Universidad de Granada

Department of Computer Architecture
and Technology

Arquitectura eficiente de condensación de información visual dirigida por procesos atencionales

Efficient architecture to condensate visual information driven by attention processes

PhD Thesis Dissertation

María Sara Granados Cabeza

Granada, October 2012

Editor: Editorial de la Universidad de Granada
Autor: María Sara Granados Cabeza
D.L.: GR 1055-2013
ISBN: 978-84-9028-492-6

Universidad de Granada

Efficient architecture to condensate visual information driven by attention processes

Arquitectura eficiente de condensación de información visual dirigida por procesos atencionales

Presented By

María Sara Granados Cabeza

To apply for the

International PhD Degree in Computer and Network Engineering

October 2012

ADVISORS

Javier Díaz Alonso

Sonia Mota Fernández

Alberto Prieto Espinosa

D. Javier Díaz Alonso, D^{a}. Sonia Mota Fernández y D. Alberto Prieto Espinosa, Ayudante Doctor, Ayudante Doctora y Catedrático de Universidad respectivamente del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN

Que la memoria titulada "Arquitectura eficiente de condensación de información visual dirigida por procesos atencionales" ha sido realizada por D^{a}. M^{a} Sara Granados Cabeza, bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor Internacional en Ingeniería de Computadores y Redes.

Granada, a 21 de October de 2012

Fdo. Javier Díaz Alonso, Sonia Mota Fernández, Alberto Prieto Espinosa

Directores de la Tesis

Editor: Editorial de la Universidad de Granada
Autor: M^{a} Sara Granados Cabeza
D.L: En trámite
ISBN:En trámite

Agradecimientos

Muchas son las personas a las que debo, en mayor o menor medida, esta tesis doctoral. No sólo me habéis ayudado en los aspectos más técnicos de ella, sino que me habeis influido tanto dentro como fuera de la investigación; con el apoyo de todos vosotros he crecido como investigadora, ingeniera y persona. Por ello, gracias de todo corazón. Esta tesis es posible gracias a vosotros y cada vez que la mire pensaré en los momentos compartidos y sonreiré, a pesar del esfuerzo que ha costado.

Por creer en mí y saber que conseguiría terminar esta tesis incluso cuando yo misma dudaba de ello, quiero agradecérselo a mis padres, mis hermanos (Esther, Jose y Javi), a mi abuela y mi familia en general. Gracias por estar ahí cada día, para lo que necesite, incluso cuando no sé qué necesito o lo que me hace falta es no estar. Gracias por entenderme y por quererme a pesar de ello, sin vosotros no sería nadie.

Quiero agradecérselo especialmente a Manu: por estar ahí siempre que lo he necesitado, por ayudarme con todo lo que ha podido (desde hacer figuras y etiquetar imágenes hasta hacerme la comida cuando se me olvidaba hasta que tenía hambre), por tener fe en mí y quererme cada día.

En los momentos más duros de investigación y escritura de esta tesis, mis apoyos más cercanos fueron mis directores de tesis, Javi, Sonia y Alberto. Es bueno saber que no estás "luchando" sola, que hay alguien detrás que tiene una idea de por dónde debes seguir, aunque no siempre estés de acuerdo con ese camino. Muchas gracias por vuestras ideas, comentarios y correcciones. También me gustaría agradecer a Edu, al que yo siempre consideraré mi director honorífico, por darme la oportunidad de trabajar en un proyecto como DRIVSCO y por apoyarme durante todos estos años no sólo con contratos sino con consejos y ánimos.

De igual forma, quiero agradecer al Departamento de Arquitectura y Tecnología de Computadores en general por el apoyo recibido, y en concreto a Encarni, Julio, Manolo y Paco, por ayudarme con los papeleos varios que son ese mal necesario del que no podemos huir. También me gustaría agradecerle a la Universidad Católica de Lovaina la oportunidad de realizar una estancia con ellos y en concreto a Marc por acogerme en su departamento, Nick por ayudarme a encontrar una aplicación concreta para mis investigaciones y Karl por las cervezas y los buenos ratos.

Conoces a muchas personas cada día, pero nunca te planteas si llegan para quedarse o cuánto pueden influirte. A todas esas personas que he ido conociendo en el camino y que han cambiado mi vida de alguna forma, gracias por apoyarme en los malos momentos, reír conmigo en los buenos y estar siempre ahí. Esta tesis no habría sido posible sin vosotros.

Gracias a “mis niñas” (Isa, Yami, Noe y Lucy) que han aguantado mis locuras y devaneos desde hace más de 15 años y por ello se merecen un monumento... o varios.

Gracias a mis compañeros de facultad que poco a poco han pasado a ser mis amigos y sin los que el camino hasta aquí hubiera sido mucho más aburrido: Alberto, Delia, Iván, Jesús, Jose Enrique (aka Dpp), Juampa, Laura, Quique, Pablo (aka Fergu), Sebas y especialmente a Aída, por aguantarme a lo largo y ancho del mundo, y a Dani (aka Atún), por ofrecerse a leer y buscar las erratas de esta tesis.

Gracias a mis compañeros de despacho que han avanzado estos años conmigo tanto físicamente -desde Ciencias al CIE y de ahí al CITIC- como psíquicamente hasta convertirse en amigos e incluso en ejemplos a seguir: Fran, Jarno, Juanma, Leo, Matteo, Mauricio, Niceto, Raquel, Richard, Silvia. Hemos llegado hasta aquí juntos y poco a poco, ayudándonos, lo conseguiremos todos.

Gracias a todos los compañeros, de departamentos amigos o “enemigos”, e incluso de fuera de la universidad, que han formado parte de mi día a día durante estos años: Ana, Antonio, Belén, Curro, David, Javi A., Javi P., Jose Luis, Juanlu, María, Migue M., Nieves, Paco, Sandra, Trini, Urquiza.

Gracias para mis nuevos amigos de RTI que me apoyaron e incluso empujaron a poner el punto final a esta tesis: Abhi, Antonio, Edu, Fernando, Gianpiero, Ken, Secho, Vishal.

Dobles gracias a todos aquellos que no se dejaron engañar por las apariencias y quisieron “escarbar” hasta conocerme y aceptarme como soy... o casi. No voy a repetir nombres, pero sabed que soy mejor persona por teneros cerca.

Contents

List of Figures

List of Tables

List of Abbreviations

a.k.a.	Also known as
AAE	Average Angular Error
DoG	Difference of Gaussians
DRIVSCO	European Project “Learning to Emulate Perception-Action Cycles in a Driving School Scenario”
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
fps	Frames per second
FT	Fourier Transform
GLOH	Gradient location-orientation histogram
GPU	Graphic Processing Units
GUI	Graphical User Interface
HA	High accuracy
HDL	Hardware Description Language
Hw/Sw	Hybrid Hardware and Software system
IMO	Independently Moving Object
LoG	Laplacian of Gaussian
MA	Medium accuracy
MCU	memory control unit
MSE	Mean Squared Error
NaN	Not a number

PCA-SIFT	Principal Components Analysis applied to SIFT descriptors
ROC	Receiver Operating Characteristic
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SoC	System-on-a-Chip
STD	Standard deviation
SURF	Speeded Up Robust Features
TTC	Time To Contact
VHA	Very high accuracy
WTA	Winner-Take-All

Abstract

This dissertation presents an innovative semidense representation map that condense visual features into sparser maps, reducing memory, bandwidth and computing resource requirements. This semidense map, contrary to existing ones, efficiently includes not only salient-area information, but also non-salient one which increases versatility and allows us to employ it in multiple applications.

This dissertation is structured in three parts.

In the first part, we review the state of the art, paying special attention to dense and sparse visual features. After that, we focus on our novel representation. First, we study how sparse features extract relevant information from the images and which would be our best option as saliency indicator. Then, we focus on the non-salient part of the image, assessing different filter operations as regularization tools and deciding how many of these non-salient points we should include. And finally, we efficiently integrate both kinds of points in a unique representation.

In the second part of this dissertation we assess our semidense representation map in different scenarios. First we employ it in an attention system in which we receive and integrate not only top-down signals but also bottom-up ones, which are not usually included in other systems. This integration presents our representation as a very well suited framework to incorporate target-driven information in a vision system. Then, we use a semidense map as input in a real-world application based on a driving scenario, reducing execution time while achieving similar results as using the original dense map.

The third part corresponds to the design and implementation of our algorithm in a FPGA, incorporating it to a low-level-vision system with memory, bandwidth and computational resources constrains. Our semidense maps solves those constrains and allows for the integration of feedback from mid-/high-level algorithms in real-time. All this without introducing any penalty in the system.

As conclusion, our results show that our semidense representation map is versatile (i.e. condenses any dense visual feature), achieves real-time constrains, inherently regularizes the input features and integrates feedback from other stages. All these characteristics make our solution a very useful representation map for any real-time embedded system that processes images.

Resumen

Esta tesis doctoral presenta un innovador mapa de representación disperso que condensa características visuales hasta conseguir una representación más dispersa, reduciendo los requisitos de memoria, ancho de banda y recursos computacionales. Este mapa semidenso, al contrario que otros existentes, incluye eficientemente tanto información de zonas salientes como no salientes, lo cual incrementa la versatilidad y permite su utilización en múltiples aplicaciones.

Esta tesis doctoral está divida en tres partes.

En la primera parte revisamos el estado del arte, poniendo especial atención a las características visuales densas y dispersas. A continuación, nos centramos en nuestra novedosa representación. Primero estudiamos cómo las características dispersas extraen la información relevante de una imagen y cuál sería la mejor opción como indicador de saliencia en nuestro mapa de representación. Después nos centramos en la parte no saliente de la imagen, estudiando los distintos filtros disponibles como herramientas de regularización y decidiendo qué cantidad de puntos no salientes deberíamos incluir. Y por último integramos eficientemente estos dos tipos de puntos en una única representación.

En la segunda parte de esta tesis doctoral estudiamos el comportamiento de nuestro mapa de representación semidenso en distintos escenarios. Primero lo utilizamos en un sistema atencional en el que recibimos e integramos señales tanto *bottom-up* como *top-down*, estas últimas no son normalmente incluidas por otros sistemas. Esta integración muestra nuestra representación como un marco muy adecuado para la incorporación de información dirigida por objetivos en un sistema de visión. A continuación, hemos utilizado nuestro mapa semidenso como entrada a una aplicación real basada en un escenario de conducción de vehículos, reduciendo el tiempo de ejecución a la vez que obtenemos resultados similares a los conseguidos con el mapa denso original.

La tercera parte corresponde al diseño e implementación de nuestro algoritmo en una FPGA, incorporándolo a un sistema de baja visión con limitaciones de memoria, ancho de banda y recursos computacionales. Nuestro mapa semidenso soluciona estas limitaciones y permite la integración, en tiempo real, de señales realimentadas procedentes de algoritmos de medio y alto nivel. Todo esto sin introducir ninguna penalización en el sistema.

Como conclusión, los resultados muestran que nuestro mapa de representación semidenso es versátil (es decir, condensa cualquier característica visual densa), soporta condiciones de tiempo real, regulariza inherentemente las características de entrada e integra retroalimentación procedente de otras etapas de procesamiento. Todas estas características hacen que nuestra solución sea una representación muy útil para cualquier sistema empotrado de tiempo real que procese imágenes.

Introducción

Motivación

La curiosidad es una de las características más importantes de los seres humanos. A lo largo de la Historia, los humanos nos hemos maravillado por el mundo que nos rodean, incluso llegando a luchar para defender nuestros descubrimientos o nuestras ideas. Fue la curiosidad la que hizo que la Ciencia avanzara y que pasáramos del descubrimiento del fuego al aterrizaje en la Luna, de los dibujos de Da Vinci sobre la anatomía humana hasta el Proyecto Genoma Humano. Y la chispa que inicia cualquier Tesis Doctoral es, y siempre será, la curiosidad.

"¿Cómo funciona el cerebro?" es sólo una de las muchas preguntas que la humanidad ha tratado de contestar desde el principio de los tiempos. E incluso hoy en día, no podemos dar una respuesta clara a esta pregunta. Sin embargo, podemos emular su comportamiento para desarrollar, por ejemplo, un cerebelo virtual [50] y utilizarlo para mejorar nuestros conocimientos en Medicina o Robótica. Nuestro grupo de investigación se centra en descubrir cómo funcionan el cerebro y la vista humanos, aplicando ese conocimiento en el área de las Ciencias de la Computación con el fin de mejorar los sistemas actuales y ayudar a la cura de múltiples enfermedades.

Una de las primeras cosas que descubres cuando trabajas en Visión Artificial es lo complejo y eficiente que es el sistema visual humano. Desde el sistema de recepción, la retina, nuestro sistema visual supera cualquier sistema artificial existente. La retina no sólo captura la intensidad y el color de la escena, sino que reduce las tareas que deberá realizar el cortex visual [44] ya que lleva a cabo cierto pre-procesamiento en el plano focal. Este pre-procesamiento es realizado por los ganglios retinales que extraen información relacionada con transiciones espacio-temporales para, posteriormente, mandarlo al cortex en forma de impulsos neuronales [12]. Además, la retina envía estos datos siguiendo un esquema de comunicación dirigido por eventos (*event-driven communication*) [14], reduciendo así el ancho de banda que necesita. Por ello, encontramos muchos estudios importantes que actualmente se dedican a desarrollar un sistema de visión bio-inspirado capaz de emular el comportamiento de la retina humana y de sustituir a los sistemas convencionales de captura de imagen (es decir, las cámaras). En concreto, Boahen [12, 21] ha desarrollado una retina artificial hecha en silicio que se comporta de manera similar a la retina de los mamíferos. Algunos investigadores de nuestro grupo han desarrollado entornos de trabajo que simulan el comportamiento de sensores retinomórficos [55, 34], llegando incluso a extraer información sobre flujo óptico[4].

A pesar de estas prometedoras investigaciones, para entender completamente el funcionamiento del sistema visual humano, tenemos que ir más allá de la retina, tenemos que entrar en el cerebro. El cortex visual está localizado en la parte posterior del cerebro, mas concretamente, en el lóbulo occipital y se encarga de procesar la información visual. Existen multitud de estudios relacionados con el cortex visual de los primates, intentando entender cómo interpretamos lo que vemos. En su artículo [47], Logothetis describe las distintas estructuras que permiten la visión de la siguiente forma:

> El sistema visual humano empieza con los ojos y se extiende a lo largo de diversas estructuras internas del cerebro, hasta llegar a las distintas regiones del cortex visual primario (V1, etc.). En el quiasma óptico, los nervios ópticos se cruzan, entre otras razones, para permitir que cada hemisferio del cerebro reciba información de ambos ojos. Dicha información es filtrada por el núcleo geniculado lateral, que a su vez está formado por varias capas de células nerviosas que reaccionan sólo a estímulos procedentes del ojo. El cortex temporal inferior se encarga de ver formas. Algunas células de cada una de estas áreas sólo se activan cuando una persona o mono es consciente de un estímulo, no debido al propio estímulo.

La Figura 1 muestra tres esquemas de dónde está localizado el cortex visual, cuáles son sus subdivisiones funcionales y cómo la información es transferida desde el ojo hasta la primera de esas subdivisiones, el cortex visual primario o V1.

La mayoría de las neuronas del cortex visual reaccionan ante información sobre la orientación lineal de los bordes siguiendo una organización jerárquica similar a la de un modelo de procesamiento serie, es decir, cuando las *células simples* reacciona a un estímulo luminoso, las *células complejas* reaccionan a la orientación [82]. Las neuronas están organizadas por columnas donde aquellas células que se encuentran en la misma columna posee los mismos atributos. Usando esta información, muchos investigadores importantes se han centrado en las prótesis neuromórficas. En concreto, nuestro grupo de investigación ha formado parte de varios proyectos que pretendían desarrollar prototipos de este tipo de prótesis en el ámbito de la

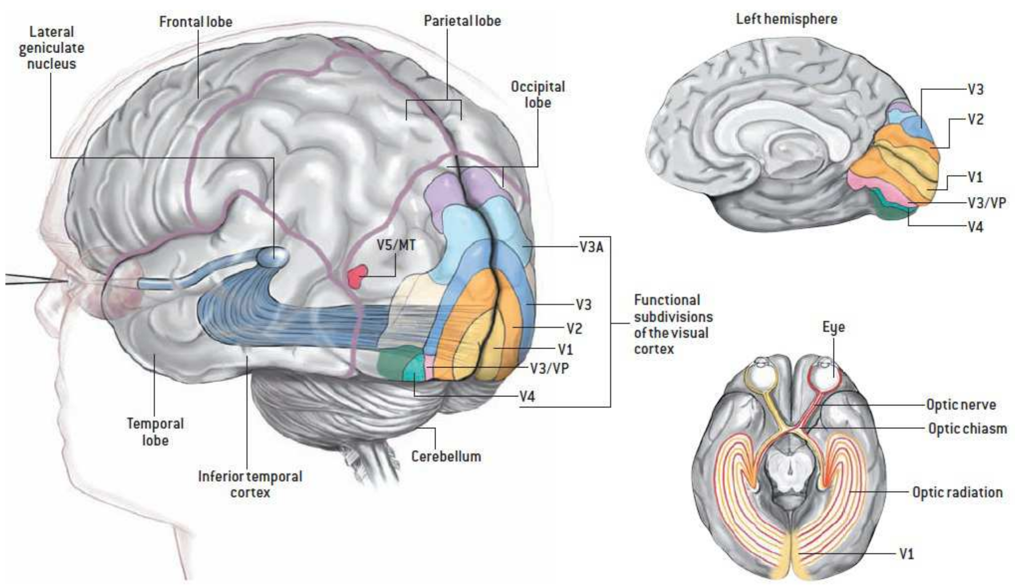


Figure 1: Esquema del sistema visual humano, extraído del trabajo de Logothetis [47].

rehabilitación visual, demostrando que las neuro-prótesis corticales (las interfaces con el cortex visual) son factibles para pacientes con ceguera profunda [65, 64].

Otro parámetro a tener en cuenta en el proceso de visión es la *atención*. Cuando un jugador de fútbol del equipo contrario se aproxima a la portería, el portero se centra en él y todo lo demás se emborrona. Esto se debe a que la atención del portero se ha centrado en el otro jugador, intentando calcular a dónde y cuándo va a tirar. Itti y Koch proponen dos procesos distintos de atención que trabajan simultáneamente [39]. El primero es un proceso voluntario que posee un criterio de selección que se adapta al objetivo que perseguimos en cada momento (en nuestro ejemplo, el portero elige al otro jugador como objetivo). El segundo proceso, sin embargo, es independiente de la tarea a realizar (*bottom-up* en inglés), extrayendo aquella información que resalta intrínsecamente con respecto a su entorno [67]. En nuestro ejemplo, el portero sigue procesando la información sobre el resto de jugadores que se mueven alrededor de la portería. De esta manera, si alguno de ellos se aproxima demasiado a la portería, pueden convertirse en un objetivo, por ejemplo, si el otro jugador decide pasarles el balón, provocando una cambio de objetivo dependiente de la atención (*top-down* en inglés). No obstante, los sistemas actuales basados en atención no incluyen este tipo de información *top-down* porque es más complicada de procesar (llegando a ser su proceso de extracción cerca de ocho veces más lento que en los casos *bottom-up* [89]).

Todos estos conceptos no sólo se pueden aplicar a la visión humana, sino que muchos de ellos puede utilizar fácilmente en robótica, escenarios de conducción, video vigilancia, seguridad, etc. De esta manera reducimos las barreras entre las distintas disciplinas y creamos un marco perfecto para colaborar con otros investigadores. Por ello, nuestro grupo ha colaborado activamente en varios proyectos multidisciplinares europeos y nacionales. Algunos de ellos son: ECOVISION, CORTIVIS, DINAM-VISION. De hecho, el trabajo que presentamos en esta tesis doctoral surgió en el ámbito del proyecto europeo DRIVSCO [90] (IST-016276-2). Además de DRIVSCO, esta investigación ha sido financiada mediante el proyecto nacional ARC-VISION (TEC2010-15396) y la beca de Formación de Personal Universitario (FPU).

Proyecto Europeo DRIVSCO

DRIVSCO deriva del nombre en inglés del proyecto *Learning to Emulate Perception-Action Cycles in a Driving School Scenario*, que significa aproximadamente "Aprendiendo a emular los ciclos percepción-acción en un escenario basado en una autoescuela". Este proyecto empezó en 2001 y terminó en 2009. El consorcio estaba formado por varias universidades y compañías a lo largo de Europa:

- Universidad Vytautas Magnus, en Lituania.
- Universidad de Granada.
- Hella KGaA, empresa de Alemania.
- Universidad Católica de Leuven, en Bélgica.
- Universidad del Sur de Dinamarca.
- Universidad de Munich, en Alemania.
- Universidad de Göttingen, en Alemania.
- Universidad de Génova, en Italia.

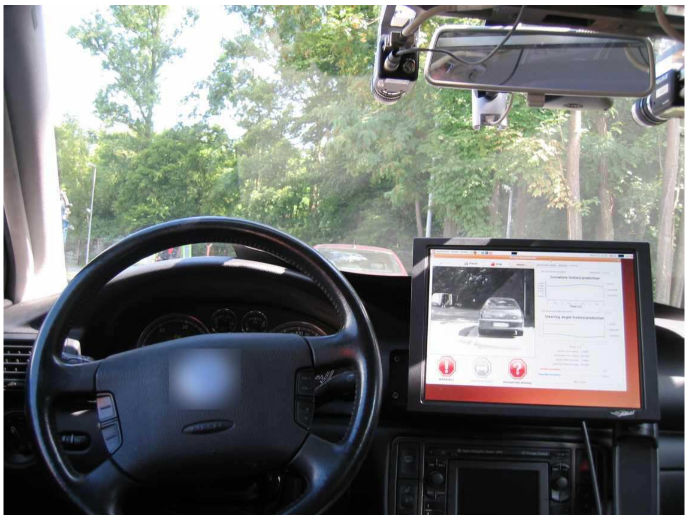
Formar parte de un consorcio internacional facilita la transferencia de conocimiento, por lo que el principal objetivo de DRVSCO está definido así [90]:

> *El objetivo principal de DRIVSCO es diseñar, probar e implementar una estrategia que permita combinar diferentes sistemas*

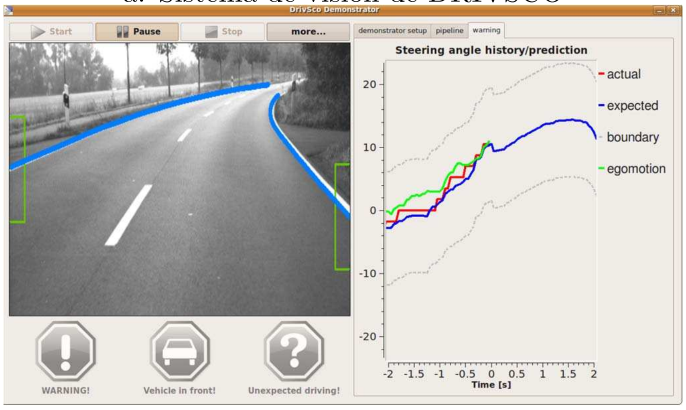
> *de aprendizaje adaptativo con sistemas de control convencional. Partiendo de un sistema basado en una interfaz hombre-máquina completamente operacional, el proyecto pretende llegar a un sistema autónomo, mejorado a través del aprendizaje, que pueda actuar de manera proactiva empleando distintos mecanismos predictivos. DRIVSCO pretende emplear aprendizaje y control basados en un ciclo cerrado de percepción acción que extrae su información tanto de coches como de los conductores; combinando, por primera vez, técnicas avanzadas de análisis visual de la escena (realizado principalmente en hardware) con mecanismos de aprendizaje secuencial supervisado con el fin de conseguir un sistema de control semi-autónomo y adaptativo para coches y otros vehículos. La idea central de este proyecto es que el coche debería aprender a conducir autónomamente simplemente correlacionando la información de la escena con las acciones realizadas por el conductor.*
>
> *En el contexto de este proyecto, el sistema deberá ser probado y utilizado en escenarios de visión nocturna empleando para ello sistemas de visión infrarroja, que es nuestro dominio de la aplicación principal y comercialmente más relevante. Aquí hemos concebido un sistema que puede aprender a conducir un coche durante el día y aplicar las estrategias de control aprendidas de manera autónoma durante la noche.*

Desde un punto de vista más práctico, el principal objetivo de DRIVSCO era instalar un sistema artificial de visión en un coche y, en vez de informar al conductor de una situación peligrosa (como por ejemplo un peatón cruzando en una carretera poco iluminada), predecir cómo actuaría el conductor y reaccionar de manera similar. En la Figura 2a vemos el coche de pruebas utilizado en DRIVSCO, incluyendo las cámaras y la interfaz gráfica con el usuario (GUI). El sistema visual humano, aunque es muy fiable durante el día, no lo es tanto durante la noche. En la Figura 3 podemos ver cómo durante el día nuestro sistema es capaz de percibir información procedente de objetos más lejanos. Por la noche, sin embargo, la distancia a la que percibimos objetos se ve dramáticamente reducida y, por lo tanto, conducir de noche es mucho más peligroso. En DRIVSCO hemos intentado reducir la diferencia entre la visión de día y de noche instalando cámaras capaces de grabar tanto a la luz del día como por la noche (es decir, cámaras infrarrojas). La interfaz gráfica con el usuario permite que éste seleccione entre diferentes algoritmos: detección de la línea de la carretera, cálculo del tiempo para el impacto con otros objetos, extracción del plano de la carretera, etc. tal y como se muestra en la Figura 2 [51].

Por su experiencia en SoC (siglas del término inglés *System-on-a-Chip*) y Visión Artificial, nuestro grupo se encargó del diseño e implementación de un



a. Sistema de visión de DRIVSCO



b. Interfaz gráfica de salida de DRIVSCO

Figure 2: Sistema de visión de DRIVSCO. a) muestra las cámaras y el monitor de visualización instalados en el coche de pruebas. b) muestra la interfaz gráfica de salida donde vemos señalada la opción de detección de la línea de la carretera.

sistema híbrido hardware-software (Hw/Sw) que extrae distintas primitivas visuales (tanto dispersas, por ejemplo la energía y la fase; como densas, por ejemplo estéreo y flujo óptico) en tiempo real tal y como se muestra en la Figura 4. Además, es necesario que este SoC combine las primitivas extraídas y la información enviada por niveles superiores. Por otro lado, las primitivas visuales extraídas por medio de hardware son más ruidosas que las obtenidas en software, principalmente debido a las limitaciones en la precisión en punto flotante y la memoria. Por lo tanto, todo el proceso se beneficiaría de un paso de regularización capaz de suavizar, aunque sólo sea

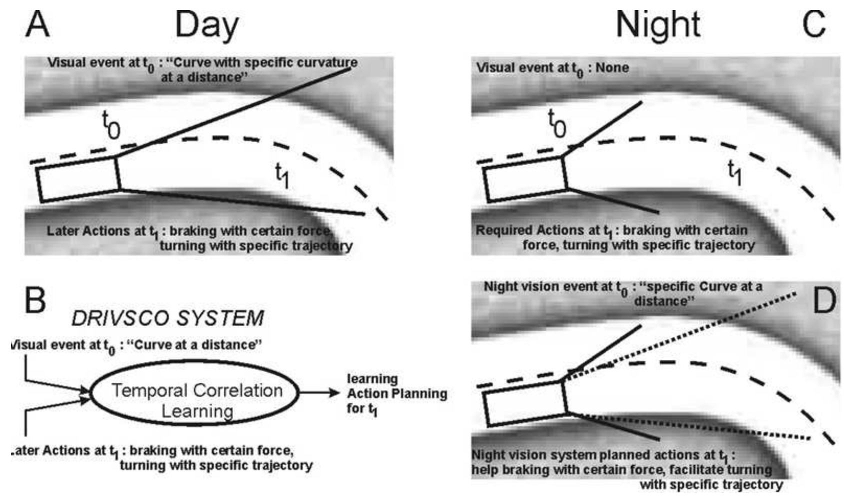

A
Day
Visual event at t_0 : "Curve with specific curvature at a distance"
t_0
t_1
Later Actions at t_1 : braking with certain force, turning with specific trajectory
Night
C
Visual event at t_0 : None
t_0
t_1
Required Actions at t_1 : braking with certain force, turning with specific trajectory
B
DRIVSCO SYSTEM
Visual event at t_0 : "Curve at a distance"
Temporal Correlation Learning
learning Action Planning for t_1
Later Actions at t_1 : braking with certain force, turning with specific trajectory
Night vision event at t_0 : "specific Curve at a distance"
D
Night vision system planned actions at t_1 : help braking with certain force, facilitate turning with specific trajectory

Figure 3: Limitaciones de la conducción por la noche. A) muestra una situación de conducción diurna. B) ejemplifica el aprendizaje realizado en DRIVSCO que correlaciona los eventos visuales en el instante t_0 y las acciones del conductor en el instante t_1. C) muestra una situación de conducción nocturna. EN t_0 el conductor no ha visto la curva todavía. D) ejemplifica el comportamiento del sistema de visión nocturna de infrarrojos, permitiendo que el sistema DRIVSCO "vea" la curva. Tras haber aprendido la correcta acción durante el día, el sistema DRIVSCO será capaz de ayudar al conductor.

parcialmente, dicho ruido [86]. Además de este co-diseño Hw/Sw, Pauwels y otros desarrollaron una solución basada en procesadores gráficos (GPU, siglas en inglés del término) [63]. No obstante, esta solución no puede ser empotrada en un SoC y, por lo tanto, sólo es útil como comparación.

En resumen, DRIVSCO necesitaba integrar distintas características o primitivas visuales de manera eficiente y apta para ser implementada en hardware, a la vez que las regularizaba para suavizar el ruido. Sin embargo, estos requisitos no son los únicos a los que hay que enfrentarse ya que los SoCs normalmente tienen memoria y ancho de banda limitados.

En la Figura 4 podemos ver cómo nuestro sistema recibe pares de imágenes (izquierda y derecha), analizando la escena a partir de ellas. Este análisis consiste en extraer diversas características visuales utilizando como entrada las imágenes capturadas. En la figura también mostramos el número de bits por píxel que necesita cada una de estas características (hasta 24 en el caso del movimiento). De hecho, para extraer algunas de estas características, nuestro sistema necesita almacenar en memoria varias máscaras de convolución[1] y, en algunos casos, varias imágenes (como por ejemplo para

[1]Estas máscaras se explican en la sección 2.1

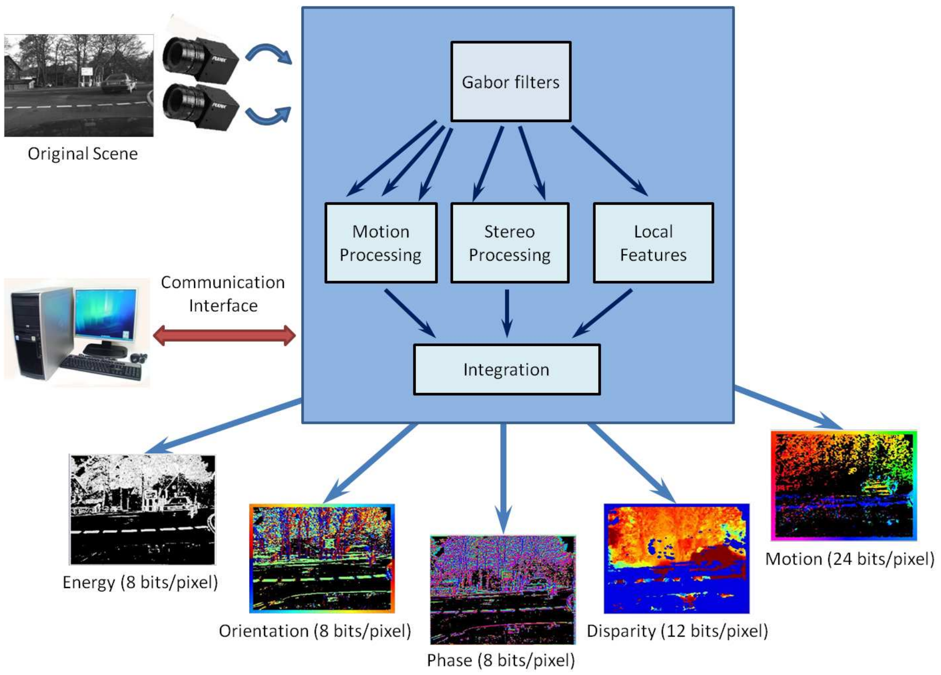


Figure 4: Sistema de baja visión de DRIVSCO. Dos cámaras estéreo capturan la escena y la envía al sistema de baja visión que se encuentra empotrado en una plataforma de co-procesado basada en FPGAs. El sistema de baja visión extrae varias características visuales: energía (*energy*), orientación (*orientation*), fase (*phase*), movimiento (*motion*) y disparidad (*disparity*); y las envía a un ordenador (PC) a través de una interfaz de comunicación (flecha roja)

el flujo óptico). Si sumamos todo esto nos damos cuenta de la cantidad de memoria que necesitamos. Además, una vez calculadas estas características, tendremos que mandarlas al co-procesador (el PC en nuestro caso) en el que se extraerán características de más alto nivel. Vamos a hacer algunas cuentas con números reales para que queden más claras las restricciones de nuestro sistema.

Memoria Asumiendo que todas las características pueden ser extraídas por el SoC, lo cual no es un problema trivial [86], necesitaríamos almacenar: varias imágenes (hasta un máximo de 5 para el flujo óptico[2]); las características calculadas[3], cada una de las cuales es extraída utilizando una aproximación multi-escala [75]; y varios filtros[4] utilizados durante este proceso. La Figura 5 muestra estos requisitos de memoria

[2]Más detalles en la sección 2.4.2
[3]Explicadas en el capítulo 2
[4]Explicados en la sección 2.1

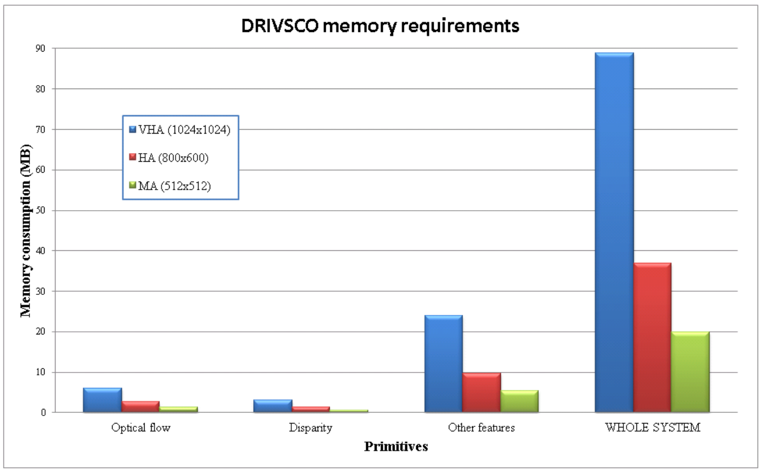


Figure 5: Requisitos de memoria para DRIVSCO considerando tres resoluciones de imagen: muy alta (VHA con 1024x1024 píxeles) en azul, alta (HA con 800x600) en rojo y media (MA con 512x512) en verde.

de nuestro sistema si utilizamos cuatro (4) escalas y consideramos tres resoluciones de imagen: media (512x512), alta (800x600) y muy alta (1024x1024). Podemos ver cómo necesitaríamos más de 100MB para almacenar estas imágenes y características, mientras que los sistemas hardware normalmente tienen menos de 50MB disponibles [93].

Ancho de banda. De igual forma, si asumimos que todas estas características deben ser enviadas al co-procesador, podríamos calcular el ancho de banda que necesitamos en DRIVSCO. En la Figura 6 vemos el ancho de banda necesario para nuestro sistema si utilizamos las mismas resoluciones y escalas que en el caso de la memoria.

De estos requisitos podemos deducir que la integración de las características visuales precisa de algún tipo de compresión o condensación (en la sección siguiente discutimos la diferencia entre estos dos términos). Por otra parte, la mayoría de los algoritmos de medio y alto nivel no necesitan un mapa denso para trabajar, sino que normalmente realizan una selección previa de los puntos más interesantes [53]. Por lo tanto, la mejor solución sería una que integrara, regularizara y seleccionara las características *antes* de mandarlas a niveles superiores, imitando el comportamiento del sistema visual humano.

Otro problema a tener en cuenta es si el coprocesador será capaz de procesar esta cantidad de datos. El proceso de recibir dichos datos y aplicar sobre ellos los distintos algoritmos para extraer características de más alto nivel no es trivial y requiere cálculos complejos y gran cantidad de memoria,

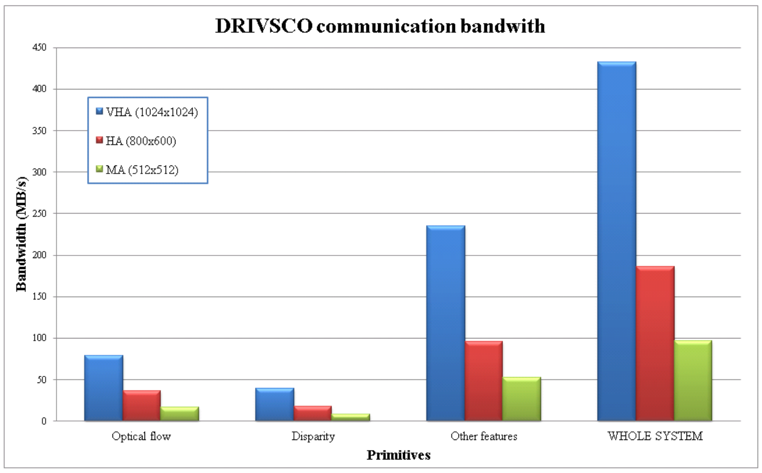


Figure 6: Requisitos de ancho de banda para DRIVSCO considerando tres resoluciones de imagen: muy alta (VHA con 1024x1024 píxeles) en azul, alta (HA con 800x600) en rojo y media (MA con 512x512) en verde.

no pudiendo ser asumida por procesadores empotrados. De hecho, esta carga de trabajo puede llegar a ser excesiva incluso para muchos procesadores estándar.

