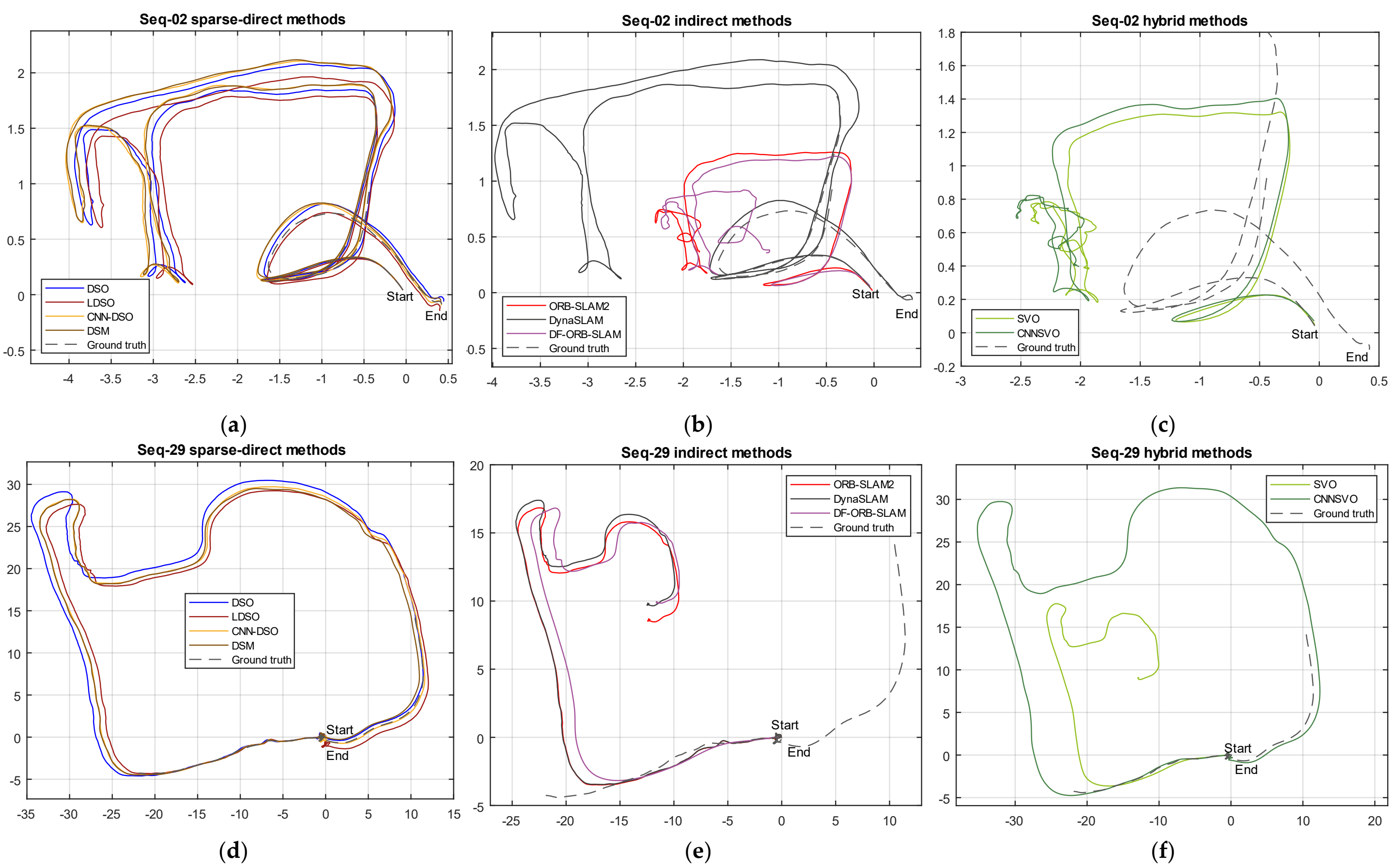


Figure 67. Trajectories in the TUM-Mono dataset for the compared sparse-direct (**a**,**d**), indirect (**b**,**e**), and hybrid methods (**c**,**f**). The top row displays the results for the indoor sequence *seq-02*, and the bottom row displays the results for the outdoor sequence *seq-29*. The solid lines represent the trajectory estimated by each algorithm; the dashed lines represent the aligned ground truth.

As depicted in Figure 67, the algorithm's observed behavior ratified this comparative analysis of quantitative results. On the top row, for the indoor sequence, it can be noticed that the sparse-direct methods outperformed the other evaluated methods starting and ending their trajectory pretty close to the ground truth. The indirect methods behaved completely differently, where the system constantly lost the trajectory, accumulating drift and obtaining the wrong scale measures concatenated erroneously when the system achieved relocalization. The DynaSLAM represents an important contribution to the ORB-SLAM2 system because it estimated the trajectory better than the rest, closing the trajectory close to the ground truth, while the rest of the indirect systems lost their trajectories. On the other hand, the hybrid methods performed considerably worse indoors; so, many of the algorithms' runs did not complete the full frame sequence, and the algorithm typically finished its trajectory pretty far from the end-segment ground truth. In the bottom row of Figure 67, it can be noticed again that the sparse-direct methods outperformed the rest of the evaluated systems, with an appropriate bootstrapping in the start segment and a small amount of accumulated drift in the end segment.

In Figure 67e, similar to indoors, the indirect methods suffered from trajectory loss issues despite the fact that the relocalization module typically was able to continue the system execution, accumulating a critical amount of drift during relocalization, making the estimated trajectory end far from the groundtruth end segment. In the hybrid methods, the SVO suffered from similar issues to the indirect methods. However, it can be noticed that the CNN version of the SVO improved its performance outdoors, differently from indoors, which can be explained because the added CNN MonoDepth module was trained in the Cityscapes dataset, which was mainly trained from outdoor sequences.

4. Discussion

In the previous studies of [16], [180], the authors compared the DSO, LDSO, ORB-SLAM2, and DynaSLAM algorithms on the same TUM-Mono benchmark, and their findings mostly matched what we observed during this evaluation. However, we extended their study considerably by implementing six additional methods following the taxonomy described in [179] and performed an appropriate statistical analysis to determine the significant differences in each system's performance. Thus, we could observe the classification advantages and limitations for classic geometric-based approaches. The classic approaches can be classified as sparse-indirect, dense-indirect, dense-direct, sparse-direct, and hybrid. For the sparse-indirect, we selected its gold standard, ORB-SLAM2, and we can report that, in our experience, it is an excellent method that demands low computational power and has an average performance not being the best but still working well outdoors. Its main limitations are its poor indoor performance and large drift and scale error accumulation during relocalization. We believe the ORB-SLAM2 has achieved high popularity due to its low computational power consumption and ease of implementation.

For the dense-indirect category, we selected the DF-ORB-SLAM, an ORB-SLAM2 open-source implementation with an additional optical flow estimation module that allows the ORB-SLAM2 to work with more image information. In this case, we observed that the optical flow module increased the computational cost of the algorithm and slightly reduced the occurrence of trajectory loss issues. However, adding image information for feature extraction also introduced noise in the estimation steps, significantly increasing the translation, rotation, scale error metrics, and RMSE. For dense-direct approaches, we selected their gold standard, the LSD-SLAM, one of the first proposed direct methods. In this case, we found that the performance was significantly the poorest of all the evaluated methods. This situation could be due, in particular, to the frequent initialization errors and trajectory loss in most of the benchmark sequences matching what Engel et al. reported in their study [2]. For the sparse-direct classification, we selected the most iconic VO system, the DSO. This system significantly outperformed the methods of the rest of the classic classifications by a large margin, demonstrating an impressive performance indoors and outdoors. The DSO was slightly outperformed by its neuronal version on the scale metric and slightly outperformed by the LSDO in the translation and RMSE metric, but the differences were not significant. This behavior lets us conclude that even its LDSO, CNN-DSO, and DSM extensions do not significantly outperform the DSO, and this method can still be considered a great avenue for future work, improvements, and contributions.

Additionally, it must be mentioned that, in contrast to what was reported in [84], the DSM method did not outperform its predecessor, the DSO, in the TUM-Mono dataset, even after implementing a complete SLAM pipeline to extend the DSO. We believe the DSM is a robust system that can contribute to the sparse-direct category. However, the authors used the extended pinhole radial-tangential camera model instead of the complete photometrical camera calibration provided in the original DSO, including the intrinsic, photometric, and nonparametric vignette calibration. As reported in [180], this situation considerably contributed to the correct execution of the algorithm and allows it to obtain the best results. Then, for the hybrid approaches, we tested the SVO, which combines direct and indirect formulation paradigms in its pipeline. The SVO was second to last in our comparison, being significantly outperformed by most of the methods in this study, only performing significantly better than the LSD-SLAM, in line with what was reported in [2]. However, with this analysis, we could observe that it at least performed better than the dense-direct method. The SVO is also a popular method in the robotics research field. We believe that this is caused by its early appearance in 2014, low computational power requirement, and open-source availability for implementation with C++ or ROS.

For machine learning classification, there are many ML implementations available in the literature [3], [6], [113], [114], [116]–[118], [135], [136], [147], [8], [9], [12], [28], [49], [50], [111], [112], and many of them include open source code implementations [6], [156], [168], [170]–[173], [159]–[164], [166], [167]. Nevertheless, most methods were formulated to work with more than one input mode, like RBD-D or INS, and their code implementations did not include monocular running instructions or did not include their monocular RGB pipeline, e.g., [6], [168], [172]. Other open-source code implementations required additional external information for running, like optical flow or feature extractors, that were not included as open source, e.g., [156]. For such reasons, we selected three ML versions of the classic approaches of the DSO, ORB-SLAM2, and SVO: CNN-DSO, DynaSLAM, and CNN-SVO. Therefore, we concluded that the CNN-SVO significantly outperformed its predecessor in all the metrics, the DynaSLAM significantly outperformed the ORB-SLAM2 in all the metrics except for the end-segment alignment error where the difference was not significant, and the CNN-DSO significantly outperformed its classic version only in the rotation error metric. Here, we can mention that in contrast to what is reported in the CNN-DSO official repository [174], where the algorithm was evaluated in the first eleven sequences of the KITTI dataset, after testing the algorithm in a larger dataset indoors, outdoors, and a large variety of motion patterns, the CNN-DSO only slightly outperformed the DSO in the scale error metric, but the observed difference was not significant, while the DSO still outperformed it in rotation and in the start- and end-segment alignment error metrics. In addition, during the evaluation, it was observed that the algorithm introduced a considerable number of outlier points in the obtained 3D reconstruction. Thus, we can point out that machine learning studies are making vital contributions to enhancing the monocular VO, SLAM, and 3D reconstruction systems. Table 8 summarizes the observed advantages and limitations of the evaluated methods based on the experience of implementing and running the algorithms.

Table 8. Practical advantages and limitations of the evaluated methods.

Method	Category	Advantages	Limitations
ORB-SLAM2 [54]	Classic sparse-indirect	Low computational cost. Multiple input modes. Ease of implementation. Robustness to multiple environments.	Trajectory loss issues. Accumulation of drift while relocalizing. Sparse 3D reconstruction.
DF-ORB-SLAM [16]	Classic dense-indirect	Low computational cost. Reduction in trajectory loss issues.	Introduction of noise for trajectory estimation. Accumulation of drift on relocalization. Sparse 3D reconstruction. Significant reduction in the performance of ORB-SLAM2.
LSD-SLAM [29]	Classic dense-direct	Low computational cost. More detailed 3D reconstruction, but with the presence of outliers. More information in the final 3D reconstruction.	Poorest performance of the evaluated methods. Initialization issues. Trajectory loss issues.
DSO [21]	Classic sparse-direct	Low computational cost. Ease of implementation. More detailed and precise 3D reconstruction. Robust to multiple environments and motion patterns. Best performance of all methods in most of the metrics.	Requirement for a specific complex camera calibration. Slightly, but not significantly, lower performance than the LDSO in the translation and RMSE metrics.
SVO [13]	Classic hybrid	Low computational cost. Good documentation and open-source availability for implementation in diverse configurations.	Frequent trajectory loss issues. Initialization issues. Critical execution errors due to the absence of a relocalization module.
LDSO [30]	Classic sparse-direct	Low computational cost. Similar to DSO, detailed and precise 3D reconstruction. Ease of implementation. Loop closure module. Slightly but not significantly better performance than the DSO in translation and rotation error. Best performance in the translation	Requirement of a specific complex camera calibration. Significantly worse performance than the DSO in the end-segment error metric.

		and RMSE metrics (compared to DSO), but without considerable difference.	
DSM [31]	Classic sparse-direct	Detailed and precise 3D reconstruction. Robust execution in most of the environments and motion patterns. Complete and interactive GUI.	Requirement of more computational capabilities than the rest of the sparse-direct methods. Significant underperformance compared to most of the sparse-direct methods.
CNN-DSO [15]	ML sparse-direct	Detailed and precise 3D reconstruction. Robust to multiple environments and motion patterns. Best performance in scale error metric.	Presence of outliers in the 3D reconstruction. Significantly better performance in the rotation error metric by the DSO. Difficult to implement. Specific hardware requirement.
DynaSLAM [32]	ML sparse-indirect	Multiple input modes. Ease of implementation. Robustness to multiple environments. Ability to detect, segment, and remove information of moving objects. Especially recommended for dynamic environments. Fewer trajectory loss issues than ORB-SLAM2.	Accumulation of drift while relocalizing. Sparse 3D reconstruction. Increase in complexity over the ORB-SLAM2. Specific hardware requirement.
CNN-SVO [54]	ML hybrid	Considerable reduction in the trajectory loss issues compared to the SVO. Initialization issues. Reduction in the number of execution issues compared to the SVO. Improved performance over the ORB-SLAM2 in the rotation, translation, scale, and RMSE metrics. Significant improvement over its classic version in all the metrics.	Considerable presence of outliers in the 3D reconstruction. Imprecise and sparser 3D reconstruction. Complex implementation. Specific hardware requirement.

5. Intermediate conclusions

In this article, the most representative open source monocular RGB SLAM and VO available implementations were tested following a taxonomy to determine the advantages and limitations of each method and classification, providing the reader a guide to correctly select the method that fits their needs or to select a path to make future contributions to the tested methods and classifications. After experimentation, it can be concluded that the monocular SLAM and VO methods need to be evaluated on larger datasets in a large variety of environments, motion patterns, and illumination conditions to be effectively compared with the state of the art, as demonstrated in this study for methods like the DSM, CNN-DSO, and DF-ORB-SLAM that did not match the expected results on the TUM-Mono dataset.

The sparse-direct category of the taxonomy achieved the significantly best results among all the ten methods in the translation, rotation, scale, and RMSE metrics outputting the most detailed and precise 3D reconstructions of the tested methods. At second best, the sparse-indirect category achieved good ego-motion estimation but output sparser 3D reconstructions that might not be suitable for many applications, presenting trajectory loss issues and evidencing worse performance indoors. Additionally, by including three machine learning-based methods and comparing them with their classic versions, we can conclude that the integration of machine learning significantly improves the performance of the SLAM or VO systems and should be considered as a future research direction to overcome the limitations of each system. Integrating CNN information for the estimation steps contributes to mitigating monocular systems' commonly known scale ambiguity issue. This behavior was demonstrated in each ML method's significant scale error reduction compared to their classic versions.

Through experimentation, refs. [103], [180] concluded that the great majority of the alignment error originated in the accumulated drift, independent from the noise in the ground truth that can be registered with any SLAM or VO, which allows using all the metrics using the ground truth of only the start and end segments. We agree and confirm that this conclusion allowed us to compare a wide variety of methods coming from different configurations and classifications that output trajectories in different scales and orientations, which can be efficiently compared after the proposed benchmark alignment method.

Chapter IV - Proposal

1. Introduction

Three-dimensional (3D) reconstruction is a complex, ill-posed challenge that has been approached through various techniques and perspectives to accurately replicate the geometry of real-world settings. Numerous scholars have attempted to resolve this issue by employing diverse sensors, including lasers, infrared, radar, LIDAR, cameras, and their combinations, supplemented by additional sensors like GPS and INS [18], [29], [38]. With the advancements in computer vision over recent years, cameras have emerged as a precise tool for this task. For 3D reconstruction, cameras are utilized in several ways: in stereo configurations that facilitate pixel triangulation; alongside active sensors such as RGB-D cameras; in conjunction with inertial INS sensors; and as standalone monocular cameras [1]. Among these, the monocular RGB camera is particularly intriguing to many researchers due to its cost-effectiveness, widespread availability in mobile devices, and independence from the limited range of additional sensors, which restricts their use in expansive areas. However, relying solely on a monocular camera for information introduces complexity due to the absence of depth or scale data, necessitating increased computational effort. This is why purely visual monocular methods are inherently scale-ambiguous, a significant drawback when contrasted with Stereo, RGB-D, or RGB-INS methods, and a primary cause of drift. Therefore, this study seeks to mitigate these issues by incorporating depth-prior information into a popular monocular RGB system.