Condensación frente a Compresión

Hemos estado hablando de lo eficiente que es el sistema visual humano y de lo útil que sería imitarlo. Sin embargo, desde un punto de vista práctico, el principal problema de nuestro problema son las restricciones de memoria, ancho de banda y capacidad computacional. ¿Sería suficiente con comprimir las imágenes?

La mayoría de las técnicas de compresión utilizan niveles de color o grises y redundancias espacio-temporales en la información como bases de reducción. Cuando comprimimos sin pérdidas, los datos se agrupan teniendo en cuenta el nivel de gris o color, de manera que ocupan menos espacio en memoria; si la compresión es con pérdidas, el número de bits asociados a cada píxel se reduce y sus valores son agrupados utilizando técnicas estadísticas [78]. Sin embargo, estas técnicas se basan simplemente en la información de color/gris de la imagen, que no es suficiente si lo que queremos es integrar la realimentación procedente de los algoritmos de más alto nivel.

Frente a este método basado puramente en la compresión, nuestro objetivo es satisfacer los requisitos de memoria y ancho de banda a la vez que compensamos las estimaciones ruidosas regularizando las características visuales. En la literatura encontramos algunas soluciones que aplican comprensión para regularizar imágenes [60]. Sin embargo, estas soluciones precisan gran cantidad de procesamiento eminentemente iterativo, que no es muy apropiado para nuestro sistema, ya sobrecargado de por sí. Esto nos lleva a otro de nuestros problemas: la carga de trabajo. Nuestra propuesta tiene que reducir la carga computacional del coprocesador de manera que estos algoritmos de alto nivel puedan ser integrados en un sistema empotrado.

Por lo tanto, nuestra mejor opción no es simplemente comprimir, sino *condensar*, es decir, seleccionar aquellas zonas de la imagen que son más relevantes para nuestro sistema dependiendo de la tarea que estemos realizando en cada momento. Nuestro algoritmo de condensación deberá encargarse no sólo de almacenar y enviar la información con el mínimo de recursos posible, sino que además deberá seleccionar los datos más importantes. Junto con las propiedades antes mencionadas, nuestro algoritmo de condensación es capaz de preservar y regularizar las zonas planas de las características visuales. Además, hemos prestado especial atención en permitir la fusión información visual, facilitando la realimentación de información [72].

Objetivos Principales

De los párrafos anteriores podemos extraer los siguientes requisitos que componen nuestros objetivos principales:

Condensación. Necesitamos reducir los requisitos de memoria y ancho de banda, así como la posterior carga computacional, extrayendo para ello la información relevante de las características visuales densas y almacenándola eficientemente.

Versatilidad. Debemos condensar cualquier característica visual que nuestro sistema de visión pueda extraer.

Implementable en hardware. El algoritmo de condensación que diseñemos debe incluirse como un módulo hardware en nuestro sistema empotrado de visión, por lo que debemos tener en cuenta los requisitos de tiempo real que esto conlleva.

Realimentación. Vamos a recibir realimentación desde niveles superiores de procesamiento (atención entre otros), por lo que nuestro módulo debe ser capaz de incorporarlos en el proceso de condensación.

Representación eficiente. Los niveles superiores de procesamiento van a recibir nuestra salida condensada como entrada. Por lo tanto, esta salida debe ser fácil de utilizar por dichos niveles.

Recuperación de la información. Dado que no sabemos a priori qué información va a ser más importante para los niveles superiores de procesamiento, debemos almacenar suficientes datos como para recuperar cualquier detalle que puedan necesitar.

En esta tesis doctoral presentamos, explicamos y probamos nuestra solución para satisfacer estos requisitos: un *mapa de representación semidenso* que condensa características visuales densas en una representación más dispersa a la vez que mantiene la mayoría de la información contenida originalmente. Esta representación reduce las necesidades de memoria y ancho de banda y además disminuye la carga computacional.

Herramientas y Métodos Utilizados

Los algoritmos presentados en esta tesis doctoral han sido implementados, inicialmente, en el entorno de programación MATLAB. Hemos elegido MATLAB porque favorece una implementación rápida de los algoritmos, así como su posterior análisis. De esta forma hemos sido capaces de probar distintas soluciones para nuestro problema, ampliando la versatilidad de la versión final. Además, MATLAB proporciona gran cantidad de funciones para la visualización de imágenes y otros resultados. Otro punto a favor es que muchos de los algoritmos de alto nivel que utilizarán características visuales condensadas están implementados en MATLAB.

No podemos olvidarnos de nuestro proyecto, DRIVSCO. El algoritmo tiene que ser implementado en un lenguaje de descripción hardware para que podamos sintetizarlo e integrarlo en nuestro SoC. Por lo tanto hemos elegido Handel-C y el entorno de programación de Mentor Graphics (anteriormente conocido como Celoxica) para implementar la versión hardware de nuestro algoritmo. Dentro de los lenguajes de descripción hardware, Handel-C es un lenguaje de alto nivel, utilizado principalmente para la programación de FPGAs, que incluye la mayoría de las características comunes del lenguaje C junto a un conjunto de instrucciones, tipos de datos y sentencias de control diseñadas para la implementación en hardware, poniendo especial énfasis en las sentencias de paralelización. La diferencia entre Handel-C y otros lenguajes de descripción hardware es el nivel de abstracción. Handel-C facilita un mayor nivel de abstracción a la hora de definir los circuitos complejos [35], acelerando el ciclo de diseño/implementación.

Contenido de la Tesis

Vamos a continuar esta tesis doctoral con una pequeña introducción a los principales algoritmos de Visión Artificial que hemos usado durante nuestra investigación. Por lo tanto, el capítulo 2 explica operadores de filtrado, características visuales densas y dispersas, sistemas atencionales y descriptores multi-modales (la solución más prometedora existente en la literatura para resolver nuestro problema). En el capítulo 3 estudiamos por qué ninguna de las opciones existente satisface todos nuestros requisitos y presentamos nuestra solución: un mapa de representación semidenso. Este capítulo incluye además distintos métodos para obtener nuestro mapa semidenso, resultados usando secuencias conocidas (*benchmarks* en inglés) y un estudio de las capacidades inherentes de regularización. En el capítulo 4 mostramos el resultado de utilizar nuestro mapa semidenso en varias aplicaciones reales, desde sistemas atencionales (*top-down* y *bottom-up*) a escenarios de conducción (con ejemplos de deteción de la carretera y de objetos). En el capítulo 5 nos centramos en la implementación de nuestro algoritmo de condensación en hardware específico, es decir, en la FPGA, incluyendo resultados de consumo de recursos, rendimiento e integración con el sistema de visión de DRIVSCO. Finalmente, el capítulo 6 resumen las principales aportaciones de nuestra investigación y sugiere varias líneas de trabajo futuro.

Chapter 1

Introduction

1.1 Motivation and Framework

Curiosity is one of the main characteristics of human beings. All over the History, people have wondered about the world, even fighting to defend their discoveries and opinions. It was curiosity what propelled Science from the discovery of fire to the landing on the Moon, from Da Vinci's human anatomy drawings to the Human Genome Project. And the spark of any PhD dissertation is, and will always be, curiosity.

"How does brain work?" is just one of the many questions mankind has tried to answer since the beginning of time. And even today we cannot give a straightforward answer to it. We can, however, emulate its behavior; developing, for instance, a virtual cerebellum [50] and using it to improve Medicine or Robotics. Our research group focuses on applying human brain and vision knowledge into Computer Science, improving the current systems and achieving results that could help existing diseases.

One of the first things you realize when working on Computer Vision is how complex and efficient human visual system is. From the reception system, the retina, our visual system overcomes current robotic systems. The retina not only grabs intensity and color but also reduces the visual cortex workload [44], taking advantage of significant focal-plane processing. This reduction is mainly due to the preprocessing performed by the retinal ganglions which extract directly spatio-temporal transition related information and transfer it using neural spikes [12]. Moreover, the retinal system shrinks data bandwidth because of its event-driven communication scheme [14]. Therefore, some important research efforts are currently concentrated on the development of bio-inspired vision systems that will provide an alternative to the conventional sensors (i.e. cameras) and grabbing systems. In particular, Boahen's artificial retina [12, 21] mimics mammal retina on a silicon chip. Some of our group inputs in this field are several frameworks that allow to work with retinomorphic sensors [55, 34], e.g. extracting optical flow [4].

Regardless of these research lines, to completely understand how the human visual system works, we need to go beyond the retina, we need to get into the brain. The visual cortex, which is located in the occipital lobe, in the back of the brain, is responsible for processing visual information. Many books and papers study how primate cortex works, how we interpret what we see. In his article [47] Logothetis describes the structures for seeing this way:

> Human visual pathway begins with the eyes and extends through several interior brain structures before ascending to the various regions of the primary visual cortex (V1, and so on). At the optic chiasm, the optic nerves cross over partially so that each hemisphere of the brain receives input from both eyes. The information is filtered by the lateral geniculate nucleus, which consists of layers of nerve cells that each respond only to stimuli from one eye. The inferior temporal cortex is important for seeing forms. Some cells from each area are active only when a person or monkey becomes conscious of a given stimulus.

Fig. 1.1 shows three simple schemes of where the visual cortex is located, which are its functional subdivisions and how the information is transfer from the eye to the first of them, the primary visual cortex or V1.

Most of the visual cortex neurons respond to linear edge orientation in a hierarchical organization following a serial processing model, i.e. as the simple cells react to light stimulus, the complex ones respond to orientation parameters [82]. Neurons are organized in a columnar architecture where the ones in the same column have the same attributes. Based on this idea, many important researchers are focused on neuromorphic prosthesis. In particular, our research group has been part of several projects that aim to develop prototypes in the field of visual rehabilitation and demonstrate the feasibility of a cortical neuro-prosthesis (interfaced with the visual cortex) as an aid to deep blind people [65, 64].

Another parameter to consider in the vision process is the *attention*. When a soccer player from the opposite team gets close to the goal, the goalie focuses on him and everything else blurs. This is because the goalie attention is focused on the other player, trying to calculate where and when he is going to shoot. Itti and Koch propose that there are two processes of attention working simultaneously [39]. The first one is a voluntary process whose selection criteria change depending on the target application (in our example, the goalie chooses the other player as target). The second one, on

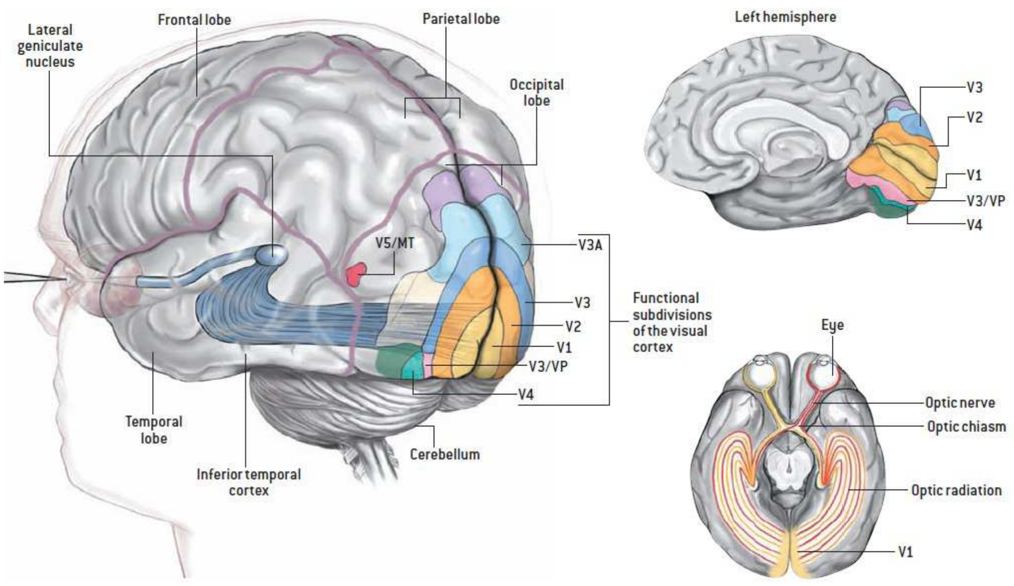


Figure 1.1: Logothetis [47] schema of the human visual pathway.

the other hand, is a task-independent process driven in a bottom-up way that extracts intrinsically salient information from the context [67]. In our example, the goalie is still processing information about the other players moving around the goal. If any of them get too close they might become a target, triggered by a top-down decision (such as detecting that the player with the ball looking for somebody to pass it). Current attention-based systems, however, do not usually include top-down information because it is a complicate process (its extraction process is circa eight times slower than bottom-up one [89]).

All of these concepts are not limited to human vision, they can easily be applied to practical scenarios such as robotics, driving scenarios, video surveillance, security, etc. Hence, we thin the boundaries between disciplines and provide the perfect framework to explore the possibilities of our research. Thus, our group has actively joined several European and National multi-disciplinary projects. Some of them are: ECOVISION, CORTIVIS, DINAM-VISION. In fact, the work presented in this dissertation came out at the European Project DRIVSCO [90] (IST-016276-2). Apart from DRIVSCO, this research have also been funded by the national project ARC-VISION (TEC2010-15396).

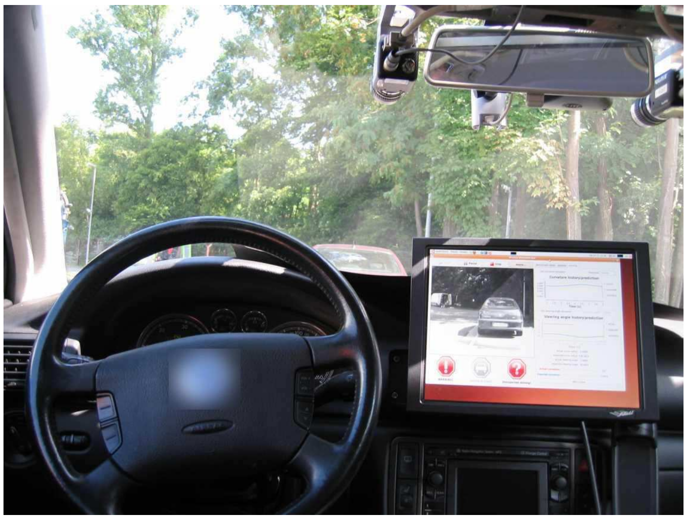
1.1.1 European Project DRIVSCO

DRIVSCO stands for *Learning to Emulate Perception-Action Cycles in a Driving School Scenario*, and it was a project that started in 2001 and finished in 2009. The consorcium involved several universities and companies around Europe:

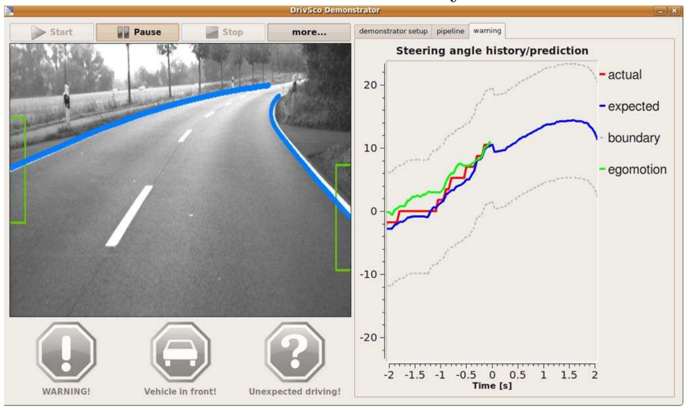
- Vytautas Magnus University, Lithuania.
- University of Granada, Spain.
- Hella KGaA, Germany.
- Katholieke Universiteit Leuven, Belgium.
- University of Southern Denmark.
- Westfälische Wilhems-University Münster, Germany
- Georg-August-University, Germany.
- University of Genoa, Italy.

Being part of an international consortium facilitates knowledge sharing, therefore, the main goal of DRIVSCO was defined in [90] as follows:

> *The main goal of DRIVSCO is to devise, test and implement a strategy of how to combine adaptive learning mechanisms with conventional control, starting with a fully operational human-machine interfaced control system and arriving at a strongly improved, largely autonomous system after learning, that will act in a proactive way using different predictive mechanisms. DRIVSCO seeks to employ closed loop perception-action learning and control to cars and their drivers; combining for the first time advanced (largely hardware based) visual scene analysis techniques with supervised sequence learning mechanisms into a semi-autonomous and adaptive control system for cars and other vehicles. The central idea of this project is that the car should learn to drive autonomously from correlating scene information with the actions of the driver.*
>
> *In the context of this project this system shall be tested and applied in night-vision scenarios with infra-red illumination, which is our main and commercially very relevant application domain. Here we envision a system that can learn to drive a car during daylight and apply the learned control strategies in an autonomous way to the system and augmented field of infrared night-vision.*



a. DRIVSCO Vision System



b. DRIVSCO GUI output

Figure 1.2: DRIVSCO Vision system. a) shows the cameras and the monitor installed in the test car. b) shows the GUI output when using the road lane tracking option.

From a more practical point of view, the main goal of DRIVSCO was to install an artificial vision system in a car and, instead of informing the driver of a dangerous situation (such as a crossing pedestrian in a poorly lighted road), predict how driver would act and react as he would do. Fig. 1.2a shows the test car used in DRIVSCO, including the cameras and the Graphical User Interface (GUI) . The human visual system, although very accurate during the day light, is very unreliable at night. In Figure 1.3 we can see how during the day our visual system is able to perceive information from farther objects. At night, however, our perception distance is dramatically reduced and, therefore, driving at night is more dangerous. In DRIVSCO framework we try to reduce this difference between day and night vision by installing cameras able to record at both, daylight and night (i.e. infrared cameras). The GUI allows the user to select different algorithms: lane detection, ground-plane extraction, time to contact (TTC), etc. (as Fig. 1.2b shows) [51].

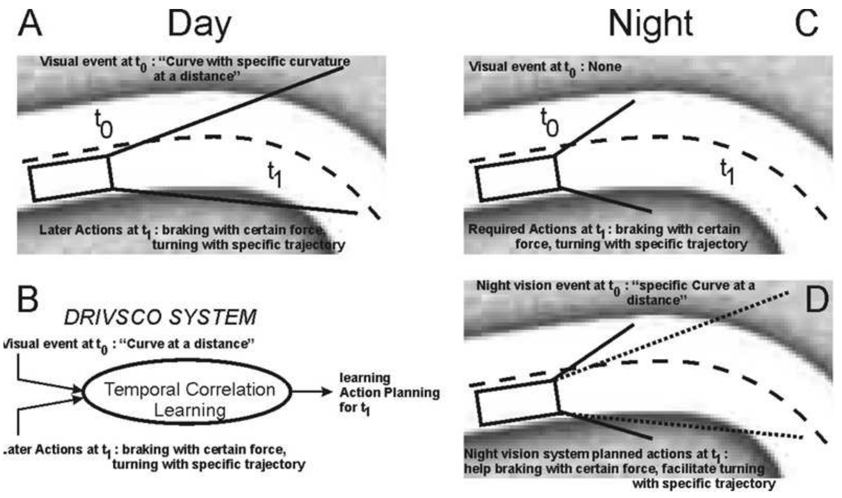

A
Day
Night
C
Visual event at t_0 : "Curve with specific curvature at a distance"
t_0
t_1
Later Actions at t_1 : braking with certain force, turning with specific trajectory
Visual event at t_0 : None
Required Actions at t_1 : braking with certain force, turning with specific trajectory
B
DRIVSCO SYSTEM
Visual event at t_0 : "Curve at a distance"
Temporal Correlation Learning
learning Action Planning for t_1
Later Actions at t_1 : braking with certain force, turning with specific trajectory
Night vision event at t_0 : "specific Curve at a distance"
D
Night vision system planned actions at t_1 : help braking with certain force, facilitate turning with specific trajectory

Figure 1.3: Night driving constrains. A) shows the daylight driving situation. B) depicts what the DRIVSCO system will do, namely learning the correlation between visual events at t_0 and driver actions at t_1. C) shows the night driving situation. At t_0 the driver does not yet see the curve. D) depicts the IR night-vision situation where the DRIVSCO system already "sees" the curve. After having learned the correct action planning during daylight the DRIVSCO system will be able to help the driver.

Due to its experience on System-on-a-Chip (SoC) and computer vision, our group was on charged of designing and implementing a hybrid hardware-software (Hw/Sw) system that extracts different visual cues (sparse, like energy and phase; and dense, like stereo and optical flow) on real-time as shown in Figure 1.4. Moreover, the SoC needed to combine its extracted cues and the middle-level information. Hardware obtained cues are noisier than software ones -mainly due to the float-point precision and memory constrains-, hence the whole process would benefit from a regularization step able to smooth some of that noise [86]. In addition to Hw/Sw co-design, Pauwels et al. developed a GPU-based solution to DRIVSCO problem [63]. This solution, however, cannot be embedded in a SoC and was only useful for comparison purposes.

In short, DRIVSCO needed a hardware-friendly efficient way to integrate different features or cues whilst regularizing them to smooth the noise. These

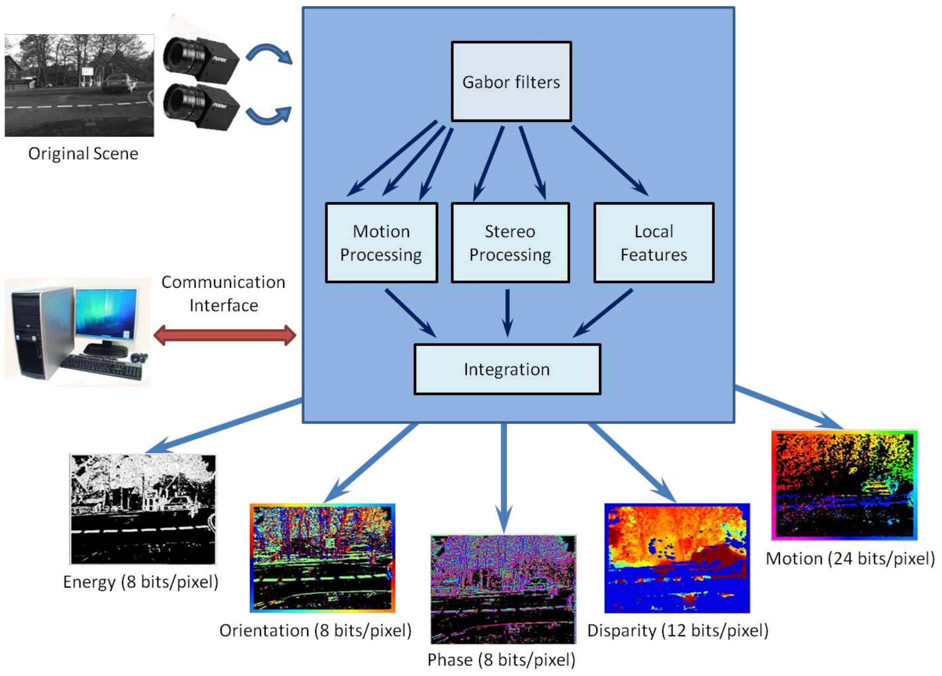


Figure 1.4: DRIVSCO low-level vision system. Two stereo cameras capture the scene and send it to the low-level-vision engine embedded in a FPGA based co-processor device. This engine extracts several features (energy, orientation, phase, motion and disparity) and sends them to a PC through a communication interface (red arrow)

requirements, however, were also limited by the memory and bandwidth constrains typical of any SoC.

In Figure 1.4 we can see how our system receives left-right image pairs and analyses the scene. This analysis consists on extracting several features using those input images. Each feature, however, needs to be stored in memory. As shown in the figure, these features need several bits per pixel to be represented (up to 24). Moreover, to extract them, our system needs to keep in memory several convolution masks[1] and, sometimes, multiple frames (such as when extracting optical flow). If we sum everything up, we need a big amount of memory. In addition, we need to transfer all extracted features back to the co-processing engine (a PC in our case) so higher-level features can be extracted. Let us make some actual numbers to provide a better idea of our system constrains.

Memory. Assuming all features are extracted by the SoC, which is not a trivial problem [85], we would need to store: several frames (up to

[1]See section 2.1

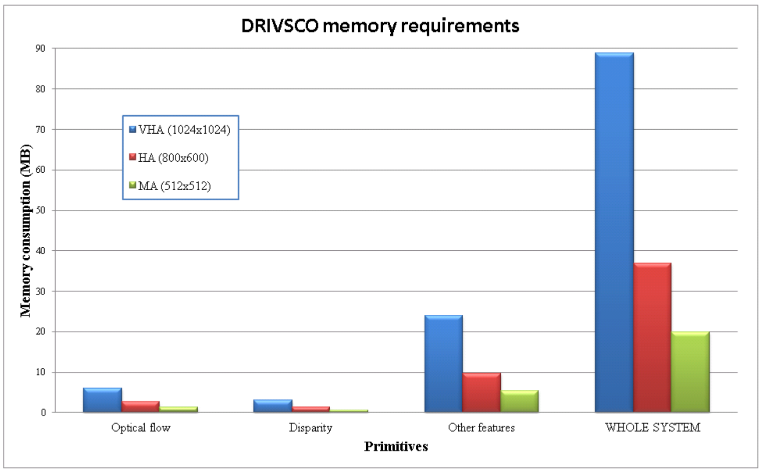


Figure 1.5: DRIVSCO memory constrains for three different image resolutions: VHA (1024x1024) in blue, HA (800x600) in red and MA (512x512) in green.

5 for the optical flow[2]) of the current stereo images; the calculated features[3] (energy, phase, orientation, disparity and optical flow), each of which is extracted using a multi-scale approach [75]; and several filters used in this process[4]. Fig. 1.5 shows the memory requirements of our system if we consider three different image resolutions: middle (512x512), high (800x600) and very high (1024x1024) for a 4-scale approach. As we can see, we would need up to 100 MB to store the images and features of our system. The hardware systems, however, typically have less than 50MB [93].

Bandwidth Similarly, if we assume we would need all the features to be transfered to higher-level stages of our system, we could calculate the bandwidth needs of DRIVSCO. Fig. 1.6 shows the bandwidth constrains we confront in our system for the same three input image resolutions and scales.

Seeing these constrains, we came to the conclusion that the integration step must include some kind of compression or condensation (see section 1.1.2 for details about the difference between these two concepts). Moreover, the majority of these mid- and high-level algorithms do not need a dense map to work; in fact, they usually perform a selection of interesting points [53]. Thus, the best solution would be one that integrates, regularizes and

[2]See section 2.4.2 for further details
[3]See chapter 2
[4]See section 2.1

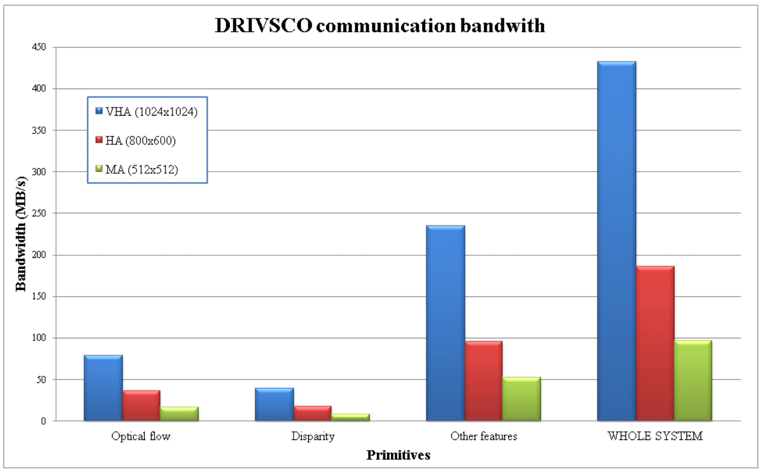


Figure 1.6: DRIVSCO bandwidth constrains for three different image resolutions: VHA (1024x1024) in blue, HA (800x600) in red and MA (512x512) in green.

selects the features before sending them to higher levels, just like the human visual system does.

Another important question is whether the co-processor will be able to handle so many data. Receiving and understanding (i.e. applying algorithms to extract higher-level features) all these features is fraught with complex computations and memory requirements that cannot be assumed by embedded processors. In fact, this workload will be excessive for many standard processors.

1.1.2 Condensation vs Compression

We have been talking about how efficient human visual system is and how nice it would be to emulate it. From a practical point of view, however, the main problem of our system are memory, bandwidth and workload constrains. Could we just solve it using a compression module?

Common compression techniques use color or gray levels and spatio-temporal information redundancy as reduction bases. When information is compressed without loss the data are grouped by taking into account gray or color levels, so they take up less space in the memory for transmission tasks; if there is any loss the number of bits associated to each pixel are reduced and their values are grouped on a statistical basis [78]. Nevertheless, these techniques merely take heed of the color/gray characteristics of the image, which is not enough if we want to integrate high-level-vision feedback.

Contrary to this purely compression method, we aim to fulfill memory and bandwidth requirements, whilst compensating for noisy estimations by locally regularizing the visual features obtained. We can find some approaches to a recovery algorithm that use a compressed input to regularize images [60]. These solutions, however, require a massive iterative processing that cannot be handled in our system. What drives us to another of our system problems: workload. Our approach has to reduce the co-processor computational requirements to allow embedded solutions to extract higher-level features.

Our best option is, therefore, not to compress, but to *condense* (i.e. select those areas that are relevant to our system depending on the current task). Our condensation algorithm is not only focused on sending and storing information at a minimum cost in terms of computational/hardware resources, but also selecting the most relevant data. Moreover, our condensation approach preserves plain areas and regularizes dense visual features. In addition, our algorithm allows the fusion of different vision information, facilitating a signal-to-symbol loop [72].

1.2 Main Goals

From the previous paragraphs we extract the following requirements that compose our main goals:

Condensation. We need to reduce memory and bandwidth requirements, as well as the ulterior computational load, by extracting relevant information from dense visual features and storing it efficiently.

Versatile. Our solution must condense any dense visual feature extracted by our vision system.

Hardware-friendly. We need to include our condensation module in an embedded solution, therefore it must be implemented following real-time constrains.

Feedback from higher levels. Attention and other information will be sent from higher levels and our module has to incorporate them in the condensation process.

Efficient representation. Higher stages of the vision system receive our condensed output as input. Our output, therefore, has to be easy to handle and facilitate feedback information from those higher stages.

Information recovery. Since we do not know a priori which information is needed by higher stages, we have to store enough data to recover anything they might need.

In this dissertation we present, explain and test our approach to fulfill these goals: *a semidense representation map* that condenses dense visual features into a sparser map, keeping most of the information they contain. This representation reduces memory and bandwidth whilst diminishing computational load.

1.3 Tools and Methods Used

The algorithms presented in this dissertation have been implemented in MATLAB in their first approach. We have chosen MATLAB because it allows for a quick implementation and testing of algorithms. This way, we have been able to test several solutions to our problem, adding versatility to the final version of our algorithm. Furthermore, MATLAB provides a perfect environment for visualizing images and other results. In addition, many of the mid-/high-level algorithms that would use our semidense representation map are implemented in MATLAB.

We cannot forget, however, DRIVSCO framework. Our algorithm needed to be implemented in a Hardware Description Language (HDL) in order to synthesize it and integrate it in our SoC. Thus, we have used Handel-C language and Mentor Graphics (previously known as Celoxica) programming environment to implement a functional version of our algorithm. Handel-C is a high level HDL, mainly used for FPGA programming, that includes all common C language features and a set of statements, types and expressions to control hardware instantiation, focusing on parallelism. The difference between Handel-C and other HDLs is the abstraction level. Handel-C facilitates a higher abstraction level when defining custom datapaths [35], speeding up design-implement cycle of complex algorithms, such as computer vision ones.

1.4 Dissertation Outline

We want to continue this dissertation with a short introduction to the main computer vision algorithms that we have used in our research. Hence, chapter 2 presents filtering operators, sparse and dense visual features, attention processes, and multi-modal descriptors (the most promising option to solve our problem within existing methods). In chapter 3 we justify why the existing options do not fulfill our requirements and present our solution: a semidense representation map. This chapter also includes different methods to obtain our semidense map, some benchmark examples and a study of its inherent regularizing capabilities. Chapter 4 presents several real-world applications of our innovative map: from attention systems (top-down and bottom-up) to driving scenarios (ground-plane and obstacle detection). In chapter 5 we discuss and explain the implementation of our condensation algorithm in specific hardware, i.e. in an FPGA, including resource consumption, performance and integration with DRIVSCO vision system. Finally, chapter 6 sums up the main contributions of our research and suggest future work.

Chapter 2

Introduction to Computer Vision

Before moving forward explaining our condensation algorithm, there are a few concepts about computer vision we would like to clarify. A wide explanation of each of them is out of discussion, although we want to present an outline. Hence, this chapter does not pretend to be a detailed explanation of visual features, edge extractors or filters. Many books can be found to extend the concepts and algorithms introduced here [22, 41, 87], but we include the description of the main ones used in our research.

In the previous chapter we have used the term *feature* to refer low-level visual primitives (such as the ones extracted by DRIVSCO system and shown in Figure 1.4). A more general definition of feature could be the one proposed by Trucco[87]:

> Image features (a.k.a visual features) are local, meaningful, detectable parts of the image.

From this definition, we deduce that an edge or a corner are features, as well as the disparity computed for a pixel in a stereo system. As previously mentioned, one of the main characteristics of our semidense map is that it condenses visual features, instead of images.

Vision features are usually obtained by applying filtering operations to an input image (as we explain in section 2.1). The extracted feature is usually represented as a map as big as the input image. This map can be dense, if almost every pixel has a valid value, or sparse, if only few of them have a valid value. From now on we use “dense feature” to refer to those features that present a dense representation map and, similarly, we use “sparse feature” to refer to those ones with a sparse map. The main goal of our condensation algorithm is to translate these “dense features” into a sparser representation.

In this chapter we introduce both kinds of features. First we include a quick sum-up of useful sparse features: energy, orientation, phase, edges, and local descriptors. Then, we introduce the main dense features extracted in the DRIVSCO system: disparity and optical flow.

In the literature we do not find many options to reduce dense features into sparse representations. In section 2.5 we introduce multi-modal descriptors, which are the most promising solution to our problem.

Another characteristic we want to include in our system is the capacity of integrating attention information in a traditional vision system. This attention is obtained thanks to saliency maps that are anything but sparse features. Due to their importance in our system, they deserve a independent subsection in this chapter.

As we already mentioned, most of these features are extracted using a filtering operation, so let us start explaining these operations.

2.1 Image Filtering Operations

In signal processing, a filter is a method to reduce or remove unwanted components of a signal. Depending on their use and implementation, we find different kinds of filters such as low- and high-pass, linear or non-linear, etc. In image processing, filters are mainly used to suppress either the high frequencies in the image, i.e. smoothing (see section 2.1.1), or the low frequencies. For simplicity we focus on *linear* filtering in this introduction, although some non-linear filters are also mentioned.

An image can be filtered either in the frequency domain or in the spatial domain. The first involves transforming the image into the frequency domain, multiplying it with the frequency filter function and re-transforming the result into the spatial domain. The filter function is shaped so as to attenuate some frequencies and enhance others. For example, a simple low-pass function is 1 for frequencies smaller than the *cut-off frequency* and 0 for all others.

The corresponding process in the spatial domain is to **convolve** the input image $f(i,j)$ with the filter function $g(i,j)$, obtaining $h(i,j)$. Since digital images are discrete, we can define the convolution as follows:

$$h(i,j) = g(i,j) \star f(i,j) = \sum_{k=1}^{n} \sum_{l=1}^{m} g(k,l) f(i-k, j-l) \tag{2.1}$$

Contrary to frequency domain where most of the filtering operations are linear, in spatial domain implementing non-linear filters is quite common.

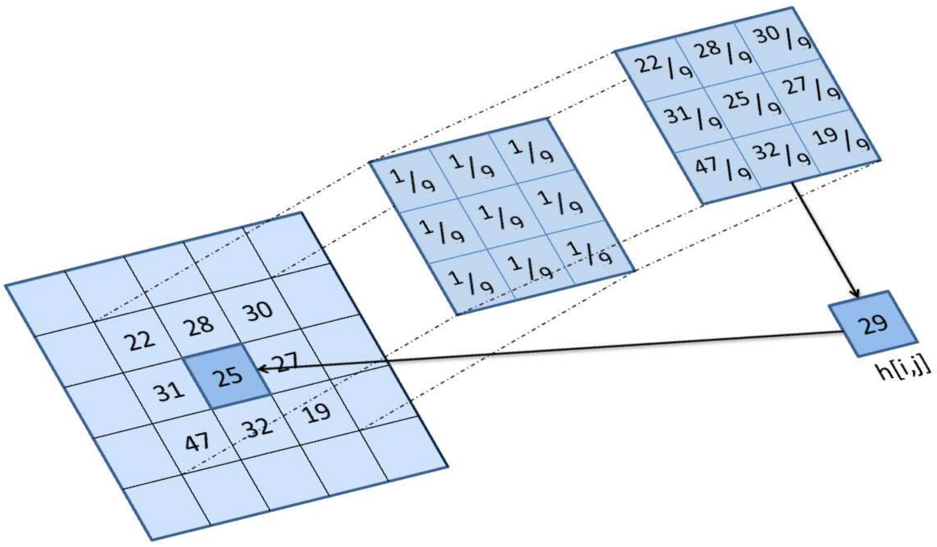


Figure 2.1: Convolution mask example. We apply a 3x3 convolution mask to an image to extract the average value of an element, i.e. we apply a mean filter using a convolution mask

In this case, (2.1) cannot be used as it is, and we need to define some kind of non-linear operator based on the same idea.