The process of 3D reconstruction is often a product of Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), and Structure from Motion (SFM). Various research efforts have contributed to enhancing this process, as precise mapping has significantly improved the performance of SLAM, VO, and SFM systems, particularly those focused on ego-motion estimation. This is because these systems often rely on tracking from the geometry map, and it has been established that tackling simultaneously tracking and mapping yields better results [18], [38]. As delineated in preceding research [193], SLAM, VO, and SFM can be categorized into three groups: direct versus indirect, dense versus sparse, and classical versus Machine Learning-based. Indirect methods preprocess images to extract visual features or optical flow, reducing data volume and producing sparse scene representations. Conversely, direct methods operate on pixel intensities, allowing for more image data but at a higher computational cost. Dense methods aim to reconstruct extensive scene geometry, whereas sparse methods recover minimal point clouds. Classical methods rely on mathematical, geometric, optimization, or probabilistic approaches without ML integration. ML methods, on the other hand, enhance classical approaches with ML techniques to improve performance from various angles. The taxonomy for SLAM, VO, and SFM suitable for 3D reconstruction includes: classic-sparse-indirect [7], [14], [31], [33], [71], [90], classic-dense-indirect [74], [77], classic-dense-direct [4], [5], [80], classic-sparse-direct [2], [23], [101], classic- hybrid [13], ml-sparse-indirect [50], [118], [161], ml-dense-indirect [113], [114], [116], [135], [136], ml-dense-direct [3], [6], [8], [9], ml + classic-sparse-direct [21], [28], [112], [147] and ml hybrid [54][1]. Furthermore, a prior study [194] evaluated various open-source methods across this taxonomy using a comprehensive monocular benchmark [103], revealing that sparse direct methods notably surpassed others in terms of translation, rotation, scale errors, and alignment RMSE across the TUM-Mono dataset's 50 sequences [103].

Consequently, we have chosen the classic-sparse-direct method DSO, recognized as one of the superior classic methods, and aim to enhance its depth map estimation by integrating a Single Image Depth Estimation (SIDE) CNN module. This integration is intended to augment the mapping capabilities of the DSO method and yield an accurately scaled reconstruction through the pixel-wise depth information estimated by the network. The SIDE technique selected is NeW CRFs, which stands at the forefront of CNN monocular depth estimation. Our experimental findings indicate that incorporating the

[1] For further details of the cited algorithms and the described taxonomy, we encourage the reader to take a look at our previous work [193]

NeW CRFs CNN module allows DeepDSO to surpass its classic counterpart in terms of rotation, translation, scale, and RMSE metrics.

2.1. Related works

Several initiatives have been undertaken to enhance the efficacy of sparse-direct methods by incorporating Convolutional Neural Networks (CNNs) into their frameworks [21], [28], [112], [147]. These integrations have been applied to various tasks, such as improving the computation of disparity maps[170], substituting the depth estimation component with a CNN [171], concurrently determining depth, pose, and uncertainty [112], identifying and segmenting moving objects through depth masks [147], and calculating pose transformations [28]. Yet, these proposals have not extensively investigated the potential of infusing prior depth knowledge to restrict the point search range along the epipolar line, an idea akin to the method presented in [54], which was successfully applied to the SVO hybrid model[54]. The most closely related work to our method is the CNN-SVO, as described by [54], which adapted the SVO hybrid system by incorporating a single image depth estimation module to refine its depth estimation process. The original SVO system utilized a Gaussian model for depth estimation and a Beta distribution to gauge the inlier ratio, with the Beta distribution parameters being updated incrementally through Bayesian methods and depth filters that converge when the variance falls below a set threshold. In contrast, the authors of CNN-SVO suggested narrowing the epipolar search interval in subsequent frames by using the depth of each pixel, as determined by the SIDE CNN, to calculate the mean and variance of the inverse depth. This modification enabled the SVO method to more rapidly converge on the accurate depth and achieve a better-scaled reconstruction. The open-source CNN-DSO code, referenced in [174], applied the CNN-SVO's methodology to the DSO algorithm, utilizing MonoDepth [195] for SIDE depth estimation. However, it did not adhere closely to the approach outlined in [54], resulting in performance that did not markedly surpass that of the original DSO algorithm [194]. In contrast, our approach has successfully integrated the Loo [54] method into the original DSO system and employed the latest and most advanced SIDE CNN NeW CRFs. This integration has led to a substantial enhancement in the performance of the sparse-direct method.

2. Materials and Methods

Within the scope of this research, the materials and methodologies employed encompassed the Direct Sparse Odometry (DSO) algorithm for visual odometry, the NeW CRFs method for depth estimation, and an enhanced technique for initializing point maps. These components are foundational to our study and will be detailed in subsequent sections.

2.1. Review of the DSO algorithm

In the preceding investigation [194], Direct Sparse Odometry (DSO) has been recognized by computer vision and robotics experts for its excellent capabilities in monocular RGB visual odometry and monocular 3D reconstruction. That study involved an assessment of ten monocular RGB SLAM and VO algorithms[2], [4], [7], [23], [54], [84], [118], [174], [181], in line with the taxonomy suggested in [193]. The aim was to identify the most fitting category within the taxonomy for the task of 3D reconstruction, leading to the conclusion that sparse-direct methods significantly excel beyond the other methods evaluated. Therefore, the DSO method was chosen for this research due to its distinction as one of the foremost classic-sparse-direct methods and its accessibility as open-source. DSO operates directly on pixel data in a sparse manner, incorporating point selection and sampling techniques to diminish the data volume for processing since increasing pixel data does not necessarily improve map quality, as shown in [2]. DSO, inspired by the studies [4], [22], is a direct method that utilizes five to seven keyframes to create a sliding window, within which parameters are optimized jointly by minimizing photometric error. In such a window, there are m keyframe poses $T \in SE(3)$ and n points $\boldsymbol{p}$, constituting the set $\mathcal{W} = \{\boldsymbol{T}_1, \dots, \boldsymbol{T}_m, \boldsymbol{p}_1, \dots, \boldsymbol{p}_n\}$. The photometric error to be minimized is expressed as:

$$\min \sum_{\boldsymbol{T}_i,\boldsymbol{T}_j,\boldsymbol{p}_k \in \mathcal{W}} E_{i,j,k}\,, \text{where}$$

$$E_{i,j,k} = \sum_{\boldsymbol{p}\in\mathcal{N}_{\boldsymbol{p}_k}} \omega_p \left\| \left(I_j[\boldsymbol{p}'] - b_j\right) - \frac{t_j e^{a_j}}{t_i e^{a_i}} \left(I_i[\boldsymbol{p}'] - b_i\right) \right\|_\gamma, \tag{54}$$

where $\mathcal{N}_{\boldsymbol{p}_k}$ is the sum of the weighted photometric differences across the neighboring pixels of $\boldsymbol{p}_k$, factoring in the affine lighting parameters a and b, exposure time t, and the image I. The weight ω_p is a heuristic factor, and $\boldsymbol{p}'$ is the reprojected pixel position for point $\boldsymbol{p}$ in the current image I_j, calculated by:

$$\boldsymbol{p}' = \Pi\left(\boldsymbol{R}\Pi^{-1}\left(\boldsymbol{p}, d_{\boldsymbol{p}_k}\right) + \boldsymbol{\tau}\right) \tag{55}$$

with Π and Π^{-1} being the projection and back-projection functions, $\boldsymbol{R}$ and $\boldsymbol{\tau}$ representing the relative rotation and translation between frames as determined by transformations$\boldsymbol{T}_i$ and $\boldsymbol{T}_j$, and d being the inverse depth of a point.

The DSO frontend selects keyframes and points for photometric error optimization, initialises parameters, and decides which elements to marginalize. Keyframe management involves estimating the initial pose of a new image by projecting active points from the sliding window and aligning them directly. New keyframes are generated in response to changes in the field of view, occlusions, disocclusions, and exposure time, detected by computing optical flow and relative brightness. A new keyframe is added to the sliding window if it meets the criteria. Unlike ORB-SLAM's persistent co-visibility graph, DSO's sliding window is dynamic, with old or redundant keyframes being marginalized. Keyframes are distributed throughout the 3D space, with marginalization favouring the keyframe that maximizes a distance score $s(I_i)$, based on the Euclidean distance $d(i,j)$, between I_i and I_j keyframes, and a small constant ε:

$$s(I_i) = \sqrt{d(i,1)} \sum_{j\in[3,n]\setminus\{i\}} \left(d(i,j) + \varepsilon\right)^{-1} \tag{56}$$

DSO significantly subsamples data across frames for sparse point management to reduce computation and redundancy. Unlike indirect methods that rely on specific features like edges and corners, DSO can sample from all available data, enabling tracking in less textured areas. Candidate points are selected from image blocks based on a gradient threshold, with the process repeated at multiple scales to detect points with weaker gradients. These points are then tracked by searching along the epipolar line to minimize photometric error, with depth and variance calculated from the best match to constrain the search in subsequent frames. Following the approach in [54], our method posits that the search interval for each point's depth can be further constrained by depth and variance from a SIDE CNN depth estimator, leading to better depth initialization, faster convergence, and improved scaled 3D reconstruction. Active candidate points are then used to maintain a uniform spatial distribution, replacing old marginalized points in each image.

2.2. Review of NeW CRFs single image depth estimation

Depth estimation is an extensively researched area within robotics and computer vision, tackled using technologies such as lasers, radar, LIDAR, and cameras. A notable purely visual technique for this task is Single Image Depth Estimation (SIDE). Numerous strategies have been proposed to solve this challenge, employing traditional machine learning (ML) methods and a fusion of both. Traditional methods have explored the depth estimation of scene objects through feature utilization. However, studies such as [196] have demonstrated that local features alone are insufficient for accurately determining the depth of each pixel. Subsequent efforts have utilized discriminatively trained Markov Random Fields (MRF) [197] and Conditional Random Fields (CRFs) [198], which effectively model the depth of individual pixels and their interrelations, thereby facilitating improved inference of depth maps from various cues such as colour, pixel location, occlusions, known object sizes, haze, and defocus.

Neural networks have also been applied to depth map estimation, directly regressing a continuous depth map from an image [151], [155], [195], [199]–[203]. This approach heavily relies on training with extensive datasets featuring diverse motion patterns and lighting conditions. Due to the complexity of obtaining accurate ground truth depth maps for training, some methods have incorporated auxiliary information to support the neural network training, introducing additional sparse depth [204] or segmentation data [205], [206]. These methods aim to directly derive the depth map from image features, posing a challenging fitting problem that necessitates intricate neural network architectures.

The Neural Window (NeW) CRFs method [207], utilized in this study, merges traditional and ML approaches by incorporating an energy function with fully connected CRFs, optimized to yield precise depth maps. Given the efficiency of CNNs for depth estimation tasks, some methods have attempted to integrate them with neural networks, using CRFs to refine CNN outputs [208] or incorporating them into CNN layers [209]. However, this increases computational complexity, as fully connected CRFs require all nodes to be interconnected across the entire graph. To address this, the creators of NeW CRFs proposed dividing the graph into multiple sub-windows, connecting nodes only within each window, thus rendering CRFs computationally feasible.

MRFs and CRFs are often employed for complex prediction tasks like depth estimation and semantic segmentation due to their proven effectiveness in error correction based on neighbouring node information. Fully connected CRFs, in particular, are adept at expanding the receptive field [207], which is why they were employed in the NeW CRFs method. Here, the energy function of fully connected CRFs is formulated as:

$$E(x) = \sum_{i} \psi_u(x_i) + \sum_{ij} \psi_p(x_i, x_j) \tag{57}$$

where x_i denotes the predicted depth for node i, and j represents other nodes in the graph. The unary potential function ψ_u is computed for each node by the predictor, while the pairwise potential function ψ_p connects node pairs and is defined by:

$$\psi_p = \mu(x_i, x_j) f(x_i, x_j) g(I_i, I_j) h(p_i, p_j) \tag{58}$$

with $\mu(x_i, x_j) = 1$ for $i \neq j$ and zero otherwise. I_i and p_i represent the colour and position of node i, respectively. The unary potential is based on the distribution over predicted values, and the pairwise potential relies on the colour and position of pixel pairs:

$$\begin{gathered} \psi_u(x_i) = -logP(x_i|I) \\ \psi_p(x_i, x_j) = \mu(x_i, x_j)\|x_i - x_j\| e^{-\frac{\|I_i - I_j\|}{2\sigma^2}} e^{-\frac{\|p_i - p_j\|}{2\sigma^2}} \end{gathered} \tag{59}$$

Where I represents the input image, and P represents the probability distribution of the prediction. The pairwise potential applies heuristic penalties based on position and colour information, promoting the distinction of distant colours and positions while penalizing similarities in colour and proximity in pixels. This is computed for each sub-window rather than the entire graph and to relay information between non-overlapping windows by $(N/2, N/2\,)$, another energy function is calculated by successively shifting the windows by half their size. To incorporate this formulation into a neural network decoder, allowing the network to compute potential functions, the unary and pairwise potentials were redefined to be directly derived by the network:

$$\begin{gathered} \psi_u(x_i) = \theta_u(I, x_i) \\ \psi_p(x_i, x_j) = \omega(\mathcal{F}_i, \mathcal{F}_j, p_i, p_j)\|x_i - x_j\| \end{gathered} \tag{60}$$

Here, θ represents the parameters of the unary network, $\mathcal{F}$ denotes each feature map, and ω is the weighting function. For each node i, all pairwise potentials are aggregated as:

$$\psi_{p_i} = \alpha(\mathcal{F}_i, \mathcal{F}_j, p_i, p_j)x_i + \sum_{j \neq i} \beta(\mathcal{F}_i, \mathcal{F}_j, p_i, p_j)x_j \tag{61}$$

The network calculates the weighting functions α and β. A query vector $\boldsymbol{q}$ and a key vector $\boldsymbol{k}$ are implemented for each patch window as matrices $\boldsymbol{Q}$ and $\boldsymbol{K}$. The dot product of these matrices with the predicted values $\boldsymbol{X}$ retrieves the potential weight for any node pair. By incorporating the relative position $\boldsymbol{P}$, the potential for node i is computed as:

$$\sum_i \psi_{p_i} = SoftMax(\boldsymbol{Q} \cdot \boldsymbol{K}^T + \boldsymbol{P}) \cdot \boldsymbol{X} \tag{62}$$

In summary, the NeW CRFs method is constructed with a bottom-up-top-down architecture featuring four levels of neuronal CRFs optimization performed by decoder modules. Based on the Swin-Transformer [210], the encoders extract feature maps to feed into the decoder CRFs. The FC-CRFs module's structure and the network's overall architecture are depicted in Figure 68.

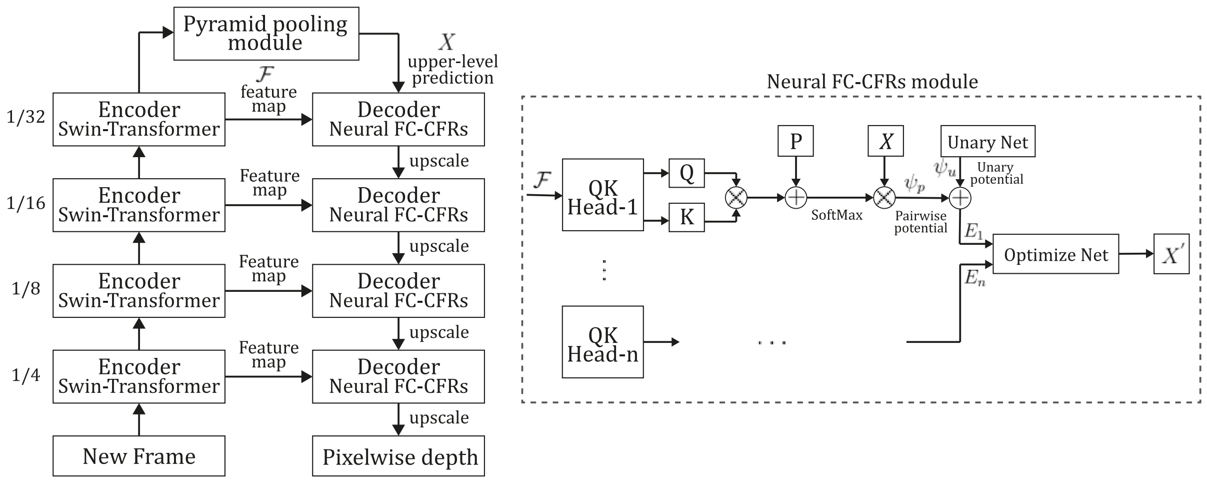


Figure 68. Bottom-up-top-down convolutional neural network (CNN) structure of the NeW CRFs, including its fully connected CRFs (FC-CRFs) module. Here, $\mathcal{F}$ denotes the feature map output from the encoder, X symbolizes the prediction from the higher level, and ψ_u and ψ_p represent the unary and pairwise potentials, respectively. The dashed lines indicate the decoder's FC-CRFs module, which constructs a multi-head energy function using the upper-level predictions and the feature map to enhance the prediction accuracy. This design is derived from the framework presented in [207].

2.3. Improved point's depth initialization for DSO

The integration of Convolutional Neural Network (CNN) outputs as prior information can significantly enhance the performance of Simultaneous Localization and Mapping (SLAM) or Visual Odometry (VO) tasks. This can be achieved through various strategies, such as substituting the depth estimation component with a Single Image Depth Estimation (SIDE) CNN [3], [112], [113],, or employing machine learning modules for refined parameter estimation [28], [49], semantic segmentation [3], [118], dynamic object detection and segmentation [118], [147], and enhancing depth estimation with depth priors. Previous research [194] has shown that incorporating a SIDE CNN to provide prior information can markedly improve the convergence of depth estimation modules, thereby boosting the overall efficacy of SLAM or VO systems [173], [174]. Our approach has adopted the strategy outlined in [54] and tailored it to fit within the Direct Sparse Odometry (DSO) framework.

This method capitalizes on the fact that, unlike feature-based methods that rely on feature matching, direct methods necessitate the recovery of point depths for tracking in a direct formulation, making precise depth estimation paramount. Furthermore, all monocular RGB SLAM and VO systems grapple with scale ambiguity. Loo's method [54] addresses both challenges by incorporating a point initialization strategy that leverages depth prior information from a single image depth estimation module. This strategy constrains the search interval for estimating the true depth of each point.

Traditionally, in the DSO method, depth estimation begins by identifying point candidates in subsequent frames through a discrete search along the epipolar line, minimizing photometric error. The best match's depth and variance are then computed using a probabilistic Gaussian framework, which encapsulates the uncertainty inherent in stereo-based depth estimation. However, rather than initializing these parameters randomly as many direct methods do, the depth and variance can be precisely initialized using data from a SIDE CNN. This reduces uncertainty and narrows the search interval, enabling the probabilistic framework to more rapidly converge on the true depth. Large uncertainties can lead to the selection of incorrect correspondences along the epipolar line in adjacent frames and a multitude of depth measurements that fail to converge on their true depths. It has been shown that providing depth prior information to constrain the search interval, thereby diminishing depth uncertainty, improves initialisation and enables new points to more swiftly converge on their true depth in a scaled fashion. This is because the search interval is now centred around the depth estimated by the SIDE, which also facilitates scaled reconstructions. This concept is illustrated in Figure 69.

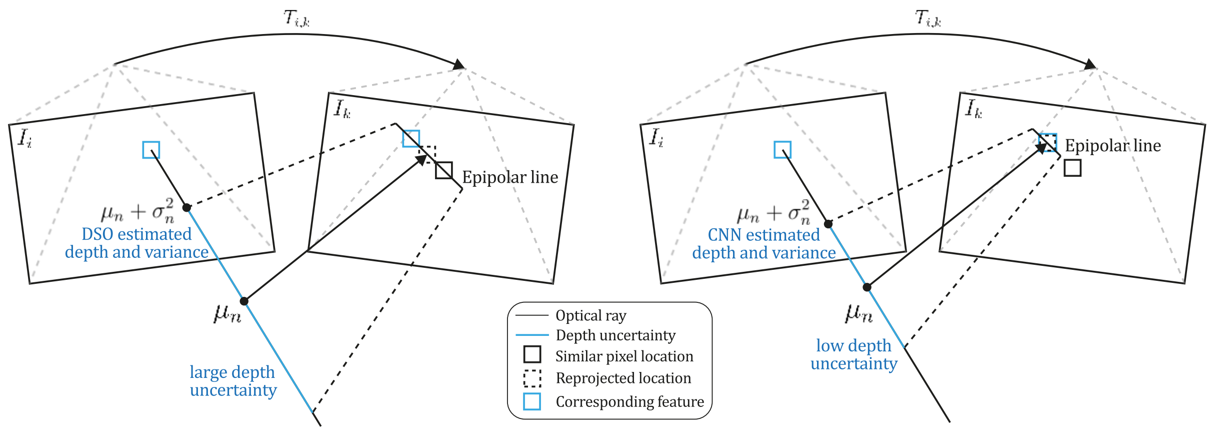


Figure 69. Map point initialization strategy. The DSO's search interval over the epipolar line is refined by incorporating CNN-derived prior information, such as estimated depth and variance. As shown in part b) of the figure, the epipolar search interval in subsequent frames is condensed, which helps each point to converge on its scaled true depth while excluding similar points from the search interval.

Thus, each point is initialized based on the depth provided by the CNN without discarding the DSO's proven depth estimation framework. This preserves the uncertainty-based inference capabilities inherited from [4], [22]. In this process, the inverse depth value of the best match from the previous frame is replaced with the depth estimation for that pixel position obtained by the NeW CRFs method, where $\mu_n = 1/d_{CNN}$ equals the inverse of the CNN-derived depth, and the depth variance follows the approach of [54], with $\sigma_n^2 = 1/(6d_{cnn})^2$ being inversely proportional to the square of six times the CNN-derived depth. This coefficient ensures that the search interval maintains a degree of uncertainty without compromising the scale provided by the CNN. Using these initialized parameters, μ_n and σ_n^2, the depth search interval $[p_i^{min}, p_i^{max}]$ is defined as follows:

$$p_i^{min} = \mu_n + \sqrt{\sigma_n^2}$$
$$p_i^{max} = \begin{cases} 0.00000001 & \text{if } \mu_n - \sqrt{\sigma_n^2} < 0 \\ \mu_n - \sqrt{\sigma_n^2} & \text{otherwise} \end{cases} \tag{63}$$

In this manner, depth prior information can direct the depth estimation process of the direct formulation by delimiting the search interval for each point along the epipolar line in each new frame. The incorporation of the depth module into the DSO algorithm is detailed in Figure 70.

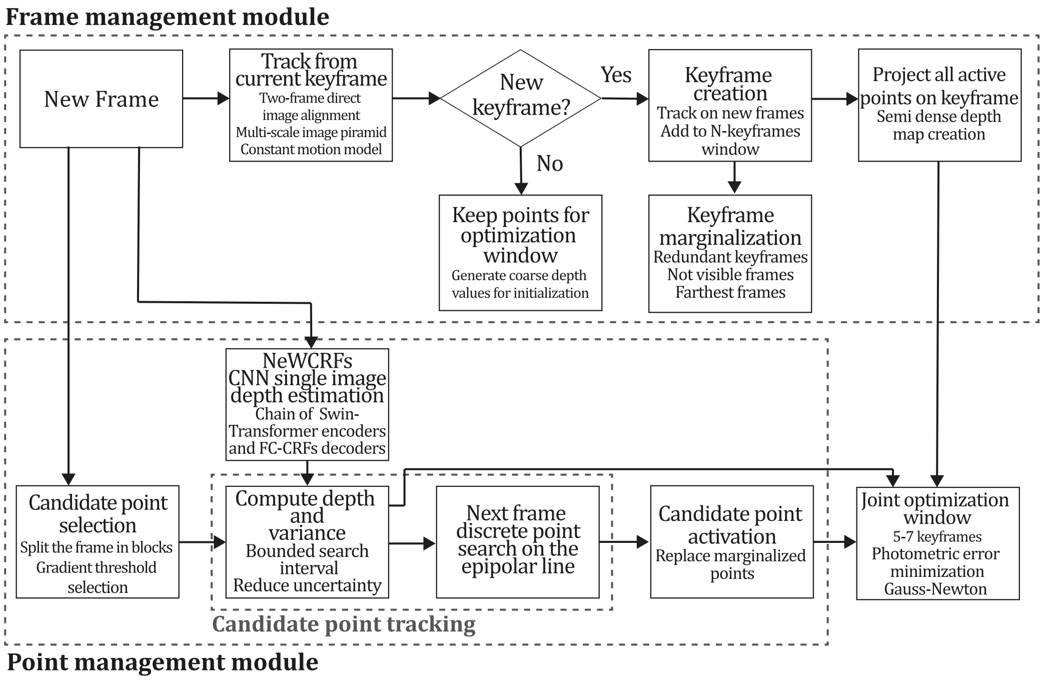

Frame management module
New Frame
Track from current keyframe
Two-frame direct image alignment
Multi-scale image piramid
Constant motion model
New keyframe?
Yes
No
Keyframe creation
Track on new frames
Add to N-keyframes window
Project all active points on keyframe
Semi dense depth map creation
Keep points for optimization window
Generate coarse depth values for initialization
Keyframe marginalization
Redundant keyframes
Not visible frames
Farthest frames
NeWCRFs
CNN single image depth estimation
Chain of Swin-Transformer encoders and FC-CRFs decoders
Candidate point selection
Split the frame in blocks
Gradient threshold selection
Compute depth and variance
Bounded search interval
Reduce uncertainty
Next frame discrete point search on the epipolar line
Candidate point activation
Replace marginalized points
Joint optimization window
5-7 keyframes
Photometric error minimization
Gauss-Newton
Candidate point tracking
Point management module

Figure 70. Diagram of DeepDSO algorithm. Integrating the NeW CRFs single image depth estimation module in the DSO method.

3. Results

The integration of Direct Sparse Odometry (DSO) with the Neural Window (NeW) CRFs CNN was executed using the C++ and Python programming languages on the Ubuntu 18.04 operating system. For the implementation and assessment, we opted for widely accessible and cost-effective computer hardware components, which were procured to construct a desktop system powered by an AMD Ryzen™ 7 3800X processor and an NVIDIA GEFORCE RTX 3060 GPU. The hardware specifications utilized for the evaluation are delineated in Table 9.

Table 9. Specifications of the hardware used during experimentation.

Component	Specifications
CPU	AMD Ryzen™ 7 3800X. 8 cores, 16 threads, 3.9 – 4.5 GHz.
GPU	NVIDIA GEFORCE RTX 3060. Ampere architecture, 1.78 GHz, 3584 CUDA cores, 12 GB GDDR6X. Memory interface width 192-bit. 2nd generation Ray Tracing Cores and 3rd generation Tensor Cores.
RAM	16 GB, DDR 4, 3200 MHz.
ROM	SSD NVME M.2 Western Digital 7300 MB/s
Power consumption	750 W[1]

[1] Hardware did not reach max power consumption. Avg. load was close to 550 W during experiments.

In a preceding study [194], we evaluated ten open-source algorithms from each taxonomy category, following the monocular benchmark established in [103]. It was found that sparse-direct methods significantly surpassed other methods in state-of-the-art performance. Consequently, this research was predicated on the classic sparse-direct method DSO, chosen for its remarkable capabilities in monocular reconstruction tasks. We considered the original DSO [2] and its neural enhancement, CNN-DSO [174], for comparative analysis to ascertain whether our proposed DeepDSO method substantially surpasses its classic counterpart and whether the advanced NeW CRFs single image depth estimation CNN notably augments the performance of the sparse-direct method.

DSO and CNN-DSO are publicly accessible on GitHub as open-source code [99], [174], and can be executed with minimal requirements. These include Pangolin V0.5 [211], OpenCV V3.4.16, TensorFlow V1.6.0, and the MonoDepth C++ version [212], which offers a speed advantage over its original Python implementation. Figure 71 presents examples of the 3D reconstructions obtained with the DeepDSO algorithm, and Figure 72 compares the three implementations running in the same sequences indoors and outdoors.

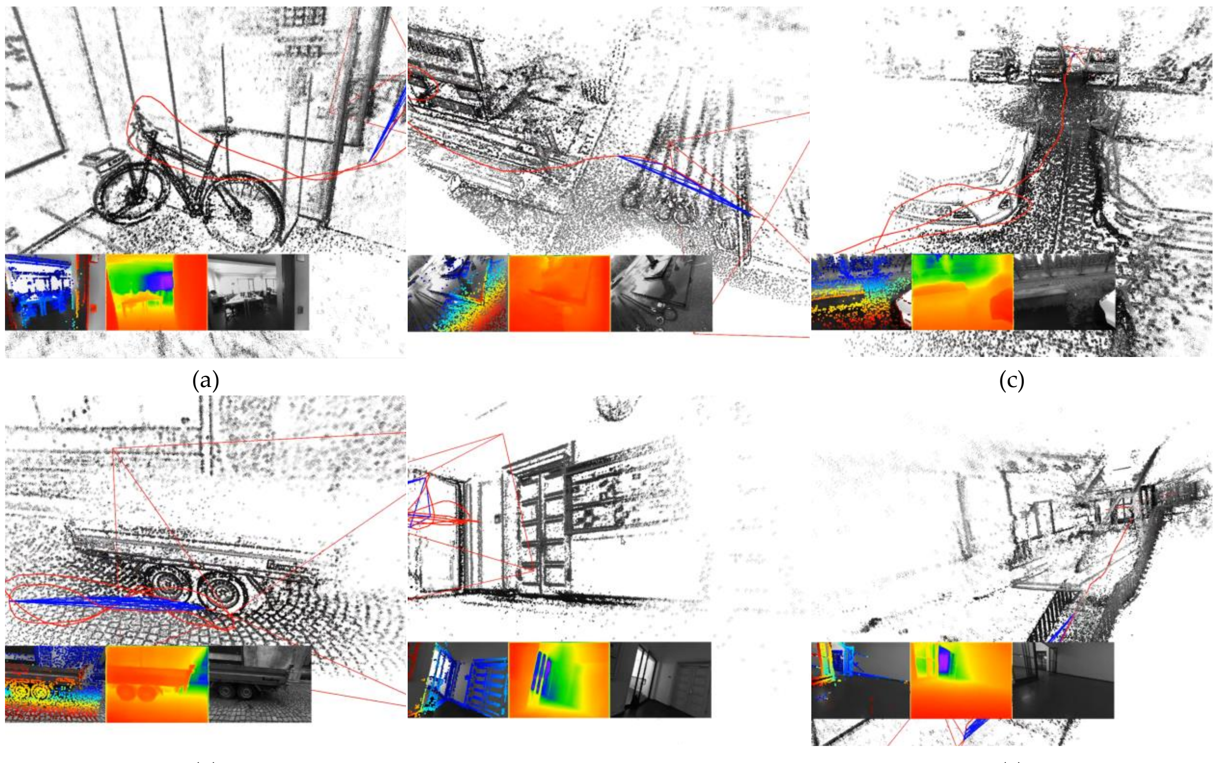


Figure 71. Depth maps generated using the DeepDSO method, with examples from sequences 01, 20, 25, 29, 38, and 40 of the TUM-Mono dataset.

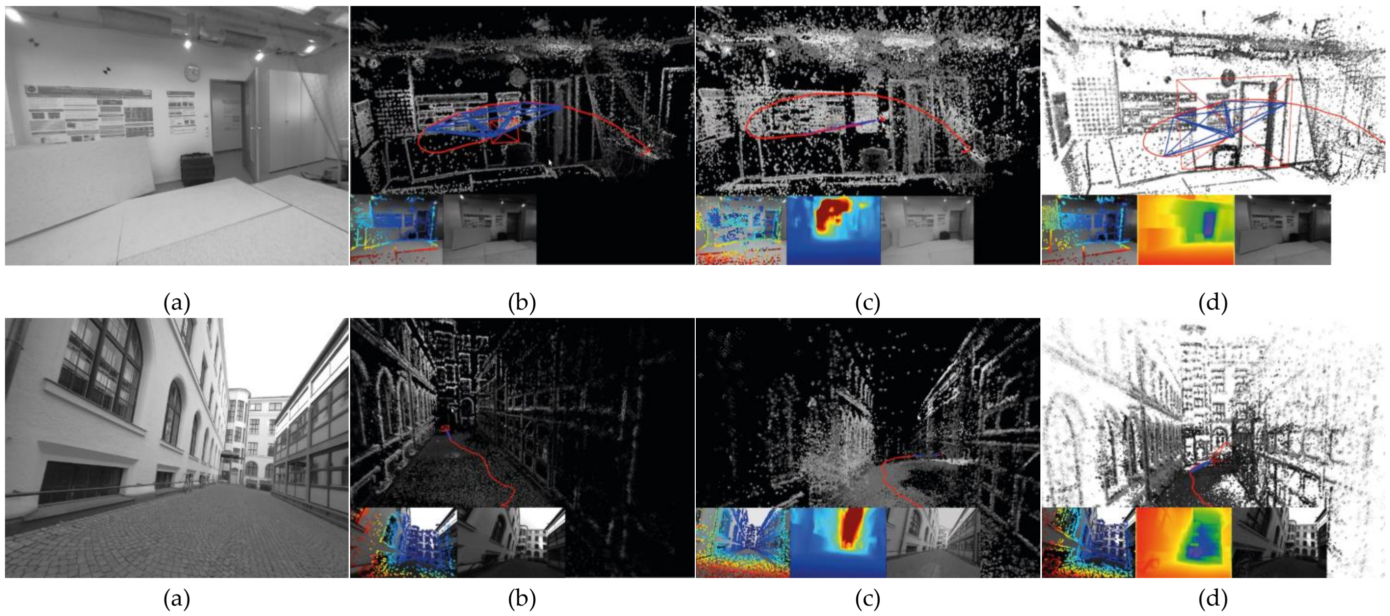


Figure 72. Visual comparisons of reconstructions obtained with the algorithms under evaluation. In this figure, a) represents the input image, while the geometric reconstructions for b) DSO, c) CNN-DSO, and d) DeepDSO methods are also presented. The top row illustrates indoor examples, and the bottom row features outdoor examples from sequences 01 and 29 of the TUM-Mono dataset [39].