Many filters define a **convolution kernel** or **convolution mask** of a given size (also called window or neighborhood) and reduce the convolution to a 'shift and multiply' operation, where we shift the mask over the image and multiply its value with the corresponding pixel values of the image. In (2.1), the size of the convolution mask would be *mxn*. Fig. 2.1 shows an example of the convolution when using a convolution mask. Various standard masks exist for specific applications, where the size and the form of the kernel determine the characteristics of the operation [57].

Different steps of our contributions are based on filters, therefore, in next subsections we explain those we used more frequently. Again, this explanation is far from being complete, and we encourage the reader to follow the references for more details.

2.1.1 Image Smoothing

In computer vision, *noise* may refer to any entity, in images, data or intermediate results, that is not interesting for the purposes of the main computation [87].

There are different kinds of noise, such as white, Gaussian, salt and pepper, etc. Smoothing filters are generally used to reduce the noise in an image. These filters, however, can potentially remove some of the information in the image, losing some of the image detail. For example, step changes will be blurred into gradual changes, and the ability to accurately localize an edge will be sacrificed [41], such as when using a mean filter (see subsection 2.1.1.1). A spatially varying filter can adjust the weights so that more smoothing is done in a relatively uniform area of the image, and little smoothing is done across sharp changes in the image. In addition, the size of the neighborhood controls the amount of filtering. A larger neighborhood, corresponding to a larger convolution mask, will result in a greater degree of filtering. As a trade-off for greater amounts of noise reduction, larger filters also result in a loss of image detail [22].

2.1.1.1 Mean Filter

This is one of the simplest examples of a linear smoothing filter. The goal is to obtain, for each point, the average of its neighborhood. For a nxn neighborhood N with a total of M pixels, we can rewrite (2.1) as follows:

$$h(i,j) = \frac{1}{M} \sum_{(k,l) \in N} f(k,l). \tag{2.2}$$

Figure 2.1 shows an example of how to apply this filter using a convolution mask. The main limitations of this filter are [87]:

- Signal frequencies shared with noise are lost. This implies sharp variations are softened and, therefore, image is blurred.
- Impulsive noise, such as salt-and-pepper, is attenuated but not removed.

2.1.1.2 Gaussian Filter

A Gaussian blur (a.k.a. Gaussian smoothing) is a particular case of the averaging filter, in which the kernel is a 2-D Gaussian, as in:

$$g(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{2.3}$$

where σ determines the width of the Gaussian. A larger σ implies a greater smoothing. The main advantage of this filter is its separability. It means that convolving an image with a 2-D Gaussian kernel is the same as convolving first all the rows, then all the columns wit a 1-D Gaussian having the same standard deviation σ. Thanks to this property, the complexity increases linearly with the mask size, instead of quadratically, making the Gaussian filters very likely for a specific hardware implementation. Thus, we can rewrite 2.1 as follows:

$$h(i,j) = \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \underbrace{\left\{ \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f(i-k, j-l) \right\}}_{\textit{vertical convolution}}. \tag{2.4}$$

This filter is not appropriate for attenuating impulsive noise or working on neighborhoods with edges as the mean one was. Nevertheless, it is useful to reduce noise following a normal distribution where mean filter finds more trouble. This is mainly due to the distribution of weights in the convolution mask which are bigger in the center and, therefore, adapt better to that kind of noise, reducing blurring effect. If our images present both kinds of noise, we could solve it by using a non-linear or a more complex filter such as bilateral one.

2.1.1.3 Median Filter

This is the simplest example of a non-linear filter. Contrary to mean and Gaussian filters, this one suppresses impulsive noise and preserve the sharp variations of the image. The median filter runs through the feature pixel by pixel, replacing each pixel with the median of a window centered in that pixel [91]. The following pseudo code explains its behavior for an image I:

```
edge_x := (window_width / 2) rounded down
edge_y := (window height / 2) rounded down
   for i = edge_x to (I_width - edge_x)
       for j = edge_y to (I_height - edge_y)
           allocate neighborhood[window_width][window_height]
           for k = 0 to window_width
               for l = 0 to window_height
                   neighborhood[k][l] := I[i+k-edge_x][j+l-edge_y]
           sort all entries in neighborhood[][]
           median_filter_output[i][j] :=
            neighborhood[window_width / 2][window_height / 2]
```

2.1.1.4 Bilateral Filter

Another way to reduce the noise preserving the edges is using a combination of different linear filters. Bilateral filter is a normalized convolution in which the weighting for each pixel p is determined by the spatial distance from the center pixel c, as well as its relative difference in intensity [83]. By this, the edges are preserved whilst smoothing the noise. In the literature [91] the spatial and intensity weighting functions f and g are typically Gaussian. The bilateral filter is defined as:

$$H_c = \frac{\sum_{p \in N} f(p-c) g(I_p - I_c) I_p}{\sum_{p \in N} f(p-c) g(I_p - I_c)} \tag{2.5}$$

where I is the input image, H the output filtered image, and N the neighborhood centered in c [91]. Note that the weights depend not only on Euclidean distance but also on the radiometric differences (differences in the range, e.g. color intensity). This preserves sharp edges by systematically looping through each pixel and according weights to the adjacent pixels accordingly.

Bilateral filtering has been widely used in literature [83, 91, 3]. As we have mentioned, it smooths plain regions and reduces impulse and normal noises, while preserving edges. These are characteristics that any vision system will benefit from. Optimizing the filter output (i.e. adapting a bilateral filter to remove the noise of our images), however, is not as easy as it would be with a simpler filter.

2.1.1.5 Anisotropic Diffusion

The goal of this “filter” is also to reduce the noise preserving the significant parts of the image. The process is a bit more complicated than the ones already described. The idea is to create a scale-space, i.e. a parameterized family of successively more and more blurred images. Each of the resulting images in this family are given as a convolution between the image and a 2D-Gaussian filter, where the width of the filter increases with the parameter. This process is a *linear* and *space-invariant* transformation of the original image. In the anisotropic diffusion, however, each of the images of the created family is a combination between the original image and a filter that depends on the local content of the original image. As a consequence, anisotropic diffusion is a *non-linear* and *space-variant* transformation of the original image. Perona and Malik [66] defined it, for an image I, as:

$$\frac{\partial I}{\partial t} = div(c(x,y,t)\nabla I = \nabla c \cdot \nabla I + c(x,y,t)\Delta I \tag{2.6}$$

where Δ denotes the Laplacian[1], ∇ denotes the gradient, $div(...)$ is the divergence operator, and $c(x,y,t)$ is the diffusion coefficient. $c(x,y,t)$ controls the rate of diffusion and is usually chosen as a function of the image gradient

[1]Laplacian operator is a differential operator given by the divergence of the gradient of a function on Euclidean space [80]

so as to preserve edges in the image. Perona and Malik also proposed two functions for the diffusion coefficient:

$$c(\|\nabla I\|) = e^{-(\|\nabla I\|/K)^2} \tag{2.7}$$

and

$$c(\|\nabla I\|) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{K}\right)^2} \tag{2.8}$$

Where the constant K controls the sensitivity to edges and is usually chosen experimentally or as a function of the noise in the image.

Anisotropic diffusion is an iterative process: each time we apply this operator, the information is diffused. The more times we apply the operator, the more spread data are and, likewise, errors reduced. This means we can control our smoothing process by applying this operator a variable number of times. Nevertheless, this approach is too complicated to be implemented in an embedded system.

2.1.2 Bio-inspired Filtering

Our filtering operations are growing in complexity, becoming too complicated to be handled by our system. We need to find an approach that allows us to remove the error without losing important information (such as edges) and as simple as possible so we can implement it in diverse platforms.

If we focus on our visual system, we find the receptive fields of the neurons of the primary visual cortex (V1) group following a very handy pattern. They have an ON-center band that responds positively to light flanked by two OFF side bands that respond to darkness. This pattern responds especially well to rays of light that are oriented in a particular direction. The cells whose receptive fields thus respond to light with a specific orientation are called *simple cells* [47].

In literature we find oriented filters that emulate simple cells behavior. Here we present Gabor filters as an example of them.

2.1.2.1 Gabor Filters

Gabor filters are linear filters widely used for feature detection. An interesting characteristic is their similarity with the human visual system, as they model the simple cells of the visual cortex. These filters are basically a Gaussian kernel modulated by a sinusoidal plane wave. Let us define the

1-D Gabor filter in the spatial domain to better understand this:

$$G(x - x_0) = e^{\frac{-(x-x_0)^2}{2\sigma^2}} \cdot e^{i\omega_0(x-x_0)}, \tag{2.9}$$

where x_0 is the spatial location of the filter, and ω_0 is the frequency of the harmonic component which will be the central spatial frequency of the power spectrum [76]. The Fourier Transform (FT) has the same functional form:

$$g(\omega - \omega_0) = e^{\frac{-(\omega-\omega_0)^2}{2\tau^2}} \cdot e^{ix_0(\omega-\omega_0)}, \tag{2.10}$$

where σ and τ are the spatial half-width and spatial frequency half-bandwidth of the filter, and the product $\sigma\tau$ is 1. This is the theoretical minimum for all complex valued linear filters [30]. The real (even) and imaginary (odd) components of the filter are given by:

$$G_{even}(x - x_0) = e^{\frac{-(x-x_0)^2}{2\sigma^2}} \cdot cos(\omega_0(x - x_0)), \tag{2.11}$$

$$G_{odd}(x - x_0) = e^{\frac{-(x-x_0)^2}{2\sigma^2}} \cdot sin(\omega_0(x - x_0)). \tag{2.12}$$

And, therefore, we can define a set of Gabor filters with different frequencies and orientations as the one shown in Fig. 2.2.

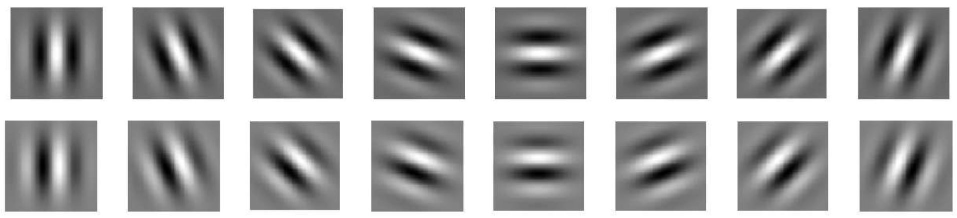


Figure 2.2: Example of a 8-orientation Gabor filter bank. First row shows the even filters and second row the odd filter for each orientation.

This kind of sets is very helpful for extracting useful features. In fact, just by combining the odd and even responses of Gabor filters we obtain magnitude and orientation of an image [75], which are the base to extract more complex features such as disparity, edges, etc.

2.2 Sparse Visual Features

Filters are not only useful to reduce the noise in an image or feature, but also provide a way to enhance their *meaningful* parts. By meaningful parts we understand all parts of the image that are associated to scene elements, such as edges, corners, surfaces, etc. There are many algorithms in the literature [54, 48, 9, 70] focused on extracting *descriptors*, i.e. describing a sparse representation of an image by selecting some of those meaningful parts. Most of them, however, involve at least one filtering step.

The visual cortex, as we have already mentioned, has evolved effectively to cope with visual information. This efficiency is critical for our survival, therefore, natural systems has discovered coding strategies for representing visual information [47]. Just like the simple cells capture oriented information (as explained in section 2.1.2.1), other visual cortex cells reduce all the environment information into sparse representations, removing redundant and non-relevant data. This reduction allows our brain to handle important information with a higher priority. These representations, however, contain enough information for our brain to recover any detail we might need [82].

In computer vision literature, we find several sparse visual features. Some of them aim to emulate the visual cortex, processing the scene and extracting useful information such as phase, orientation and energy. Other algorithms, on the other hand, just point which areas of the image are more likely to contain important information. Edge detectors and intrinsic dimension are example of this. More complex descriptors, such as SIFT, aim to define invariant sparse features to characterize a scene and facilitate object recognition and matching.

Our condensation algorithm uses sparse visual features to provide relevant information to our semidense representation map. Moreover, our representation map is the perfect framework to easily integrate several of these features as we explain in section 3.1.

2.2.1 Energy, Orientation and Phase

Based just on their response to even and odd Gabor filters, images can be characterized by local features such as energy (a.k.a. magnitude), orientation and phase. At this point, energy refers to the change in the intensity of the image, i.e. the boundaries of the image (see section 2.2.2 for details). As previously mentioned, oriented filters, such as Gabor, efficiently extract orientation information. Nevertheless, if we just consider 8 oriented filters (as the ones in Fig. 2.2), is likely that the local orientation of some objects do not fit this discrete number of orientations. This is why we need to interpolate filter outputs to estimate a more accurate orientation. Phase, on the other hand, has been proved to be robust to scene variations in contrast and brightness, as well as small affine distortions [84]. Local phase also requires an interpolation process similar to the one followed by the orientation to estimate a more accurate feature.

Different methods can be used to extract these sparse features. If we use an oriented filter h_i whose angle is defined as $angle = i \star \pi/N$, we extract real(c_i) and imaginary (s_i) response:

$$h_i = c_i + js_i. \tag{2.13}$$

Using them, we define the magnitude E_i and the phase P_i as:

$$E_i = c_i^2 + s_i^2 \tag{2.14}$$

$$P_i = arg(c_i, s_i) \tag{2.15}$$

After applying the 8 oriented filters in each point, we calculate the final energy, phase and orientation of that point using one of the following methods:

i Winner-take-all (WTA). For each pixel we take the phase, energy and orientation of the filter with maximum energy.

$$E_{local} = E_{max} \qquad P_{local} = P_{max} \qquad \theta_{local} = \theta_{max} \tag{2.16}$$

ii Weighted-average:

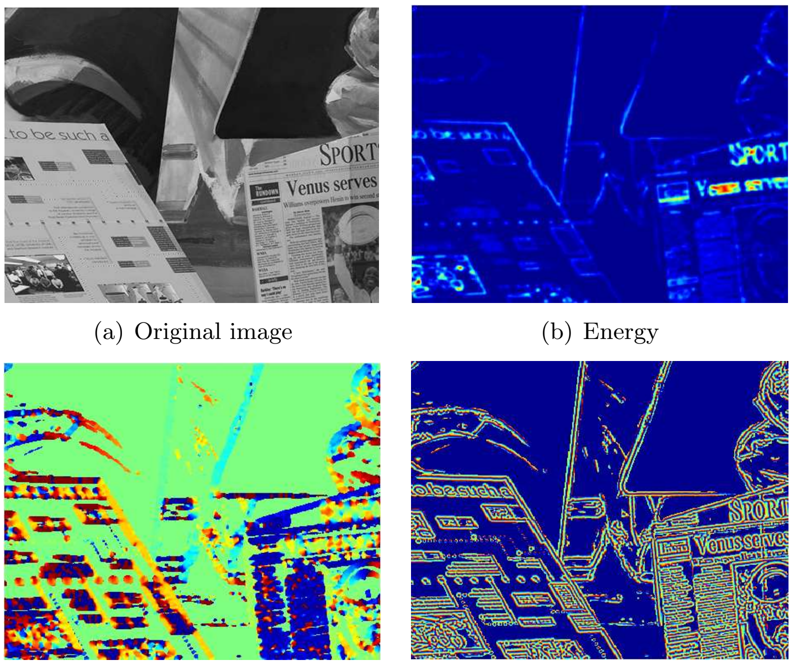
$$E_{local} = \sum_i E_i^N \qquad P_{local} = \frac{\sum_i P_i E_i}{\sum_i E_i} \qquad \theta_{local} = \sum_i \frac{\theta_i E_i}{\sum_i E_i} \tag{2.17}$$

where all angles are properly shifted for avoiding angle wrapping effects.

iii Tensor-based method. Based on a local tensor that projects the different orientations, information can be computed as follows (where j stands for the complex unit):

$$\begin{aligned}
E_{local} &= \sum_i E_i^N && (2.18)\\
\theta_{local} &= \frac{1}{2}\arg\left(\sum_i \frac{4}{3}\sqrt{c_i^2 + s_i^2}\exp(j2\theta_i)\right) && (2.19)\\
P_{local} &= \arctan\left(\frac{s}{c}\right) && (2.20)\\
c &= \sum_i c_i \cos^2\theta_i - \theta_{local} && (2.21)\\
s &= \sum_i s_i \cdot sign\cos(\theta_i - \theta_{local}) \cdot \cos^2(\theta_i - \theta_{local}) && (2.22)
\end{aligned}$$

In DRIVSCO framework we based our approach on iii because it is independent from filtering stage (Gabor) and it achieves a high accuracy.



(a) Original image (b) Energy

(c) Orientation (d) Phase

Figure 2.3: (b) Energy, (c) orientation and (d) phase extracted from (a) using [25]

However, a hardware simplification was made and magnitude and phase are calculated as follows:

$$E_{local} = \sqrt{\frac{\sum_{\theta=1}^{N} E_\theta}{N}} \tag{2.23}$$

$$P_{local} = atan2(\sum_\theta C_\theta, \sum_\theta S_\theta) \tag{2.24}$$

Where N is the number of different θ orientations and E_θ is the energy calculated for the orientation θ:

$$E_\theta = C_\theta^2 + S_\theta^2 \tag{2.25}$$

Local orientation is calculated starting from mean energy along orientations:

$$\theta_{local} = \frac{1}{2} atan2^2(\sum_\theta E_\theta \sin 2\theta, \sum_\theta E_\theta \cos 2\theta) \tag{2.26}$$

In Figure 2.3 we can see an example of these sparse visual features extracted using the DRIVSCO algorithm [85] from Middlebury benchmark image *venus* [79].

[2]atan2 computes the arctangent of y / x given y and x, but with a range of $(-\pi, \pi]$

2.2.2 Edges

Edges are pixels around which the image values undergo a sharp variation [87]. However, we prefer to consider edges as a connected chain of those sharp variations, i.e. boundaries between regions of an image. Defined like this, edges are pretty similar to magnitude. The main difference between them is that edges entail an ulterior selection process. Hence, we can understand magnitude as a first step in edge detection.

A more general definition describes edge detection as a two-step process. First we extract the discontinuities using magnitude (i.e. Gabor filters), derivatives of a Gaussian, convolution masks, etc. Then, we select those points that we consider more promising as edges by thresholding, edge thinning, or even non-linear methods. The output of an edge detector is a binary map where each pixel is marked as either an edge pixel or a non-edge pixel.

This sparse visual feature is pretty handy in computer vision, as we can use it to segment the image, detect objects, calibrate a camera, analyze the motion in a sequence, etc.

One of the main problems when detecting edges is the noise. As noise can cause intensity variations, a noisy input could cause false positives in the output. This is why most of the algorithms that detect edges need to perform a smoothing step. Smoothing, as we have already mentioned, can potentially remove discontinuities, and therefore, edges.

There are multiple algorithm that detect edges: Canny, Sobel, Prewitt, Robert Cross, etc. However, we are going to focus on the first two: Canny, the most used one, based on Gaussian filters, and Sobel, based on applying small separable convolution masks to the vertical and horizontal components.

2.2.2.1 Canny Edge Detector

Canny's [16] aim was to discover the optimal edge detection algorithm, i.e. an algorithm fulfilling these characteristics:

- Good detection: the algorithm should minimize the number of false positives detected, whilst marking as many real edges in the image as possible.
- Good localization: edges marked should be as close as possible to the edge in the real image.
- Single response: A given edge in the image should only be marked once, i.e. minimize the number of local maxima around the true edge created by the noise.

To satisfy these requirements Canny used the calculus of variations, defining a function that optimized them. This function, however, can be approximated by the first derivative of a Gaussian with only a slight error from the original output [87]. Nevertheless, this trade-off solution introduces some noise in the localization and single response requirements. Hence we need to introduce some steps to solve this noise: nonmaximum suppression and thresholding.

Noise reduction and Boundary Detection The Canny edge detector is susceptible to noise present on input images, so to begin with, the input image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

Once reduced the error, we need to extract the edges of the image. An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. Another approach could be to compute the gradient of the image evaluating the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient magnitude and direction can be determined:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2} \tag{2.27}$$

$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right). \tag{2.28}$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

In our low-level vision system, however, we do not need gradient as we already computed orientation and magnitude of the image intensity using Gabor filters (as explained in section 2.2.1).

Nonmaximum suppression In this step, we compare the strength ($\mathbf{G}$) of the current point with its neighbors. At this step we only consider neighbors those points following the same direction as current point, i.e. a maximum of two neighbors. If current point's magnitude is smaller than at least one of this two neighbors, we consider it was not a maximum and we suppress it by changing it magnitude to 0.

Thresholding with Hysteresis The output from the nonmaximum suppression step still contains the local maxima created by noise and background regions. Large intensity gradients are more likely to correspond to

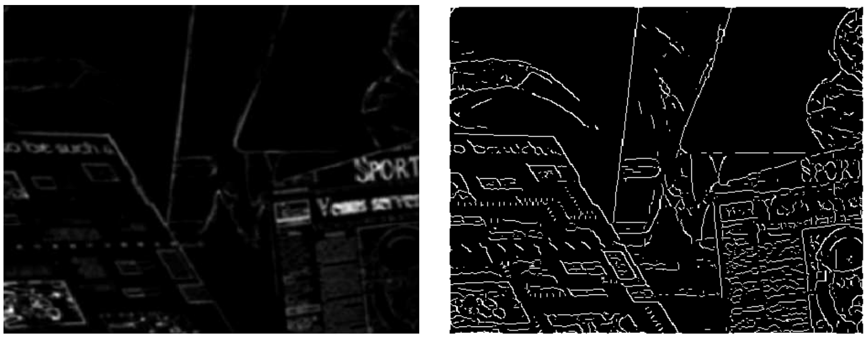


Figure 2.4: Canny detector applied to Middlebury's venus image (Fig. 2.3(a)). (a) shows the edges detected filtering the image. (b) shows the result after applying non-maximum suppression and hysteresis thresholding to (a).

edges than small intensity gradients. Nevertheless, it is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

We can assume that important edges should be along continuous curves in the image. Thus if we follow a faint section of a given line, we can discard a few noisy pixels that do not constitute a line but have produced large gradients.Thresholding with hysteresis requires two thresholds: high and low. The high threshold marks out the points most likely to be part of an edge. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

In Figure 2.4 we can see an example of Canny edge detector using *venus* sequence. We can see the output from the first stage of the algorithm (Fig. 2.4(a)), i.e. after the noise/enhacing step. We can also see the final binary map corresponding to the algorithm output (Fig. 2.4(b)).

2.2.2.2 Sobel Edge Detector

Mathematically less accurate than Canny, this edge detector is based on Sobel's discrete differentiation operator that computes an approximation of the gradient of the image intensity function. This operator, although inaccurate, is of sufficient quality to be of practical use in many applications.

At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. This operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

The operator approximates the derivatives using two 3×3 convolution kernels, one for horizontal changes and one for vertical, and applying them to the original image I. If we define G_x and G_y as the horizontal and vertical derivative approximations, the computations are as follows:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \star \mathbf{I} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \star \mathbf{I}. \tag{2.29}$$

The x-coordinate is here defined as increasing in the "right"-direction, and the y-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude and orientation, using 2.27.

Several improvements to this algorithm are found in literature, such as Logarithmic Image Processing (LIP-Sobel)[43] or Parameterized LIP (PLIP-Sobel)[92], that increase Sobel's robustness to local intensity changes.

In fact, LIP-Sobel implementation is very handy in real-world scenes where there are multiple situations with different brightness ranges (such as dusk or dawn)[56]. This model defines a logarithmic space where image intensity can be completely represented by a gray-level function $\widetilde{f}$. The obtained values are real numbers within the range $[0, M)$, where M is positive (for instance, in 8-bit gray-scale images M is 255). In the linear space to convolve an area A such as:

$$A = \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix} \tag{2.30}$$

we use the convolution mask G_x defined in (2.29) (and similarly with G_y). In the logarithmic space, on the other hand, we apply the operator L defined as follows [56]:

$$L = M - M \left(\frac{\widetilde{f_3}\widetilde{f_6}^2\widetilde{f_9}}{\widetilde{f_1}\widetilde{f_4}^2\widetilde{f_7}} \right) \tag{2.31}$$

where $\widetilde{f_i} = M - f$.

LIP-Sobel edge detection is robust to small local intensity changes in the image, working better than the linear Sobel operator in real-world conditions.

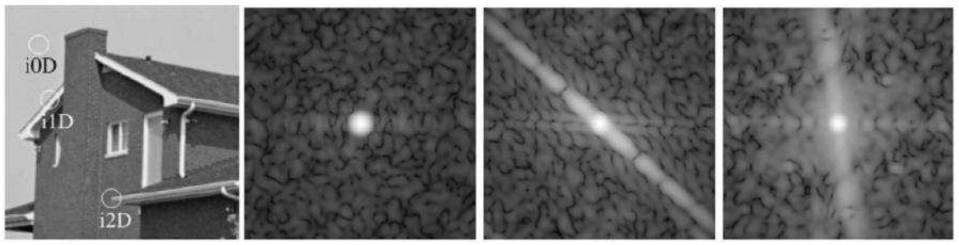


Figure 2.5: Illustration intrinsic dimensionality. In the image on the left, three neighborhoods with different intrinsic dimensionalities are indicated. The other three images show the local spectra of these neighborhoods, from left to right: i0D, i1D, and i2D. Figure extracted from [45].

2.2.3 Intrinsic Dimension

Not only edges present sharp variations but also corners (or junctions) and textures. However, when interpreting these areas there are some concepts that might not be applicable for all of them. For example, the idea of orientation does make sense for edges but not for a junction or most textures. Hence, before we apply concepts like orientation or position, we want to classify image patches according to their junction-ness, edge-ness or homogeneous-ness. The intrinsic dimension has proven to be a suitable descriptor in this context. Homogeneous image patches have an intrinsic dimension of zero (i0D), edge-like structures are intrinsically 1-dimensional (i1D) while junctions and most textures have an intrinsic dimension of two (i2D)[45].

Krüger and Felsberg [45] define the intrinsic dimension (a.k.a. intrinsic dimensionality) based on the spectrum of the image. In figure 2.5 we can see how:

- if the spectrum is concentrated in a point, the image patch has an intrinsic dimensionality of null (i0D),
- if the spectrum is concentrated in a line, the image patch has an intrinsic dimensionality of one (i1D), and
- otherwise the image patch has an intrinsic dimensionality of two (i2D).

Each of these three cases can be characterized more vividly. Constant image patches correspond to i0D patches. Edges, lines, and sinusoid-like textures obtained by projecting 1D functions correspond to i1D patches. All other structures like corners, junctions, complex textures, and noise correspond to i2D patches.

Nevertheless, a classification like this, based on a local image patch without taking the context into account, always faces the problem of the high degree of ambiguity of visual information [52]. Taking into account this, [45] assess confidences instead of binary decisions.

Hence, image structure can be classified only by extracting its spectrum and matching the different regions as previously mentioned. The spectrum of the image can be extracted using oriented Gabor filters (see section 2.1.2.1 for details).

2.2.4 Local Descriptors

Although useful, previously mentioned sparse features are not enough on real-world sequences. On those sequences we need features at least partially invariant to illumination, 3D projective transforms, and common object variations. These features must also be sufficiently distinctive to identify specific objects among many alternatives [48]. Moreover, mid- and high-level visual feature extraction, such as object recognition, could benefit from sparse features unaffected by nearby clutter or partial occlusion. This is why local photometric descriptors are computed for interest regions.

As a general rule, local descriptors extract interest points, similarly to the previous algorithms. The most valuable property of an interest point detector is its repeatability, i.e. whether it reliably finds the same interest points under different viewing conditions [9]. Next, the neighborhood of every interest point is represented by a feature vector, the descriptor. This descriptor has to be distinctive and, at the same time, robust to noise, detection errors, and geometric and photometric deformations.

The evaluation of the descriptors is performed in the context of matching and recognition of the same scene or object observed under different viewing conditions [54]. The best descriptors are those that are invariant to noise and to changes in scale, rotation and brightness. We present here some well-known descriptors.

2.2.4.1 SIFT

Lowe's patented method [48] can robustly identify objects even among clutter and under partial occlusion, because his scale-invariant feature transform (SIFT) descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. This approach is based on a model of the behavior of complex cells in the cerebral cortex of mammalian vision.

As many other local descriptors, SIFT was initially designed to solve object recognition and matching problems. SIFT key points (a.k.a. interest points) of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence [48].

The main stages of SIFT algorithm are the followings:

- **Key localization or detection**. To achieve rotation invariance and a high level of efficiency, Lowe selects key locations at maxima and minima of a difference of Gaussian function applied in scale space. This is a computationally efficient approximation of the Laplacian of Gaussian (LoG). This can be computed very efficiently by building an image pyramid with resampling between each level. Specifically, a DoG image $D(x, y, \sigma)$ is given by

  $$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma), \tag{2.32}$$

  where $L(x, y, k\sigma)$ is the convolution of the original image $I(x, y)$ with the Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$, i.e.,

  $$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{2.33}$$

  Hence a DoG image between scales $k_i\sigma$ and $k_j\sigma$ is just the difference of the Gaussian-blurred images at those scales.

- **SIFT key stability**. To characterize the image at each key location, the smoothed image L at each level of the pyramid is processed to extract image gradients and orientations. At each pixel, $A_{i,j}$, the image gradient magnitude, $M_{i,j}$, and orientation, $\theta_{i,j}$, are computed using pixel differences:

  $$M(i, j) = \sqrt{\left(L(x+1, y) - L(x-1, y)\right)^2 + \left(L(x, y+1) - L(x, y-1)\right)^2} \tag{2.34}$$

  $$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \tag{2.35}$$

Each key location is assigned a canonical orientation so that the image descriptors are invariant to rotation. In order to make this as stable as possible against lighting or contrast changes, the orientation is determined by the peak in a histogram of local image gradient orientations, i.e. the descriptor is normalized by the square root of the sum of squared components.

- **SIFT descriptor** is a 3D histogram of gradient locations and orientations. First a set of orientation histograms are created on 4x4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16 x 16 region around the key point such that each histogram contains samples from a 4 x 4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are 4 x 4 = 16 histograms each with 8 bins the vector has 128 elements.

Due to its stability and invariance, this descriptor is useful to detect not only objects, but also any other feature based on relevant points. Therefore, these descriptors have been widely used in robot localization and mapping, 3D scene modeling and tracking, etc [54].

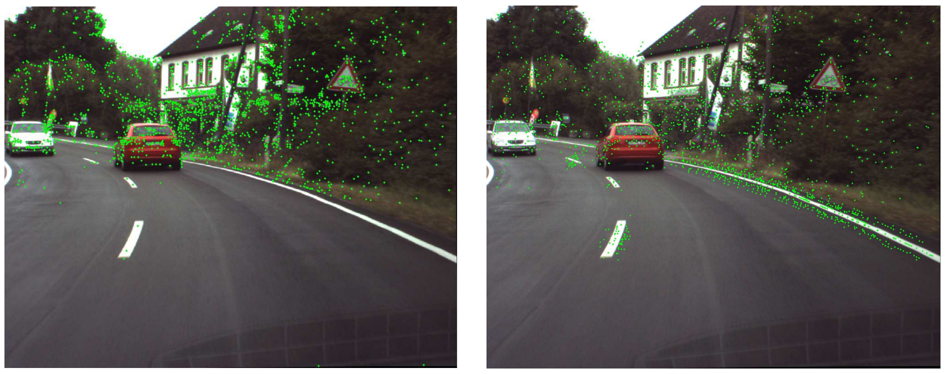
2.2.4.2 SURF

The good performance of SIFT made it the base of many other descriptors. SURF (Speeded Up Robust Features) is one of them.

The differences between SURF and SIFT are the region detector and the descriptor. On one hand, instead of using Lowe's DoG approximation to LoG, SURF defines a Hessian matrix based detector, called 'Fast-Hessian' detector. The descriptor, on the other hand, describes a distribution of Haar-wavelet responses within the interest point neighborhood. Moreover, only 64 dimensions are used, reducing the time for feature computation and matching, and increasing simultaneously the robustness. In addition, SURF includes a new indexing step based on the sign of the Laplacian, which increases not only the matching speed, but also the robustness of the descriptor.

Figure 2.6 compares these two algorithms' outputs[3] in a driving scenario image, extracted from [8].

[3]Descriptors extracted using L.Rubio and F.Naveros code.



(a) SIFT Local Descriptors (b) SURF Local Descriptors

Figure 2.6: SIFT and SURF local descriptors obtained using a real-world image (resolution 1280x1024). The number of points obtained by SIFT is 2511 while SURF extracts 2057.

2.2.4.3 Other Descriptors

In [54] we find a sum-up of other SIFT based descriptors:

Principal Components Analysis applied to SIFT (PCA-SIFT) is a vector of image gradients in x and y direction computed within the support region. The gradient region is sampled at 39 x 39 locations, therefore,the vector is of dimension 3,042. The dimension is reduced to 36 with principal components analysis [42].

Gradient location-orientation histogram (GLOH) is an extension of the SIFT descriptor designed to increase its robustness and distinctiveness. It computes the SIFT descriptor for a log-polar location grid with three bins in radial direction (the radius set to 6, 11, and 15) and 8 in angular direction, which results in 17 location bins. The gradient orientations are quantized in 16 bins. This gives a 272 bin histogram. The size of this descriptor is reduced with PCA.

Shape context is similar to the SIFT descriptor, but is based on edges. Shape context is a 3D histogram of edge point locations and orientations. Edges are extracted by the Canny detector (see section 2.2.2). Location is quantized into nine bins of a log-polar coordinate system with the radius set to 6, 11, and 15 and orientation quantized into four bins (horizontal, vertical, and two diagonals). We therefore obtain a 36 dimensional descriptor. Many implementations also weight a point contribution to the histogram with the gradient magnitude.

2.3 Saliency Maps: Attention Processes

Saliency maps are based on primate visual system. Primates have a remarkable ability to interpret complex scenes in real time, despite the limited speed of the neuronal hardware available for such tasks. Intermediate and higher visual processes appear to select a subset of the available sensory information before further processing, most likely to reduce the complexity of scene analysis. This selection is implemented in the form of a spatially circumscribed region of the visual field, the so-called ’focus of attention’, which scans the scene both in a rapid, bottom-up, saliency-driven, and task-independent manner as well as in a slower, top-down, volition-controlled, and task-dependent manner [38].

Itti and Koch proposed this bottom-up and top-down duality in the visual cortex processing [39]. They also suggested that the bottom-up process extracts those stimuli that are intrinsically salient in a context, obtaining bottom-up saliency maps. The second process is voluntary, it means it has a selection criteria that changes depending on the target application. This top-down process is as a feedback from higher stages of the visual system and confers it a high power. Nevertheless, it is approximately slower than the first one (circa 8x) [89]. Due to this property, this process is usually avoided for the most efficient implementations [67].

Their algorithm extracts a saliency value for each point of the image. Thus, to obtain a sparse feature from a saliency map, we just fix a threshold and only compute the most salient points. This approach, therefore, has been proved biologically accurate and can dramatically reduce the workload as is demonstrated in human visual system [38], but less interesting areas are usually ignored. Moreover, their implementation does not take into account top-down information.

2.3.1 Bottom-Up Saliency Maps

Figure 2.7 shows how the bottom-up saliency maps are extracted in [67].

First, low-level-vision features (color channels tuned to red, green, blue and yellow hues, orientation and magnitude) are extracted from the original color image at several spatial scales (i.e. Gaussian pyramids), using linear filtering.

Then, each feature is computed in a center-surround structure akin to visual receptive fields. Center-surround operations are implemented in the model as differences between a fine and a coarse scale for a given feature. This way seven types of features, for which wide evidence exists in mammalian visual systems, are computed: one encodes on/off image intensity

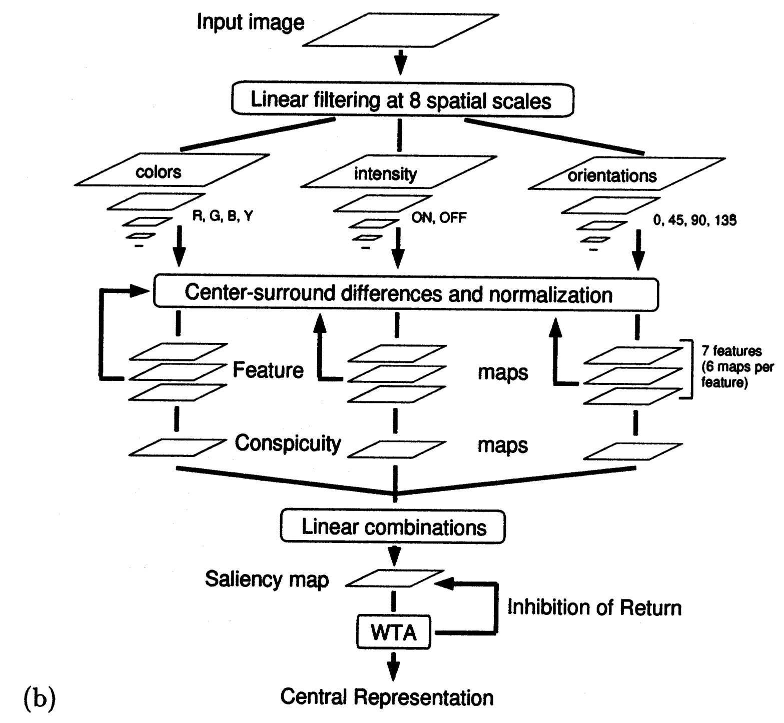
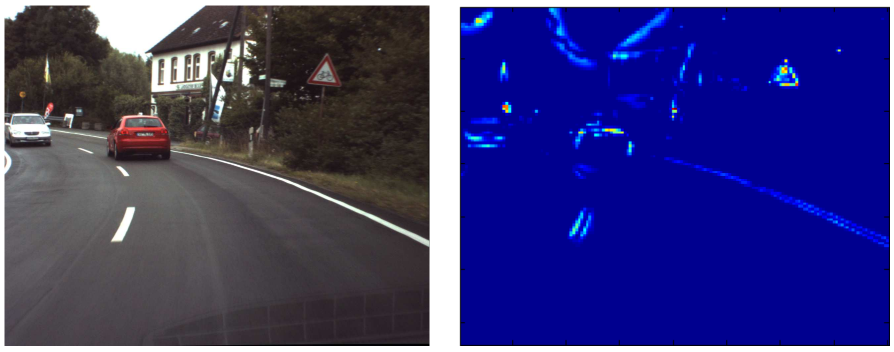


Figure 2.7: Itti and Koch saliency extraction model [67]. Visual features are computed using linear filtering at eight spatial scales, followed by center-surround differences, which compute local spatial contrast in each feature dimension for a total of 42 maps. An iterative lateral inhibition scheme instantiates competition for salience within each feature map. After competition, feature maps are combined into a single 'conspicuity map' for each feature type. The three conspicuity maps then are summed into the unique topographic saliency map. The WTA module detects the most salient location and directs attention towards it. An inhibition-of-return mechanism transiently suppresses this location in the saliency map, such that attention is autonomously directed to the next most salient image location. Image extracted from [40].

contrast, two encode for red/green and blue/yellow double-opponent channels and four encode for local orientation contrast [67].

From these features, we need to extract a unique saliency map. Each feature map is first normalized (between 0 and 1). Then, each map $\mathcal{M}$ is iteratively convolved by a large 2-D DoG filter, the original image is added to the result, and negative results are set to zero after each iteration:

$$\mathcal{M} \leftarrow |\mathcal{M} + \mathcal{M} \star DoG - C_{inh}|_{\geq 0} \tag{2.36}$$



(a) Input image (b) Saliency Map

Figure 2.8: Saliency map of a driving scenario image. The saliency of the points is represented using a color scale. Most salient points are marked in red and less interesting areas are dark blue.

where DoG is the 2D difference of Gaussian filter described above, $||_{\geq 0}$ discards negative values, and C_{inh} is a constant inhibitory term that introduces a small bias towards slowly suppressing regions that typically correspond to extended regions of uniform textures, which we would not consider salient. This process is iteratively repeated several times, and then we obtain three different conspicuity maps: one for intensity, one for color and one for orientation (see Fig. 2.7).

Finally, these three conspicuity maps are summed into an unique saliency map. Following a WTA approach, the most salient stimuli are then sent to be processed and then inhibited using a return signal. This process is repeated for all the points in the saliency map under a given threshold.

Figure 2.8 shows the saliency map output of a real-world image. In this representation, the most salient stimuli are marked with red or yellow. Less important areas are represented in dark blue. We could use a threshold to obtain a sparse saliency map. If we compare figures 2.6 and 2.8, we see how we obtain different information, but the most salient parts are always represented.

Sparse features, although useful to extract other higher level features, are not enough to achieve a complete understanding of the scene. This understanding will benefit from denser features.

2.4 Dense Visual Features

The goal of dense visual features is to provide information for each point of the image, no matter whether they are redundant. These features, contrary to sparse ones, are considered better when they obtain a denser representation. In the framework of these dissertation, disparity and optical flow are considered.

2.4.1 Disparity

Known as disparity, depth, stereoscopy (or just stereo), this visual feature calculates the distance of various points in the scene relative to the position of the camera. This task is useful at several levels in computer vision, from calibrating a camera system to providing information to TTC algorithms, for instance.

A common method for extracting such depth information from intensity images, such as the ones we are studying, is to acquire a pair of images using two cameras displaced from each other by a known distance. As an alternative, two or more images taken from a moving camera can also be used to compute depth information. In contrast to intensity images, images in which the value at each pixel is a function of the distance of the corresponding point in the scene from the sensor are called range images. Range images, however, are out of the scope of this dissertation [41].

In the literature we find many ways to extract disparity information from intensity images. DRIVSCO system, as shown in figure 1.2, captures the scene using two cameras separated a known distance. This is why we are focus on this approach. Even within this approach several algorithms can be found. In [25], Díaz analyzes different techniques to compute disparity using this approach. Table 2.1 includes a sum-up of his analysis. Local techniques are those that estimate disparity at a point by comparing a small region around that point in an image with a series of small regions extracted from the other image. Global ones, on the other hand, try to compute the disparity field based on prior assumptions [73].

When working on real camera systems, input images contain distortions due to the capturing system. This is why the images must first be corrected for distortions, such as barrel distortion. Then we need to project the image back to a common plane to allow comparison of the image pairs, known as image rectification [73].

As this chapter is merely an introduction, we present local-phase-based technique as an example of disparity extraction method and we assume input images are rectified. We are focusing on this approach because it is hardware-friendly (since it is local) and robust to intensity changes, a very useful trait when working in a real-world scenario. In fact, those characteristics make this technique perfect to be implemented in DRIVSCO framework. However, before moving to the example, let us point out the main problems of the stereo systems.

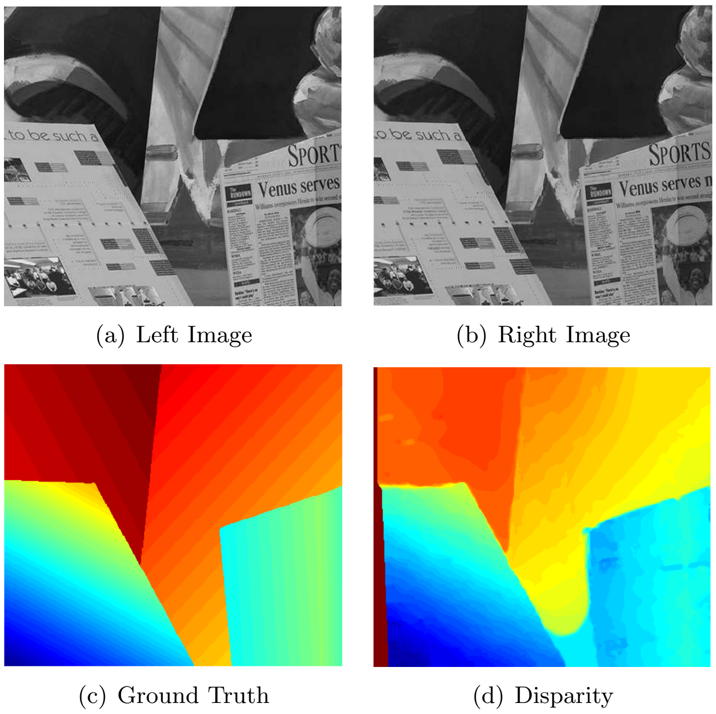


Figure 2.9: Disparity example. (a) and (b) are the left and right original images from Middlebury's ´venus´ sequence. (d) is the disparity calculated using [73] where warm colors codify closer objects and cold colors, farther objects.

Figure 2.9 shows an example of a pair of images from Middlebury benchmark [79], the disparity calculated using Ralli et al. algorithm [73] and the real stereo information of the scene (ground-truth). Warm colors (like red or orange) codify closer objects while cold colors encode farther objects.

2.4.1.1 Stereo System Problems

A stereo system must solve two problems [87]:

Correspondance Determine which item in the left eye corresponds to which item in the right eye. Moreover, some parts of the image are only visible with one eye.

Reconstruction Our brain interprets the scene thanks to the disparity map between corresponding items. If the geometry of the scene is known, our brain translate the disparity map into a 3-D reconstruction.

Technique	Description
	Local Methods
Region-Based Matching	Search for maximum match score or minimum error over a small region, typically using variants of crosscorrelation or robust rank metrics.
Differential	Minimize a functional, typically the sum of squared differences, over a small region.
Energy- or Phase-Based	Analyze the image on the Fourier domain focusing on the energy or phase of the signal after convolving with quadrature filters.
Feature Matching	Match specific features rather than intensities themselves.
	Global Methods
Dynamic Programming	Determine the disparity surface for a scanline as the best path between two sequences of ordered features. Typically, order is defined by the epipolar ordering constraint.
Global optimization	Determine the disparity surface as energy minimization process.

Table 2.1: Local and Global Disparity Techniques

2.4.1.2 Phase-Based Disparity

As already mentioned, phase-based algorithms rely on structure information instead of intensity. This is why this approach is unbiased to luminance changes and behaves better against affine transformations (for instance due to different camera perspectives) [25].

As a general approximation, a mono-scale phase-based algorithm follows these steps:

1. Even (C) and odd (S) filtering of left and right images (we could use Gabor filters as in section 2.1.2.1).

2. For each orientation, compute the phase difference using (2.37) and apply a thresholding operation.

3. Chose of final disparity estimation between different orientations (for instance, median value).

For oriented filters, the phase difference has to be projected on the epipolar line. Since we assume rectified images, this is equal to the horizontal [63].

For a filter at orientation θ_q, the disparity is obtained as follows:

$$\delta_q(x) = \frac{\left[\phi_q^L(x) - \phi^R(x)_q\right]_{2\pi}}{\omega_0 cos\theta_q} \tag{2.37}$$

where ϕ_q^L and ϕ_q^R are the left and right image phases, $[]_{2\phi}$ depicts reduction to the $]-\pi;\pi]$ interval, and ω_0 represents the filter peak frequency.

2.4.2 Optical Flow

Horn and Schunck [37] defined optical flow as follows: *"The optical flow is a velocity field in the image which transforms one image into the next image in a sequence. As such it is not uniquely determined. The motion field, on the other hand, is a purely geometric concept, without any ambiguity it is the projection into the image of three-dimensional motion vectors."*

As we did in the disparity section, we focus on a phase-based approach to solve this problem. Finally we introduce the problems found when computing optical flow.

2.4.2.1 Phase-Based Optical Flow

Starting with the convolution output (from several oriented filters), we calculate the phase for each orientation. Then we extract the temporal phase gradient in time t for every orientation θ and location x, noted as $\phi_\theta(\mathbf{x}, t)$, through a linear least-squares fit:

$$\phi_\theta(\mathbf{x}, t) \approx c_\theta(\mathbf{x}) + \phi_{t,\theta}(\mathbf{x})t \tag{2.38}$$

A simple unwrapping technique is used to cope with the periodicity of the phase.

Next, for each orientation θ a component velocity is computed directly from $\phi_{t,\theta}(x)$:

$$v_{c,\theta}(\mathbf{x}) = \frac{-\phi_{t,\theta}(\mathbf{x})}{2\pi(f_{x,\theta}^2 + f_{y,\theta}^2)}(f_{x,\theta}, f_{y,\theta}) \tag{2.39}$$

Where f$_{x,\theta}$ and f$_{y,\theta}$ are the spatial frequency values at θ orientation. If we assume phase linearity, we can translate the sum of squared frequency values to a constant characteristic which is the peak frequency of the filter used. Note that the spatial phase gradient is substituted by the radial frequency vector.

The reliability of each component velocity at each orientation θ is measured by the Mean Squared Error (MSE):

$$MSE = \sum_t \frac{(\Delta\phi_\theta(x,t))^2}{n}, \tag{2.40}$$

where n is the number of frames and $\Delta\phi_\theta(\mathbf{x},t) = (c_\theta(\mathbf{x}) + \phi_{t,\theta}(\mathbf{x})t) - \phi_\theta(\mathbf{x},t)$. Please note that, for uniform motions (no acceleration), the flow computation benefits of the use of a large number of temporal frames. Unfortunately, real-time and (accessible) memory constraints reduce this number to few frames, using typically 3-5 frames [84]. Using this information, we discard those orientations that are not reliable, i.e. that are under a given threshold.

Finally, if a minimal number of reliable component velocities are obtained, an estimate of the full velocity v^* is computed for each pixel:

$$v^*(\mathbf{x}) = argmin_{v(\mathbf{x})} \sum_{\theta \in O(\mathbf{x})} \left(\|v_{c,\theta}(\mathbf{x})\| - v(\mathbf{x})^T \frac{v_{c,\theta}(\mathbf{x})}{\|v_{c,\theta}(\mathbf{x})\|} \right)^2 \tag{2.41}$$

Where $O(x)$ is the set of orientations at which the valid component velocities have been obtained for pixel x. For those points without enough reliable oriantations, the optical flow vector or a tag indicating *non valid* data is generated.

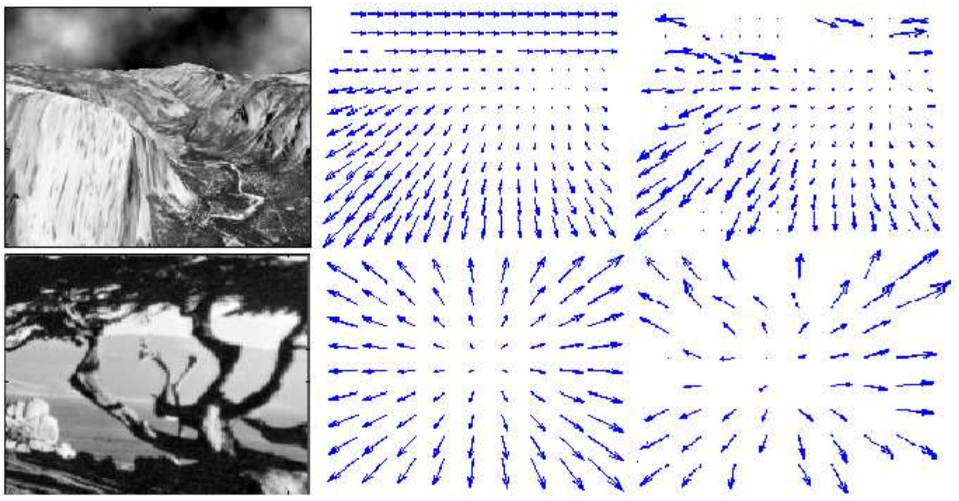


Figure 2.10: Optical flow example. Left column shows one of the three frames of Middlebury's ´yosemite´ (top) and ´diverging tree´ sequences used as input images. The central column represents the ground truth of the optical flow. Left column shows the optical flow calculated using [6].

Figure 2.10 shows an example the optical flow computation for Middlebury sequences [79]. This optical flow has been calculated using Barranco et al. algorithm [6, 5] and compared to the ground-truth (central column).

As we did with the disparity, we are going to point out the main problems we have to handle when extracting optical flow.

2.4.2.2 Optical Flow Problems

As with the stereo, motion computation presents the *correspondence* and *reconstruction* problems (see section 2.4.1.1 for details). In addition, optical flow presents other problems, such as aperture.

Aperture problem Since we extract the optical flow of a point considering a limited region, inspired by the V1 cells in the visual cortex [47], we can only detect a 1D motion: the velocity component perpendicular to the local line feature inside its receptive field. This means that if we only see a portion of an object, we cannot detect the true motion of the 2D object directly from its many local motions.

Aperture problem is usually solved by using a larger window in a local method or employing a global approach that take into account the whole image.

2.5 Multi-Modal Visual Features

Now that we have studied the features extracted in our system, it is the moment to introduce the existing solutions to our memory, bandwidth and performance problems. In fact, very few approaches handle visual features instead of images. In literature we find multi-modal descriptors that use a sparse representation to condense several visual features into an unique descriptor.

A large amount of evidence suggests that the human visual system processes a number of visual features in its first cortical stages [47, 44, 10]. In [70], Pugeault et al. suggest a representation to bundle the different features in one visual descriptor which can be interpreted as a functional abstraction of a specific repetitive pattern at the first stage of cortical visual [46]. This representation, that they call multi-modal descriptor, has been used in a number of applications such as the learning of object representations [69], pose estimation [23], motion estimation, and vision based grasping [68].

This multi-modal representation is based on image structures, it accumulates edge information in terms of an abstracted representation by local multi-modal descriptors covering geometric and appearance. Thus, we understand it as a condensed sparse representation, i.e. only a few descriptors are transfered to higher level, but those points contain information concerning several visual features.

The extraction process of this multi-modal descriptor can be summarized as follows (a more detailed description can be found in [69, 46, 70]):

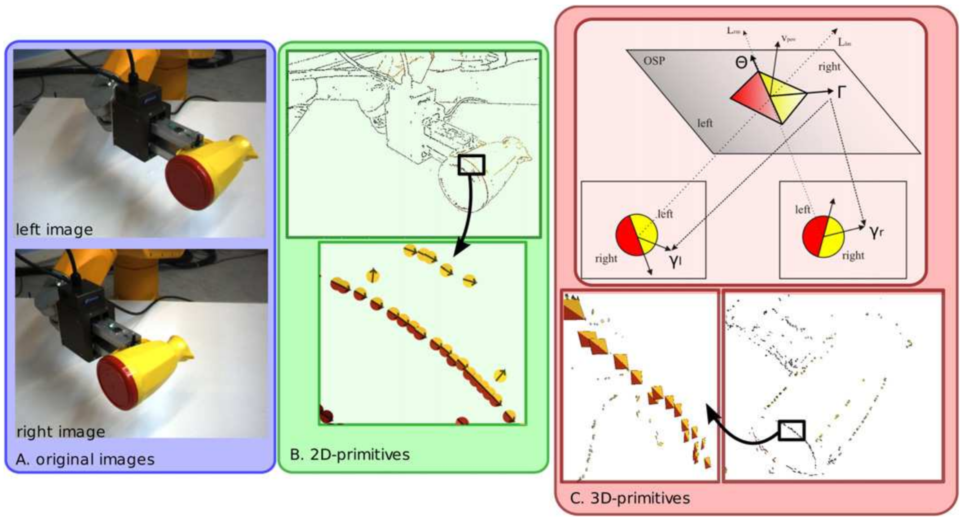


Figure 2.11: Multi-modal descriptor extraction process. (A) Original images (left and right); (B) extraction of 2D-primitives; and (C) stereo reconstruction of 3D-primitives. Image from [69]

1. Their system receives a stereo-pair of images obtained from a precalibrated stereo rig as shown in Figure 2.11A.

2. The left and right images are then processed independently in the low-level vision layer extracting: magnitude, orientation, phase, color, and optical flow.

3. For each pixel, its local flow is represented by its color: the hue[4] indicates the orientation of the flow vector and the intensity, the magnitude of the flow (where black stands for zero flow).

4. The local signal is classified using the intrinsic dimension (see section 2.2.3) computed for each pixel in the image. This way we extract a confidence value about the image structure

5. Pixel-wise information provided by early vision is combined in a sparse, condensed set of feature vectors called *2D-primitives* (Figure 2.11B).

6. 2D-primitives are then matched across the two stereo-views and correspondences allow for the reconstruction of 3D-primitives that extend the primitive representation into 3D space (Figure 2.11C). This representation is then provided to higher levels.

[4]The quality of a color as determined by its dominant wavelength

Despite of all the advantages of this descriptor (see [69]), the main disadvantage of this sparse visual feature, as all the others studied in this chapter, is that plain regions (i.e. low textured areas) are ignored.

Chapter 3

Semidense representation map for visual features

Sparse visual features introduced in the previous chapter look like the perfect solution to our bandwidth and memory problems. Let us check if they fulfill all our goals:

Condensation. All of them will dramatically reduce our memory and bandwidth requirements. For instance, local descriptors extract less than 1% of the original image points as relevant, as shown in Fig. 2.6. Even edges or saliency maps will provide a good reduction.

Versatile. Except multi-modal descriptor, sparse features receive intensity images as input. This means they are not prepared to handle dense visual features. However, we could use them as a mask and apply it to condense any dense visual feature.

Hardware-friendly. Most of these algorithms are designed and implemented to be used in a software system and, therefore, they need several iterations to complete their calculations. Only those based only in filtering operations, such as energy, orientation, phase and edge detectors, have a straightforward implementation in hardware [25].

Feedback from higher levels. Multi-modal descriptors have successfully included feedback information to improve their behavior [69]. Although attention systems include top-bottom targets in their design, not many of them include those kind of signals in their representations [89].

Efficient representation. Our most promising options (local and multi-modal descriptors) are too complicated to be handled by mid- and high-level algorithm without changing those algorithm or including a translation module.

Information recovery. These options do not include enough plain-region information, in fact, most of them include none.

All options and constrains considered, none of these sparse visual features satisfies all the requirements of our system. We need to design and implement our own representation map.

Contrary to sparse and dense approaches, we have designed a *semidense representation map* combining them. Figure 3.1 shows an example of the semidense representation map achieved from a disparity input. It also shows how it can receive top-down information from higher levels. As we can see, our innovative map is mainly composed of two different parts: relevant points (RPs) containing salient information, and a filter-based subsampling grid, integrating plain region data at a minimum cost [33].

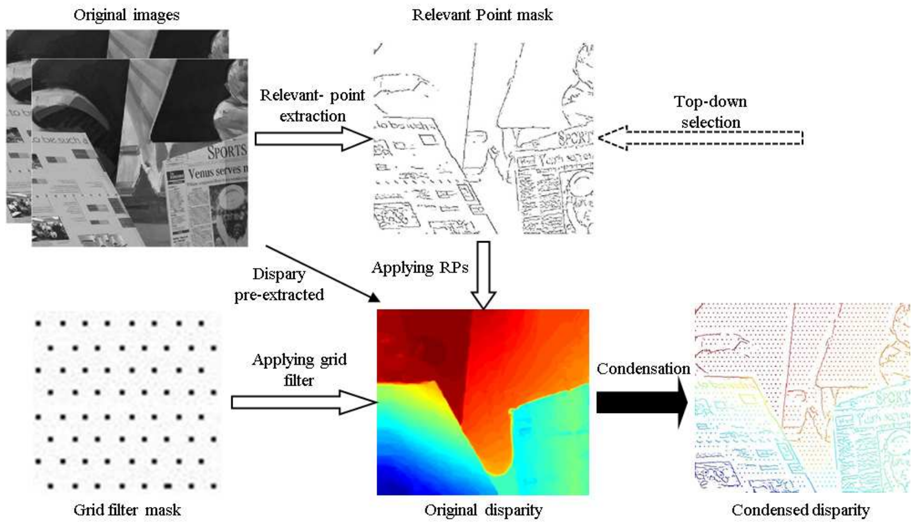


Figure 3.1: Condensation process example. Condensed disparity map size is around 6% of the original. We have used a Canny-based extractor for the salient areas and 9x9 bilateral-based extractor for the plain regions.

In the first part, one or several of the sparse features previously mentioned is used to detect salient regions, that we call relevant points (RPs). In section 3.1 we explain in detail this process, using a edge-based relevance detector as example, and how to integrate it with other sparse features.

In the second part of our algorithm, 'context' information is extracted. Instead of sending only the RPs (as other descriptors typically do), we apply a filter to the dense feature we are condensing and subsample a regular grid of that context information. The filtering process regularizes and spreads the information contained in the dense feature. In the grid extraction (section 3.2) we assess different filters and grid sizes.

Section 3.3 includes some tests using real-world sequences to test the regularization capacity of our semidense representation map.

3.1 Relevant Points

The process followed to extract these points starts by using an enhancing signal to select salient areas in the image. Then, we use those selected points as a mask and extract RPs from a dense visual feature.

As a general rule, our condensation algorithm can use any method that selects a branch of salient points in an image as the ones described in chapter 2. For vision algorithms, however, the early cognitive features of an image are based on its local structure, which is related to front-end vision. Structure information allows us to discard entire regions that lack interest and center our attention (and processing power) on the relevant parts in order to transfer them. Most of the sparse feature extractors we have mentioned follows this idea. This is why, instead of employing a more complicate algorithm that requires more complex computations, we have chosen a simpler method based on structure detection. In fact, structure information is usually applied in applications such as vehicle navigation, 3D scene/object reconstruction and robotic grasping.

We have designed this salience-based part of our representation so it is ready to integrate several sparse features at the same time. In fact, it is ready to receive feedback data from low-,mid- or high-level stages of our system, i.e. top-down targets, and integrate them with our RP mask. In the following sections we introduce a structure-based RP extractor and how to integrate different sparse signals. Section 4.1.1, on the other hand, introduce saliency-map-based RP extraction, while sections 4.1.2 and 4 shows a real-world example of signal integration.

3.1.1 Structure-Based Selection

In chapter 2 we have studied several structure-based sparse features that we could use as base to our condensation algorithm. Since we are looking for a real-time hardware-friendly approach, we focus on edge detectors and intrinsic dimension.

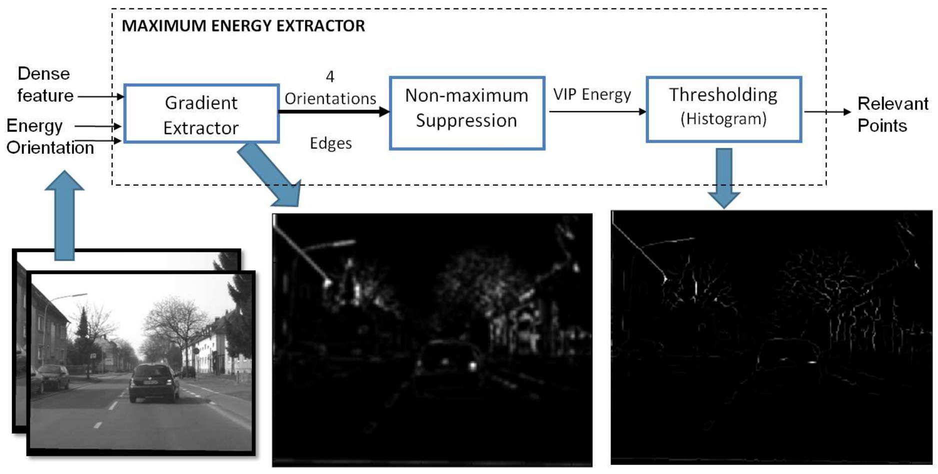


Figure 3.2: Relevant points extractor based on Canny edge detector. The scheme shows the main steps of the RP extractor using Canny non-maximum suppression and hysteresis thresholding. Bottom part corresponds to a real-world example of this process, from the original images to the final edge mask.

3.1.1.1 Canny Edge Detector Approach

Gabor filters are used to extract local energy in our vision system (as shown in Fig.1.4). Our algorithm re-utilizes their outputs and uses magnitude as main salience detector. Obtained edges, however, are too thick. This is why, as explained in section 2.2.2.1, we apply non-maximum suppression and hysteresis to further reduce the representative data. Since our resources are limited, we do not use gradient orientation. Our solution employs local orientation calculated by our system to compute the non-maximum suppression. We establish four orientations:

- Orientation 1: angles within one of these two groups $[0, 22.5)$ or $[157.5, 180)$.
- Orientation 2: angles within $[22.5, 67.5)$.
- Orientation 3: angles within $[67.5, 112.5)$.
- Orientation 4: angles within $[112.5, 157.5)$.

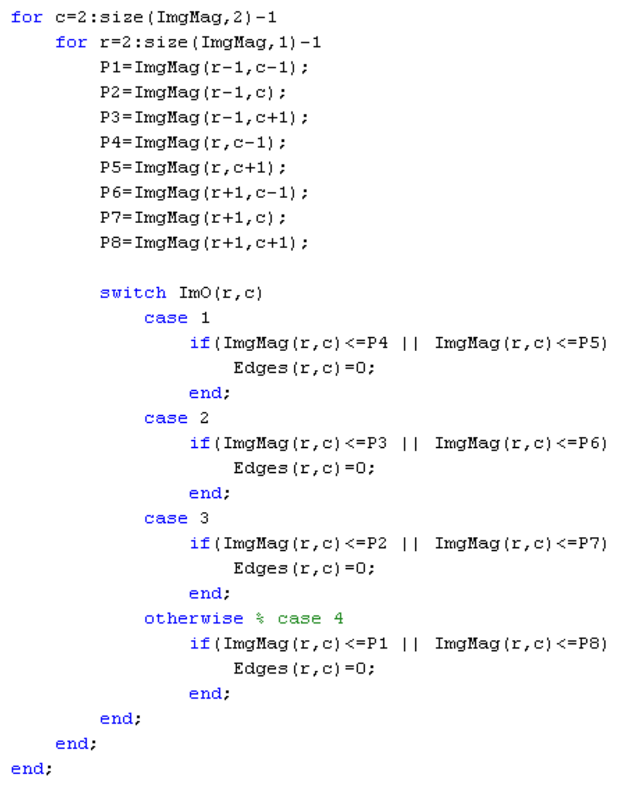
Our relevant point algorithm based on Canny edge detector is shown in Figure 3.2 as well as an example output when used on a real-world sequence (bottom left images). In Figure 3.4, we compare it with the other methods assessed. The main steps of this process are the followings:

1. Apply image magnitude a given threshold. Store it in `Edges`.

2. Associate each pixel one of the four "gradient orientation". Store it in `ImgO`.

3. Discard those pixels that are not a maximum. Figure 3.3 shows the MATLAB code for this step.

4. Apply hysteresis thresholding using a given threshold.



```
for c=2:size(ImgMag,2)-1
    for r=2:size(ImgMag,1)-1
        P1=ImgMag(r-1,c-1);
        P2=ImgMag(r-1,c);
        P3=ImgMag(r-1,c+1);
        P4=ImgMag(r,c-1);
        P5=ImgMag(r,c+1);
        P6=ImgMag(r+1,c-1);
        P7=ImgMag(r+1,c);
        P8=ImgMag(r+1,c+1);

        switch ImO(r,c)
            case 1
                if(ImgMag(r,c)<=P4 || ImgMag(r,c)<=P5)
                    Edges(r,c)=0;
                end;
            case 2
                if(ImgMag(r,c)<=P3 || ImgMag(r,c)<=P6)
                    Edges(r,c)=0;
                end;
            case 3
                if(ImgMag(r,c)<=P2 || ImgMag(r,c)<=P7)
                    Edges(r,c)=0;
                end;
            otherwise % case 4
                if(ImgMag(r,c)<=P1 || ImgMag(r,c)<=P8)
                    Edges(r,c)=0;
                end;
        end;
    end;
end;
```

Figure 3.3: Non maximum suppression code implemented in MATLAB.

Threshold used could be fixed or vary to adapt to the intensity changes in the scene as explained in the hardware implementation (see section 5.2).

3.1.1.2 Sobel Edge Detector Approach

Our system has a convolution module available that we can easily adjust to apply Sobel's kernels (see equation (2.29)). Since Sobel-based edges are usually thinner than the ones obtained by Canny detector, no farther processing is needed [56]. Nevertheless, we could use the same non-maximum suppression and thresholding to obtain even thinner edges. In Figure 3.4, we compare it with the other methods assessed.

3.1.1.3 Intrinsic Dimension Approach

Three differentiated masks are obtained in this this approach: i0D, i1D and i2D. They can also be calculated using Gabor filters to emulate image gradient. This algorithm's steps can be summed up as follows:

1. Extract image gradient using oriented Gabor filters.
2. Calculate gradient squared magnitude using soft-thresholding [26].
3. Apply local averaging to the soft-thresholded magnitude.
4. Create the three masks using averaged magnitude and angle information.

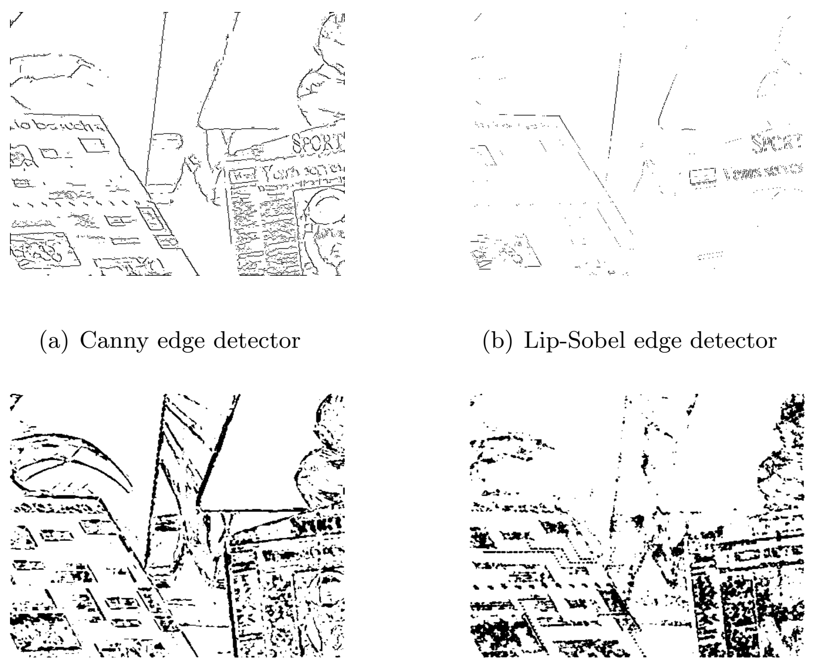
The complexity of this option, as we can see, is bigger than the previous approaches and its implementation in real-time is more challenging. In Figure 3.4, we compare i1D and i2D outputs with the other methods assessed.

3.1.2 RP Extractor Assessment

We have assessed these three options (Canny, LIP-Sobel and intrinsic dimension) as enhancing signals to show the versatility of our algorithm. In Figure 3.4 we show the RP mask for each of the assessed options, using as input the same image as in Figure 3.1. As previously mentioned, these point selectors can be considered bottom-up saliency enhancers [67].

To evaluate the accuracy and the condensation capacity of our semidense map, we have used as input the disparity obtained from several Middlebury benchmark sequences [79]: Tsukuba, Venus, Teddy and Cones. The dense disparity has been obtained using Ralli et al. algorithm [73]. We have also followed Middlebury accuracy measurement convention, called 'bad point error'. A bad point is a point whose error is above a given threshold (typically 1 pixel of disparity extraction) [79]. Hence, 'bad point error' is the percentage of bad points in our representation.

Figure 3.5 shows the behavior of our algorithm for each of these RP extractors and sequences. The condensation columns (labelled as 'Cond.') correspond to the percentage of the original disparity we are actually using. We can see that the error for the majority of the sequences is bigger than the original one. This is because we select a few points from the most complex areas of the image (edges and corners). At a first glance, LIP Sobel extractor performs better reduction (although its error is bigger). The number of point extracted by this algorithm, however, is too small and they would result insufficient for most of the higher-level algorithms.



(a) Canny edge detector (b) Lip-Sobel edge detector

(c) Intrinsic Dimension 1 (id1) (d) Intrinsic Dimension 2 (id2)

Figure 3.4: Relevant point mask for the different structure-based extractors assessed. We have used Middlebury [79] sequence 'Venus' to graphically compare them.

Moreover, Figure 3.5 shows how the condensation percentage depends on the sequence: more textured images profit from id2 advantages, while simpler ones work better with the edge detectors (Canny and Lip-Sobel). It also depends on the ulterior high-level application. Thus, several of these extractors can be combined whenever necessary, providing a more versatile approach. Since one of our goals is to obtain a hardware-friendly solution, we have selected Canny as a trade-off approach. Although edges are prone to larger disparity estimation errors, this approach's increase is quite limited. In addition, Canny uses a threshold that is really useful to control intensity variations, providing an easy way to adapt to changing environments.

In our software implementation, this step output is basically a binary matrix containing '1' if the point is relevant or '0' otherwise. After integrating this mask with the plain-region grid, we transfer our RPs using a bio-inspired approach: an event-driven communication protocol similar to [13] (more details can be found in section 5.1.1).