As observed in Figure 71, the mapping outcomes are significantly enhanced when employing the DeepDSO algorithm. The top row, showcasing *sequence*_01 from the TUM-Mono dataset, reveals that integrating the MonoDepth CNN into CNN-DSO diminishes the mapping efficacy of DSO. This reduction in quality may stem from a deviation from the recommended procedures for point search along the epipolar line, as suggested in [54]. Furthermore, the precision of the MonoDepth CNN is notably inferior to that of NeW CRFs, resulting in a higher incidence of outliers within the CNN-DSO reconstructions. Additionally, DeepDSO achieves reconstructions that are more accurate and marginally denser, thanks to an enhanced strategy for selecting point candidates that accelerate the convergence to their authentic depths. Correspondingly, the lower row of Figure 72, depicting an outdoor scene from *sequence*_29, illustrates that DeepDSO produces reconstructions that are denser, more accurate, and properly scaled in comparison to the sparser and outlier-prone reconstructions of its predecessor, CNN-DSO. It is apparent that the depth maps generated by the MonoDepth CNN are of lesser quality than those produced by NeW CRFs, a distinction further expounded in Figure 73.

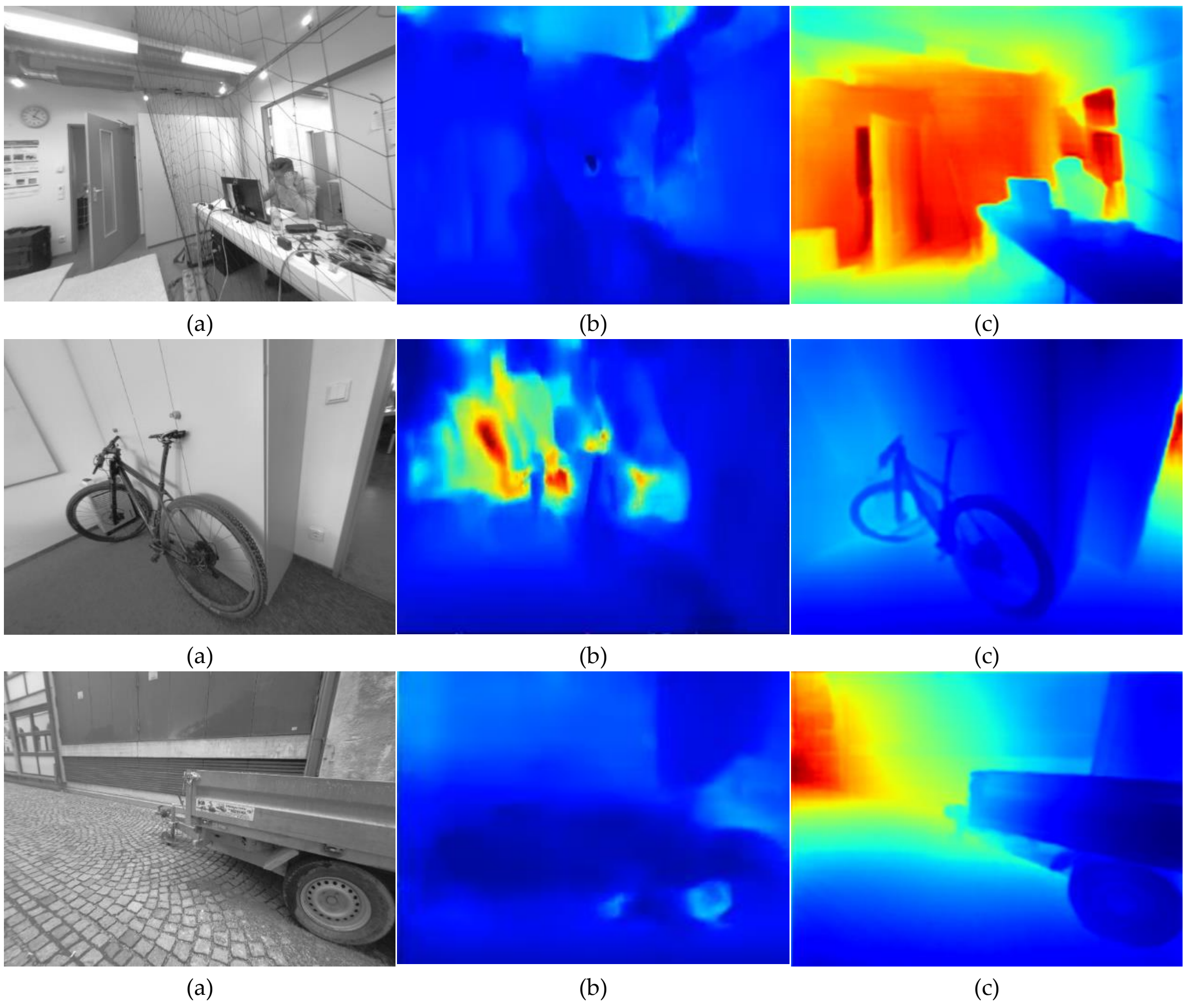


Figure 73. Comparative examples of per-pixel depth estimations derived from two distinct single image depth estimation CNNs. In this figure, part a) displays the original image provided for analysis, part b) reveals the depth estimations ascertained by the MonoDepth CNN, and part c) depicts the results obtained from the NeW CRFs CNN.

Figure 73 showcases the advancements in per-pixel depth estimation, particularly with the newer SIDE NeW CRFs, which outperforms MonoDepth by delivering more precise, detailed, consistent, and dependable depth estimations. Conversely, the examples in Figure 73b, derived from the MonoDepth SIDE, exhibit poor depth estimations, especially when applied to the TUM-Mono dataset, which possesses a distinct camera calibration from the Cityscapes Dataset [186] where MonoDepth was trained.

Despite applying a camera calibration compensation formula (equation 11) from [54], CNN-DSO still shows incompatibility with the TUM-Mono's 640×480 rectified images.

$$d_{current} = \frac{f_{current}}{f_{trained}} d_{trained} \tag{64}$$

Here, d denotes the scaled estimated depth, and f represents the focal length ratio between the current and the training dataset cameras. The preeminence of NeW CRFs over preceding SIDE methodologies is substantiated in [207].

In the realm of visual SLAM and VO system assessments, a multitude of benchmarks have emerged in recent times, such as those referenced in [67], [69], [186], [79], [103], [123]–[127], [158]. Among these, only a select few are tailored to evaluate monocular RGB systems captured using monocular cameras. Notable datasets include the KITTI dataset [69], featuring 21 sequences from an automobile perspective; the EUROC-MAV dataset [67], with 11 sequences from a quadcopter; the TUM-Mono dataset [103], encompassing over 190,000 frames from 50 sequences, both indoors and outdoors; and the ICL-NUIM dataset, which offers eight sequences across two ray-traced environments. The TUM-Mono dataset stands out for its comprehensive nature and particular suitability for sparse-direct systems due to the inclusion of exhaustive photometric calibration data for the cameras. This dataset has been the benchmark of choice for evaluating a range of contemporary monocular sparse-direct approaches, such as those found in [2], [16], [23], [180], [194].

Evaluating the efficacy of monocular RGB methods can be approached from various angles. Scene geometries, for instance, might be represented as structures, surfaces, or point clouds with varying densities. A robust and well-defined ground truth is essential to facilitate a fair comparison. However, dense and precise point clouds are often unattainable in existing benchmarks. Consequently, as suggested in [16], [68], [103], [180], trajectory estimation stands as the most reliable metric for monocular RGB methods, as the accuracy of the trajectory is indicative of the precision of the map constructed by the SLAM and VO systems. The TUM-Mono benchmark is particularly comprehensive, offering a multi-dimensional evaluation framework for purely visual systems, in contrast to the two-dimensional metrics proposed in [81], which are the Absolute Trajectory RMSE (ATE) and the Relative Pose RMSE (RPE).

The TUM-Mono benchmark introduces several metrics, including:

- **The start segment alignment error**, which aligns the experimental trajectory with the ground truth for the initial 10-20 seconds to calculate a relative transformation:

$$\boldsymbol{T}_s^{gt} := \underset{\boldsymbol{T} \in Sim(3)}{\operatorname{argmin}} \sum_{i \in S} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2 \tag{65}$$

- **The end segment alignment error,** aligning the final 10-20 seconds of the trajectory to determine the relative transformation and accumulated drift:

$$\begin{gathered} \boldsymbol{T}_e^{gt} := \underset{\boldsymbol{T} \in Sim(3)}{\operatorname{argmin}} \sum_{i \in E} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2, \\ \boldsymbol{T}_{drift} := \boldsymbol{T}_e^{gt} \left(\boldsymbol{T}_e^{gt}\right)^{-1} \end{gathered} \tag{66}$$

- **The translation error**, isolating the translational component of the accumulated drift:

$$e_t := \left\| translation(\boldsymbol{T}_{drift}) \right\| \tag{67}$$

- **The rotation error,** isolating the rotational component of the accumulated drift:

$$e_r := rotation(\boldsymbol{T}_{drift}) \tag{68}$$

- **The scale error,** isolating the scale component of the accumulated drift:

$$e_s := scale(\boldsymbol{T}_{drift}) \tag{69}$$

- **The translational RMSE,** TUM-Mono benchmark authors also introduced a metric designed to uniformly consider the effects of translation, rotation, and scaling, termed the alignment error e_{align}, which applies to both the initial and final segments of the trajectory. Furthermore, when this metric is applied to the aggregate of the initial and final segments, it corresponds to the translational Root Mean Square Error (RMSE).

$$\begin{gathered} e_{align} := \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{T}_s^{gt}\boldsymbol{p}_i - \boldsymbol{T}_e^{gt}\boldsymbol{p}_i\right\|_2^2}, \\ e_{rmse} := \sqrt{\min_{\boldsymbol{T}\in Sim(3)} \frac{1}{|S \cup E|} \sum_{i\in S\cup E} (\boldsymbol{T}\boldsymbol{p}_i - \widehat{\boldsymbol{p}}_i)^2}, \end{gathered} \tag{70}$$

where $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n \in \mathbb{R}^3$ are the estimated tracked positions for the 1 to n frames, $S \subset [1;n]$ and $E \subset [1;n]$ are the frame indices corresponding to the start- and end-segments for the ground truth positions $\widehat{\boldsymbol{p}} \in \mathbb{R}^3$. These metrics collectively facilitate a nuanced comparison of SLAM and VO systems, particularly for sparse-direct systems. The TUM-Mono benchmark's suitability for monocular evaluations is further enhanced by its monocular camera origin and ground truth registration via a monocular SLAM system. This benchmark was employed to assess the performance of our proposed DeepDSO system.

This way, DeepDSO was benchmarked against the traditional DSO and the enhanced CNN-DSO. Following the guidelines outlined in the literature [2], [16], [103], [180], we subjected each algorithm to rigorous testing across all 50 sequences of the TUM-Mono benchmark. This entailed executing each algorithm forward and in reverse ten times, culminating in 1000 trials per algorithm and a comprehensive 3000 trials for the entire evaluation. The outcomes of each trial were meticulously recorded in a text file, formatted as $\boldsymbol{P}_i = (t_i, x_i, y_i, z_i, q_{x_i}, q_{y_i}, q_{z_i}, q_{w_i})$, which documented the tracked position (x_i, y_i, z_i) alongside the associated quaternion $(q_{x_i}, q_{y_i}, q_{z_i}, q_{w_i})$ at each time stamp t_i. The trajectory data were then processed using MATLAB with the official scripts from the TUM-Mono benchmark. Figure 74 illustrates each algorithm's aggregate translation, rotation, and scale errors from 500 iterations.

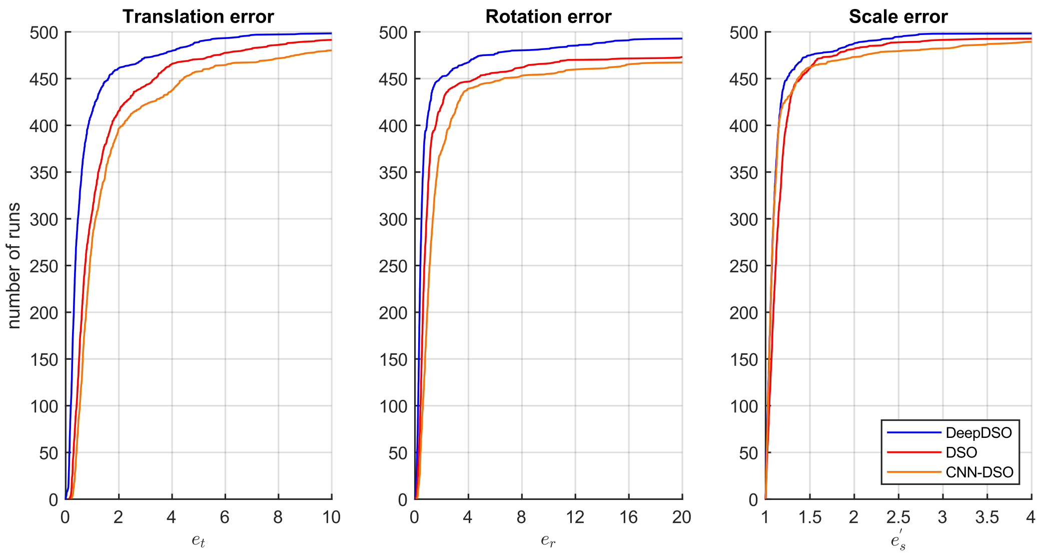


Figure 74. Cumulative errors in translation e_t, rotation e_r, and scale e_s' for the evaluated algorithms DeepDSO, DSO, and CNN-DSO.

The diagram in Figure 74 indicates the accumulated error across 500 iterations, with the y-axis denoting the number of iterations and the x-axis indicating the error magnitude. The optimal algorithms are those positioned towards the upper left quadrant of each plot. It is evident from Figure 4 that DeepDSO surpasses both DSO and CNN-DSO in terms of translation, rotation, and scale errors, implying that the integration of the NeW CRFs SIDE CNN has enhanced the DSO framework, yielding not only more accurately scaled reconstructions but also more precise trajectories and rotations. This is reflected in the diminished translation, rotation, and scale errors. Conversely, it is noteworthy that the CNN-DSO update, which incorporated the MonoDepth SIDE module, did not surpass the original DSO, accruing the highest errors in translation, rotation, and scale. Furthermore, Figure 75 presents the alignment error, synthesising these effects into a single metric for the initial and final segments.

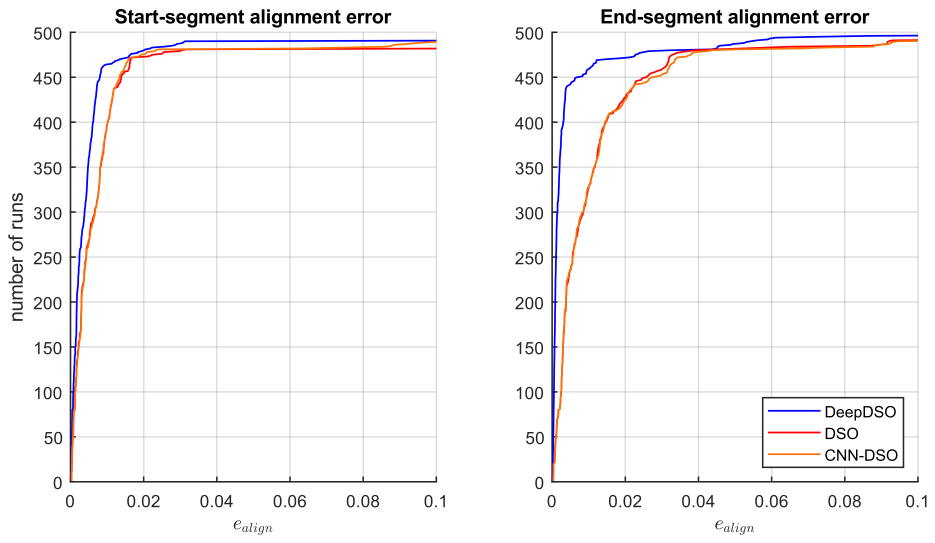


Figure 75. Initial and final segment alignment discrepancies corresponding to the Root Mean Square Error (RMSE) of the alignment error when juxtaposed with the ground truth of the initial and final segments.

Figure 75 demonstrates that DeepDSO outperformed in the alignment error metric for both the initial and final segments, signifying that the NeW CFRs SIDE CNN has substantially contributed to the DSO system, particularly during the initialization phase. This was anticipated due to its refined point initialization technique that expedites convergence to the true depth by narrowing the search range along the epipolar line for subsequent frames. Moreover, DeepDSO exhibited a significant error reduction in the final segment, suggesting that the enhanced depth initialization of points improved overall algorithm performance by diminishing drift in each pose through more precise depth and pose estimations. This is ultimately manifested in a notable drift reduction in the final segment. It is also worth mentioning that CNN-DSO showed a marginal improvement in alignment error at the start segment compared to DSO, indicating that the MonoDepth SIDE does aid in improving the DSO initialization process. However, the prevalence of outliers adversely impacted its performance throughout the trajectory, resulting in a higher end-segment alignment error. Subsequently, we examined the motion bias effect for the three algorithms by contrasting each method's performance in forward and reverse runs and assessing their collective impact. The motion bias for DeepDSO, DSO, and CNN-DSO is depicted in Figure 76.

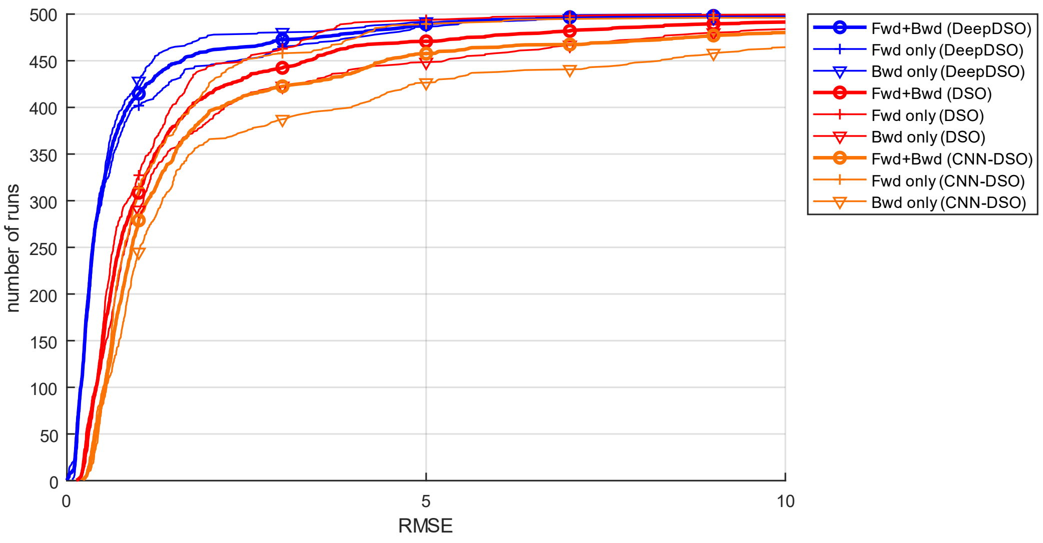


Figure 76. Motion bias in the dataset for each evaluated method; assessed by executing all sequences in both forward and reverse directions, as well as their combined effect (bold).

Motion bias, as delineated by [103], refers to the differential response of a monocular SLAM or VO system when subjected to varied environmental contexts and motion dynamics. Observations from Figure 76 elucidate that DeepDSO exhibits a reduced motion bias effect compared to other evaluated algorithms, signifying heightened robustness and dependability across diverse motion scenarios, including rapid manoeuvres, pronounced rotations, and environments with minimal textural information. This enhanced capability is likely due to the integration of a CNN within the DSO framework, which augments the system's ability to discern more accurate and scaled environmental details, thereby curtailing the incidence of outliers and imprecise estimations and, consequently, refining the overall efficacy of the DSO system. Additionally, Figure 76 reveals that DeepDSO registers a lower RMSE in reverse operation, a notable deviation from the original DSO's performance, which tends to degrade during backward movements. Figure 77 illustrates the aggregate alignment error for each system when deployed on the 50 sequences of the TUM-Mono benchmark in both forward and reverse directions, to better showcase the comparative system performances.

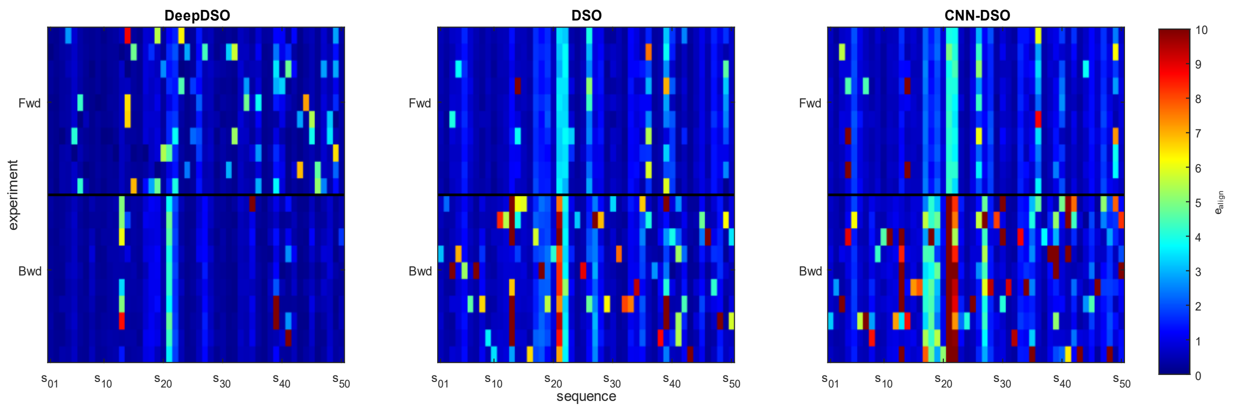


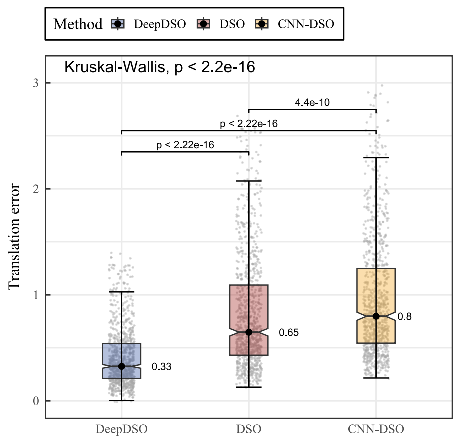
Figure 77. Color-coded alignment error e_{align} for each algorithm in the TUM-mono dataset.

In the visual representation of Figure 77, it is noticeable that the DeepDSO approach outperforms the other two methods in terms of performance across the entire dataset. Although DeepDSO incurs more errors in the forward direction (indicated by yellow, orange, and green pixels), it significantly diminishes such errors when operating in reverse, indicating a more consistent and trustworthy performance relative to DSO. While DSO shows proficiency in forward execution, it is beset by numerous instances of failure in reverse, particularly in sequences 13, 21, and 22. DeepDSO's performance is markedly superior in these instances, albeit with some minor complications in sequence 21. Conversely, CNN-DSO exhibits the most suboptimal performance, with a pronounced failure rate, especially in reverse runs, where it fails almost entirely in sequences 21 and 22.

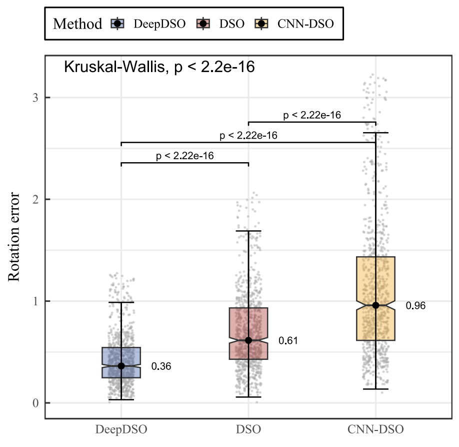
In pursuit of a comprehensive evaluation, as delineated in the referenced work [38], a thorough statistical examination of the outcomes was conducted following applying the TUM-Mono benchmark across various algorithms. This analysis commenced with the aggregation of error metrics data into a repository. This included translation error e_t, rotation error e_r, scale error e'_s, start-segment alignment error e^s_{align}, end-segment alignment error e^e_{align}, and translational RMSE e_{RMSE}. These served as the dependent variables, while the algorithmic approach and the directionality of the modality—forward and backwards—were the categorical variables under consideration.

Subsequently, the Mahalanobis distance was employed as a method for data purification, specifically for the exclusion of outliers. A threshold of 22.45774 was established, correlating with the chi-square χ^2 distribution at a confidence level of 99.999%, facilitating the identification and removal of 94 anomalous data points, culminating in a refined dataset of 2906 observations.

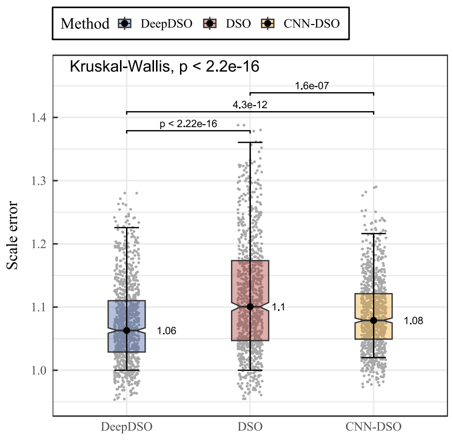
The assumptions of normality and homogeneity for each dependent variable were then scrutinized using the Kolmogorov-Smirnov and Levene's tests. Taking the translation error as an instance, the Kolmogorov-Smirnov test yielded p-values of 2.2e-16 for the algorithms DeepDSO, DSO, and CNN-DSO, leading to the rejection of the normality assumption. Similarly, a p-value of 2.2e-16 from Levene's test necessitated the dismissal of the homogeneity assumption. Analogous outcomes were observed for the remaining dependent variables, concluding that the sample was non-parametric. Consequently, the Kruskal-Wallis test was selected for the overall analysis, with the Wilcoxon signed-rank test applied for pairwise comparisons post hoc to ascertain the significance of the observed disparities. The results of this statistical analysis are visually represented in Figure 78 and numerically detailed in Table 10, which depicts the medians and Kruskal-Wallis comparisons for the cumulative errors after 1000 iterations of each algorithm. These include translation error, rotation error, scale error, alignment errors at the start and end segments, and the RMSE for the combined effect on these segments.



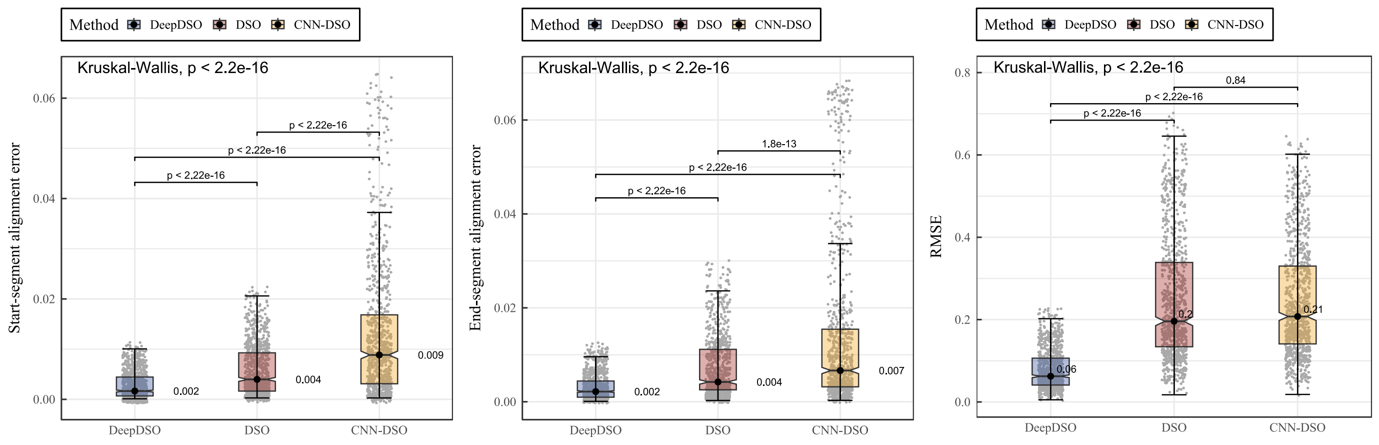
(a)



(b)



(c)



(d) (e) (f)

Figure 78. Distribution and central tendencies of cumulative errors via box plots and error bars, alongside Kruskal-Wallis tests for median values, following 1000 iterations of each algorithm. This includes a) translation error, b) rotation error, c) scale error, d) start-segment alignment error exclusively, e) end-segment alignment error exclusively, and f) the RMSE reflecting the aggregate impact on both start and end segments.

Table 10. Medians and Kruskal-Wallis comparisons for each algorithm's translation, rotation, and scale errors.

Method	**Translation error**	**Rotation error**	**Scale error**	**Start-segment Align. error**	**End-segment Align. error**	**RMSE**
Kruskal-Wallis general test	$\chi^2 = 3582.9$ $p_{val} = 2.2e-16$	$\chi^2 = 2278.4$ $p_{val} = 2.2e-16$	$\chi^2 = 2419.1$ $p_{val} = 2.2e-16$	$\chi^2 = 2419.1$ $p_{val} = 2.2e-16$	$\chi^2 = 2419.1$ $p_{val} = 2.2e-16$	$\chi^2 = 2419.1$ $p_{val} = 2.2e-16$
DeepDSO	**0.3250961**^{a}	**0.3625576**^{a}	**1.062872**^{a}	**0.001659008**^{a}	**0.002170299**^{a}	**0.06243667**^{a}
DSO	0.6472075^{b}	0.6144892^{b}	1.100516^{b}	0.003976057^{b}	0.004218454^{b}	0.19595997^{b}
CNN-DSO	0.7978954^{c}	0.9583970^{c}	1.078830^{c}	0.008847161^{c}	0.006651353^{c}	0.20758145^{b}

As evidenced in Figure 78 and Table 10, the Kruskal-Wallis tests achieved statistical significance for all the comparisons, prompting the execution of the Wilcoxon signed-rank test for pairwise post hoc analysis on each dependent variable. This analysis revealed that DeepDSO markedly surpassed both DSO and CNN-DSO regarding translation error, with an average reduction of 0.3221114 compared to its classical counterpart. In the context of rotation error, DeepDSO again demonstrated superior performance, with an average reduction of 0.2519316. The scale error metric also saw DeepDSO outperforming the others, with a reduction of 0.37644 relative to DSO, while CNN-DSO notably exceeded DSO. The start-segment alignment error metric followed a similar pattern, with DeepDSO significantly reducing error. This trend was consistent with the end-segment alignment error, where DeepDSO's performance was significantly better than the others. Lastly, in the comprehensive RMSE metric, DeepDSO continued to outshine DSO and CNN-DSO, with reductions of 0.1335233 and 0.14514478, respectively. It is noteworthy that DSO's RMSE was marginally lower than that of CNN-DSO, albeit not to a statistically significant degree.

4. Discussion

The empirical findings from the TUM-Mono benchmark, corroborated by statistical analysis, indicate that the integration of the NeW CRFs SIDE neural network module within the DSO framework markedly enhances its efficacy across various metrics, including translation, rotation, scale, alignment, and RMSE. This evidences the substantial potential of incorporating neural modules for depth estimation into traditional Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) systems, offering a significant stride towards resolving monocular SLAM and VO challenges.

It is pertinent to acknowledge that preceding endeavours, such as those referenced in [174] and inspired by the concepts in [54], have ventured to tackle these challenges. However, the experimental results from the current study reveal that the CNN-DSO adaptation fell short, with the original DSO surpassing it in rotation, translation, and both start and end-segment alignment errors. The sole metric where CNN-DSO demonstrated superiority was in scale. This suggests that while the MonoDepth module's integration into DSO facilitated scaled reconstructions, it concurrently introduced noise that adversely affected the DSO system's performance. This could be ascribed to the MonoDepth module's limited efficacy within the TUM-Mono dataset sequences and the suboptimal integration of depth prior into the DSO process.

Further, statistical analyses have confirmed that DeepDSO not only excels in scale error—implying that the CNN module effectively aids in achieving accurately scaled reconstructions—but also shows that enhanced initialisation of depth points is crucial in the context of a sparse-direct approach like DSO. This improvement significantly bolsters the tracking and mapping capabilities, as DSO relies on depth information to manage and track every point and keyframe. Such advancements are manifest in the substantial reduction of translation and rotation errors.

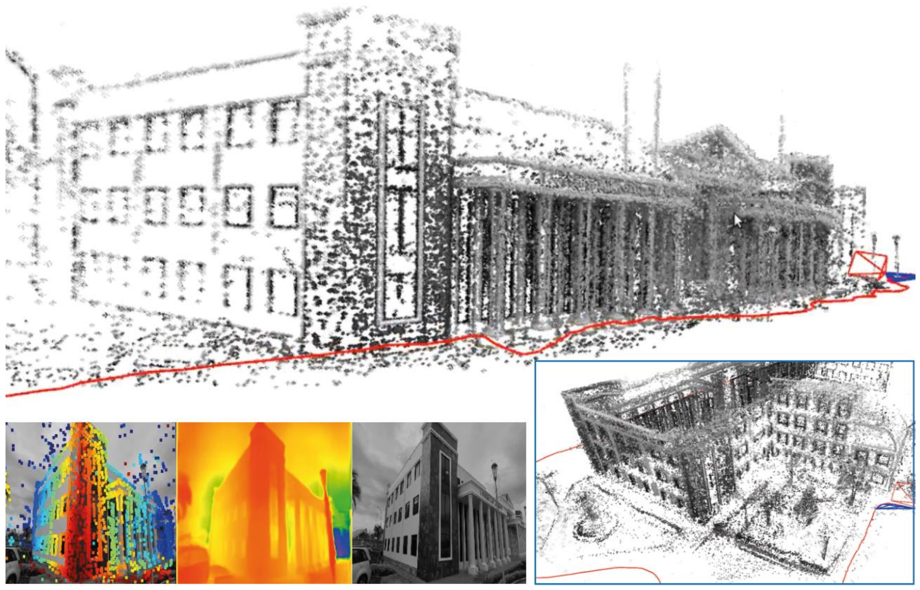


Figure 79. Three-dimensional representation of the administrative edifice of Carchi State Polytechnic University. Acquired using a standard consumer smartphone, specifically the Xiaomi Redmi Note 10 Pro, without photometric calibration. It is observable that DeepDSO can generate 3D reconstructions and trajectories utilizing commodity cameras that are not specialized for computer vision tasks, and this is achieved without the need for specialized photometric calibration.