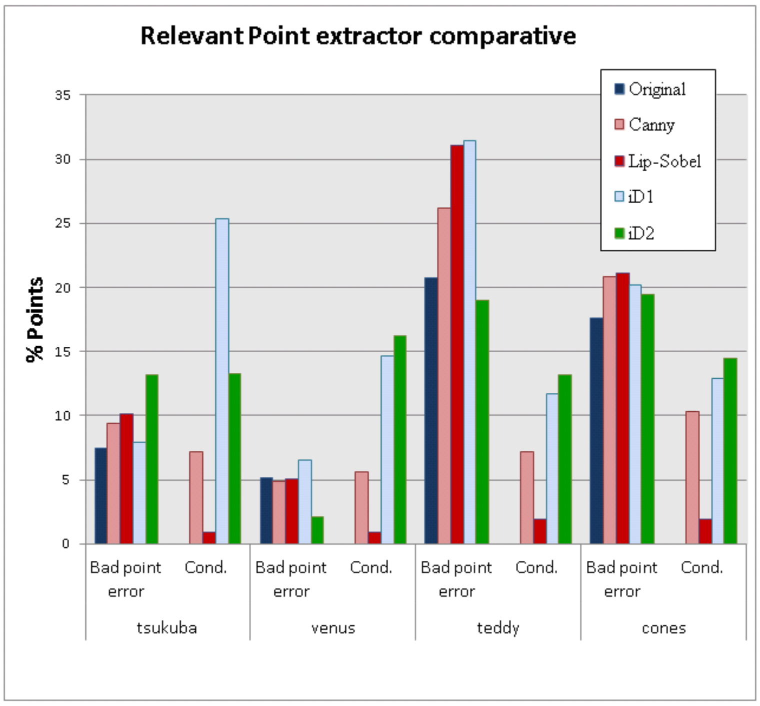


Figure 3.5: Relevant-point extractor comparison. 'Bad point error' means the percentage of points whose absolute error is above a given threshold, that is, over 1 in this example; this measurement has been widely used [79]. 'Cond.' is the percentage of the condensed disparity points compared to dense ones.

3.1.3 Integration with Other Sparse Features

Although we have used a structure-based RP extractor, our model is prepared to receive any other sparse visual feature as bottom-up signal. We might want to add the new signal to our extractor or replace the RP extractor completely.

On one hand, integrating a sparse feature consists on transforming the new feature into a binary matrix that follows the same scheme as our representation. This way, we can just add (binary AND) them to obtain a single mask. Note that this process can also be used to inhibit regions or points by sending a mask that contains '0' in the areas to suppress. We have successfully used this approach to integrate not only several low-level features (section 5.3.2) but also mid- and high-level ones (section 4.2).

On the other hand, if we want to replace our RP extractor by another one, the simplest solution is to replace this module by the sparse feature extractor and transform it into a binary matrix. Some of those feature descriptors, however, provide more information than just the salient point position. If we wanted to include that information, a bigger adaptation would be needed.

3.2 Plain or Context Regions

In contrast to sparse features, our semidense representation map also includes plain-region information (i.e. i0D regions according to intrinsic dimensionality, as explained in section 2.2.3 and [45]). This operation is the key that allows higher-level stages to evaluate the scene as accurately as using a dense map. Our algorithm extracts representative points using an homogeneous grid to select the points' coordinates. This grid is combined with the RPs to obtain our final semidense representation map (as shown in Figure 3.1).

To obtain this regular grid, we apply locally (in a window Ω) a filter to the dense feature and select a representative element of that window, following a regular pattern. An scheme of this process is shown in Figure 3.6. Formally, the grid extraction process can be defined as follow:

$$\forall c/c \in \mathbb{S}, \qquad V_c = f_\Omega(c) \tag{3.1}$$

where c is a grid point, $\mathbb{S}$ is the set of grid points and f_Ω is a filter locally applied in a window Ω centered in c.

The number of points in $\mathbb{S}$ is determined by the size of the window Ω. Their relative position, however, could vary: we could select a squared or hexagonal grid, for instance. Hence, there are two main elements to assess: Ω, the grid window size; and f_Ω, the filter applied.

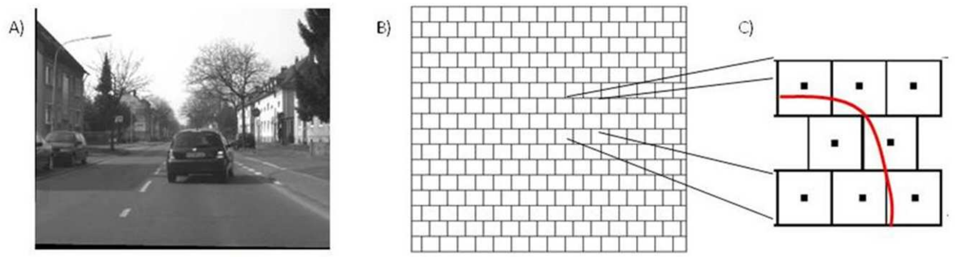


Figure 3.6: Grid extraction scheme. A) original image example; B) grid windows used in the condensation process (hexagonal to reduce distances between grid points); C) the black points represent the grid points extracted from the minor-relevance areas and the red line represents the relevant points in that area.

Before studying Ω and f_Ω, let us focus on the grid shape, i.e. the relative position of points contained in $\mathbb{S}$. We have decided to fix it, following an hexagonal pattern, as shown in Figure 3.6. This distribution, contrary to a squared one, reduces the Euclidean distance between grid points. In fact, compound eyes of many insect are distributed following this hexagonal pattern [56]. Nevertheless a more complex grid shape, such an adaptative one could be designed (see future work in section 6.2).

3.2.1 Grid Window Size Ω

As we deduce from (3.1), Ω depends on the image size and on the final number of non-relevant points we want to obtain (size of $\mathbb{S}$). A 5x5 grid window extracts around 4% of the dense feature points; a 7x7 one, around 2%; and a 9x9 one, around 1,2%. Moreover, $\mathbb{S}$ size determines the number of subsampling operations we will need to perform. A bigger grid size, however, could produce insufficient points and result useless for higher-level stages. Nevertheless, it is important to notice that the original dense feature may be non-dense, for instance there is no disparity information available when a region has no texture. Thus, some of the grid points will be NaN (not a number), whenever the majority of the points in the window are NaN. Since dense maps present NaN points, their presence in our semidense representation is expected and should not affect higher-stage algorithm. Our main concern, however, is wheter we are marking too many points as NaN and therefore loosing too many data.

We need a way to measure whether we are losing too much information due to the condensation process or not. We have used Receiver Operating Characteristic (ROC) curves to assess different Ω values [94]. Appendix A explains these curves, including specificity and sensitivity equations. In this case we label *true positive* those points that keep an actual value, *true negative* points that being NaN in the dense features are kept as NaN, *false positive* points that being NaN in the original are given a value in the condensed one, and *false negative* points having a value that are marked as NaN when condensed (i.e. points we loose when condensing).

Apart from considering NaN values, we have marked as invalid those condensed points too noisy to be considered as correct. Hence, we have calculated the disparity of a test sequence, then we have condensed it using different grid widows and median filter (see next section for further details). Finally we have calculated the difference between original and condensed values and the standard deviation (std). This way, if the difference is bigger than a given threshold we consider that point invalid. Formally this process can be defined as follows:

$$|Original\_Disp - Condensed\_Disp| \leq std/i, \tag{3.2}$$

	3x3 Window		13x13 Window		23x23 Window		33x33 Window	
2-9	**Sens.**	**Spec.**	**Sens.**	**Spec.**	**Sens.**	**Spec.**	**Sens.**	**Spec.**
Std / 1	0.9521	0.7460	0.8170	0.6685	0.8069	0.6485	0.6702	0.5655
Std / 2	0.4774	0.4507	0.3047	0.4215	0.3075	0.4210	0.2438	0.3742
Std / 3	0.3183	0.4465	0.1744	0.4129	0.1950	0.4132	0.1414	0.3629
Std / 4	0.2437	0.4452	0.1251	0.4104	0.1350	0.4099	0.1004	0.3593
Std / 5	0.1920	0.4444	0.1003	0.4093	0.1007	0.4083	0.0724	0.3570
Std / 6	0.1551	0.4439	0.0826	0.4085	0.0854	0.4076	0.0592	0.3560
Std / 7	0.1269	0.4436	0.0741	0.4081	0.0753	0.4071	0.0520	0.3554
Std / 8	0.1118	0.4434	0.0673	0.4078	0.0649	0.4067	0.0460	0.3550
Std / 9	0.0996	0.4432	0.0608	0.4076	0.0590	0.4064	0.0416	0.3547
Std / 10	0.0891	0.4431	0.0559	0.4074	0.0543	0.4062	0.0379	0.3544

Table 3.1: Grid window sensitivity (Sens.) and specificity (Spec.) for different grid widow sizes calculated using (3.2)

where $i = [1, 10]$.

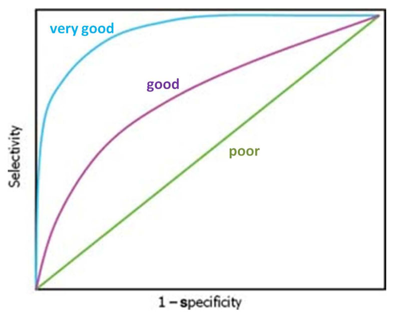
Table 3.1 shows an example of this process using a few of the assessed window sizes. Using these data (and similar ones calculated for other window sizes) we have generated several ROC curves shown in Figure 3.7. In 3.7(b) we see how smaller sizes provide a better behavior. Although we have assessed big window sizes, Figure 3.7(d) shows how the bigger the grid, the worse the curve. For smaller sizes (Figure 3.7(c)), however, most of the options are good enough.

3.2.2 Filter Selection

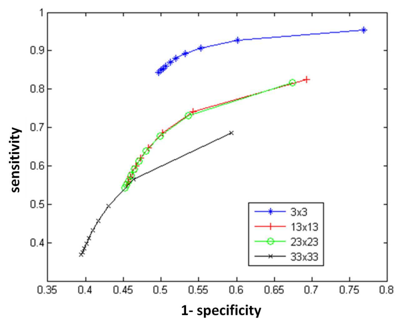
We use a filter instead of simply sub-sampling so that existing errors are smoothed and information is spread throughout the feature [3]. When working on real-time applications, features are noisier, therefore, a smoothing step produces evident improvements [86]. This process can be considered a regularization step. We have matched filter size and grid size to achieve a simpler implementation.

We can see that f_Ω depends on the input feature noise and the image size. To reduce this noise, we have assessed median and bilateral filters, anisotropic diffusion, and subsampling (i.e. no filtering, just selecting one point per window). Figure 3.8 compares these filters when used to extract the representative value of the grid. We have also assessed several filter window sizes: 5x5, 7x7 and 9x9. We use Middlebury sequences and error measurement as in Figure 3.5.

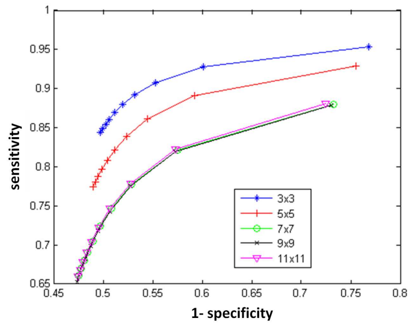
We can see that, once again, the final results depend on the sequence, but also that a bigger grid window usually produces a higher error. Even though we are using a very accurate disparity [73] and a regularization step is not



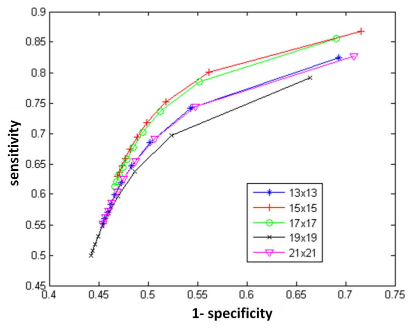
(a) ROC curves behavior



(b) Grid window size: general view



(c) Small grid window sizes



(d) Big grid window sizes

Figure 3.7: Grid window size comparison using ROC curves. (a) shows how to interpret ROC behavior. (b) compares different window size. (c) focuses on the smaller sizes while (d), on the bigger ones.

essential, we can see how the error is slightly reduced when using bilateral (Tsukuba and Venus) or median (Teddy) filtering. This regularization is more evident when noisy or less accurate primitives are used, as in real-time implementations that have to meet real-time constraints [1, 86] (see section 3.3.2).

From figures 3.5 and 3.8 we determine that every application requires customized configurations. Nevertheless, a 5x5 hexagonal grid using bilateral filter achieves great results as a general rule. Moreover, there is a very profitable advantage when using the bilateral filter: it preserves edges. This is very important since we are already extracting the edge information in the previous stage of our algorithm. If we use a median filter, for instance, we would include this information again, adding inaccurate data to our representation.

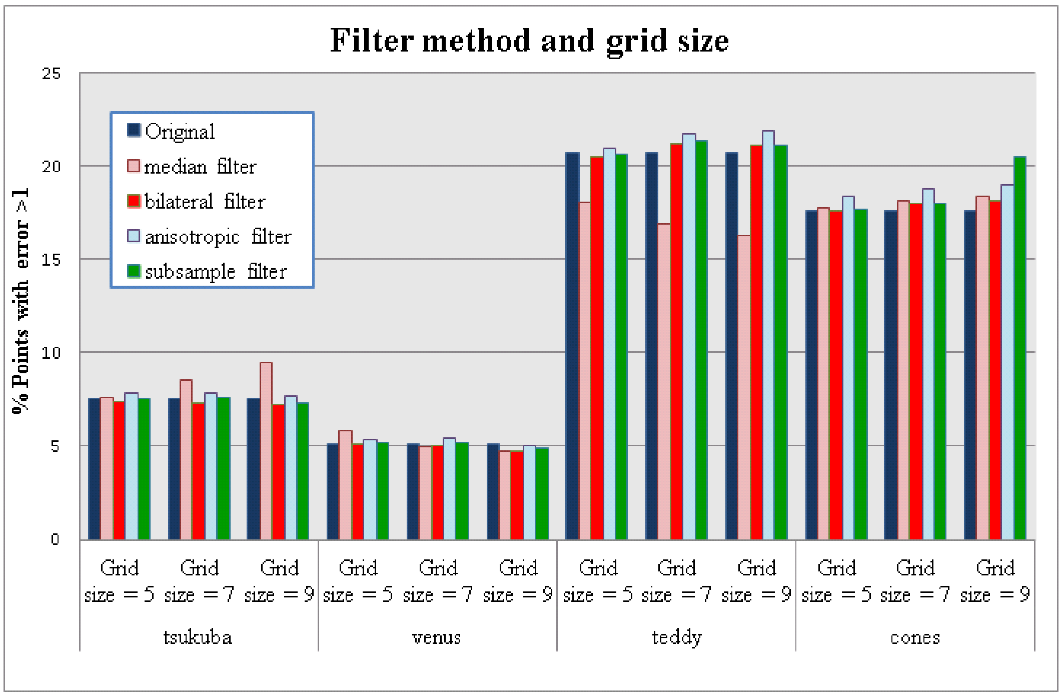


Figure 3.8: Filter and grid-size comparison. The error used corresponds to the percentage of points whose absolute error compared to ground truth is above a given threshold, that is, over 1 in this example [79].

3.3 Results and Validation

So far, we have studied how to extract salient information using RP extractor and how to keep the less salient one by using a regular grid. Now we need to translate those separate processes into an unique representation, a semidense representation map.

3.3.1 Semidense Representation Map

The final step of our condensation algorithm is to combine and transfer efficiently the points extracted, relevant and non-relevant. Since grid points have fixed positions, when transferring our semidense representation map, we do not need to include their coordinates; it is enough to provide a binary mask or a simple algorithm to the higher-level stages. By this, we only need to transfer the list of feature values. A special case are those grid positions where we find a RP. These positions' feature value would be marked as NaN, and hence ignored by other algorithms. Relevant points, however, have no pre-fixed positions. Therefore we represent them as the feature value and its coordinates within the image (in section 5.1.1 we explain this transfer process in detail).

Although this is not a compression algorithm, our condensation process reduces the amount of data to be transferred between co-processors. Instead of sending a whole image, we send 10% of the original map for a grid size of 5x5 (this percentage depends on the sequence spatial structure). Table 3.2 shows the 'bad point' error for the previously assessed sequences (used to obtain Fig. 3.8 and 3.5). As we can see, the error is slightly bigger than the original one. This increase, however, is worth it for many algorithms due to bandwidth, workload, and computating resources reduction. As previously mentioned, when using dense features from real-time systems, the filtering steps is pretty useful to reduce the error as we prove in the next section.

Sequence	Tsukuba	Venus	Teddy	Cones
% Bad point error (original)	7,5	5,11	20,72	17,59
% Bad point error (condensed)	8,31	5,32	25,7	19,29
% Condensed	7,98	8,13	8,54	10,81

Table 3.2: Condensation output.

Many high-level algorithms work with sparse representation maps; therefore they could receive this semidense map as input, reducing the amount of data to be transferred between stages and allowing feedback from higher-levels. We successfully tested these semidense maps as an input of high-level algorithms with encouraging results (see section 4 for farther details).

Other high-level algorithms, however, are prepared to receive a dense feature (see section 4.2.2.2), thus our representation would imply some changes in their implementation. A simple module can be implemented to translate our semidense map to a representation containing as many point as the dense feature by filling the non-defined points with a NaN value. In addition, we have implemented a straightforward interpolation module that recovers a dense map from a semidense one. Appendix B explains the different options assessed to interpolate a dense map using our semidense one as input.

In addition, as an example of this process, we include here Figure 3.9 that shows how we recover the dense map by interpolating condensed disparity. The interpolation method used is a simple one: linear. In short, we spread the grid point information in its neighborhood, weighted by the distance. This process acts as a regularization mechanism producing more homogeneous fields which could improve the field accuracy when working with noisy features (a capability we explore in next section).

Our semidense representation map is versatile, which means we can condense different dense features. Figure 3.10 shows an optical flow example,

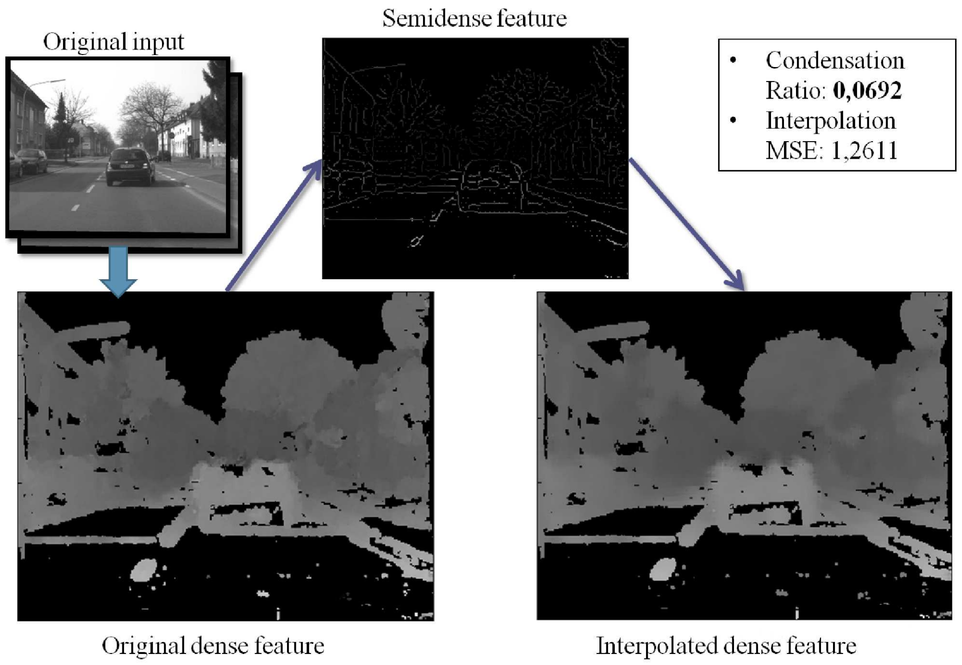

Original input
Semidense feature
• Condensation Ratio: **0,0692**
• Interpolation MSE: 1,2611
Original dense feature
Interpolated dense feature

Figure 3.9: Condensation process including interpolation. Using the input images (top left), dense disparity (bottom-left) is extracted using [86]. After condensing it, we recover a dense map using simple calculations.

calculated for a real-world sequence, as well as its condensation and interpolation. This example is just a frame of a 300-frame sequences. In addition, we also present the MSE (comparing original and interpolated) and condensation ratio achieved for the whole sequence (Figure 3.11(a) and 3.11(b) respectively). Again, read Appendix B for farther details about interpolation techniques.

We tested our condensation method with several sequences as the example above, extracting different visual features and leading to final condensed representations requiring 7-15% of the original image size. Note that the de-condensation process allows us to recover the original data density with only small errors. These errors, however, have been proved to be more than just degradation in many cases, as we explain in next section.

3.3.2 Inherent Regularization

As we have seen in Fig. 3.9 and 3.11, the difference between original visual feature and de-condensed one has an error (for instance MSE: 1.2611 in the former figure). Nevertheless, this difference cannot be interpreted as degradation of the results (due to the condensation), but as a regularization.

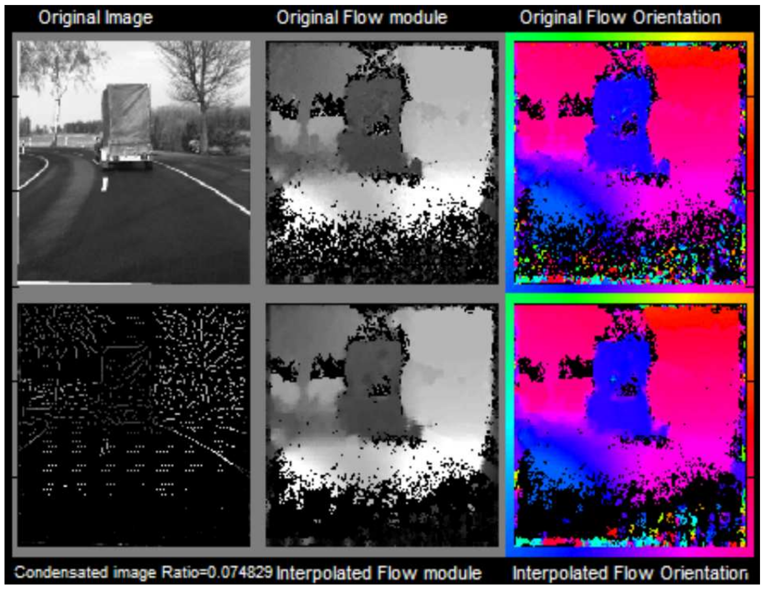
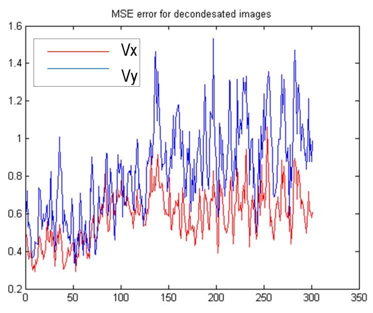
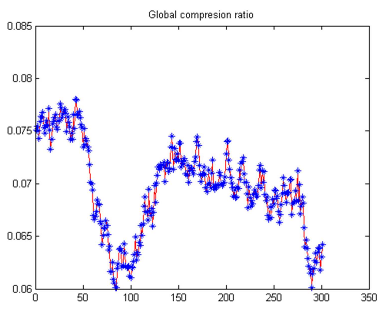

Original Image
Original Flow module
Original Flow Orientation
Condensated image Ratio=0.074829
Interpolated Flow module
Interpolated Flow Orientation

Figure 3.10: Using the original image and optical flow (first row), we extract our semidense map and we recover a dense representation by applying a simple interpolation method



(a) MSE error per component (Vx and Vy)



(b) Condensation ratio

Figure 3.11: (a) MSE and (b) condensation ratio of a real-world sequence when extracting, condensing and decondensing optical flow. (a) shows the error between the original dense feature and the interpolated one for each velocity component.

Currently, a large amount of papers about visual feature regularization can be found [17, 31, 49], but none of them focus on achieving an efficient algorithm that fits in a hardware implementation and at the same time reduces transfer load. Our semidense representation map provides this regularization as an inherent quality, joining hardware and regularization effectively.

To evaluate the regularization capabilities of our condensation algorithm we applied it to an optical-flow field computed with a FPGA device. We used several Middlebury benchmark sequences whose ground truth is known [2]. The raw hardware engines obtained a low-cost dense map. Then we condensed and decondensed it using our proposed approach. Finally we compared these three maps (low-cost hardware, condensed and interpolated) with the ground-truth ones, obtaining the error of each of them. This example is of special interest for low-cost hardware implementations in which the results provided by the hardware engines are noisier (due to computation precision constraints or model simplifications) as in Table 3.3.

	Yosemite with clouds			Yosemite without clouds		
	AAE	STD	Density	AAE	STD	Density
Low-cost hardware based	18,6	18,62	99,53	15,65	16,67	99,4
Condensed	16,1	16,62	11,13	13,83	15,13	11,08
Interpolated	14,71	14,96	99,53	11,91	13,25	99,4

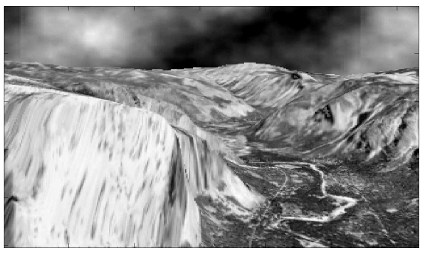
Table 3.3: Low-cost hardware optical flow condensed and decondensed (interpolated). A significant improvement in accuracy can be seen when condensing a low-cost hardware engine for front-end estimate extraction.

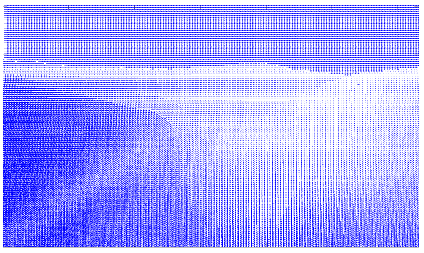
Table 3.3 shows the results of using the synthetic Yosemite sequence to obtain a low-cost hardware version of the optic-flow extractor. It also shows the condensation error when using Canny-based RP extractor and 3x3 median-filter grid. In addition we have decondensed the semidense map using a linear interpolation method. The density shown highlights the compression achieved by the condensation algorithm. The measurement used is the average angular error (AAE) and its standard deviation (STD) as suggested by [2]. The angular error (AE) between a flow vector (u, v) and the ground-truth flow (u_{GT}, v_{GT}) is the angle in 3D space between $(u, v, 1.0)$ and $(u_{GT}, v_{GT}, 1.0)$. The AE can be computed by taking the scalar product of the vectors, dividing by the product of their lengths, and then taking the inverse cosine:

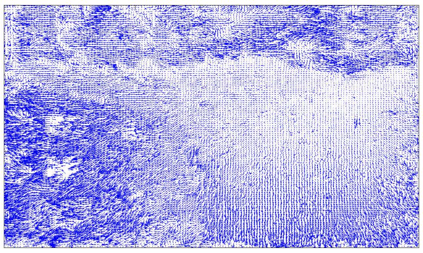
$$AE = \cos^{-1}\left(\frac{1.0 + u \times u_{GT} + v \times v_{GT}}{\sqrt{1.0 + u^2 + v^2}\sqrt{1.0 + u_{GT}^2 + v_{GT}^2}}\right). \tag{3.3}$$

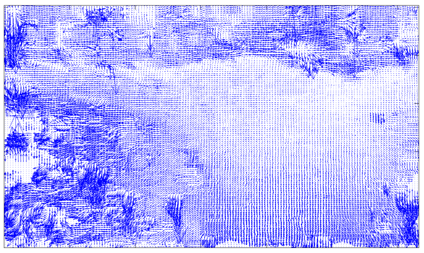
Therefore, from Table 3.3 we deduce that our semidense representation gains in accuracy as well as the interpolated map derived from it (see Appendix B).

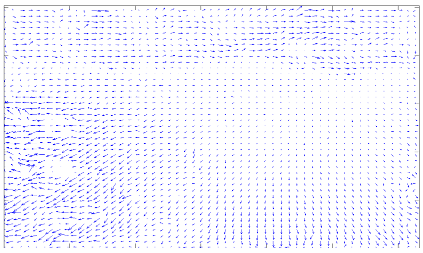
Figure 3.12 shows Yosemite sequence and the different optical flows calculated in table 3.3. To compare our output with other algorithms, we also show Chessa et al. affine-based regularization [17] when applied to the original noisy feature (Fig. 3.12(d)). Although our algorithm performance is better using a 5x5 window size (as shown in Fig. 3.7), to fairly compare the two approaches, we fixed the same window size (3x3) and we did not extract RP. This figure corroborates the previous table, showing condensation and de-condensation outputs and their accuracy gain. Moreover, we can see that our solution not only outperforms affine-based one but also requires less computational workload. Therefore, it may be concluded that our algorithm has inherent regularization capabilities.


(a) Yosemite original image


(b) Optic-flow ground truth


(c) Noisy optical flow


(d) Affine-based regularization


(e) Condensed optical flow

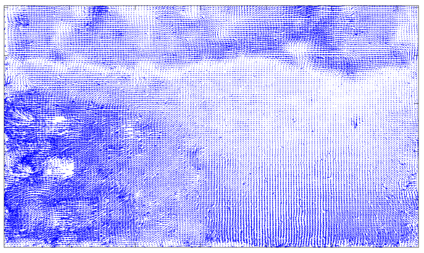

(f) Decondensed optical flow

Figure 3.12: Inherent regularization example. Using a low-cost hardware implementation, we have extracted (c) from Middlebury Yosemite sequence (a). (d) is an example of Chessa et al. regularized output [17] for (c). (e) shows our condensed output extracting no RPs and a 3x3 median-based grid. (f) shows decondensed (e) using linear approach.

Table 3.4 shows a more accurate hardware implementation of the optic-flow extractor [85] for several Middlebury sequences and how the condensation process works on them. Apart from the Yosemite sequence, we used the non-synthetic benchmark sequences Hydrangea and Rubber Whale, frequently used in computer vision [79]. As can be seen in these tables, the behavior of our condensation scheme is very interesting when a noisy input is provided (Table 3.3), but it can also reduce the error slightly when less noisy inputs are used (Table 3.4). In fact, we can see a worsening in one of the condensed cases, Hydrangea. This is due to the image complexity: more complex images has a higher number of RP and therefore, the regularizing capacities of our algorithm are diminished. After interpolating it, however, we see how the error is smaller than the hardware one. A similar behavior was obtained for the disparity field.

	Yosemite with clouds			Yosemite without clouds		
	AAE	STD	Density	AAE	STD	Density
Hardware based	11,98	15,07	78,52	7,92	6,93	92,02
Condensed	11,67	15,25	12,6	7,79	7,1	15,03
Interpolated	11,28	13,98	78,52	7,33	6,47	92,02
	Rubber Whale			Hydrangea		
	AAE	STD	Density	AAE	STD	Density
Hardware based	22,36	21,96	78,93	51,84	43,38	86,81
Condensed	20,47	20,21	13,38	47,41	42,00	12,41
Interpolated	21,53	19,95	78,93	51,82	41,42	86,81

Table 3.4: An improvement in accuracy can be seen when using a more accurate hardware engine as optic-flow extraction module [85]. Therefore the gain in accuracy of the condensation/interpolation process is very low in this case because regularization is explicitly carried out during a previous stage (embedded in the hardware extraction engine, which in this case includes median filters)

Figure 3.13 compares Tables 3.3 and 3.4. We have represented the AAE and its standard deviation. In the figure we can see how the condensed output error is less or equal than the original one, proving the regularization capacity of our semidense representation map.

From this analysis, it is clear that our semidense map not only preserves visual feature information and allows recovering it without error increase, but also regularizes noisy features without workload increment.

3.4 Conclusions

We have obtained a new semidense representation map that translate dense visual feature (such as optical flow or disparity) to a sparser representation. This semidense map is composed of two different kinds of points: relevant ones, which are obtained by bottom-up selection over the original image

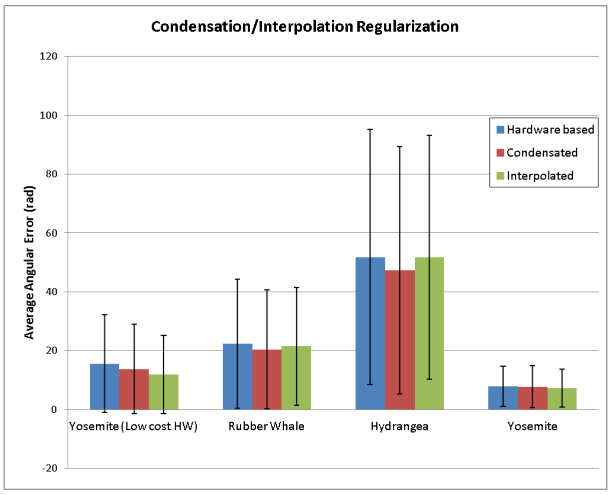


Figure 3.13: Regularization results from Tables 3.3 and 3.4.

(e.g., a Canny edge detector [16] or saliency maps [39]); and context ones, which are extracted by filtering the original feature (e.g., applying a bilateral filter [83]) and selecting a regular grid of points. The latter is a very important contribution of this new representation since other non-dense representations do not include non-salient information [54]. This information, despite of its name, can be useful in some algorithms such as ground-plane extraction, as shown in section 4.2.

We have assessed different extractors for both, salient and non-salient regions. For that we have compared the behavior of several structure-based RP extractors and different filters for non-salient areas to explore the possibilities of our condensation algorithm. We conclude that our semidense map response depends on the image structure and, therefore, we can modify the extraction process so it adapts easier to its ulterior application. Thus, we achieve better results when working on known situations (very textured sequences, driving scenarios,). Nevertheless, a trade-off configuration would be to use a Canny-based relevance extractor and a 5x5 bilateral-based grid.

Simplicity and versatility are the main advantages of this map: it reduces any dense visual feature to a map whose size is around 10% of the original one (using grid window of 5x5 pixels), with minimal error and workload.

Since the non-salient extraction is based on filtering, our condensation algorithm inherently performs a regularization that helps to improve accuracy (mainly in low-cost hardware-based feature extraction engines). To prove this, we have compared our condensation/decondensation process to an existing regularization algorithm, outperforming it. Moreover, since low-level features obtained from hardware implementations are noisier than the ones achieved by software ones [63], many higher-level algorithms include a regularization step. Our semidense map, however, inherently regularize these features, freeing them from this step and improving their performance.

At the beginning of these chapter we summed up the different goals of our representation. Let us check if we fulfill all of them:

Condensation. Our semidense map obtain around a 10% of the original map. Moreover, we can reduce bandwidth and memory requirements as much as we want by modifing RP extractor threshold and grid window size.

Versatile. We have assessed our algorithm with the main dense features of our system, i.e. disparity and optical flow. Other features (such as color) can easily be condensed.

Hardware-friendly. We reuse our system filters in both steps of our algorithm. In chapter 5 we present its implementation in a FPGA.

Feedback from higher levels. Our RP extractor is ready to receive higher-level signals as feedback and integrate them in the semidense map.

Efficient representation. We have obtained a simple representation that can be directly used by mid- and high-level algorithms as proved in next chapter.

Information recovery. We include information from not only salient regions but also plain ones. We have developed a simple interpolation method to recover a dense map with slight error.

In summary, our condensation algorithm translates dense representation maps into a semidense representation that can easily be handled by standard processors for further computing by higher level modules. Therefore the main goal of the next chapter is to integrate this representation map in mid- and high-level vision stages. This idea will reduce data flow through the system and free the higher processing stages from having to analyze the whole image.

Chapter 4

Method Validation: Experimental Results and Applications

So far we have a condensation algorithm that translates dense representation maps into a semidense one, requiring 7-15% of the original image size. We have tested this algorithm using different visual features (as input and relevance enhancers) and filtering operations. Notwithstanding the previous examples, we still have to assess its actual performance in real-world situations, i.e. we need to integrate it in a real application and evaluate its output, execution time and functionality.

In chapter 5 we assess bandwidth and memory reductions achieved by our condensation algorithm when included in a hardware system (see section 5.3). Nevertheless, our aim in this chapter is to prove that our semidense representation map can be used as input in higher processing stages on a standard processor. As we have mentioned, many mid- and high-level algorithms are usually based on sparse approaches [70]. In current systems, however, they usually receive dense maps. This means that either these algorithms perform a selection process as a first step or they process the entire dense map. Any of these approaches entails an unnecessary increment in their workload. Moreover, low-level features obtained from hardware implementations are noisier than the ones achieved by software ones [63]. Hence, mid-/high-level algorithms should perform a regularization step before doing any computation. This, again, increases their workload unnecessarily.

Working with our semidense map not only reduces input errors thanks to regularization (as explained in section 3.3.2), but also higher-level workload and execution time since we send them the minimum information they need to achieve their goals. This reduction has been successfully assessed in a driving scenario where we used condensed disparity to extract the ground plane of a road and detect obstacles using that plane [32] (see sections 4.2.1 and 4.2.2).

Before moving to that example, however, we want to focus on another of our goals: integrate attention processes in a traditional vision system. In the next section we show how relevant points can be extracted using saliency maps (as the ones explained in section 2.3) and therefore include bottom-up attention in our system. Moreover, within our driving scenario application, we generate top-down attention signals -such as Independently Moving Objects (IMOs) or TTC- and send them back to our condensation algorithm (see section 4.1.2).

4.1 Integration of Attention Processes

As we explained in chapter 2, Itti et al. [39] difference two kinds of attention processes: bottom-up and top-down. Their definition states that bottom-up selection is target-independent and based on the image saliency. Although the algorithms assessed in the previous section as RP extractor follow a bottom-up approach, we introduce here saliency maps as bottom-up attention process, i.e. as relevance indicators in our condensation algorithm.

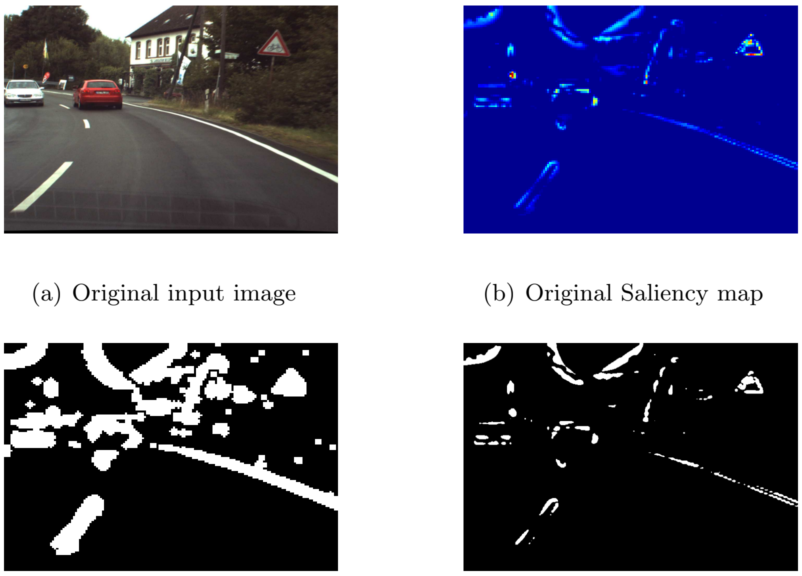
On the other hand, top-down signals are generated by higher levels and give more priority to some areas due to their importance in a given application, i.e. they are target driven. In a changing scenario, such as our driving one, adapting to different situations is critical and therefore one of our goals was to integrate this kind of signals in our system. This approach, however, is not followed by many algorithms due to its slowness [89] extracting and integrating that information. Nevertheless, we have successfully included top-down processes (such as feedback from higher levels) in our condensation algorithm without additional latency or workload, improving algorithms accuracy whilst reducing bandwidth constrains.

4.1.1 Bottom-Up Attention Processes: Saliency Maps

Instead of using a structure-based RP extractor, we are employing saliency maps obtained from the input images as relevance indicators. As we have described in chapter 3, our semidense representation map is able to integrate as RP extractor any signal that selects salient points in the image and, therefore, saliency maps are a logical option we want to assess [40].

We present in this section a saliency-based approach, although a hybrid solution that includes both kinds of RPs (structure- and saliency-based ones) could be implemented. These saliency maps are calculated using Barranco et al. code [5]. In short, using low-level vision features (color -3 hues-, orientation and magnitud), they calculate a branch of features in several scales. Then they apply DoG iteratively until they obtain three different conspicuity maps: one for intensity, one for color and one for orientation. Finally, they are summed into an unique saliency map. In section 2.3 we explain this process in detail.

Although Itti et al. approach sends only the most salient stimuli to higher levels [67], we are going to create a saliency map as big as the original image and represent each stimuli by its saliency value. In Figure 4.1(b) we see the saliency map we receive as input in our RP extractor[1].



(a) Original input image (b) Original Saliency map

(c) Zero-threshold RP extractor (d) RP mask extracted

Figure 4.1: RP extractor using saliency maps. (b) shows the saliency map extracted from (a) red areas represent the most salient areas. (c) shows the binary mask extracted from (b) using all the information provided. (d) is the result of applying a threshold to (c)

As we can see, this map present some points with high saliency (red in Figure 4.1(b)) surrounded by smaller saliency areas (blueish colors). If we use all those points as RPs, we would have redundant information as shown in Figure 4.1(c). This behavior is similar to the one we solved in Canny edge detector (see section 2.2.2.1). In that case, we thinned the energy applying non-maximum suppression and thresholding based on the

[1]Original saliency map extracted using Barranco et al. [5] implementation of Itti et al. algorithm

orientation. Nevertheless, if we want to repeat that process here, we need to extract some structure-based features. Since we want to avoid using any additional structure information (such as orientation or magnitude), we cannot apply non-maximum suppression. Hence, we use a threshold to thin the saliency "bolbs" and discard the less salient areas. Figure 4.1(d) shows a saliency-based RP mask after the thresholding process.

We have tested this saliency-based condensation in disparity and optical flow sequences. In Figure 4.2 we see the disparity calculated using Barranco et al. algorithm [7, 5] and its condensed version. We have condensed that disparity using saliency-based RP extractor (i.e. Fig. 4.1 final mask) and a 9x9 bilateral-based grid, obtaining Fig. 4.2(b). Semidense map final size is 4.21% of the original image size. We have assessed this for several frames of different sequences achieving similar results.

In Figure 4.3 we can see the output of the optical flow calculated using [6, 5]. This optical-flow representation, however, is a subsampled version of the original one. We are showing only 1 of every 10 points so the representation is easier to understand. Figure 4.3(b) shows semidense optical flow using saliency maps as RP extractor and a 9x9 median-based grid. Again, we have subsampled the optical-flow representation, showing only one of every 5 points.

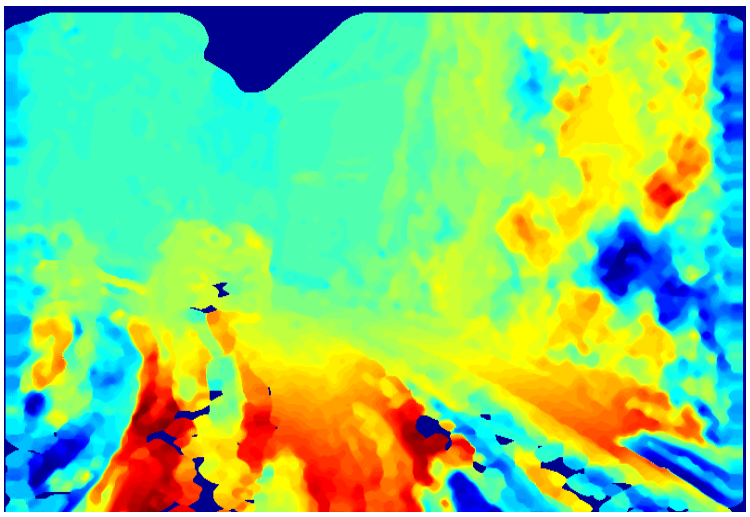
These examples show that saliency maps work as good as structure-based algorithms (such as the ones assessed in section 3.1) as relevance indicators. In fact, this implementation introduces color information (used to extract the saliency maps) into our system that otherwise was ignored. Moreover using our semidense representation, attention can be integrated in any vision system. This integration requires minimal changes in the vision algorithms, mainly due to the presence of non-relevant information in the semidense map as explained in sections 3.2 and 4.2.1.

Now that we have bottom-up attention introduced in our system, we are going to include target-driven one, i.e. top-down attention.

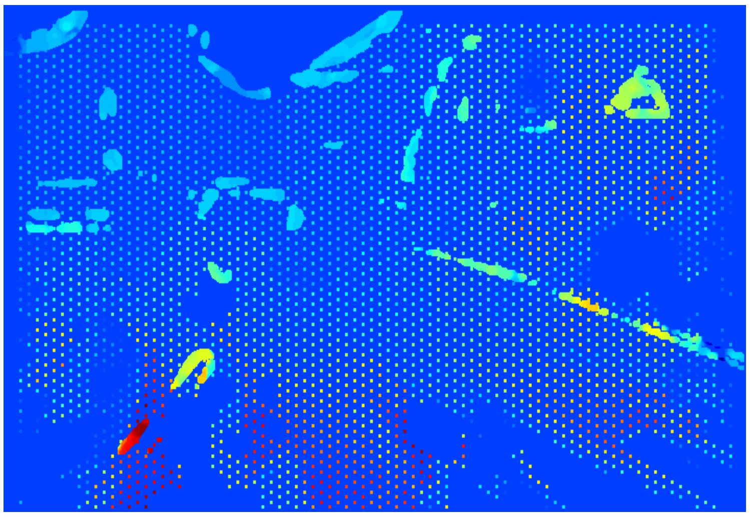
4.1.2 Top-Down Attention Processes: IMOs

As we explained in the previous section, we introduce top-down attention in our system using feedback signals from higher-level algorithms. In this section we illustrate this idea in a driving scenario.

Top-down attention signals are very handy in real-world applications. They are useful to adapt system priorities to changing conditions and facilitate reactions in real-time constrains. In our driving scenario we find several mid-/high-level features that provide top-down information. For instance, when we drive in a city, information from pedestrians walking on

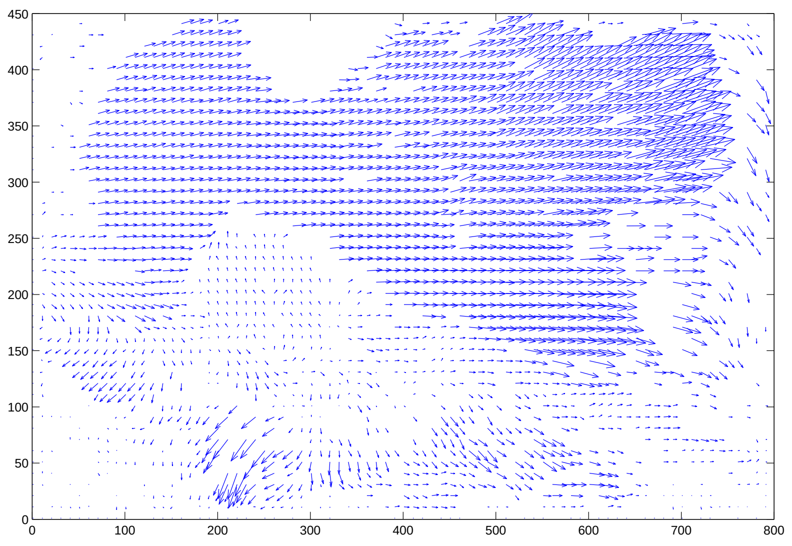


(a) Original disparity

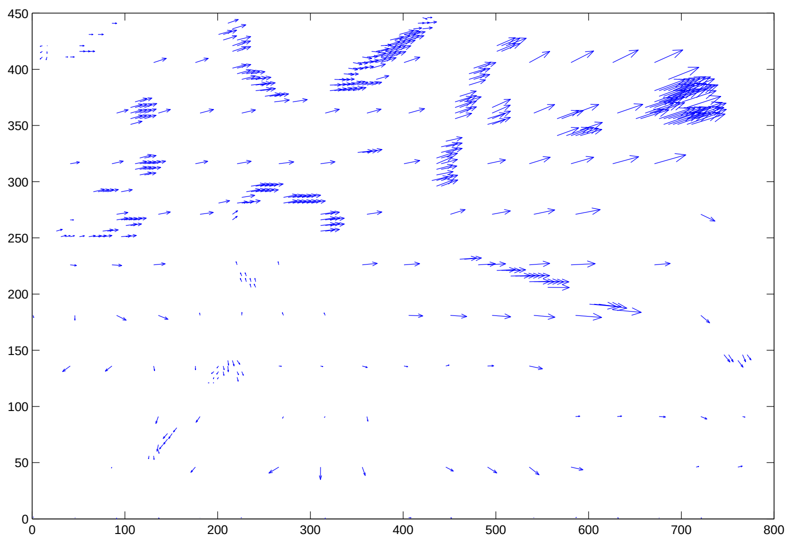


(b) Semidense disparity

Figure 4.2: Disparity condensation using saliency maps. We obtain disparity information (a) using [7, 5] algorithm and then condense it using 4.1 saliency map as RP extractor and a 9x9 bilateral-based grid for the plain areas, obtaining the semidense map (b) (we have dilated the semidense map to improve printing quality).



(a) Original optical flow



(b) Semidense optical flow

Figure 4.3: Optical flow condensation using saliency maps. We obtain optical flow (a) using [6, 5] algorithm and then condense it using 4.1 saliency map as RP extractor and a 9x9 median-based grid for the plain areas, obtaining the semidense map (b). Both images are subsampled to facilitate visualization. (a) shows only 1 point of every 10 while (b) shows 1 of every 5.

the sidewalk is usually less relevant than the one from the road. However, if all of a sudden one of those pedestrians starts crossing the street a few meters from our car, our priorities change. The pedestrian will become part of our interest area and our brain will process it with a higher priority [39]. Similarly, if the car going ahead of us breaks suddenly, our system should receive that information as soon as possible. If we are able to integrate these kinds of information into our system, we would create a signal-to-symbol loop, improving its response. Hence, our goal in this section is to integrate top-down attention signals in our system at run-time, using semidense map as framework.

As a example of this top-down information integration we are going to use Independently Moving Objects (IMOs). We understand that an IMO is a blob that indicates the presence of an object that is moving in a different direction than the rest of the scene [62]. In next section we introduce several approaches to IMO extraction problem. Then we select and explain one of them, IMO extraction based on optical-flow information and egomotion.

4.1.2.1 Independently Moving Object Extraction

Several approaches toward the IMO-extraction problem can be found in literature [27, 58, 62, 71]. Some of them are based on navigation information, such as Simultaneous Localization and Mapping (SLAM) approaches -that create a map of the environment and keep track of a robot in it [27]- and visual odometry ones -that estimate egomotion[2], tracking a large number of features over a small number of frames [58]-. Other approaches has incorporate appearance-based object detection to the IMO extraction [28]. In addition, based on visual features, we find approaches that detect independent motion by removing a global component due to egomotion (as in the flow-parsing approach) and methods that identify clusters of consistent (3D rigid) motion [62, 71].

In this example we use Pauwels et al. algorithm [62]. This approach is based on same low-level-vision features that our low-level-vision system. In Figure 4.4 we can see a scheme of how it works. It receives left and right images and extracts edges, disparity and optical flow. Then, it computes the egomotion of the scene based on disparity and optical flow. Finally, extracts the scene general movement from the optical-flow fields, obtaining those object that moves independently.

Hence, using this approach, we obtain the IMOs in a scene, translate them to top-down signals and integrate them in the system through our semidense map. In Figure 4.5 we can see how the IMO information is inte-

[2]Egomotion refers to estimating a camera's motion relative to a rigid scene.

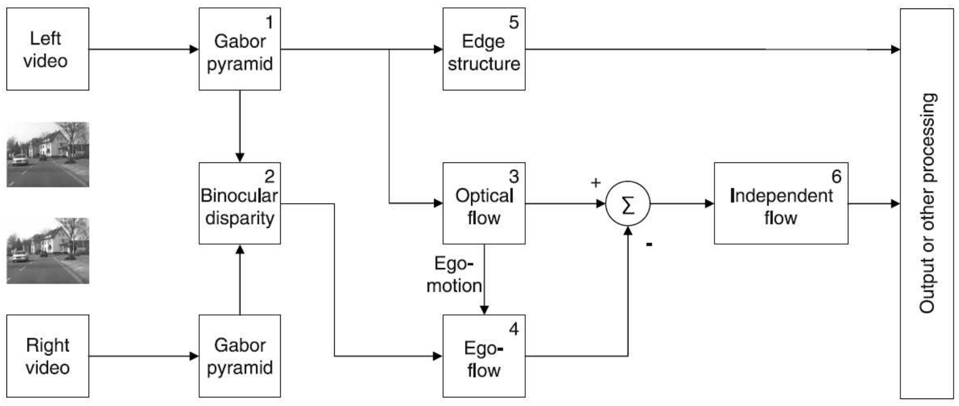


Figure 4.4: Scheme Pauwels et al. IMO-extraction algorithm. We can see how it uses Gabor pyramid, disparity and optical flow as our low-level-vision does. Image extracted from [62].

grated as RPs, i.e. we sent those points relative to moving objects without condensation and at a higher priority.

In our application we cannot obtain the IMOs of the current frame as fast as our real-time system would need. Pauwels et al. algorithm computes 640x512 resolution images at 21 frames per second (fps) [62], while our system processes more than 49 fps for that resolution (see section 5.3). Hence we integrate IMO information from frame N in the semidense map of frame $N+1$. This one-frame top-down integration latency is affordable due to the inherent temporal coherence of the sequence. In other words, since we are processing at 49 fps a blob can change its shape or position, but is not likely that an object only appears in one frame. If it does, it is likely to be an error[3].

Once condensed, the semidense map could be used, for instance, to compute the TTC. This way we would include IMO information without any further computations and using just a small percentage of the original points. This approach reduces computational load (specially important on embeded devices) and bandwidth (sections 4.2.2 and 5.3 include an analysis of these reductions).

With this example of attention integration, we have proved that our system can receive and integrate top-down information without any modification or additional workload. Our next step, therefore, is to assess our semidense map in a complete real-world application: obstacle detection based on ground plane extraction.

[3]otherwise, please contact NASA, UFO division, as soon as possible.

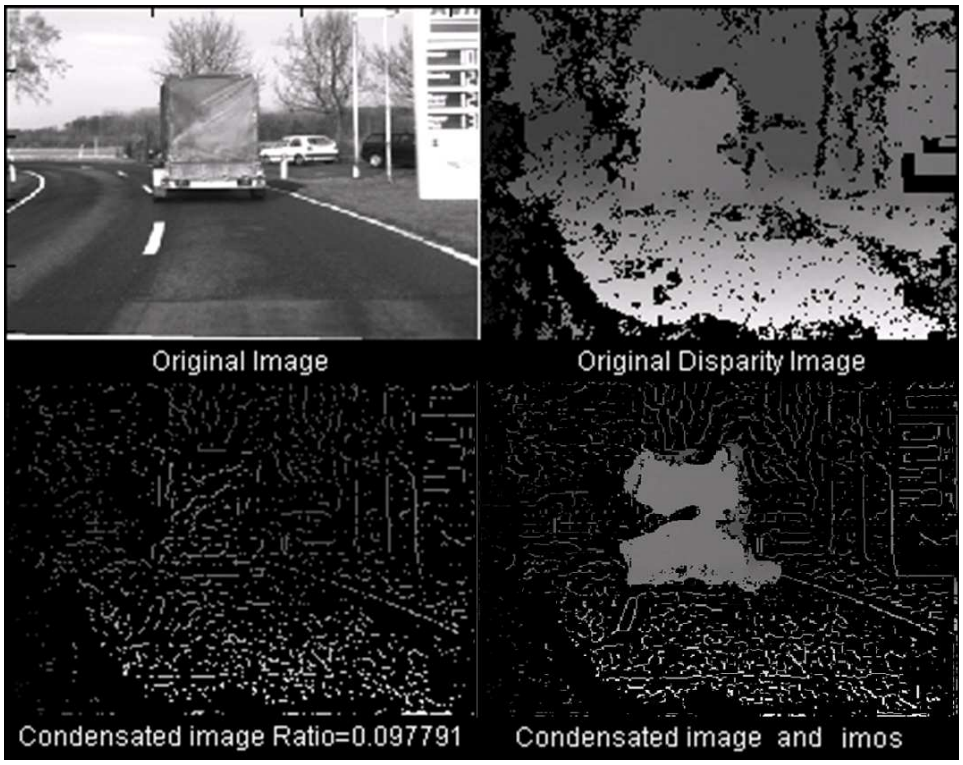


Figure 4.5: IMOs integration in semidense maps. We have included the IMOs calculated using Pauwels et al. algorithm in our semidense representation as RPs.

4.2 Obstacle Detection on a Driving Scenario

Within DRISVCO framework we can find many high-level algorithms that could benefit from our semidense representation map: TTC [15], affine-based regularization [17], egomotion [62] and so. Most of these algorithms has difficulties to be implemented in hardware due to their memory and computational requirements [63]. Hence, using a semidense representation map could be a solution to their integration on embedded systems.

In this section we present one of them, an algorithm to extract the ground plane and use it to detect obstacles [19]. First we are going to focus on the ground-plane detection algorithm. The main goal in that case is to show how non-relevant areas are important to some algorithms and how our semidense map provides a differentiating quality by including a regular grid. With this example we show the limitations of sparse representations and prove that our solution solves them whilst providing a solution as accurate as a dense map.

Then, we move to the obstacle detection algorithm, using the ground-plane extracted in the previous step. This way we show that even higher-levels algorithm benefit from our semidense map.

4.2.1 Ground-Plane Extraction

The ground plane on a road scenario is quite useful when detecting obstacles, whether on robotics applications or on driving ones. Just by looking at its name, it is clear ground-plane extraction is based on plain regions of the image and, therefore, a perfect candidate to test our semidense map. Let us start with a formal definition of the ground plane and how to implement a ground-plane extractor using Chumerin et al. algorithm[4] [19].

4.2.1.1 Implementation Details

As a general rule, we can denote disparity map as a set $\mathcal{D} = \{(x_i, y_i, \delta_i)\}_i$, where δ_i is disparity of the image pixel (x_i, y_i). A plane Π in 3D world coordinate system attached to the nodal point of the left camera can be defined by:

$$\Pi : aX + bY + cZ + d = 0 \tag{4.1}$$

Without loss of generality we assume that $a^2 + b^2 + c^2 = 1$, otherwise one can divide coefficients a, b, c by $\sqrt{a^2 + b^2 + c^2}$. In this case vector $n = (a, b, c)^T$ represents the normal unity vector of the plane Π and coefficient d represents the distance from the camera nodal point to the plane (to avoid front-parallelism of Π, we assume that $b > 0$). Using simple algebraic manipulations it is easy to show, that each disparity map element (x, y, δ), corresponding to the point (X, Y, Z) of the ground plane Π also satisfy a linear model:

$$\Delta\,\delta = \alpha x + \beta y + \gamma, \tag{4.2}$$

where x, y are pixel coordinates in the frame coordinate system, δ is disparity of the pixel (x, y) and coefficients α, β and γ can derived as:

$$\begin{cases} \alpha = -aL/d, \\ \beta = -bL/d, \\ \gamma = -cL/d. \end{cases} \tag{4.3}$$

[4]Thanks to N. Chumerin for the algorithm and images

Where L is the baseline length. The inverse mapping from the disparity domain to 3D world domain is also possible :

$$\begin{cases} a = -\alpha/\sqrt{\alpha^2+\beta^2+(\gamma/f)^2}, \\ b = -\beta/\sqrt{\alpha^2+\beta^2+(\gamma/f)^2}, \\ c = -\gamma/(f\sqrt{\alpha^2+\beta^2+(\gamma/f)^2}), \\ d = -L/\sqrt{\alpha^2+\beta^2+(\gamma/f)^2}. \end{cases} \tag{4.4}$$

From the last two equations (4.3, 4.4) follows that the estimation of the ground plane model (4.1) parameters is equivalent to the estimation of the disparity plane model (4.2) parameters and *vice versa.* That is why we can use the disparity map $\mathcal{D}$ to fit the disparity plane model (4.2) and then convert it to the desired ground plane model (4.1) using (4.4).

If we apply directly the conventional linear regression methods (e.g., least squares) to the disparity plane fitting problem, we can obtain inaccurate results, because the basic assumptions of these methods are not always met:

1. The noise distribution of the estimated disparity is unknown, while it is expected to be Gaussian with zero mean.
2. There are always a lot of outliers in the disparity map.

Robust regression extends classic regression methods making them less sensitive to the outliers or other small deviations from the model assumptions. Among popular methods of robust regression (e.g., Iteratively Reweighted Least-Squares (IRLS) [36], Least Median of Squares Regression (LMedS) [74], Random Sample Consensus (RANSAC) [29]) Chumerin et al. choose IRLS due to its speed and low computational complexity.

In Figure 4.6 we can see a typical driving scenario. In this example, the dense disparity is obtained using a phase-based algorithm [63]. As a preprocessing step before the robust linear regression Chumerin et al. intersect the disparity map (Figure 4.6b) with a heuristically designed predefined ground mask (Figure 4.6c). By this step, they filter out the majority of the pixels which do not belong to the ground plane and are outliers in the disparity plane model. Then the data from the masked disparity map (see figure 4.6d) are used by IRLS to estimate the desired disparity plane model parameters [19].

Finally, the algorithm considers the ground plane estimated correctly if the following conditions are met:

$$\begin{cases} \|\mathbf{n}_t - \mathbf{n}_0\| < \theta_0, \\ \|\mathbf{n}_t - \mathbf{n}_{t-1}\| < \theta_1, \end{cases} \tag{4.5}$$

Where n_k is the normal vector of the ground plane for kth frame, n_0 is normal vector of the canonical ground plane.

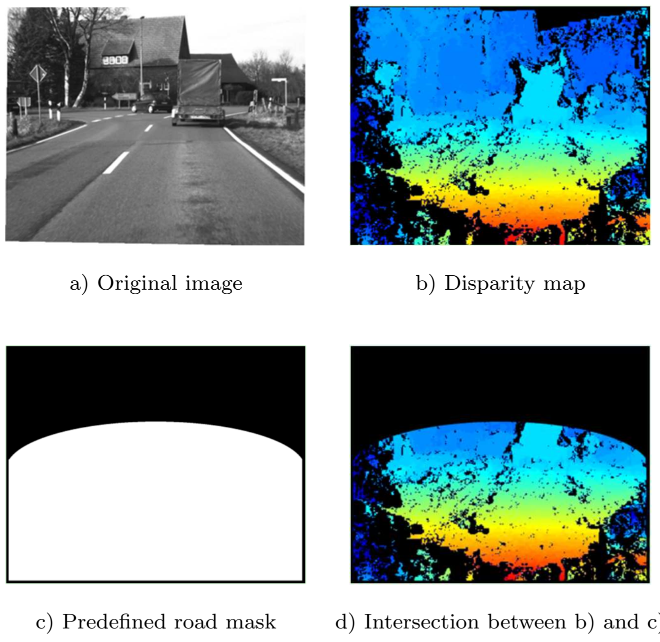


Figure 4.6: Ground-plane extraction scheme. The color in b) and d) indicates magnitude of the disparity: red - large, blue - small. Image extracted from [19]

4.2.1.2 GP-Extraction using Semidense Maps

In [28], Ess et al. developed a system that integrates depth and appearance information for robust pedestrian detection and simultaneous ground-plane estimation from video streams. Nevertheless, this approach uses a dense map and its workload is excessive for its implementation on a real time application (such as on embedded automotive processors). On the other hand, Santana et al. introduced saliency maps to speed up ground plane and obstacle detection on an off-road scenario [77]. Their solution, however, omit non-relevant regions and top-bottom attention information.

Our approach condenses the disparity and use its semidense map as input to the algorithm previously explained. Figure 4.7 compares the dense (original), the semidense (condensed) and the sparse (only RP) approaches. It uses three different measurements: the distance to the ground plane, the azimuth and the zenith.

Figure 4.7 shows how the sparse representation (green dotted line), which is actually a Canny edge detector, has difficulties to extract the ground plane since the algorithm needs information from the non-relevant regions. Other sparse methods would have this very same problem. On the other hand, the grid points (magenta dashed line) are extracted by a bilateral filter approach

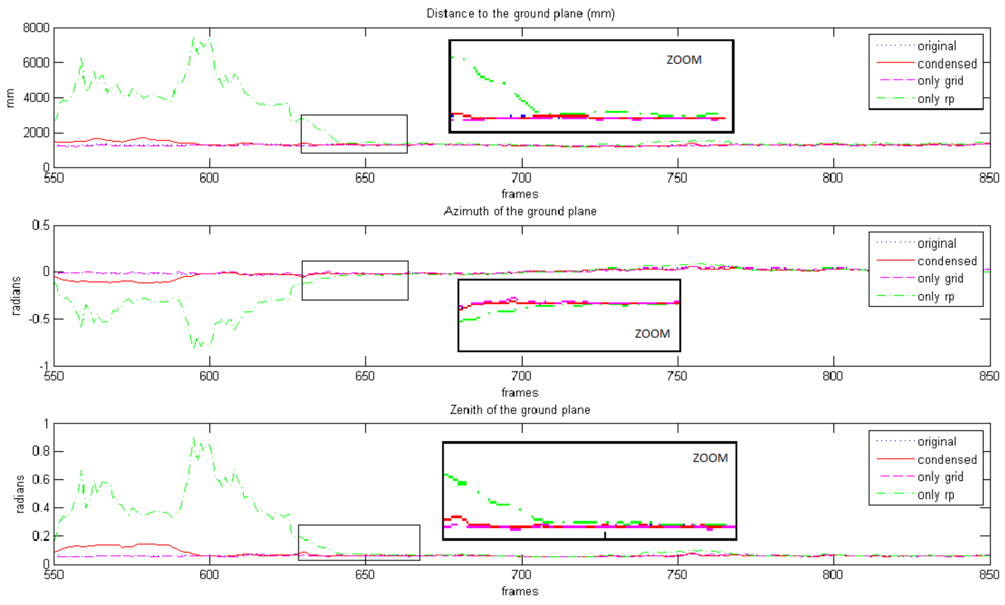


Figure 4.7: Ground-plane output for dense and semidense maps.

(5 × 5 window) and show a much better behavior: their response is similar to the dense map one. The semidense map (red line) smooths some of the noise introduced by the RPs, but cannot match the performance of the grid. These details can be better seen in the zoomed area of Figure 4.7.

This example proves how "non-relevant" regions can be important depending on the application. The versatility of our algorithm is really handy in this kind of applications; since we know beforehand that the ground-plane extractor works better when working only with the grid information, we could inhibit the RPs and avoid processing them. In this stage, we could also add the information from the previous frame as enhancing trigger to the RP extractor; this way, we would include top-down information to our process (as shown in seciton 4.1.2).

4.2.2 Obstacle Detection

We find in literature several algorithms to detect obstacles. Chumerin et al. based on ground-plane to obtain the elevation information and, therefore, detect the obstacles in the scene[5] [19].

[5]Thanks to N. Chumerin for the algorithm and images

4.2.2.1 Implementation Details

Once they extract the ground plane, Chumerin et al. use it to calculate the elevation of the points in the scene, and, this way, detect the obstacles. For each element (x, y, δ) of disparity map $\mathbf{D}$ they define the elevation as the distance from each point $P(X, Y, Z)$ in 3D to the ground plane Π. Based on the classic formula of the distance from a point to a plane and assuming $a^2 + b^2 + c^2 = 1$, the elevation $e(x, y)$ can be estimated as:

$$e(x,y) = \frac{|aX + bY + cZ + d|}{\sqrt{a^2 + b^2 + c^2}} = |(ax + by + f)L/\delta + d| \,. \qquad (4.6)$$

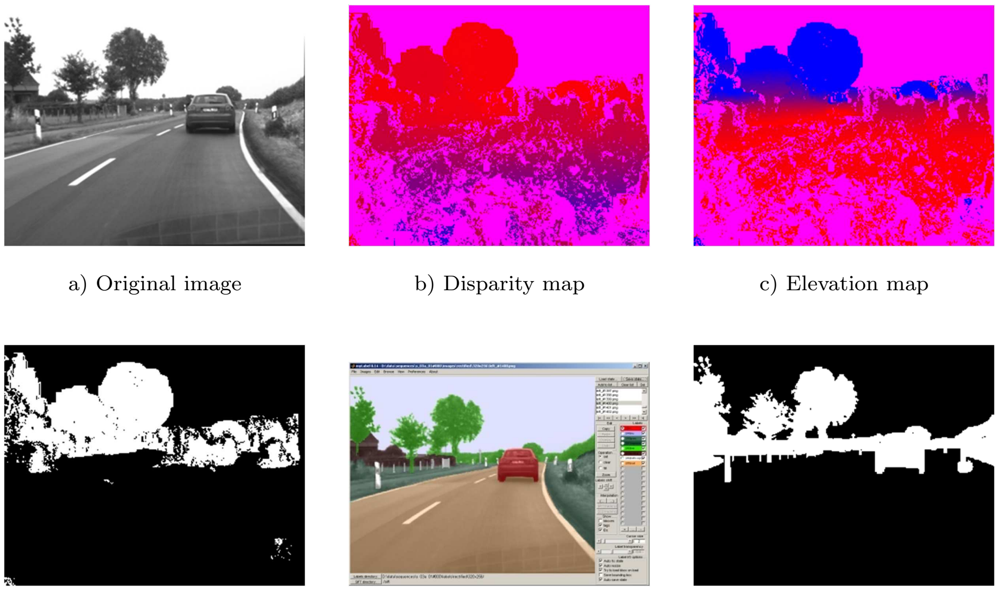
Following this definition it is possible to estimate elevation for each element of the disparity map, and consequently generate the elevation map. An example of elevation map is shown in Figure 4.8c.

The most straightforward application of the elevation map is obstacle segmentation. Indeed, any pixel with positive elevation could be classified/marked as an obstacle. In order to reduce the amount of misclassifications in the above mentioned obstacle segmentation procedure (e.g., due to non-planarity of the ground surface), Chumerin et al. propose to mark a pixel as an obstacle only if its elevation is larger than some positive threshold. In their simulations they use a threshold equivalent to elevation of 0.3 m. An example of obstacle map is depicted in Figure 4.8d. Since this a real sequence, there is no ground truth information for it. We have manually marked them, as shown in Figure 4.8f, using a MATLAB tool (Figure 4.8e) [18].

4.2.2.2 Results using Semidense Maps in Obstacle Detection

Using the semidense disparity and the ground plane obtained in section 4.2.1, we improve the performance of this obstacle detection algorithm since the algorithm would need to process less than a 10% of the initial data. Figure 4.9 compares the output of the obstacle detection algorithm using dense (Fig. 4.9(c)) and semidense (Fig. 4.9(c)) maps. These obstacle masks are obtained using the information from the ground plane previously mentioned, which is represented here as a blue line, and the elevation maps calculated using (4.6) (Fig. 4.9(a) and 4.9(b)). Finally, we represent the original input image, the ground plane and the obstacle mask in one figure for both cases: dense (Fig. 4.9(e)) and semidense (Fig. 4.9(f)). We can see how both outputs are equivalent.

We have also assessed the performance improvement of our representation map for different driving sequences. Using a PC with an Intel Core 2 Duo T6400 at 2.00 GHz processor and 4 GB of RAM memory, we have



a) Original image b) Disparity map c) Elevation map

d) Detected obstacles e) Labeled image f) Obstacle ground truth

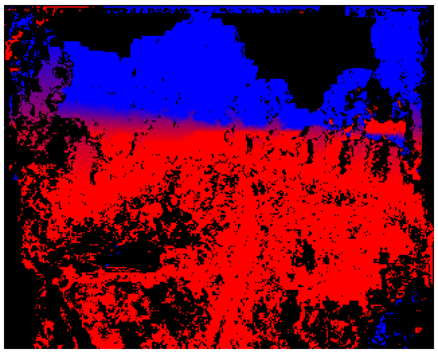
Figure 4.8: Result of the elevation map (c) and the obstacle map (d) estimation for the original frame (a) based on the disparity map (b).

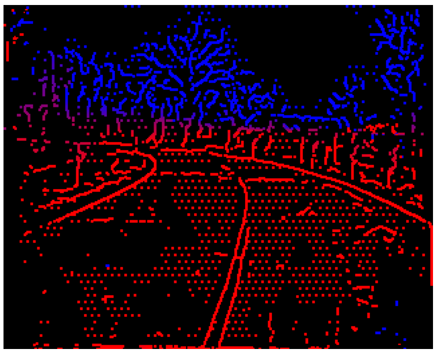
assessed the obstacle detection algorithm for three real sequences. Each sequence is composed of images of 320x256 pixels. Disparity computing time has not been included in the measurements. Table 4.1 shows the difference of execution time (in seconds) for the ground-plane and obstacle detection algorithms with and without condensing the dense feature. As we can see in the table, we have reduced the memory consumption (semidense maps under 9% of the originals) and the execution time (ground-plane extraction, for instance, is around 10 times faster). These results show clearly how our representation map suits better for embedded systems than a dense one.

Sequence		**Original dense map**		**Semidense map**		
Name	#frames	Ground pl.	Obstacle	Ground pl.	Obstacle	% Conden
Seq. 1	284	7.886 s	23.418 s	0.693 s	18.769 s	8.105
Seq. 2	362	7.754 s	42.053 s	0.807 s	28.973 s	7.912
Seq. 3	550	13.429 s	58.199 s	0.970 s	38.145 s	7.513

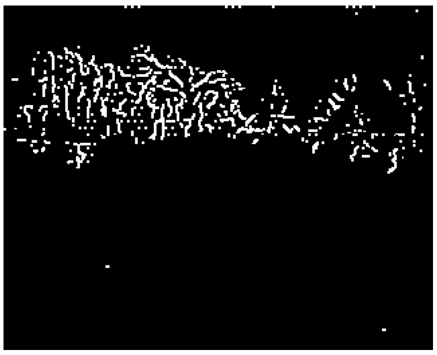
Table 4.1: Performance test. 'Ground pl.' corresponds to the ground plane algorithm; 'Obstacle', to the obstacle detection algorithm; and '% Condensation', to the percentage of points of the original feature we are actually using in the semidense approach.