5. Intermediate conclusions

In this work, we delineate an advancement of the established sparse-direct Visual Odometry (VO) framework, Direct Sparse Odometry (DSO), through the strategic incorporation of a cutting-edge depth estimation mechanism, the NeW CRFs single image depth estimation Convolutional Neural Network (CNN). This module is adept at introducing depth prior knowledge, which substantially diminishes the search scope along the epipolar line, a critical step in the VO process. Integrating this module into the DSO framework significantly enhances the initialization process of depth points, thereby expediting the convergence of each point towards its true depth with greater accuracy.

Upon rigorous testing, which included in excess of 3000 executions on the TUM-Mono benchmark—a freely available dataset to the scientific community—we have substantiated that our proposed methodology enhances the fidelity of 3D scene reconstructions. This is evidenced by the generation of reconstructions that are marginally denser and exhibit proper scaling—a notable improvement over traditional models. In a comparative analysis, the performance of our novel DeepDSO was meticulously evaluated against the foundational DSO and the CNN-DSO extended version that integrated the

MonoDepth CNN. The comparative results were revealing; DeepDSO demonstrated a pronounced superiority over the methodologies mentioned above across a spectrum of metrics. These metrics included rotation, translation, scale, start-segment alignment error, end-segment alignment error, and the Root Mean Square Error (RMSE). Such empirical evidence unequivocally suggests that our implementation contributes to the scaling of 3D reconstruction and plays a pivotal role in the recovery of more accurate depth maps and trajectory estimations, thereby pushing the boundaries of precision in monocular VO systems.

The implications of these findings are twofold. Firstly, they validate the efficacy of integrating CNN-based depth estimation modules into VO systems, which traditionally have not utilized machine learning to this extent. Secondly, they open up new avenues for further research into the optimization of depth initialization in VO, which could lead to even more robust and accurate navigation and mapping solutions in robotics and autonomous systems. This study, therefore, not only presents a significant enhancement in the domain of VO but also lays the groundwork for future explorations into the synergistic integration of machine learning and classical computer vision techniques.

Chapter V – Conclusions

1. The quest to accurately reconstruct 3D scenes from 2D data is an intricate and ill-posed problem that has been addressed through many methods and technological innovations. Our study has provided an in-depth examination of a particularly promising approach to 3D reconstruction: the visual reconstruction of environments using a single monocular RGB camera as the exclusive sensory input. This exploration has covered a range of input modalities, each with its unique strengths and constraints, with a concentrated focus on three core methodologies: Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO), and Structure from Motion (SFM). We have developed a comprehensive taxonomy that encapsulates the predominant configurations of systems as reported in academic research, categorizing them into three main classes that give rise to ten possible permutations within an extended taxonomy structure. Additionally, this thesis has presented a thorough review of 42 seminal monocular systems. We have identified nine critical criteria for each traditional system to facilitate the discernment and application of these systems, which are essential in deploying a 3D reconstruction system. These criteria include the algorithm type, tracking approach, map density, pixel usage, estimation technique, global optimization, relocalization capability, loop closure, and system availability. We have outlined eleven criteria for machine learning-based methods, reflecting those of the classical systems while also integrating two additional factors relevant to the neural network: the CNN architecture and the primary estimation tasks it performs. The wealth of information compiled in this study aims to assist researchers in making informed decisions when selecting an algorithm or taxonomy category that best suits their project goals. We have also articulated each category's key advantages and drawbacks within the taxonomy. An evaluation of the progression of each category within the taxonomy over the past 18 years has been undertaken, utilizing citation metrics to gauge the influence and recognition each has achieved in the research community.
2. We have also rigorously evaluated the most salient open-source monocular RGB SLAM and VO implementations, employing a structured taxonomy to discern the pros and cons of each method and classification. This analysis serves as a navigational tool for readers, guiding them to judiciously select a method that aligns with their specific requirements or to identify avenues for future enhancements to the methodologies and classifications under review. Through extensive testing, it has become evident that to facilitate a robust comparison with the state-of-the-art, monocular SLAM and VO methods must undergo assessment across more expansive datasets, encompassing a diverse array of environments, motion dynamics, and lighting conditions, as this study has shown for methods like DSM, CNN-DSO, and DF-ORB-SLAM, which reported impressive metrics but evidenced a poor performance on the TUM-Mono dataset. The sparse-direct category within our taxonomy has demonstrated superior performance, significantly outperforming the other ten methods in translation, rotation, scale, and RMSE metrics, yielding the most intricate and accurate 3D reconstructions among the evaluated methods. Following this, the sparse-indirect category showed commendable ego-motion estimation capabilities, albeit producing sparser 3D reconstructions that may not meet the criteria for certain applications, with noted deficiencies in trajectory consistency and subpar indoor performance. Moreover, the inclusion of three machine learning-based methods in our comparison—and their juxtaposition with traditional counterparts—has led us to ascertain that the integration of machine learning can markedly enhance the capabilities of SLAM or VO systems. This finding points to machine learning as a promising direction for future research aimed at transcending the limitations of existing systems. The incorporation of CNN-derived insights into the estimation process has been shown to alleviate the scale ambiguity typically associated with monocular systems, as evidenced by the significant reduction in scale error for each machine-learning method compared to their classical versions.
3. We have also articulated an enhancement to the well-established sparse-direct Visual Odometry (VO) paradigm (specifically to the Direct Sparse Odometry (DSO), which evidenced an

impressive performance among all the evaluated methods) by the judicious integration of an advanced depth estimation module, the NeW CRFs single image depth estimation Convolutional Neural Network (CNN). This addition adeptly infuses depth prior knowledge, effectively narrowing the search range along the epipolar line, a pivotal aspect of the VO methodology. The assimilation of this CNN into the DSO framework markedly improves the initialization of depth points, thus accelerating the alignment of each point to its actual depth with heightened precision. Through extensive evaluation involving over 3000 trials on the TUM-Mono benchmark—an openly accessible dataset for the academic community—we have validated that our proposed approach refines the quality of 3D scene reconstructions. The resulting reconstructions are slightly denser and properly scaled, marking a significant advancement over conventional models. In our comparative analysis, the DeepDSO, our innovative adaptation, was rigorously assessed against the original DSO and the CNN-DSO variant that incorporated the MonoDepth CNN. The outcomes of this analysis were telling; DeepDSO exhibited marked superiority over the established methods across various metrics, including rotation, translation, scale, start-segment alignment error, end-segment alignment error, and Root Mean Square Error (RMSE). This robust set of results firmly indicates that this implementation contributes to the scaling of 3D reconstructions and significantly enhances the accuracy of depth maps and trajectory predictions, thus advancing the state-of-the-art in monocular VO systems.

4. The significance of the contributions of this thesis is manifold. Firstly, they confirm the benefits of embedding CNN-based depth estimation modules within VO systems, a practice uncommon in machine learning applications. Secondly, they pave the way for continued research into the refinement of depth initialization in VO, which promises to yield more resilient and precise navigation and mapping solutions, particularly in robotics and autonomous system applications. Consequently, this research signifies a substantial improvement in the VO field and establishes a foundation for future research that will explore the collaborative fusion of machine learning with traditional computer vision techniques.

Published articles

A Comparison of Monocular Visual SLAM and Visual Odometry Methods Applied to 3D Reconstruction.

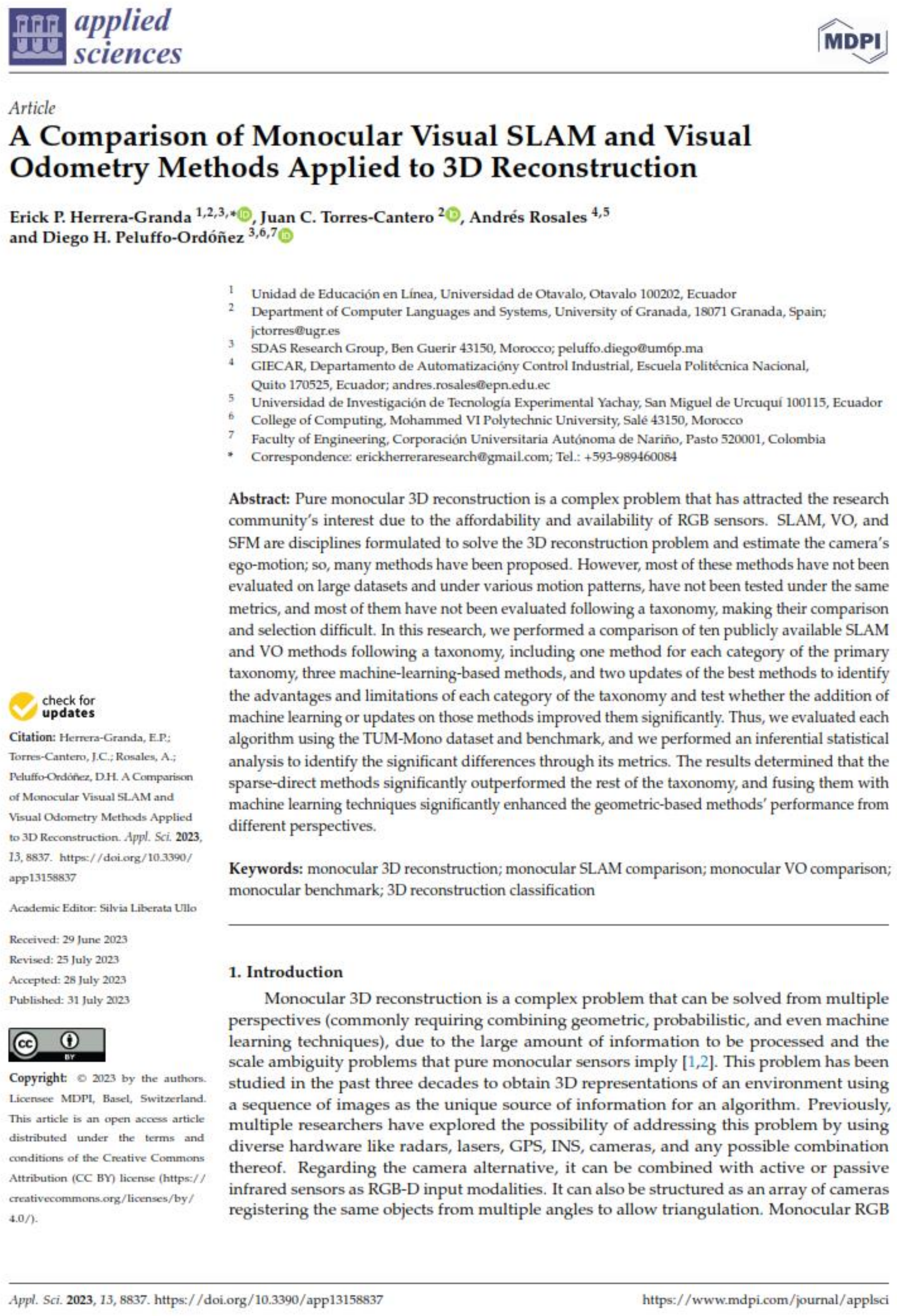
Authors: Erick P. Herrera-Granda, Juan C. Torres-Cantero, Andrés Rosales, and Diego H. Peluffo-Ordóñez. Appl. Sci. 2023, 13(15), 8837.

Status: Published

Impact Factor (JCR 2022): 2.7

Quartile: Q2

DOI: http://dx.doi.org/10.3390/app13158837



applied sciences

MDPI

Article

A Comparison of Monocular Visual SLAM and Visual Odometry Methods Applied to 3D Reconstruction

Erick P. Herrera-Granda [1,2,3,*], Juan C. Torres-Cantero [2], Andrés Rosales [4,5] and Diego H. Peluffo-Ordóñez [3,6,7]

1 Unidad de Educación en Línea, Universidad de Otavalo, Otavalo 100202, Ecuador
2 Department of Computer Languages and Systems, University of Granada, 18071 Granada, Spain; jctorres@ugr.es
3 SDAS Research Group, Ben Guerir 43150, Morocco; peluffo.diego@um6p.ma
4 GIECAR, Departamento de Automatizacióny Control Industrial, Escuela Politécnica Nacional, Quito 170525, Ecuador; andres.rosales@epn.edu.ec
5 Universidad de Investigación de Tecnología Experimental Yachay, San Miguel de Urcuquí 100115, Ecuador
6 College of Computing, Mohammed VI Polytechnic University, Salé 43150, Morocco
7 Faculty of Engineering, Corporación Universitaria Autónoma de Nariño, Pasto 520001, Colombia
* Correspondence: erickherreraresearch@gmail.com; Tel.: +593-989460084

Abstract: Pure monocular 3D reconstruction is a complex problem that has attracted the research community's interest due to the affordability and availability of RGB sensors. SLAM, VO, and SFM are disciplines formulated to solve the 3D reconstruction problem and estimate the camera's ego-motion; so, many methods have been proposed. However, most of these methods have not been evaluated on large datasets and under various motion patterns, have not been tested under the same metrics, and most of them have not been evaluated following a taxonomy, making their comparison and selection difficult. In this research, we performed a comparison of ten publicly available SLAM and VO methods following a taxonomy, including one method for each category of the primary taxonomy, three machine-learning-based methods, and two updates of the best methods to identify the advantages and limitations of each category of the taxonomy and test whether the addition of machine learning or updates on those methods improved them significantly. Thus, we evaluated each algorithm using the TUM-Mono dataset and benchmark, and we performed an inferential statistical analysis to identify the significant differences through its metrics. The results determined that the sparse-direct methods significantly outperformed the rest of the taxonomy, and fusing them with machine learning techniques significantly enhanced the geometric-based methods' performance from different perspectives.

Keywords: monocular 3D reconstruction; monocular SLAM comparison; monocular VO comparison; monocular benchmark; 3D reconstruction classification

check for updates

Citation: Herrera-Granda, E.P.; Torres-Cantero, J.C.; Rosales, A.; Peluffo-Ordóñez, D.H. A Comparison of Monocular Visual SLAM and Visual Odometry Methods Applied to 3D Reconstruction. *Appl. Sci.* **2023**, *13*, 8837. https://doi.org/10.3390/app13158837

Academic Editor: Silvia Liberata Ullo

Received: 29 June 2023
Revised: 25 July 2023
Accepted: 28 July 2023
Published: 31 July 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Monocular 3D reconstruction is a complex problem that can be solved from multiple perspectives (commonly requiring combining geometric, probabilistic, and even machine learning techniques), due to the large amount of information to be processed and the scale ambiguity problems that pure monocular sensors imply [1,2]. This problem has been studied in the past three decades to obtain 3D representations of an environment using a sequence of images as the unique source of information for an algorithm. Previously, multiple researchers have explored the possibility of addressing this problem by using diverse hardware like radars, lasers, GPS, INS, cameras, and any possible combination thereof. Regarding the camera alternative, it can be combined with active or passive infrared sensors as RGB-D input modalities. It can also be structured as an array of cameras registering the same objects from multiple angles to allow triangulation. Monocular RGB

Monocular Visual SLAM, Visual Odometry, and Structure from Motion Methods Applied to 3D Reconstruction: A Comprehensive Survey.