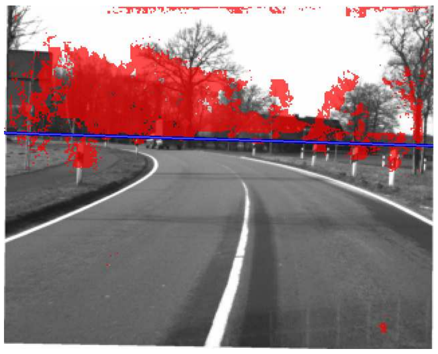
As we can see in Table 4.1, the execution-time reduction achieved by the ground-plane algorithm is bigger than the obstacle detection one. This is


(a) Dense elevation map


(b) Semidense elevation map

(c) Dense obstacle mask


(d) Semidense obstacle mask


(e) Dense output: Obstacles + ground plane

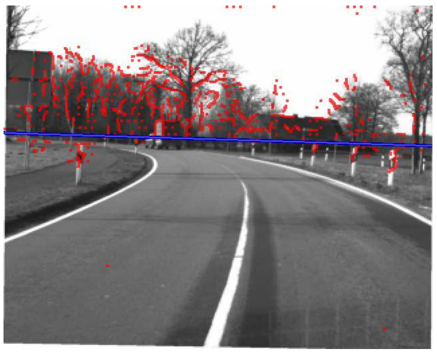

(f) Semidense output: Obstacles + ground plane

Figure 4.9: Comparison dense vs semidense obstacle detection. Using the equation (4.6), we extract the elevation maps: (a) and (b). Based on them and on the ground-plane information, obstacle masks are extracted: (c) and (d). (e) and (f) represent the final output of the algorithm, combining obstacle masks and ground plane (blue line).

because the latter one is not optimized to work with non-dense maps. Thus, the method do not take advantage of our semidense map. Although we could modify the algorithm to use our map as an input, a faster and easier approach would be to interpolate the semidense map as shown in Figure 3.10. This versatility makes our map easy to integrate in current vision systems, although minor modifications might be requiered.

4.3 Conclusions

In this chapter we have shown many of the features we wanted to include in our semidense representation (see section 3.4 for details).

First, we have shown its versatility using saliency maps as RP-extractor. The obtained results are similar to the ones achieved by the structure-based approaches presented in section 3.1.1.

Second we have successfully integrated feedback from higher levels, i.e. top-down attention signals (in this example IMO information), to our semidense representation map without workload increment. This functionality has not been explode in many applications [89], while our semidense map performs it without any modification or performance detriment. This integration capability allows for creating a signal-to-symbol loop in our system [72].

Finally, we have used our semidense representation in a real-world application, obtaining a similar response as using dense maps and dramatically reducing the execution time (up to 10 times faster). This real-world application consists on two steps: ground-plane extraction and obstacle detection. We have successfully used a semidense disparty in both steps of the application.

Another remarkable achievement of this chapter is the corroboration of the importance of the "plain-regions". Some algorithms, such as the ground-plane extraction one, employ those regions to extract information. Hence, as we have advanced, those regions are relevant for many algorithm and should not be ignored. Our semidense representation map has proved to be a very good solution for this kind of algorithms since it preserves the plain-region information.

Chapter 5

Implementation on reconfigurable hardware

In the previous chapters we have presented several examples of visual features extracted using software implementations of different algorithms. These implementations are fast to develop and easy to modify. This is why, for research purposes, we use those kinds of implementations to validate our methods. These solutions are usually implemented using high-level languages and executed in conventional general purpose processors. The processing power of these machines has grown up very quickly (following the Moore's law). Moreover, many programming languages are available, allowing the researchers to implement different approaches in short time. In our research, for instance, we have used MATLAB environment that includes several imaging libraries that have eased our design and reduced our implementation time.

Nevertheless, if we need to achieve real-time requirements, we cannot rely just on general purpose processors. We could adapt our algorithm to a specific processor, generating an optimized version of it and performing a better response [1]. Due to the versatility of our algorithm, we would need to adapt each RP extractor and grid filter, or select the best fitting one and optimize it. This approach, however, is time consuming and its optimization varies from one processor to another. Moreover, any solution based on a general purpose processor require more memory and power-consumption than any embedded system could afford.

Another possible solution is to employ a specific hardware accelerators such as Graphic Processing Units (GPU). These units have been designed to handle matrix-like data, so they work efficiently with images. The solutions based on GPU are generally implemented using a C-like environment with specific tools for parallel such as CUDA [59] that vary depending from the hardware platform. GPUs have improved very fast mainly due to the game industry, evolving to better solutions in terms of parallelism and memory access speed. In fact, some of the applications executed using these platforms achieve real-time requirements [63]. The solutions implemented in these platforms, however, depends on a high-computational functions that also require big quantities of memory to execute. Hence, the energy consumed by these processing units is too high for a SoC.

If we are planning to develop a SoC, the previous options are not suited, a better fitting option is a digital signal processor (DSP). DSPs are specialized microprocessors with an architecture optimized for the fast operational needs of digital signal processing, whilst keeping the energy-consumption from rising. The main problem is the DSP are optimized for specific simple operations. Although our condensation algorithm is simple enough to be implemented using DSPs, if we want to implement a complete low-level-vision system, we need a more powerful approach [11].

Another low-power-consumption approach that works very well with embedded applications is a FPGA. FPGAs main advantages are their parallel-processing capacities and their real-time speed [25, 84]. On the other hand, the main disadvantages of FPGAs are: limited on-chip memory, slow implementation time and high time-to-market. The limited-memory problem is the main goal of our condensation algorithm and, as can be seen in section 5.3, we have successfully solved it. In addition, many FPGA processing boards currently include co-processors (such as DSPs or even GPUs) that efficiently performs specific operations. This combination speeds up FPGA implementation time and performance. Therefore, FPGA approach seems to be the best option to develop an on-chip low-level-vision system that extracts, condenses and transfers several visual features (such as energy, orientation or optical flow). Moreover, this solution low-power-consumption facilitates its integration in a SoC (such as robots, vehicles or medical low-vision devices).

In this chapter we present the hardware implementation of our condensation algorithm and its integration in DRIVSCO FPGA solution [85]. First we introduce the architecture of our condensation core and the different submodules it is divided on. This condensation core is the adaption of our condensation algorithm to an embedded system as this one [33]. Then, we present the communication protocol designed to efficiently transfer data between the FPGA and the co-processor. After that we explain in detail the different submodules of our core and the resource consumption of the system. Finally we present some results of our hardware implementation: the memory and bandwidth reduction in the system and the integration in real-time of feedback signals. These results shows how this hardware implementation keeps the properties achieved by the software models presented in previous chapters whilst taking advantage of the embedded capabilities of the FPGA devices.

5.1 Architecture Design

Our vision system comprises a low-level-vision feature extractor and a condensation step that integrates them, whilst reducing the transfer bandwidth. The condensation system works as an independent core in an FPGA device, concretely a XircaV4 platform from Seven Solutions [81]. This platform includes a Virtex4 XC4VFX100, with 94,896 logic cells and 4 ZBT SRAM memory blocks with 8 MB each (a total of 32 MB on-board) [93]. Tomasi et al. [85] feature-extraction architecture follows a fine-grain pipeline structure capable of processing one pixel per clock cycle, using intensively the potential parallelism available in FPGA.

This vision system, as we showed in Figure 1.4 in chapter 1, generates several sparse features and two different dense visual features: optical flow and disparity information. This means we need to condense those two dense features at the same time. Hence, we have to focus on designing an scalable architecture.

In Figure 5.1 we present an simplification of this low-level-vision system that extracts optical flow, energy and orientation. On the other hand, section 5.3 includes Figure 5.6 that represents Tomasi et al. complete system. Although in section 5.3 we explain in detail the latter of these figures, we include this simplification here to give the reader a general idea of how the condensation core is connected to the low-level-vision system as well as the hierarchy of the different submodules that compose our core.

The Memory Controller Unit [88] and the interface with the PCIe were developed using VHDL, the low-level-vision system and the condensation modules were implemented using the high level Handel-C language. This tool allows us a simpler algorithmic description without a significant loss of performances [61]. The implementations and integrations of the complete system were carried out using the Xilinx ISE Design Suite and the DK Design Suite for Handel-C [93].

Figure 5.1 shows how we have divided our condensation algorithm in several submodules (represented as red boxes). Hence, our condensation core follows a multi-scalar fine-grain pipeline, i.e. it is composed of several stages that allow us to execute multiple instructions at the same time and each of those stages is composed of several units working in parallel. Thanks to this architecture, each submodule processes one pixel per clock cycle and ensures correct interconnection to the other cores of our system. Once condensed, the semidense visual features are packed in an efficient way, ready to be

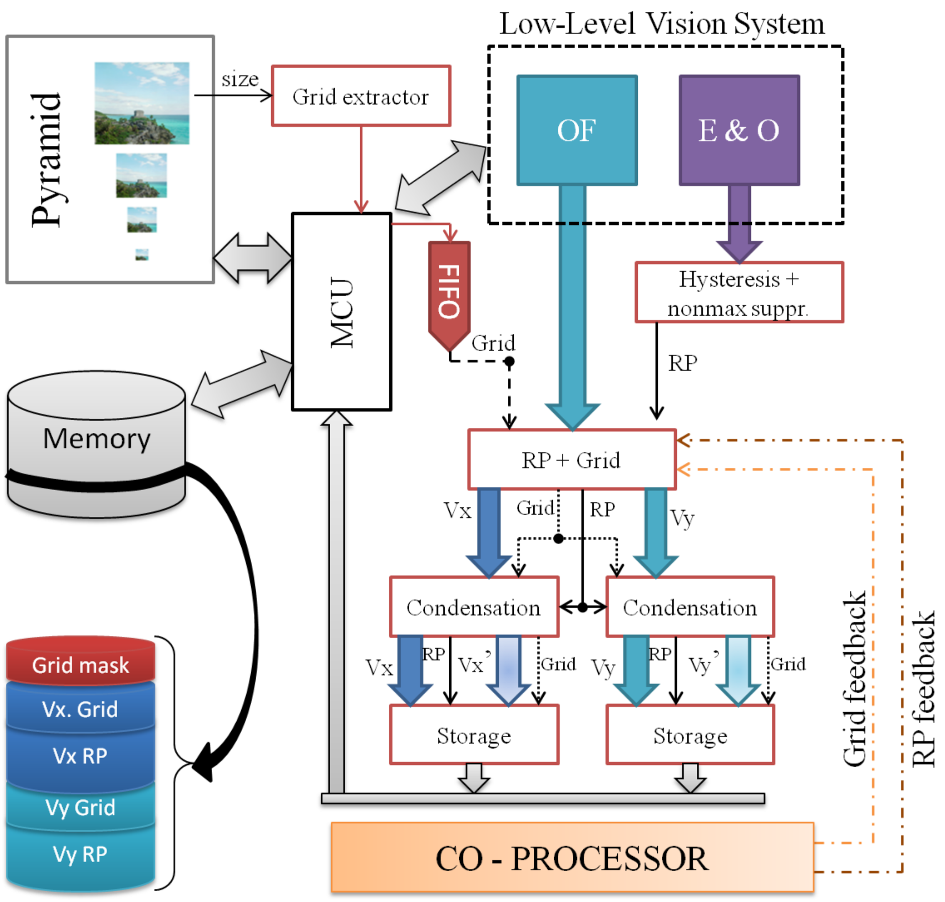


Figure 5.1: Integration of condensation modules (in red) in a optical-flow extraction system. Optical-flow extraction core is represented by a dotted box and for simplicity we assume it processes optical flow (OF), energy and orientation (E&O). We also include connections with Memory Control Unit (MCU) and a scheme of how we storage condensed information.

sent to the PC using a PCI Express interface. This interface uses share memory, so our condensed data must be stored in it as shown in the figure. In following sections we explain in detail each of these submodules.

As we have already mentioned, our condensation module receives information as feedback from other stages of the system (as shown in chapter 4). Hence, we have designed it to receive a mask of external feedback signals (see arrows from the co-processor in Figure 5.1).

5.1.1 Communication Protocol

Using our semidense map, the amount of data to transfer is highly reduced. Nevertheless, we still have to send it to upper-processing-levels efficiently. We assume that our two kinds of data (grid and RPs) can be sent using different methods in order to achieve the best result available. In this section we establish different communication schemes for both, grid data and RPs.

5.1.1.1 Grid Transfer

Since the grid extracted is regular (see section 3.2) and we fix its window size at the beginning of our system (see Figure 5.1), we know beforehand the positions of each grid point. Therefore, we could benefit from this information and avoid sending data we could easily calculate in the co-processor. We have studied two different options: sending just the grid values, and sending the grid values plus a binary mask.

On one hand, we could transfer a list of grid-point-data, without sending explicit address or position information. In fact, we could concatenate two grid values in on register, assuming a 32-bit register and a visual-feature bitwidth of up to 16 bits. In Figure 5.2 we can see a scheme of this solution. The main disadvantage of this approach is the grid size and position are hard-coded and, in case they vary, we would need to re-implement some or our cores and functions.

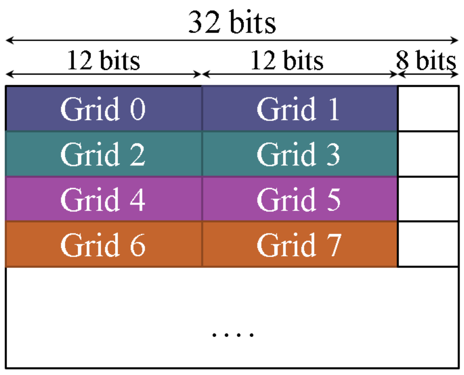


Figure 5.2: Example of how grid points can be stored without including their addresses, just concatenating their values

On the other hand, we could solve this problem if we storage and send, not only the grid values, but also a binary matrix indicating their position. This way, using a small amount of memory, we would have a more versatile storage. The binary matrix could be calculated by the higher-level algorithms or, to provide even more versatility, send it whenever it changes

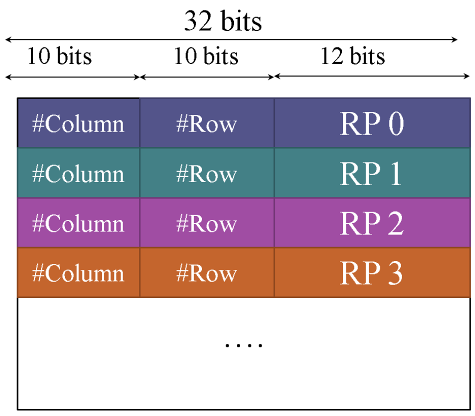


Figure 5.3: Example of how RPs can be stored and transfered using a AER protocol similar to the one defined by Boahen et al. [13].

(typically only once, when generated). This is why we have chosen this option.

5.1.1.2 RP Transfer

Since RP are extracted following a non-regular distribution (see section 3.1 for details), we cannot know beforehand their positions. In this case we have considered two options: AER protocol or binary matrix plus list of data.

On one hand, we could follow Boahen et al. bio-inspared approach [13] and use a address-event representation (AER) protocol. This protocol consists on transferring a register per each datum containing it address (row and column number) and its value. Figure 5.3 shows a scheme of this apporach. This option, therefore, would use a complete memory word per each RP. We have tested this option in our driving scenario, where structure-based RP are between 1-4% of the original image size. With these percentages, this approach is the best option, achieving an efficient and bio-inspired communication protocol. Moreover, this option would allow us to assign priority to the different RP and transfer first the most salient parts.

On the other hand, more complex input images (such as more textured ones) would not benefit so much from this AER protocol. Hence, our second approach is to use the same scheme as the grid points. Since our RPs are nothing but a binary mask similar to the one extracted by edge detectors (see section 2.2.2), we could store this mask in memory and transfer it along with a list of RP values for those positions. However, this approach is not efficient for percentages under 4% and does not allow priority transfer.

Since our condensation core is going to be integrated in DRIVSCO framework (where images usually have less than a 4% of RPs), we have decided to use AER protocol. In addition, this approach facilitates a priority classification very useful for some of the high-level algorithms of our project (such as TTC and IMOs).

5.2 Implementation Details

We need to integrate our condensation core in the low-level-vision system. The available resources in the FPGA, however, are limited. Hence, we need to design each of the submodules of our core taking into account the filters and modules available in our system so we can reuse as many as possible. For instance, although bilateral filter was the best option when extracting the plain-region information (see section 3.2), we are going to apply median filters as the ones used in the vision system to reuse the ones used by the low-level-vision system. All these simplification details are explained in this section.

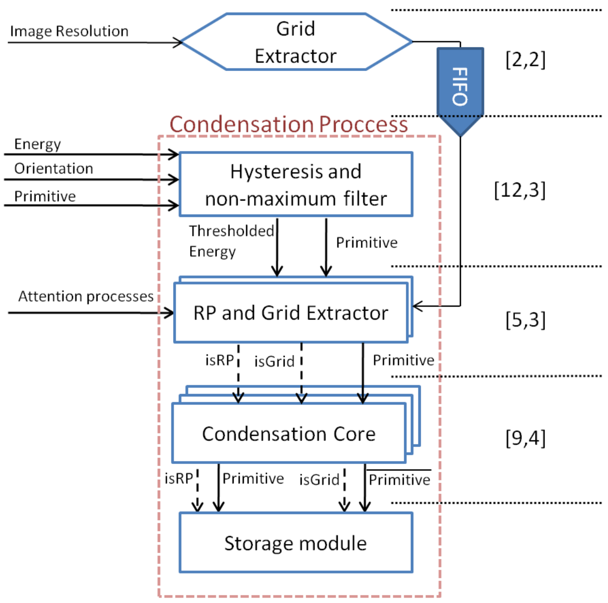


Figure 5.4: Condensation core architecture. Each box corresponds to a submodule in our architecture. Each module is implemented following a multi-scalar pipeline, where the numbers in brackets represent the number of stages (first component) and the number of scalar units (second component) of that module.

We have summed up the condensation core architecture in Figure 5.4. We have included different submodules as boxes. The first submodule, grid mask extractor, is usually executed once, when we initialized the system (see section 5.1.1 for details about its use). The other modules, enclosed in a dotted square, compose the loop of our system. We can also see some multiboxes. They represent submodules that need more than one instance to condense several features. As we mentioned, we follow a fine-grain pipeline approach, so we also include information about the number of stages and scalar units of each submodule (numbers in brackets in Figure 5.4). In the following subsections we explain in detail these submodules.

5.2.1 Grid Mask Extractor

Once we know the input image size and the window size of our grid, we can calculate the positions of the grid points (see more details in section 3.2.1). Thus, we just need to calculate once which points belong to the grid: when we initialize the system. Moreover, since the positions of the grid points are fixed, we do not need to explicitly store them (as explained in section 5.1.1). This is why this module stores in embedded memory a binary mask containing '1' for those positions of the image belonging to the grid and '0' in other case (as shown in Figure 5.6).

Nevertheless, this stored mask can be changed when needed, even at run time, adding flexibility to our system. For instance, we could adjust our grid window size in real time, this way we could use a 5x5 window and, whenever we need more (or less) data, rearrange it to send a 3x3 (or 9x9) one. This configuration allows us to easily control the bandwidth at run time (see future work in section 6.2).

In section 5.2.3 we explain how to use this mask to generate the semi-dense map. As shown in Figure 5.4, the number of units working in parallel in this function is 4, i.e. 2 stages per each feature in our system (disparity and optical flow).

5.2.2 Hysteresis and Non-maximum Filter

As mentioned in section 3.1.1, we receive as inputs energy and orientation extracted in previous stages of the vision system. The energy received, however, need a non-maximum suppression step so we remove redundant information and therefore obtain thinner edges (as explained that section).

Comparing current pixel orientation (i.e. its gradient) and its neighborhood ones, we decide whether this pixel is a local maximum or not. Then, we apply the hysteresis process using thresholds (high and low) that adapt dynamically. For this, we maintain historic information (i.e. a histogram) of the current feature frame and, once computed its last pixel, we use that information to calculate the new thresholds. This way, our thresholds use previous image information to adapt to a changing environment. This one-image threshold estimation latency is affordable due to the inherent temporal coherence of the sequence.

This module's output is a bit which is '1' if the point is really a maximum, and therefore a RP, or '0' if its been marked as non-relevant. As shown in Figure 5.4, this submodule is composed of 12 stages, and we need 3 scalar units (one for disparity and two for the optical flow velocity components, Vx and Vy), i.e. a total of 36 units of our pipeline working in parallel.

5.2.3 RP and Grid Extractor

At the beginning of this submodule we read 4 bits corresponding to 4 indicators: calculated-RP (received from the previous module), calculated-grid (read from embedded memory and calculated by "Grid extractor"), and feedback-RP and feedback-grid (sent by the co-processor). If no feedback information is available, the latter values will be '0'.

Using these bits, this module sends as output a 14-bit register. The first bit -called 'isRP' in Figure 5.4- is '1' if the point is a RP (no matter if it was selected as RP by the previous submodule or by a feedback signal). The second bit -called 'isGrid' in Figure 5.4- is '1' if its a grid point (again, independently of who marked it as grid). Finally, the last 12 bits contains the feature value for that pixel (assuming its bitwidth is 12 bits).

As we can see in Figure 5.4 we need two instances of this module, one for disparity and another one for optical flow. This way, we can receive independent feedback for each of the features. The module is composed of 15 units (5 stages · 3 scalar units) working in parallel. As we mentioned, the whole system process two instances of those scalar units, although the number of stages is the same, i.e. the whole system has a total of 30 units $(5 \cdot (3 \cdot \mathbf{2}))$.

5.2.4 Condensation Core

This module is on charge of extracting the non-salient information, as shown in Figure 5.4. It applies a 5x5 median filter to each grid point (see sections 2.1.1.3 and 3.2.2 for details). Due to the FPGA parallel nature, we apply this filter to each feature pixel and then concatenate the grid indicator received from the previous submodule.

As output, this module has two registers: one containing the input feature (unchanged) and a bit indicating whether it is a RP, and another one containing the median feature value and a bit indicating whether it is a grid point.

In this module we need four scalar units because we are deciding or extracting four values: median filter of the dense feature, input value transfer, one bit indicating whether this point is a RP, and one bit indicating whether it is a grid point. In addition, as shown in Figure 5.4, it is also necessary to have several instances of this module. Figure 5.6 shows how this instantiation is done for the optical flow, using two of this submodules. In the whole system, however, we need to triple the number of scalar units so we can condense the disparity and the two optical flow components (Vx and Vy) [1]. Hence, since the module has 9 stages with 4 scalar units, i.e. 36 units working in parallel, and our system has a total of 108 units ($9 \cdot (4 \cdot \mathbf{3})$).

5.2.5 Storage Module

Since the communication with the PC is done through shared memory, this module sends to the memory control unit (MCU) [88] the RP or grid values, that are then stored in the RAM cells. We have used a hybrid "regular and event-driven" scheme that stores data in two different areas of the memory (see Figure 5.5). The storage place and way will depend on data type (relevant or grid point).

On the one hand, the event-driven part of the protocol stores the relevant points following an AER scheme [13], as explained in section 5.1.1. This means the relevant points are stored directly; one relevant point per 32-bit-memory address as each of them contains explicitly its address (row and column, using 10 bits each) and its feature value (using 12 bits). The memory (and therefore the bandwidth) needed depends on the number of relevant points.

On the other hand, the grid data are regular, this means bandwidth requirements are low and constant due to the sub-sampling applied. Two grid data values are packed together before being stored in the memory (see section 5.1.1), since each one requires 12 bits and a register (and memory word length) has 32 bits. As the grid definition is fixed, we explicitly avoid storing their addresses because this would be redundant.

An special case are those points with double representation: RP and grid point. Since we do not want to include redundant information in our map, we need store them either as RP or as grid point. Since we consider RP more important than grid ones, we discard the latter. However, as we are storing grid points without including their coordinates (x and y), we cannot omit any value. An easy solution to this inconvenience is to substitute the calculated grid value with NaN. This way, the point will be discarded in higher levels.

Following this protocol, instead of storing and transmitting a dense feature with the same resolution as the original image, we only need the information of the relevant points and a regular grid.

The final memory organization is shown in Figure 5.5. As we can see, the regular information has a limited storage space and the relevant points' space is only restricted by the RAM size. This Figure also provides an example of memory use for a 512x512 image assuming 2% of relevant points.

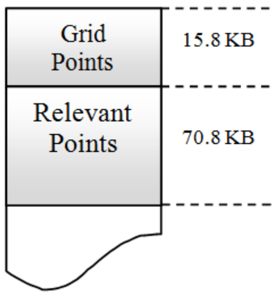


Figure 5.5: Physical memory distribution. For a 512x512 image, assuming 2% of relevant points, the regular grid-based information is stored in the first 15.8 KB of the memory and the RPs space is located in the following 70.8KB. For the example image used, the data sent to the upper levels is around 86.6 KB instead of the 576KB needed without condensation.

5.2.6 Resource Usage Analysis

The resources used by our condensation core are indicated in Table 5.1. We have not only included the consumption for one generic dense visual feature but also the resource need to condense all the dense features of our low-level-vision system, i.e. disparity and optical flow. It is interesting to compare these two cases. If we just condense disparity information, our system needs 8% of the system resources. However, since we have designed our core to be scalable, adding optical flow condensation requires just 12% of slices, instead of 24% needed to generate three complete instances of the whole core. This is because we only double the *RP and Grid Extractor* scalar units (just 1% increase) and triple the *Condensation core* ones (4% increase). Note also that instead of the 194 RAM modules expected ($64 \cdot 2 + 22 \cdot 3 = 194$) the hardware synthesis tool optimized the final architecture, simplifying it and sharing some resources by finally reducing the total embedded memory utilization.

As a comparative note, a single JPEG core [20] implemented in this architecture needs 17% of the slices available per dense feature, i.e. 51% for

Submodule	Slices (Out of 84352)	DSP (Out of 160)	Block RAM (Out of 376)	Freq. (MHz)
Grid Extractor	82 (1%)	0	1	177
Hysteresis and non-maximum	1848 (4%)	0	9	87
RP and Grid Extractor	297 (1%)	0	64	131
Condensation Core	1105 (2%)	16	22	92
Condensation module (1 feature)	3432 (8%)	16	95	90
Condensation module (2 features)	5357 (12%)	48	139	59

Table 5.1: Use of hardware resources for each submodule in a reconfigurable platform with a Virtex-4 FX100.

our whole system, which means that our system not only performs a smarter analysis of visual features, but also consumes less resources.

5.3 Results and Examples

Before moving to the results achieved by our condensation core, we want to explain in detail how we have included our condensation core in the low-level-vision system.

It is out of the scope of this dissertation to explain the details of how the low-level-vision system is implemented. A complete explanation of this system, including energy, orientation, phase, disparity and optical flow extractors, can be found in [84]. Figure 5.6, on the other hand, shows an scheme of how we have integrated our condensation core in that vision system. This image is similar to Figure 5.1, but including disparity extraction core to the low-level-vision system.

In Figure 5.6 we have represented condensation core modules in red. We can see how *Grid extractor* is called as soon as the image size is known. From then on, grid mask is read from memory (using a FIFO). For each image, *Hysteresis and non-maximum suppression* receives energy and orientation to extract RPs while the dense feature is directly received by *RP and Grid Extractor*. In this submodule, signals from higher level are integrated, transferring RP and grid indicators to the next level. If required, this submodule also divide the dense feature in simpler components, such as Vx and Vy components. We can see how the next submodules, *Condensation Core*, convolve each component and send to *Storage* the visual feature (disparity, Vx or Vy components in this example), a register indicating whether it is a RP, the convolved feature, and a register indicating whether it is a grid point. Finally, this submodule selects the corresponding value and transfers it to memory.

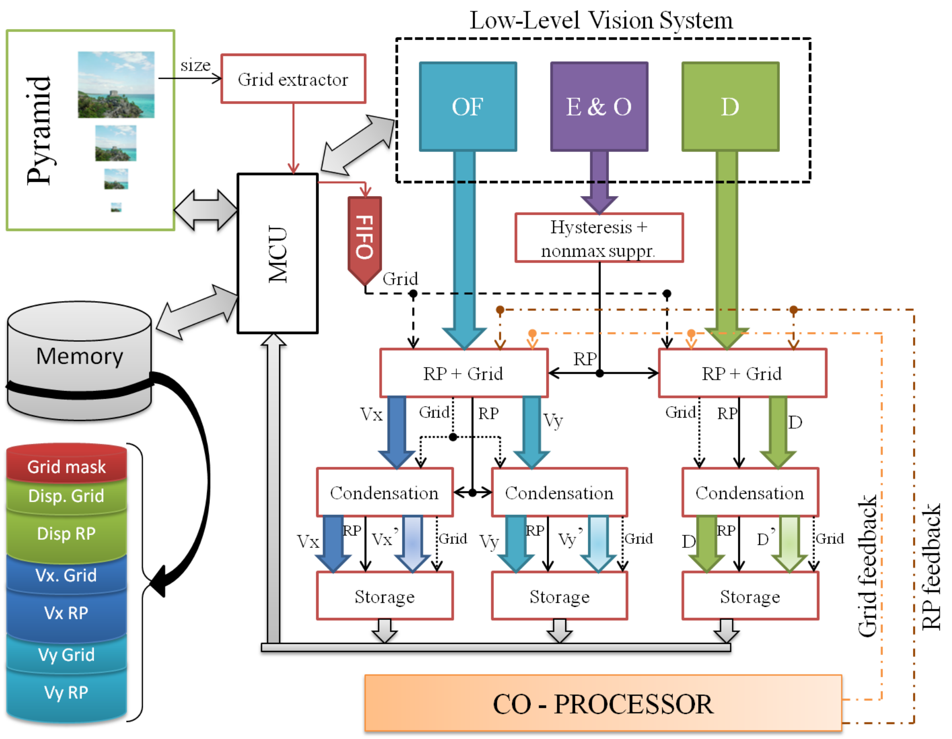


Figure 5.6: Integration of condensation modules (in red) in DRIVSCO low-level-vision system. Low-level vision is represented by a dotted box and for simplicity we assume it only processes disparity (D), optical flow (OF), energy and orientation (E&O). We also include connections with Memory Control Unit (MCU) and a scheme of how we storage condensed information.

It is important to check whether our condensation core is able to maintain the whole system throughput. Since our system processes one pixel per cycle and our core frequency is 90 MHz when condensing one feature (as shown in Table 5.1), we deduce that we can process 90 Megapixels per second. If our input image resolution is VHA, i.e. 1024x1024 pixels, we need to process 1 Megapixels. Hence, our condensation module can process 90 frames per second (fps). On the other hand, if we are condensing disparity and optical flow, our system processes 59 fps. This speed, however, is limited by the low-level-feature extractor. In fact, [85] states that the whole system frequency is 49 MHz. From this we conclude that our condensation system does not introduce any delay or frequency penalty in our vision system. Moreover, in section 3.3.2 we presented the inherent regularization capabilities of our algorithm. These capabilities are very handy in this low-cost hardware implementation as we showed in Figure 3.3.2.

Now that we know our condensation core does not include any delay and is able to condense in real-time the different visual features of the system, let us check if it solves the bandwidth constrains of our system (introduced in chapter 1), while keeping the feedback capabilities presented in chapter 4.

5.3.1 Bandwidth Reduction in DRIVSCO Framework

Although in chapter 1 we introduce the bandwidth requirements of our system, a more detailed studied is needed to fully understand the results achieved by our condensation algorithm.

As a rule, our low-level-system receives two input images, of which maximum resolution is 1024x1024 pixels, and we extract features known as local contrast descriptors (LCDs) [24]. These LCDs are energy, phase and orientation and need 8 bits per pixel. We also extract visual primitives such as optical flow [1] and disparity [86], using phase-based multi-scale models [75]. Each of these primitives requires 12 bits per component, meaning that the optical flow needs 24 bits (x and y velocity components) per pixel while disparity (depth estimation) needs 12 bits per pixel. As multi-scale models, each of them need also several scales to be computed per frame, so our system must store up to 8 different resolutions. Moreover, optical flow calculation requires storing a temporal window of 3 frames. After computing all this, the FPGA sends the results (optical flow, disparity and LCDs) to the co-processor. [85] explain in detail how we implement this system. Tables 5.2 and 5.3 show a summary of the memory and bandwidth requirements of our system.

	Memory Requirements (MB)				
	Input Pyramid	Optical flow	Disparity	Features	Whole System
VHA	8	6	3	23,98	88,92
HA	3,4	2,75	1,37	9,72	36,95
MA	2	1,25	0,625	5,31	19,8

Table 5.2: Memory requirements for different input resolutions: VHA (very high accuracy) (1024x1024), HA (high accuracy) (800x600) and MA (medium accuracy) (512x512). Input pyramid used in some primitive extraction engines (such as optical flow) includes several input images at different resolutions to build multi-scale models. Optical flow, disparity and features are results of our low-level-vision system, referred to here as the "whole system", and require 24, 12 and 8x3 bits per pixel respectively.

In Table 5.2 we show original memory requirements of our low-level-vision system introduced in Figure 1.5. If we compare those requirements with the memory needs of our condensed approach in Figure 5.5, we can see how we reduce it dramatically. This reduction allow the vision system to store more scales or more optical-flow frames, improving its ouputs [63, 75].

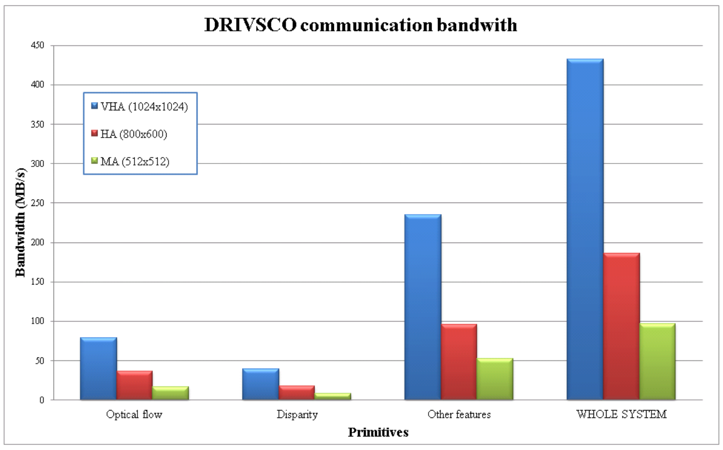
	Bandwidth Requirements (MB/s)			
	Optical flow	Disparity	Features	Whole System
VHA	78,6	39,3	235,1	432,3
HA	36	18	95,6	185,6
MA	16,4	8,2	52,2	96,4

Table 5.3: Bandwidth requirements for different input resolutions: VHA (very high accuracy) (1024x1024), HA (high accuracy) (800x600) and MA (medium accuracy) (512x512). The conditions are similar to Table 5.2.

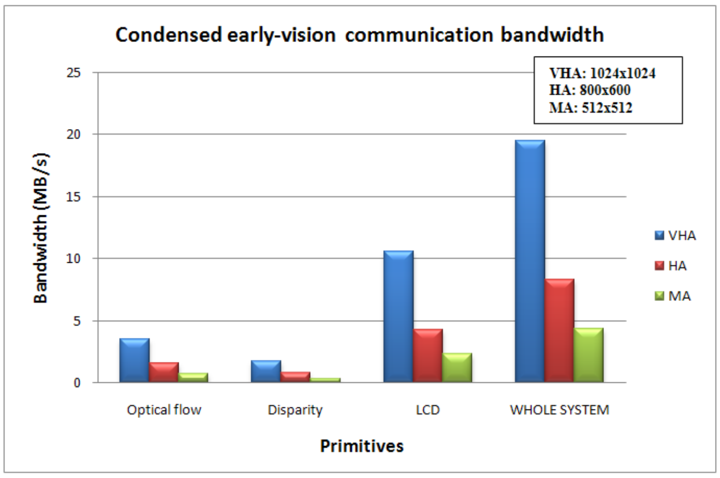
Bandwidth requirements introduced in Table 5.3, that would be unimportant in a standard processor, are critical in our system. Since we have an FPGA device with a 1-lane PCIe interface to connect with the PC, there is a bottleneck in this communication. To clarify this constrain, we include in Figure 5.7(a) the bandwidth requirements of DRIVSCO system (this graph is the same as Figure1.6). The total bandwidth requirement of the system for a standard SVGA resolution (i.e. HA resolution) is over 185 MB/s. Although multilane PCIe solutions would be used, the constraints of current multicore processors will prevent them from managing such a data flow [1].

Using our condensation core, we reduce the transfer load as shown in Figure 5.7(b). We can see that the bandwidth requirements become dramatically reduced using our condensation scheme in actual sequences. We have used a 5x5 grid, which means that 4% of the original image is sent as regular data, and around 2% sent as relevant points. It is interesting to compare Figure 5.7(a) and 5.7(b), whence the bandwidth needed for the whole system on a SVGA resolution (800x600) is reduced from over 185 MB/s to less than 20MB/s.

From these figures and tables we conclude that our condensation core reduces memory and transfer loads, allowing the vision system to perform, store and transfer data in real-time. Furthermore, due to the communication and workload reductions, our approach reduces power consumption. Although the DRIVSCO setup presented here assumes we are sending the condensed data to a commodity processor, the final on-board system could use an embedded soft-processor (if a more complex FPGA is used) or to a external DSP to post-process these data. Both options are valid as vehicular solutions and compatible with our hardware implementation of the condensation algorithm.



(a) Original bandwidth requirements



(b) Bandwidth requirements using semidense maps

Figure 5.7: Bandwidth comparison between original dense features and semidense ones for three different resolutions: VHA (very high accuracy) (1024x1024), HA (high accuracy) (800x600) and MA (medium accuracy) (512x512).

Finally, we need to assess whether the other capabilities of our semidense map (such as attention and feedback integration) are kept in this hardware implementation.

5.3.2 Low-Level Feedback

As we studied in section 4.1, when we drive in a city, the information from the pedestrians walking on the sidewalk is usually less relevant than the one from the road. However, if all of a sudden one of those pedestrians starts crossing the street, our priorities change. The pedestrian will become part of our interest area and our brain will process it with a higher priority [39].