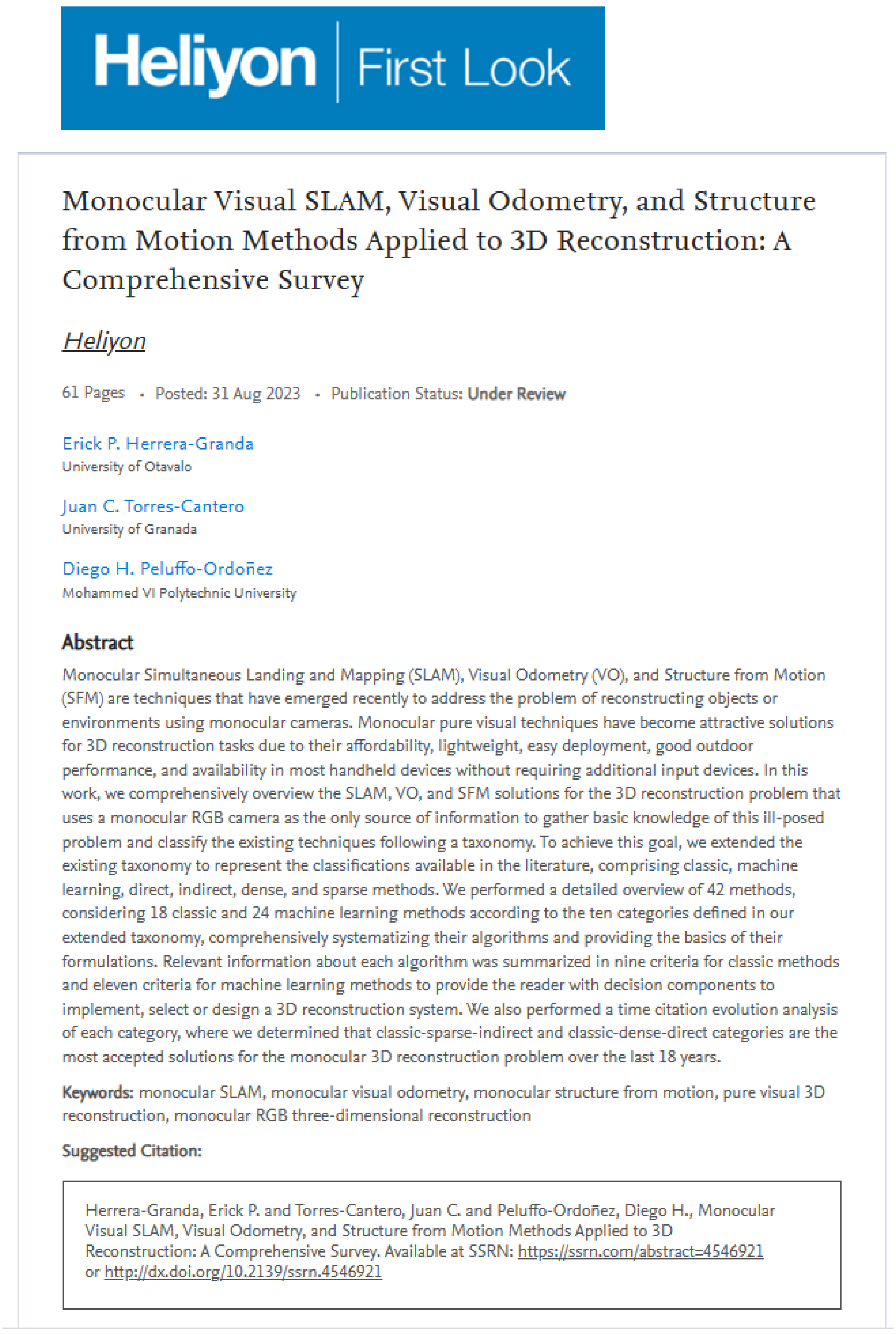
Authors: Erick P. Herrera-Granda, Juan C. Torres-Cantero, and Diego H. Peluffo-Ordóñez. Heliyon Computer Science.

Status: Preprint published and waiting for final decision.

Impact Factor (JCR 2022): 3.776

Quartile: Q1

DOI: http://dx.doi.org/10.2139/ssrn.4546921



Heliyon | First Look

Monocular Visual SLAM, Visual Odometry, and Structure from Motion Methods Applied to 3D Reconstruction: A Comprehensive Survey

Heliyon

61 Pages • Posted: 31 Aug 2023 • Publication Status: **Under Review**

Erick P. Herrera-Granda
University of Otavalo

Juan C. Torres-Cantero
University of Granada

Diego H. Peluffo-Ordoñez
Mohammed VI Polytechnic University

Abstract

Monocular Simultaneous Landing and Mapping (SLAM), Visual Odometry (VO), and Structure from Motion (SFM) are techniques that have emerged recently to address the problem of reconstructing objects or environments using monocular cameras. Monocular pure visual techniques have become attractive solutions for 3D reconstruction tasks due to their affordability, lightweight, easy deployment, good outdoor performance, and availability in most handheld devices without requiring additional input devices. In this work, we comprehensively overview the SLAM, VO, and SFM solutions for the 3D reconstruction problem that uses a monocular RGB camera as the only source of information to gather basic knowledge of this ill-posed problem and classify the existing techniques following a taxonomy. To achieve this goal, we extended the existing taxonomy to represent the classifications available in the literature, comprising classic, machine learning, direct, indirect, dense, and sparse methods. We performed a detailed overview of 42 methods, considering 18 classic and 24 machine learning methods according to the ten categories defined in our extended taxonomy, comprehensively systematizing their algorithms and providing the basics of their formulations. Relevant information about each algorithm was summarized in nine criteria for classic methods and eleven criteria for machine learning methods to provide the reader with decision components to implement, select or design a 3D reconstruction system. We also performed a time citation evolution analysis of each category, where we determined that classic-sparse-indirect and classic-dense-direct categories are the most accepted solutions for the monocular 3D reconstruction problem over the last 18 years.

Keywords: monocular SLAM, monocular visual odometry, monocular structure from motion, pure visual 3D reconstruction, monocular RGB three-dimensional reconstruction

Suggested Citation:

Herrera-Granda, Erick P. and Torres-Cantero, Juan C. and Peluffo-Ordoñez, Diego H., Monocular Visual SLAM, Visual Odometry, and Structure from Motion Methods Applied to 3D Reconstruction: A Comprehensive Survey. Available at SSRN: https://ssrn.com/abstract=4546921 or http://dx.doi.org/10.2139/ssrn.4546921

References

[1] M. Zollhöfer *et al.*, "State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 523–550, May 2018.

[2] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2017.

[3] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 6565–6574.

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct monocular SLAM," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8690 LNCS, no. PART 2, pp. 834–849.

[5] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, pp. 2609–2616.

[6] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "DeepFactors: Real-Time Probabilistic Dense Monocular SLAM," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 721–728, Apr. 2020.

[7] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[8] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2560–2568.

[9] T. Laidlow, J. Czarnowski, and S. Leutenegger, "DeepFusion: Real-time dense 3D reconstruction for monocular SLAM using single-view depth and gradient predictions," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019, vol. 2019-May, pp. 4068–4074.

[10] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, 2016, pp. 239–248.

[11] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, "Neural RGB®D sensing: Depth and uncertainty from a video camera," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 10978–10987.

[12] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep Tracking and Mapping with Convolutional Neural Networks," *Int. J. Comput. Vis.*, vol. 128, no. 3, pp. 756–769, Mar. 2020.

[13] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 15–22, Sep. 2014.

[14] H. Jin, P. Favaro, and S. Soatto, "Real-time 3-D motion and structure of point features: a front-end system for vision-based control and interaction," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, pp. 778–779, 2000.

[15] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, pp. 229–241, 2001.

[16] E. Mingachev, R. Lavrenov, E. Magid, and M. Svinin, "Comparative analysis of monocular slam algorithms using tum and euroc benchmarks," in *Smart Innovation, Systems and Technologies*, 2021, vol. 187, pp. 343–355.

[17] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2320–2327, 2011.

[18] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus 2016 51*, vol. 5, no. 1, pp. 1–26, Oct. 2016.

[19] D. Scaramuzza and F. Fraundorfer, "Tutorial: Visual odometry," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[20] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *J.*

F. Robot., vol. 23, no. 1, pp. 3–20, Jan. 2006.

[21] R. Cheng, C. Agia, D. Meger, and G. Dudek, "Depth Prediction for Monocular Direct Visual Odometry," *Proc. - 2020 17th Conf. Comput. Robot Vision, CRV 2020*, pp. 70–77, May 2020.

[22] J. Engel, J. Sturm, and D. Cremers, "Semi-dense Visual Odometry for a Monocular Camera," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1449–1456.

[23] X. Gao, R. Wang, N. Demmel, and D. Cremers, "LDSO: Direct Sparse Odometry with Loop Closure," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 2198–2204, Dec. 2018.

[24] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization:," *http://dx.doi.org/10.1177/0278364914554813*, vol. 34, no. 3, pp. 314–334, Dec. 2014.

[25] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers, "Omnidirectional DSO: Direct Sparse Odometry With Fisheye Cameras," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3693–3700, 2018.

[26] D. Schubert, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "Direct Sparse Odometry with Rolling Shutter," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11212 LNCS, pp. 699–714, Sep. 2018.

[27] J. Sun, Y. Wang, and Y. Shen, "Fully Scaled Monocular Direct Sparse Odometry with A Distance Constraint," *2019 5th Int. Conf. Control. Autom. Robot. ICCAR 2019*, pp. 271–275, Apr. 2019.

[28] C. Zhao, Y. Tang, Q. Sun, and A. V. Vasilakos, "Deep Direct Visual Odometry," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7733–7742, Jul. 2022.

[29] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, "Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking," *J. Sensors*, vol. 2021, p. 2054828, 2021.

[30] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, p. 16, 2017.

[31] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[32] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[34] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[35] G. Chahine and C. Pradalier, "Survey of Monocular SLAM Algorithms in Natural Environments," in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 345–352.

[36] Y. Chen, Y. Zhou, Q. Lv, and K. K. Deveerasetty, "A Review of V-SLAM," in *2018 IEEE International Conference on Information and Automation (ICIA)*, 2018, pp. 603–608.

[37] M. He, C. Zhu, Q. Huang, B. Ren, and J. Liu, "A review of monocular visual odometry," *Vis. Comput.*, vol. 36, no. 5, pp. 1053–1065, 2020.

[38] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, "A Comprehensive Survey of Visual SLAM Algorithms," *Robotics*, vol. 11, no. 1, 2022.

[39] I. Krešo, M. Ševrović, and S. Šegvić, "A Novel Georeferenced Dataset for Stereo Visual Odometry." 2013.

[40] N. Suenderhauf and P. Protzel, *Stereo odometry - A review of approaches (Technical Report 3/07)*. Germany: Chemnitz University of Technology, 2007.

[41] D. Valiente García, L. Fernández Rojo, A. Gil Aparicio, L. Payá Castelló, and O. Reinoso García, "Visual Odometry through Appearance- and Feature-Based Method with Omnidirectional Images," *J. Robot.*, vol. 2012, pp. 1–13, 2012.

[42] X. Ke, F. Huang, Y. Zhang, Z. Tu, and W. Song, "3D Scene Localization and Mapping Based on Omnidirectional {SLAM}," *{IOP} Conf. Ser. Earth Environ. Sci.*, vol. 783, no. 1, p. 12143, May 2021.

[43] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1015–1026, 2008.

[44] K. Tateno, N. Navab, and F. Tombari, "Distortion-Aware Convolutional Filters for Dense Prediction in Panoramic Images," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11220 LNCS, pp. 732–750, 2018.

[45] A. Cumani, "Feature Localization Refinement for Improved Visual Odometry Accuracy," *Int J Circuits Syst Signal Process*, vol. 5, no. 2, pp. 151–158, 2010.

[46] B. Kitt, J. Rehder, A. Chambers, M. Schönbein, H. Lategahn, and S. Singh, "Monocular Visual Odometry using a Planar Road Model to Solve Scale Ambiguity," in *ECMR*, 2011.

[47] "VOSviewer - Visualizing scientific landscapes." [Online]. Available: https://www.vosviewer.com/. [Accessed: 17-Sep-2022].

[48] G. Li, L. Yu, and S. Fei, "A deep-learning real-time visual SLAM system based on multi-task feature extraction network and self-supervised feature points," *Measurement*, vol. 168, p. 108403, 2021.

[49] C. Tang and P. Tan, "BA-Net: Dense Bundle Adjustment Network," *7th Int. Conf. Learn. Represent. ICLR 2019*, Jun. 2019.

[50] L. Sun, W. Yin, E. Xie, Z. Li, C. Sun, and C. Shen, "Improving Monocular Visual Odometry Using Learned Depth," *IEEE Trans. Robot.*, pp. 1–14, 2022.

[51] R. Gonzalez, F. Rodriguez, J. L. Guzman, C. Pradalier, and R. Siegwart, "Combined visual odometry and visual compass for off-road mobile robots localization," *Robotica*, vol. 30, no. 6, pp. 865–878, 2012.

[52] N. Nourani-Vatani and P. V. K. Borges, "Correlation-based visual odometry for ground vehicles," *J. F. Robot.*, vol. 28, no. 5, pp. 742–768, Sep. 2011.

[53] F. Cheng, C. Liu, H. Wu, and M. Ai, "DIRECT SPARSE VISUAL ODOMETRY with STRUCTURAL REGULARITIES for LONG CORRIDOR ENVIRONMENTS," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch.*, vol. 43, no. B2, pp. 757–763, Aug. 2020.

[54] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, "CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 5218–5223, May 2019.

[55] A. J. Davison, "Mobile Robot Navigation Using Active Vision," Univ. of Oxford, 1998.

[56] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *Computer Vision --- ECCV'98*, 1998, pp. 809–825.

[57] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, 2002.

[58] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Science and Technology Proceedings - IEEE 2009 International Symposium on Mixed and Augmented Reality, ISMAR 2009*, 2009, pp. 83–86.

[59] B. Williams, G. Klein, and I. Reid, "Real-time SLAM relocalisation," *Proc. IEEE Int. Conf. Comput. Vis.*, 2007.

[60] P. Moulon, P. Monasse, R. Marlet, and Others, "OpenMVG: An Open Multiple View Geometry library," 2013.

[61] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, "OpenMVG: Open Multiple View Geometry," in *Reproducible Research in Pattern Recognition*, 2017, pp. 60–74.

[62] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, 2012.

[63] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," in *Robotics: Science and Systems*, 2011, vol. 6, pp. 73–80.

[64] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2352–2359, 2011.

[65] C. Mei, G. Sibley, and P. Newman, "Closing loops without places," *IEEE/RSJ 2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc.*, pp. 3738–3744, 2010.

[66] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[67] M. Burri *et al.*, "The EuRoC micro aerial vehicle datasets:,"

https://doi.org/10.1177/0278364915620033, vol. 35, no. 10, pp. 1157–1163, Jan. 2016.

[68] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[69] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[70] R. Mur-Artal and J. D. Tardós, "Visual-Inertial Monocular SLAM With Map Reuse," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 796–803, 2017.

[71] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[72] C. Campos, J. M. M. Montiel, and J. D. Tardós, "Inertial-Only Optimization for Visual-Inertial Initialization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 51–57.

[73] R. Elvira, J. D. Tardós, and J. M. M. Montiel, "ORBSLAM-Atlas: a robust and accurate multi-map system," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6253–6259.

[74] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert, "Dense versus Sparse Approaches for Estimating the Fundamental Matrix," *Int. J. Comput. Vis. 2011 962*, vol. 96, no. 2, pp. 212–234, Jun. 2011.

[75] O. Chum, J. Matas, and J. Kittler, "Locally Optimized RANSAC," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2781, pp. 236–243, 2003.

[76] O. Chum, T. Werner, and J. Matas, "Two-view geometry estimation unaffected by a dominant plane," *Proc. - 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, CVPR 2005*, vol. I, pp. 772–779, 2005.

[77] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense Monocular Depth Estimation in Complex Dynamic Scenes," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 4058–4066, Dec. 2016.

[78] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," *Mult. View Geom. Comput. Vis.*, Mar. 2004.

[79] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, 2012, pp. 611–625.

[80] J. Stühmer, S. Gumhold, and D. Cremers, "Real-Time Dense Geometry from a Handheld Camera," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6376 LNCS, pp. 11–20, 2010.

[81] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image Vis. Comput.*, vol. 29, no. 7, pp. 434–441, Jun. 2011.

[82] X. Bresson, S. Esedoḡlu, P. Vandergheynst, J.-P. Thiran, and S. Osher, "Fast Global Minimization of the Active Contour/Snake Model," *J. Math. Imaging Vis. 2007 282*, vol. 28, no. 2, pp. 151–167, Jul. 2007.

[83] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance:," *http://dx.doi.org/10.1177/0278364908090961*, vol. 27, no. 6, pp. 647–665, Jun. 2008.

[84] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel, "Direct Sparse Mapping," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1363–1370, 2020.

[85] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017.

[86] C. Liu, J. Zhao, N. Sun, Q. Yang, and L. Wang, "IT-SVO: Improved Semi-Direct Monocular Visual Odometry Combined with JS Divergence in Restricted Mobile Devices," *Sensors*, vol. 21, no. 6, 2021.

[87] P. Favaro, H. Jin, and S. Soatto, “A semi-direct approach to structure from motion,” in *Proceedings 11th International Conference on Image Analysis and Processing*, 2001, pp. 250–255.

[88] C. Mei, S. Benhimane, E. Malis, and P. Rives, “Efficient Homography-Based Tracking and 3-D Reconstruction for Single-Viewpoint Sensors,” *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1352–1364, 2008.

[89] A. J. Davison, “SceneLib 1.0,” 2006. [Online]. Available: https://www.doc.ic.ac.uk/~ajd/Scene/index.html.

[90] R. Castle, “PTAM-GPL: Parallel Tracking and Mapping,” *GitHub repository*, 2013. [Online]. Available: https://github.com/Oxford-PTAM/PTAM-GPL.

[91] P. Moulon, “OpenMVG (open Multiple View Geometry),” *GitHub repository*, 2013. [Online]. Available: https://github.com/openMVG/openMVG/tree/v2.0.

[92] R. Mur-Artal, “ORB-SLAM Monocular,” *GitHub repository*, 2016. [Online]. Available: https://github.com/raulmur/ORB_SLAM.

[93] J. Schönberger, “COLMAP,” *GitHub repository*, 2016. [Online]. Available: https://github.com/colmap/colmap.

[94] R. Mur-Artal, “ORB-SLAM2,” *GitHub repository*, 2017. [Online]. Available: https://github.com/raulmur/ORB_SLAM2.

[95] J. D. Tardós, “ORB-SLAM3,” *GitHub repository*, 2021. [Online]. Available: https://github.com/UZ-SLAMLab/ORB_SLAM3.

[96] P. Foster, “OpenDTAM,” *GitHub repository*, 2016. [Online]. Available: https://github.com/anuranbaka/OpenDTAM.

[97] M. Pizzoli, “REMODE,” *GitHub repository*, 2015. [Online]. Available: https://github.com/uzh-rpg/rpg_open_remode.

[98] J. Engel, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” *GitHub repository*, 2014. [Online]. Available: https://github.com/tum-vision/lsd_slam.

[99] J. Engel, “DSO: Direct Sparse Odometry,” *GitHub repository*, 2017. [Online]. Available: https://github.com/JakobEngel/dso.

[100] N. Demmel, G. Xiang, and U. Erkam, “LDSO: Direct Sparse Odometry with Loop Closure,” *GitHub repository*, 2020. [Online]. Available: https://github.com/tum-vision/LDSO.

[101] J. Zubizarreta, “DSM: Direct Sparse Mapping,” *GitHub repository*, 2021. [Online]. Available: https://github.com/jzubizarreta/dsm.

[102] C. Forster, “Semi-direct monocular visual odometry,” *GitHub repository*, 2017. [Online]. Available: https://github.com/uzh-rpg/rpg_svo.

[103] J. Engel, V. Usenko, and D. Cremers, “A Photometrically Calibrated Benchmark For Monocular Visual Odometry,” Jul. 2016.

[104] D. DeTone, T. Malisiewicz, and A. Rabinovich, “SuperPoint: Self-Supervised Interest Point Detection and Description,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 337–33712.

[105] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, “MVSNet: Depth Inference for Unstructured Multi-view Stereo,” in *Computer Vision -- ECCV 2018*, 2018, pp. 785–801.

[106] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, “Working hard to know your neighbor’s margins: Local descriptor learning loss.” arXiv, 2017.

[107] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “LF-Net: Learning Local Features from Images,” in *Advances in Neural Information Processing Systems*, 2018, vol. 31.

[108] J. Kopf, X. Rong, and J.-B. Huang, “Robust Consistent Video Depth Estimation,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1611–1621.

[109] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, “Consistent Video Depth Estimation.” arXiv, 2020.

[110] E. Sucar, K. Wada, and A. Davison, “NodeSLAM: Neural Object Descriptors for Multi-View Shape Reconstruction,” in *2020 International Conference on 3D Vision (3DV)*, 2020, pp. 949–958.

[111] N. Yang, R. Wang, J. Stückler, and D. Cremers, “Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry.” arXiv, 2018.

[112] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1278–1289.

[113] Z. Teed and J. Deng, "DeepV2D: Video to Depth with Differentiable Structure from Motion." arXiv, 2020.

[114] B. Ummenhofer *et al.*, "DeMoN: Depth and motion network for learning monocular stereo," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 5622–5631, Nov. 2017.

[115] W. Wang, Y. Hu, and S. Scherer, "TartanVO: A Generalizable Learning-based VO," *4th Conf. Robot Learn. (CoRL 2020)*, 2020.

[116] C. Yang, Q. Chen, Y. Yang, J. Zhang, M. Wu, and K. Mei, "SDF-SLAM: A Deep Learning Based Highly Accurate SLAM Using Monocular Camera Aiming at Indoor Map Reconstruction With Semantic and Depth Fusion," *IEEE Access*, vol. 10, pp. 10259–10272, 2022.

[117] A. Steenbeek and F. Nex, "CNN-Based Dense Monocular Visual SLAM for Real-Time UAV Exploration in Emergency Conditions," *Drones*, vol. 6, no. 3, 2022.

[118] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4076–4083, 2018.

[119] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[120] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision -- ECCV 2014*, 2014, pp. 740–755.

[121] L. Madhuanand, F. Nex, and M. Y. Yang, "Self-supervised monocular depth estimation from oblique UAV videos," *ISPRS J. Photogramm. Remote Sens.*, vol. 176, pp. 1–14, 2021.

[122] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Computer Vision -- ECCV 2012*, 2012, pp. 746–760.

[123] W. Yin, Y. Liu, and C. Shen, "Virtual Normal: Enforcing Geometric Constraints for Accurate and Robust Depth Prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, p. 1, 2021.

[124] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling Task Transfer Learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3712–3722.

[125] J. Cho, D. Min, Y. Kim, and K. Sohn, "A Large RGB-D Dataset for Semi-supervised Monocular Depth Estimation." arXiv, 2019.

[126] X. Huang *et al.*, "The ApolloScape Dataset for Autonomous Driving," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1067–10676.

[127] K. Xian *et al.*, "Monocular Relative Depth Perception with Web Stereo Data Supervision," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 311–320.

[128] J. Lee, M. Back, S. S. Hwang, and I. Y. Chun, "Improved Real-Time Monocular SLAM Using Semantic Segmentation on Selective Frames," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 2800–2813, 2023.

[129] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, 2018.

[130] R. Lang, Y. Fan, and Q. Chang, "SVR-Net: A Sparse Voxelized Recurrent Network for Robust Monocular SLAM with Direct TSDF Mapping," *Sensors*, vol. 23, no. 8, 2023.

[131] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[132] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1647–1655.

[133] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *2018 IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, 2018, pp. 8934–8943.

[134] A. Dosovitskiy *et al.*, "FlowNet: Learning Optical Flow with Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.

[135] Z. Min and E. Dunn, "VOLDOR-SLAM: For the Times When Feature-Based or Direct Methods Are Not Good Enough," *CoRR*, vol. abs/2104.0, 2021.

[136] Z. Teed and J. Deng, "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras." arXiv, 2021.

[137] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2020.

[138] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields." 2022.

[139] A. Rosinol, J. J. Leonard, and L. Carlone, "Probabilistic Volumetric Fusion for Dense Monocular SLAM," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 3096–3104.

[140] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep Tracking and Mapping," in *Computer Vision -- ECCV 2018*, 2018, pp. 851–868.

[141] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632.

[142] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic Scene Completion from a Single Depth Image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 190–198.

[143] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9351, pp. 234–241, 2015.

[144] Z. Devito *et al.*, "Opt: A Domain Specific Language for Non-Linear Least Squares Optimization in Graphics and Imaging," *ACM Trans. Graph.*, vol. 36, no. 5, Oct. 2017.

[145] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6612–6619.

[146] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes." arXiv, 2013.

[147] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, "MonoRec: Semi-Supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6108–6118.

[148] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in Monocular Visual Odometry: Photometric Calibration, Motion Bias, and Rolling Shutter Effect," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2878–2885, 2018.

[149] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3923–3931.

[150] N. Mayer *et al.*, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016.

[151] C. Godard, O. Mac Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 3827–3837, Oct. 2019.

[152] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency." 2017.

[153] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[154] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "DeepMVS: Learning Multi-view Stereopsis," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2821–2830.

[155] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," in *Advances in Neural Information Processing Systems*, 2014, vol. 27.

[156] Z. Min, "VOLDOR: Visual Odometry from Log-logistic Dense Optical flow Residual," *GitHub repository*, 2020. [Online]. Available: https://github.com/htkseason/VOLDOR.

[157] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep Tracking and Mapping BT - Computer Vision – ECCV 2018," 2018, pp. 851–868.

[158] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1524–1531.

[159] B. Bescos, "DynaSLAM," *GitHub repository*, 2019. [Online]. Available: https://github.com/BertaBescos/DynaSLAM.

[160] C. Tang, "BA-Net: Dense Bundle Adjustment Network," *GitHub repository*, 2020. [Online]. Available: https://github.com/frobelbest/BANet.

[161] A. Steenbeek, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image," *GitHub repository*, 2022. [Online]. Available: https://github.com/annesteenbeek/sparse-to-dense-ros.

[162] B. Ummenhofer, "DeMoN: Depth and Motion Network," *GitHub repository*, 2022. [Online]. Available: https://github.com/lmb-freiburg/demon.

[163] Z. Teed and J. Deng, "DeepV2D," *GitHub repository*, 2020. [Online]. Available: https://github.com/princeton-vl/DeepV2D.

[164] Z. Teed and J. Deng, "DROID-SLAM," *GitHub repository*, 2022. [Online]. Available: https://github.com/princeton-vl/DROID-SLAM.

[165] A. Rosinol, "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields," *GitHub repository*, 2022. [Online]. Available: https://github.com/ToniRV/NeRF-SLAM.

[166] A. Sundar, "CNN-SLAM," *GitHub repository*, 2018. [Online]. Available: https://github.com/iitmcvg/CNN_SLAM.

[167] H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM," *GitHub repository*, 2019. [Online]. Available: https://github.com/lmb-freiburg/deeptam.

[168] S. Troscot, "CodeSLAM," *GitHub repository*, 2022. [Online]. Available: https://github.com/silviutroscot/CodeSLAM.

[169] J. Czarnowski and M. Kaneko, "DeepFactors," *GitHub repository*, 2020. [Online]. Available: https://github.com/jczarnowski/DeepFactors.

[170] S. Zhang, "DVSO: Deep Virtual Stereo Odometry," *GitHub repository*, 2022. [Online]. Available: https://github.com/SenZHANG-GitHub/dvso.

[171] R. Cheng, "CNN-DVO," *McGill repository*. McGill, 2020.

[172] F. Wimbauer and N. Yang, "MonoRec," *GitHub repository*, 2017. [Online]. Available: https://github.com/Brummi/MonoRec.

[173] S. Y. Loo, "CNN-SVO," *GitHub repository*, 2019. [Online]. Available: https://github.com/yan99033/CNN-SVO.

[174] Muskie, "CNN-DSO: A combination of Direct Sparse Odometry and CNN Depth Prediction," *GitHub repository*, 2019. [Online]. Available: https://github.com/muskie82/CNN-DSO.

[175] L. Cui and C. Ma, "SDF-SLAM: Semantic Depth Filter SLAM for Dynamic Environments," *IEEE Access*, vol. 8, pp. 95301–95311, 2020.

[176] R. A. Güler, N. Neverova, and I. Kokkinos, "DensePose: Dense Human Pose Estimation In The Wild."

[177] S. J. Lee, H. Choi, and S. S. Hwang, "Real-time Depth Estimation Using Recurrent CNN with Sparse Depth Cues for SLAM System," *Int. J. Control. Autom. Syst.*, vol. 18, no. 1, pp. 206–216, 2020.

[178] M. F. Aslan, A. Durdu, A. Yusefi, K. Sabanci, and C. Sungur, "A Tutorial: Mobile Robotics, SLAM, Bayesian Filter, Keyframe Bundle Adjustment and ROS Applications," in *Robot Operating System (ROS): The Complete Reference (Volume 6)*, A. Koubaa, Ed. Cham: Springer International Publishing, 2021, pp. 227–269.

[179] E. P. Herrera-Granda, "An Extended Taxonomy for Monocular Visual SLAM, Visual Odometry, and Structure from Motion methods applied to 3D Reconstruction," *GitHub repository*, 2023. [Online]. Available: https://github.com/erickherreraresearch/TaxonomyPureVisualMonocularSLAM/.

[180] E. Mingachev *et al.*, "Comparison of ROS-Based Monocular Visual SLAM Methods: DSO, LDSO, ORB-SLAM2 and DynaSLAM," in *Interactive Collaborative Robotics*, 2020, pp. 222–233.

[181] S. Wang, "DF-ORB-SLAM," *GitHub repository*, 2020. [Online]. Available: https://github.com/834810269/DF-ORB-SLAM.

[182] S. Y. Loo, "MonoDepth CPP," 2021. [Online]. Available: https://github.com/yan99033/monodepth-cpp.

[183] M. Gurturk, A. Yusefi, M. F. Aslan, M. Soycan, A. Durdu, and A. Masiero, "The YTU dataset and recurrent neural network based visual-inertial odometry," *Measurement*, vol. 184, p. 109878, 2021.

[184] T. Chen, F. Pu, H. Chen, and Z. Liu, "WHUVID: A Large-Scale Stereo-IMU Dataset for Visual-Inertial Odometry and Autonomous Driving in Chinese Urban Scenarios," *Remote Sens.*, vol. 14, no. 9, 2022.

[185] A. Wong, X. Fei, S. Tsuei, and S. Soatto, "Unsupervised Depth Completion from Visual Inertial Odometry." 2021.

[186] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[187] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr. 1987.

[188] S. Sarabandi and F. Thomas, "Accurate Computation of Quaternions from Rotation Matrices," in *Advances in Robot Kinematics 2018*, 2019, pp. 39–46.

[189] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Mach. Vis. Appl.*, vol. 13, no. 1, pp. 14–24, 2001.

[190] ROS.org, "ROS Camera Calibration," 2020. [Online]. Available: http://wiki.ros.org/camera_calibration. [Accessed: 26-Dec-2022].

[191] Open Source Computer Vision.org, "Camera calibration with OpenCV," 2019. [Online]. Available: https://docs.opencv.org/4.1.1/d4/d94/tutorial_camera_calibration.html. [Accessed: 26-Dec-2022].

[192] H. Ghorbani, "MAHALANOBIS DISTANCE AND ITS APPLICATION FOR DETECTING MULTIVARIATE OUTLIERS," *FACTA Univ. Ser. Math. INFORMATICS*, vol. 34, pp. 583–595, 2019.

[193] E. P. Herrera-Granda, J. C. Torres-Cantero, and D. H. Peluffo-Ordoñez, "Monocular Visual SLAM, Visual Odometry, and Structure from Motion Methods Applied to 3D Reconstruction: A Comprehensive Survey," *Heliyon - First Look*, vol. 8, no. 23, pp. 1–61, 2023.

[194] E. P. Herrera-Granda, J. C. Torres-Cantero, A. Rosales, and D. H. Peluffo-Ordóñez, "A Comparison of Monocular Visual SLAM and Visual Odometry Methods Applied to 3D Reconstruction," *Appl. Sci.*, vol. 13, no. 15, 2023.

[195] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," *CoRR*, vol. abs/1609.0, 2016.

[196] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning Depth from Single Monocular Images," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, 2005, pp. 1161–1168.

[197] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D Scene Structure from a Single Still Image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, 2009.

[198] X. Wang, C. Hou, L. Pu, and Y. Hou, "A depth estimating method from a single image using FoE CRF," *Multimed. Tools Appl.*, vol. 74, no. 21, pp. 9491–9506, 2015.

[199] S. Aich, J. M. Uwabeza Vianney, M. Amirul Islam, and M. K. Bingbing Liu, "Bidirectional Attention Network for Monocular Depth Estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11746–11752.

[200] L. Huynh, P. Nguyen-Ha, J. Matas, E. Rahtu, and J. Heikkilä, “Guiding Monocular Depth Estimation Using Depth-Attention Volume,” in *Computer Vision -- ECCV 2020*, 2020, pp. 581–597.

[201] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, “From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation.” arXiv, 2019.

[202] S. Lee, J. Lee, B. Kim, E. Yi, and J. Kim, “Patch-Wise Attention Network for Monocular Depth Estimation,” *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 3, pp. 1873–1881, May 2021.

[203] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia, “GeoNet: Geometric Neural Network for Joint Depth and Surface Normal Estimation,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 283–291.

[204] V. Guizilini, R. Ambruş, W. Burgard, and A. Gaidon, “Sparse Auxiliary Networks for Unified Monocular Depth Prediction and Completion,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11073–11083.

[205] M. Ochs, A. Kretz, and R. Mester, “SDNet: Semantically Guided Depth Estimation Network,” in *Pattern Recognition*, 2019, pp. 288–302.

[206] S. Qiao, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, “ViP-DeepLab: Learning Visual Perception with Depth-aware Video Panoptic Segmentation,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3996–4007.

[207] W. Yuan, X. Gu, Z. Dai, S. Zhu, and P. Tan, “Neural Window Fully-connected CRFs for Monocular Depth Estimation,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 3906–3915.

[208] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, “Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1119–1127.

[209] Y. Hua and H. Tian, “Depth estimation with convolutional conditional random field network,” *Neurocomputing*, vol. 214, pp. 546–554, 2016.

[210] Z. Liu *et al.*, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10002.

[211] S. Lovengrove, “Pangolin,” *GitHub repository*, 2016. [Online]. Available: https://github.com/stevenlovegrove/Pangolin/tree/v0.5.

[212] S.-Y. Loo, “MonoDepth-cpp,” *GitHub repository*, 2021. [Online]. Available: https://github.com/yan99033/monodepth-cpp.