In chapter 4 we applied our semidense representation map to different applications, helping the driver to react in a changing driving scenario (see section 1.1.1). Our goal here is to prove that this capacity is still present in our hardware implementation and we can integrate it in real-time. Our vision system, however, calculates too much information to process it in real time. For instance, we extract optical flow cues of not only other objects in the road, but also the rest of objects in the image: pedestrians on the pavement, trees, houses, etc. In fact, to compute mid- and high-level features such as TTC [15] and IMOs [62] (see section 4.1.2), background information is useless. Based on this idea, we have used our semidense representation map to integrate optical flow and disparity, using the latter to filter those areas too far to be processed by the mentioned algorithms.

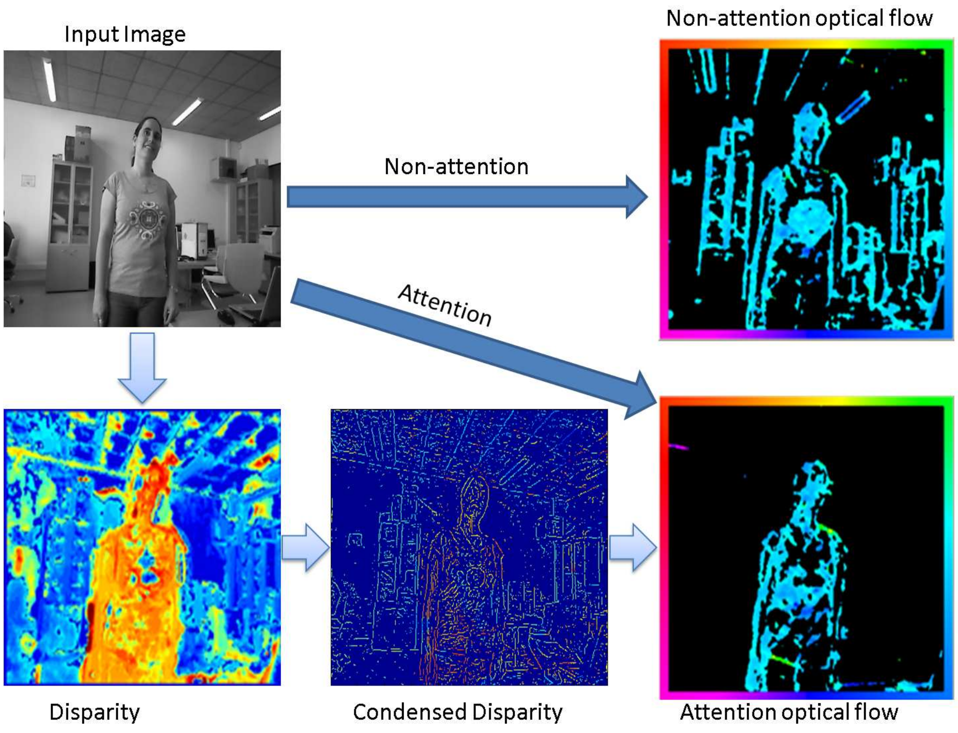


Figure 5.8: Attention feedback example. Real-time execution of our visions system using condensed disparity to inhibit background areas from the dense optic fields.

We use this simple example to illustrate how an attention process can lead our low-level-vision system, removing less-relevant areas and enhancing interesting ones. In Figure 5.8 we can see how a camera moves as if it were a robot checking the scene, while a static person is in our way, close enough to become an obstacle to avoid. In the non-attention optical flow (upper-right image) we detect background information, too far to be interesting for our TTC algorithm. Thus, after extracting disparity and optical flow using our system-on-chip (SoC) [84], we use our condensation core to condense the disparity. Then, we use this condensed disparity as attention signal and easily integrate it in the optical flow's *RP and Grid Extractor* submodule (see Figure 5.6). This way, as we condense the optical flow, we inhibit far areas, sending just the important optic fields.

Although this inhibition could be done in higher-level algorithms, doing it on chip avoids sending unnecessary information, and, therefore, improves even more our bandwidth reduction. In fact, it is not necessary to condense the optical flow since the reduction achieved by this inhibition is good enough. For instance, the condensed optical flow is around 8% of the original one; using the disparity feedback, however, we can discard useless areas (those that are further), sending to higher levels just 2.48% of the original one, i.e. 30% of the codensed flow. This way we reduce even more the bandwidth needed by our system whilst integrating low-level features on real-time. And, if we need to, we can always condense the attention optical flow. A smaller amount of optical-flow information implies a reduction in the computational workload and allows embedded processors to compute high-level algorithms (such as TTC) in real-time.

5.4 Conclusions

We have designed a hardware architecture to integrate our condensation algorithm in DRIVSCO low-level-vision system, condensing the obtained visual features in real-time. We have implemented our condensation core as a fine-grain pipeline architecture which is grouped in specific functional modules. All pipeline stages work in parallel, processing one pixel per clock cycle. This is why our low-level-vision system maintains its high performances after including the condensation core.

As indicated in Table 5.1, the hardware resource requirements indicate that the condensation system can be integrated in an image-processing scheme at affordable hardware cost. Working at 90 MHz (59MHz if we condense several features) and computing one pixel per clock cycle, this module can process 90 fps of 1024x1024 image resolution (or 59 fps if we condense several features). These results prove that our core does not introduce any delay in our low-level-vision system that works at 49 fps [85].The main advantages of this core are: real-time frequency, scalability and feedback from other stages.

We have used a hybrid 'regular and event-driven' protocol. Thus, we have developed a communication protocol that sends information with two different priorities in the framework of low-level to mid-level communication. This protocol manages to increase data transmission efficiency between the FPGA, which extracts the low-level-vision features, and the PC, which uses them for high-level processing tasks, without losing any important information (as shown in Figure 5.7). Therefore, the FPGA acts as a co-processor platform when the system needs to extract low-level-vision features. Moreover, our semidense map reduces the computational workload (as showed in the previous chapter), cutting down the execution time and facilitating the use of embedded processors, instead of external ones (such as a PC or a DSP).

This feedback is achieved due to the integration of low-, mid- and high-level signals in the condensation process. Furthermore, we have employed disparity information obtained by our system as attention signal, using it to test the feedback capabilities of our system. Moreover, our semidense map facilitates a feasible framework to implement higher-level algorithm in an embedded processor.

Chapter 6

Conclusions and Future work

This dissertation presents our contributions to the areas of computer vision and image representation. In this chapter we present a general discussion of the motivation problem and our solution. We also include some highlights of our future work. Then, we include the publications derived from our work and, finally, a summary of the main contributions achieved.

6.1 General Discussion

Local descriptors (such as SIFT, SURF or multi-modal ones) provide a selection of the most interesting points of the image, reducing the amount of data that mid-/high-level-vision algorithms need to process. However, most of these descriptors work with images, while higher-level algorithms usually employ visual features (such as disparity and optical flow) as inputs. Nevertheless, even those descriptors that extract information from visual features (such as multi-modal ones) ignore the plain regions. These regions, although less salient, also contain information that can be important depending on the algorithm (for instance, ground-plane extractor). From this we can conclude that computer vision is lacking a hybrid method that reduces dense visual feature to a sparser representation while keeping information from plain regions.

In this dissertation we present our solution to fulfill this need. We have designed, implemented and assessed an innovative hybrid representation map that integrates salient points, that we call relevant points, and plain regions, that we represent using a regular grid. We have called it *semidense representation map*, and the process to obtain it, *condensation*.

Relevant points (RPs) are obtained by an enhancing signal applied over the original image. In fact, we can use any sparse visual feature that selects salient points in the image. We have focused on structure-based selectors, such as Canny's edge detector. On the other hand, to extract information from the plain regions, we filter the original feature and select a regular grid of points. This grid is a very important contribution of this new representation since, as we have mentioned, other non-dense representations do not include the non-relevant information.

We have assessed different extractors for both relevant and non-relevant regions to explore the possibilities of our condensation algorithm. We conclude that our semidense map response depends on the image structure and, therefore, we can modify the extraction process so it adapts to its ulterior application. Thus, we achieve better results when working on known situations (very textured sequences, driving scenarios, ...).

Simplicity and versatility are two of the main advantages of this map: it reduces any dense visual feature to a map whose size is around 10% of the original one (using grid window of 5x5 pixels), with minimal error and workload.

Moreover, our representation map is a very well suited framework to easily integrate several sparse features. In fact, we have evaluated saliency maps as relevance indicator, incorporating our semidense representation map into an attention system. Furthermore, we have included top-down attention to our semidense map integrating IMO information in it.

This incorporation of top-down information is another important contribution of our semidense map. We have designed it so we can naturally receive and integrate feedback information from low-, mid- and high-level stages of a vision system, without any additional cost (in time or computational resources).

We have also assessed its performance when used as input in a ground-plane extraction algorithm based on disparity information. This algorithm uses non-relevant regions to determine the road of the sequence, taking advantage of the innovative part of our semidense map. The results are qualitative and quantitative similar as using a dense input, with remarkable memory and CPU workload reduction. This application shows the restrictions of sparse representations, useless when extracting the ground plane. Moreover, we have employed the extracted ground-plane as input in an obstacle detection algorithm with similar results. This application is a good example of how the improvements of our semidense map not only affect to the mid-level algorithms that use it as an input; but also influence the higher-level stages of the process, reducing bandwidth, memory and workload all over the system.

Our approach can be used in many other applications. For instance, we have assessed our algorithm as a bandwidth and memory reduction tool on a real-time application to communicate an FPGA with a PC using a PCI interface. The simplicity of our algorithm allows for its implementation in specific hardware and its use with real-time constraints. Hence, we have designed a hardware architecture to integrate our condensation algorithm in a actual low-level-vision system, within the framework of the European project DRIVSCO. We have implemented our condensation core as a fine-grain pipeline architecture which is grouped in specific functional modules. All pipeline stages work in parallel, processing one pixel per clock cycle. This is why our low-level-vision system maintains its high performances after including the condensation core. In fact, to condense one feature, our core works at 90 MHz and computes one pixel per clock cycle, processing 90 fps of 1024x1024 image resolution. The main advantages of this core are: real-time frequency, scalability and feedback from other stages.

Real-time applications produce simplified versions of visual features and, therefore, higher-level algorithms would benefit from a regularizing stage. Our semidense map inherently regularizes visual features when condensing them, due to the filter applied to the non-relevant regions. This way, we have developed an efficient way to regularize visual features in real-time, while freeing higher-level algorithms from doing it.

We have paid special attention to the evaluation of each of the aspects mentioned in this section. We have used benchmark and real-world sequences to compare dense maps with our semidense approach, provided as tables and graphs in this dissertation.

In summary, this condensation scheme can be understood as being a processing stage along the vision datapath that translates dense representation maps into a semidense representation that can easily be handled by standard and embedded processors for further computing by higher level modules. Therefore the main motivation of this condensation is to provide an appropriate representation format for mid- and high-level vision stages.

6.2 Future work

As future work, we consider two different aspects: include other feedback signals in our semidense representation map and apply it to new applications.

Although we have already assessed several features as relevance extractors, we would like to carry out a complete study of how to integrate several of them at the same time to improve the robustness of our solution. Moreover, we want to evaluate how to combine our semidense map with the multi-modal descriptors. We also intend to explore how to integrate signals such as TTC estimations as top-down attention signals. On the other hand, we will also explore mechanisms to dynamically adapt the grid to further optimize bandwidth requirements and regularization capabilities of our condensation algorithm, developing an adaptive grid. Furthermore, we would like to explore a bio-inspired version of this adaption, designing a fovea-like grid that uses a smaller window size for those regions that are more promising depending on the application.

As future applications, we will use our semidense representation map as input in several vision algorithms such as TTC in a driving scenario, tracking in a video surveillance application, and efficient communication and integration of multi-cameras.

6.3 Publications

The published (or submitted) works related to this research are the followings:

International Journals with Scientific Impact:

- F. Barranco, M. Tomasi, J. Díaz, S. Granados and E. Ros, *Hierarchical Architecture for Motion and Depth Estimations based on Color Cues.* [Journal of Real-Time Image Processing, IN PRESS, DOI: 10.1007/s11554-012-0294-1, 2012.
- S. Granados, F.Barranco, S. Mota, and J. Diaz. *On-chip semidense representation map for dense visual features driven by attention processes.* [SUBMITTED TO -under minor review-] Journal on Real-Time Image Processing. Special Issue on Real-Time Image Processing in Embedded Systems, 2012.
- S. Granados, N. Chumerin, S. Mota, and J. Diaz. *Obstacle detection using semidense representation maps.* [SUBMITTED TO -under major review-] Journal of Visual Communication and Image Representation, 2012.

International Conference:

- S. Granados, E. Ros, R. Rodríguez, and Javier Díaz. *Visual processing platform based on artificial retinas.* In Proceedings of the 9th international work conference on Artificial neural networks, IWANN 07, San Sebastian, Spain, pages 506-513, 2007.

National Conferences:

- S. Granados, S. Mota, E. Ros, and J. Díaz, *Condensación de primitivas visuales de bajo nivel para aplicaciones de procesamiento en tiempo real.* JCRA 2008, Madrid (Spain), pages 207-214, 2008, BEST CONFERENCE PAPER AWARD.

- F. Barranco, M. Tomasi, M. Vanegas, S. Granados, J. Díaz, *Entorno software para visualización y configuración de procesamiento de imágenes en tiempo real con plataformas reconfigurables.* JCRA 2009. Alcalá de Henares (Spain), pages 327-336, 2009.
- S. Granados, F. Barranco, J. Díaz, S. Mota and E. Ros, *Condensación de primitivas visuales de bajo nivel para aplicaciones atencionales.* Congreso Español de Informática (CEDI 2010), X JCRA. Valencia (Spain), pages 199-206, 2010.

6.4 Main contributions

We include here a sum-up of the main contributions achieved in this dissertation:

- We have designed a semidense representation map that condense dense visual features into a sparser representation with a minimal loose of information. This innovative map is composed of two different kinds of points: relevant ones and plain-region ones, providing information of salient and non-salient parts of the image. This hybrid approach fills a need in Computer Vision, where only salient parts were taken into account.
- We have assessed several sparse features as relevance indicators, from structure-based features (such as edge detectors) to salency maps. We have evaluate their performances to extract the most salient parts of the image, using benchmark and real-world sequences.
- We have evaluated several filtering operators when extracting plain-region information. Once filtered, we create a regular grid that selects one point within a window. Hence, we have assessed several filters and different grid-window sizes using benchmark and real-world sequences.
- We have proved the regularizing capabilities of our semidense representation map comparing it with existing regularization methods.
- We have employed semidense maps to condense multiple visual features. We have also used the obtained semidense maps as input in multiple real-world applications achieving remarkable improvements in bandwidth, memory and processing performance.
- We have designed and implemented a hardware architecture to integrate our condensation algorithm in a actual low-level-vision system that works on a FPGA. This architecture follows a fine-grain pipeline design, where all stages work in parallel, processing one pixel per

clock cycle. We have evaluated this architecture, reducing memory and bandwidth while maintains the system high performances.

- We have created a very well suited framework to integrate several sparse features, efficiently combining them. Moreover, this framework creates a signal-to-symbol loop since our semidense representation map receives feedback from low-/mid-/high-level algorithms and integrates them in an unique representation without additional cost. We have also evaluated this feedback integration for the two implementations of our algorithm: software and hardware.

Conclusiones y Trabajo Futuro

Esta tesis doctoral muestra nuestras aportaciones a las áreas de visión artificial y representación de imágenes. En este capítulo presentamos una discusión general de la motivación del problema y nuestra solución. También incluimos algunas sugerencias como trabajo futuro. A continuación enumeramos las publicaciones derivadas de nuestro trabajo y, por último, un resumen de las principales aportaciones.

Discusión General

Los descriptores locales (como SIFT, SURF o los multi-modales) proporcionan una selección de los puntos más interesantes de la imagen, reduciendo la cantidad de datos que los algoritmos de medio y alto nivel necesitan procesar. Sin embargo, la mayoría de estos descriptores trabaja con imágenes, mientras que los algoritmos de alto nivel suelen utilizar como entradas las características visuales (como la disparidad o el flujo óptico). No obstante, incluso aquellos descriptores que extraen información a partir de características visuales (como los descriptores multimodales) ignoran las zonas planas. Estas regiones, aunque menos salientes, también contienen información que puede ser importante para algunos algoritmos (como por ejemplo el de extracción del plano de la carretera). Podemos concluir que en visión artificial es necesario un método híbrido que reduzca las características visuales densas en una representación más dispersa que mantengan la información de las regiones planas.

En esta tesis doctoral presentamos nuestra solución para suplir esa necesidad. Hemos diseñado, implementado y validado un innovador mapa de representación híbrido que integra puntos salientes, que llamamos puntos relevantes, y zonas "planas", que representamos por medio de una rejilla regular. Lo hemos llamado *mapa de representación semidenso* y al proceso de obtención *condensación*.

Los puntos relevantes (RP, siglas del nombre en inglés) se obtienen por medio de una señal que resalta aquellas partes de la imagen más importantes. De hecho, podemos utilizar cualquier característica visual dispersa que seleccione puntos salientes de la imagen. Nos hemos centrado en aquellas señales que se basan en la estructura de la imagen, como por ejemplo el detector de bordes de Canny. Por otro lado, para extraer la información del as zonas planas, filtramos la característica original y seleccionamos una rejilla regular de puntos. Esta rejilla es una aportación muy importante de esta nueva representación ya que, como ya hemos mencionado, el resto de representaciones no densas no suelen incluir información de las zonas menos relevantes.

Hemos probado varios extractores para ambos tipos de puntos: relevantes y no relevantes con el fin de encontrar el que mejor resultado diera. Podemos concluir que la respuesta de nuestro mapa semidenso depende de la estructura de la imagen y, por lo tanto, podemos modificar el proceso de extracción de manera que se adapte a la aplicación concreta que va a recibir el mapa. De esta manera, conseguiremos mejores resultados cuando trabajemos con situaciones conocidas (secuencias con muchas texturas, escenarios de conducción,...).

Sencillez y versatilidad son dos de las ventajas más importantes de este mapa de representación: reduce cualquier característica visual densa a un mapa cuyo tamaño es alrededor de un 10% del tamaño original (usando una rejilla de ventana 5x5 píxeles) y todo ello con mínimo error y carga computacional.

Además, nuestro mapa de representación es un marco muy adecuado para integrar fácilmente varias características visuales dispersas. De hecho, hemos evaluado los mapas de saliencia como indicadores de relevancia, incorporando nuestro mapa semidenso a un sistema atencional. E incluso hemos incorporado señales atencionales *top-down* a nuestro mapa semidenso al integrar las señales de objetos en movimiento en él.

Esta incorporación de atención dependiente de la tarea a realizar (es decir, *top-down*) es otra de las principales aportaciones de nuestro mapa semidenso. Lo hemos diseñado de tal manera que recibe e integra las señales de realimentación de bajo, medio y alto nivel de manera natural y sin ningún coste adicional (en tiempo o en recursos computacionales).

Otra de las aplicaciones en las que lo hemos probado el rendimiento de nuestro mapa ha sido en el ámbito de un algoritmo que extrae el plano de la carretera utilizando información de disparidad para ello. Este algoritmo utiliza las regiones no relevantes de la imagen para calcular cuál es la carretera de una sección de conducción, de manera que se aprovecha de la parte más innovadora de nuestro mapa semidenso. Los resultandos obtenidos son cualitativa y cuantitativamente similares a los obtenidos utilizando un mapa denso, aunque la memoria y la carga computacional se ven muy reducidas. Esta aplicación nos demuestra las limitaciones de las características dispersas, incapaces de extraer el plano de la carretera. Además, hemos empleado el plano calculado como entrada a un algoritmo de detección de obstáculos con resultados similares a los anteriores. Esta segunda aplicación es un buen ejemplo de cómo las mejoras en ancho de banda, memoria y carga computacional aportadas por nuestro mapa semidenso se difunden a lo largo de todo el proceso, afectando tanto a los algoritmos de medio nivel como a los de más alto nivel que se basan en ellos.

Nuestra solución puede ser utilizada en otras aplicaciones. Por ejemplo, hemos evaluado nuestro algoritmo como herramienta para la reducción del ancho de banda y la memoria en una aplicación de tiempo real que necesita comunicar una FPGA con un PC a través de una interfaz PCI. La sencillez de nuestro algoritmo permite su implementación en hardware específico y su uso bajo condiciones de tiempo real. Por ello, hemos diseñado una arquitectura hardware que integra nuestro algoritmo de condensación en un sistema real de baja visión desarrollado en el marco del proyecto europeo DRIVSCO. Hemos implementado nuestro módulo de condensación siguiendo una arquitectura con segmentación fina del cauce de datos, lo que permite procesar un píxel por ciclo, coincidiendo así con el modo de funcionamiento del resto del sistema y facilitando su incorporación al mismo. Nuestro módulo de condensación para una única característica trabaja a una frecuencia de 90 MHz, lo que nos permite procesar 90 imágenes de resolución 1024x1024 píxeles por segundo. De esta manera mantenemos el alto rendimiento del sistema. Las principales ventajas de este módulo son: funcionamiento en tiempo-real, escalabilidad y realimentación de otras etapas del procesamiento.

Las aplicaciones de tiempo real suelen producir versiones simplificadas de las características visuales y, por lo tanto, los algoritmos de más alto nivel se verían beneficiadas si incluimos algún tipo de regularización antes de pasarle las características visuales. Nuestro mapa semidense regulariza inherentemente estas características visuales al condensarlas debido al filtro que se aplica a las zonas no relevantes. Por lo tanto hemos desarrollado un método eficiente para regularizar primitivas visuales en tiempo real, liberando a las aplicaciones de de más alto nivel de hacerlo.

Hemos prestado especial atención a la evaluación de cada uno de los aspectos mencionados en esta sección, comparando los resultados densos y semidensos obtenidos utilizando imágenes reales e imágenes de bancos de prueba (*benchmarks*) y proporcionando dichos resultados en tablas y gráficas a lo largo de esta tesis.

En resumen, el esquema de condensación puede ser entendido como una etapa más de un sistema de visión que traduce mapas de representación densos en una representación semidensa más fácil de utilizar por las etapas superiores tanto en procesadores estándar como empotrados. Por lo tanto, la principal motivación de esta condensación es proporcionar un formato de representación adecuado para las etapas bajas, medias y altas de dicho sistema de visión.

Trabajo Futuro

Como trabajo futuro, nos planteamos dos líneas diferentes: incluir otras señales como realimentación en nuestro sistema de representación y emplear los mapas semidensos en nuevas aplicaciones.

Aunque hasta el momento hemos probado diversas características como extractores de relevancia, nos gustaría realizar un estudio exhaustivo de cómo integrar varias de ellas a la vez para mejorar la robustez de nuestra solución. Además, queremos evaluar cómo combinar nuestro mapa semidenso con los descriptores multi-modales. También pretendemos explorar las mejoras que supondrían el incorporar señales más alto nivel (como TTC), utilizándolas como señales atencionales *top-down*. Por otro lado, también queremos estudiar mecanismos para adaptar dinámicamente la rejilla, es decir, obtener una rejilla adaptativa capaz de mejorar aún más los requisitos de ancho de banda y las capacidades de regularización de nuestro algoritmo de condensación. Dentro de las distintas opciones de generar una rejilla adaptativa, nos gustaría estudiar una versión bio-inspirada: diseñar una rejilla que se comporte como la fóvea y que utilice un tamaño de ventana menor para las zonas más interesantes de la imagen (según la aplicación).

Como futuras aplicaciones, queremos utilizar nuestro mapa de representación semidenso como entrada a diversos algoritmos de visión como el del cálculo del tiempo de choque en un escenario de conducción, el seguimiento de objetos en una aplicación de video-vigilancia o la integración de múltiples cámaras.

Publicaciones

Los trabajos relacionados con esta tesis doctoral publicados o en proceso de revisión son los siguientes:

Revistas Internacionales con Índice de Impacto:

- F. Barranco, M. Tomasi, J. Díaz, S. Granados and E. Ros, *Hierarchical Architecture for Motion and Depth Estimations based on Color Cues*. Journal of Real-Time Image Processing, 2012, IN PRESS, DOI: 10.1007/s11554-012-0294-1.

- S. Granados, F.Barranco, S. Mota, and J. Diaz. *On-chip semidense representation map for dense visual features driven by attention processes.* [SUBMITTED TO -under minor review-] Journal on Real-Time Image Processing. Special Issue on Real-Time Image Processing in Embedded Systems, 2012.

- S. Granados, N. Chumerin, S. Mota, and J. Diaz. *Obstacle detection using semidense representation maps.* [SUBMITTED TO -under major review-] Journal of Visual Communication and Image Representation, 2012.

Conferencias Internacionales:

- S. Granados, E. Ros, R. Rodríguez, and Javier Díaz. *Visual processing platform based on artificial retinas.* In Proceedings of the 9th international work conference on Artificial neural networks, IWANN 07, San Sebastian, Spain, pages 506-513, 2007.

Conferencias Nacionales:

- S. Granados, S. Mota, E. Ros, and J. Díaz, *Condensación de primitivas visuales de bajo nivel para aplicaciones de procesamiento en tiempo real.* JCRA 2008, Madrid (Spain), pages 207-214, 2008, BEST CONFERENCE PAPER AWARD.

- F. Barranco, M. Tomasi, M. Vanegas, S. Granados, J. Díaz, *Entorno software para visualización y configuración de procesamiento de imágenes en tiempo real con plataformas reconfigurables.* JCRA 2009. Alcalá de Henares (Spain), pages 327-336, 2009.

- S. Granados, F. Barranco, J. Díaz, S. Mota and E. Ros, *Condensación de primitivas visuales de bajo nivel para aplicaciones atencionales.* Congreso Español de Informática (CEDI 2010), X JCRA. Valencia (Spain), pages 199-206, 2010.

Aportaciones Principales

En esta sección incluimos un resumen de las principales aportaciones conseguidas en esta tesis doctoral:

- Hemos diseñado un mapa de representación semidenso que condensa características visuales densas en una representación más dispersa con una pérdida de información mínima. Este innovador mapa está formado por dos tipos de puntos distintos: relevantes y de zonas planas, proporcionando información de las partes salientes y no salientes de la imagen. Esta solución híbrida satisface una necesidad en el ámbito de la visión artificial, en el que sólo se tenían en cuenta las partes salientes.

- Hemos estudiado el comportamiento de varias características visuales dispersas como indicadores de relevancia, desde características basadas en la información estructural de la imagen (como los detectores de bordes) hasta mapas de saliencia. Hemos evaluado su rendimiento a la hora de extraer los puntos más relevantes de la imagen utilizando imágenes reales y *benchmarks*.

- Hemos estudiado distintos filtros como extractores de información de las zonas planas. Para ello, una vez filtrada la imagen, hemos creado una rejilla regular seleccionando un punto en cada ventana de dicha rejilla. Por lo tanto, hemos evaluado varios filtros y distintos tamaños de ventana de rejilla tanto en imágenes reales como en *benchmarks*.

- Hemos demostrado las capacidades regularizadoras de nuestro mapa de reprsentación semidenso, comparándolo con métodos de regularización existentes.

- Hemos utilizado mapas dispersos para condensar distintas características visuales. Además, hemos utilizado los mapas dispersos obtenidos como entradas en varias aplicaciones reales consiguiendo mejoras importantes en el ancho de banda, la memoria y la carga computacional.

- Hemos diseñado e implementado una arquitectura hardware que integra nuestro algoritmo de condensación en un sistema real de baja visión que funciona en una FPGA. Esta arquitectura sigue una segmentación fina del cauce de datos donde todas las etapas se ejecutan en paralelo, procesando un píxel por ciclo de reloj. Hemos evaluado esta arquitectura, reduciendo la memoria y el ancho de banda a la vez que mantiene el alto rendimiento del sistema.

- Hemos creado un marco muy adecuado para integrar varias características visuales dispersas, combinándolas eficientemente. Además, este marco crea un ciclo de realimentación que permite que nuestro mapa disperso incorpore información procedente de algoritmos de bajo, medio y alto nivel en una única representación sin coste adicional. Hemos evaluado esta realimentación en las dos implementaciones de nuestro algoritmo: software y hardware.

Appendix A

Receiver Operating Characteristic Curves

A Receiver Operating Characteristic (ROC) curve is a graphical representation of the trade off between the false negative and false positive rates for every possible cut off. Equivalently, the ROC curve is the representation of the trade-offs between sensitivity and specificity [94]. They describe how well a test discriminates between cases with and without a certain condition. In our case we want to decide if we are losing information, i.e. if we are marking as NaN points that contained a valid value in the original sequence and vice versa.

Sensitivity The proportion of true positives or the proportion of cases correctly identified by the test as meeting a certain condition (e.g. in our example, the proportion of valid points that remained in the semidense representation map).

Specificity The proportion of true negatives or the proportion of cases correctly identified by the test as not meeting a certain condition (e.g. in our example, the proportion of NaN that were marked as NaN in the semidense map).

This means that we need to transform our problem into a binary classification one, in which the outcomes are labeled either as positive (p) or negative (n). There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p.

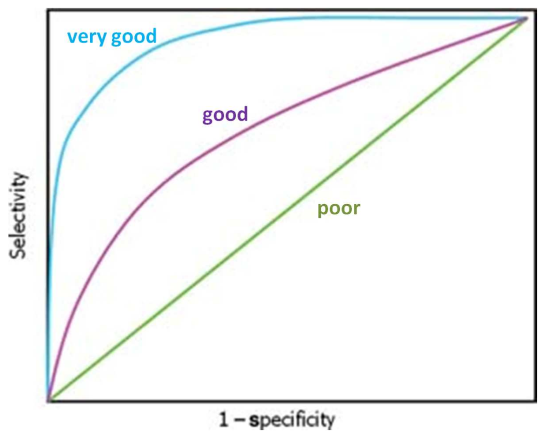


Figure A.1: ROC Curve typical values.

From that, we can calculate the sensitivity or true positive rate (TPR) as $TPR = TP/P = TP/(TP+FN)$ and specificity (SPC) $SPC = TN/N = TN/(FP+TN) = 1 - FPR$, where FPR represents false positive rate.

By tradition, the plot shows Sensitivity on the Y axis and 1-Specificity on the X axis. Figure A.1 shows how signals are classified depending on their plot.

ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making. They have been widely used in signal detection, medicine, and machine learning.

Appendix B

Decondensation

A complete information recovery is also important. For instance, some well known algorithms are not ready (or are not able) to work with sparse or semidense representation maps, therefore, we need a way to move from our representation to a dense one. For that reason, we have assessed different methods to interpolate a semidense map. We also include here some example of real-world sequences condensed and ´decondensed´ together with the error produced in the process. Figure B.1 shows the whole condensation process including the interpolation step.

B.1 Interpolation Methods

Three different methods to interpolate a semidense map have been assessed.

B.1.1 MATLAB-Function-Based Interpolation

Using MATLAB functions we have designed an interpolation method. The algorithm to follow is:

1. Using `meshgrid`, we generate two vectors containing all the pixel coordinates.
2. After reshaping them properly, we replace the NaN in the condensed image with an empty vector.
3. Function `griddata` is the appropriate to decondense our image, so we use it including the parameter "nearest" that will replicate the nearest value to interpolate it.

Figure B.2(b) shows original, condensed and interpolated maps obtained using this method. The decondensed map is less dense than the original

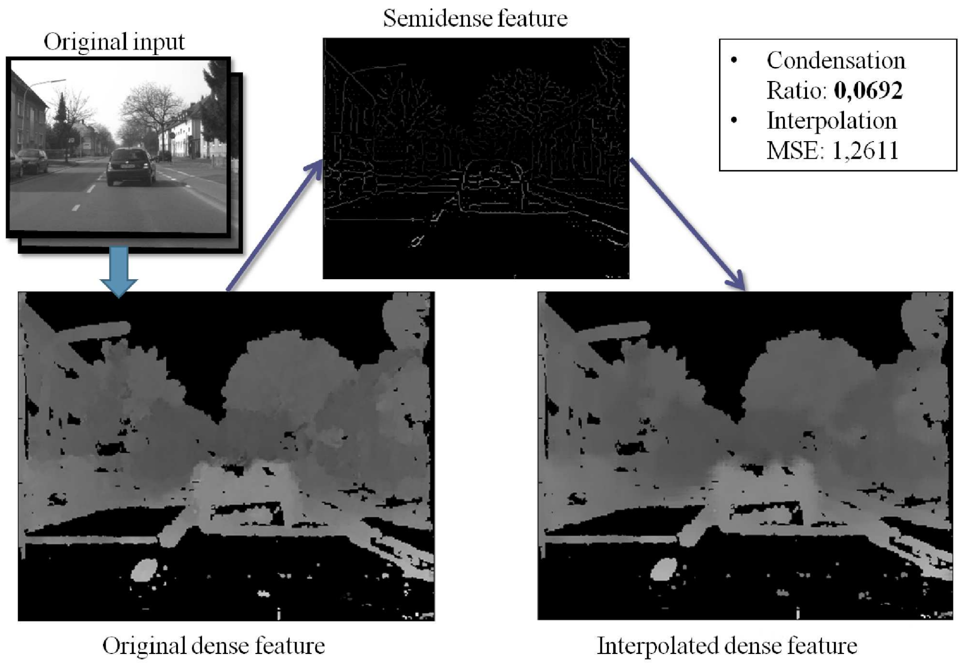

Original input
Semidense feature
• Condensation Ratio: **0,0692**
• Interpolation MSE: 1,2611
Original dense feature
Interpolated dense feature

Figure B.1: Condensation process including interpolation. Using the input images (top left), dense disparity (bottom-left) is extracted using [86]. After condensing it, we recover a dense map using simple calculations.

one. This means this solution loses information and therefore it can be improved.

B.1.2 Replicate Method

Although the MATLAB approach achieves a less dense solution than expected, it provides a good idea for improvement: replication. This second approach uses the replication idea as follows:

1. For each pixel in the image:
    (a) If it is not a local maximum:
        i. Choose the closer pixel (with distance < 5) containing a value (different of NaN).

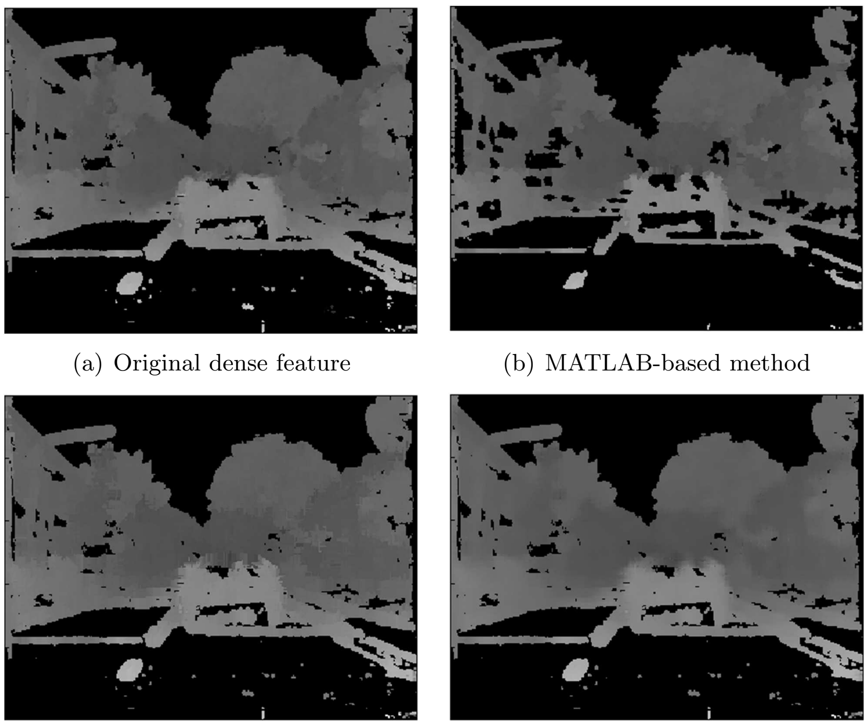
Even if a quite simple method, it achieves better qualitative results than the previous one as shown in figure B.2(c).

B.1.3 Linear Method

After testing the replicate method, we considered using linear distance between the points to ponder the values in order to improve the result. So, in this case, the algorithm is:

1. For each pixel in the image:
   (a) If it is not a local maximum:
      i. For each pixel close to it (distance ¡ 5) with a value (different of NaN):
         A. Multiply distance inverse and disparity value of that pixel.
      ii. Add all the calculated values and associate to the current point.

This algorithm is not as simple as the replicate one, but it provides an improved decondensed output as shown in figure B.2(d).

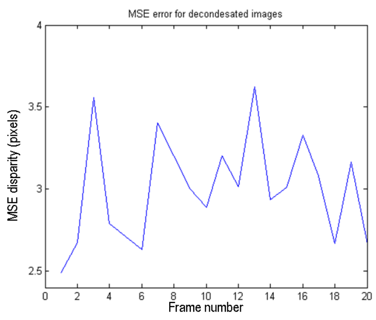

(a) Original dense feature (b) MATLAB-based method

(c) Replication method (d) Linear method

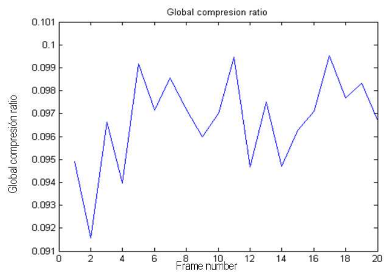
Figure B.2: Comparison between different interpolation methods to recover a dense map from a semidense one. The semidense map can be found in Figure B.1.

B.2 Interpolation Validation

Although our qualitative results are quite representative, we have measured the mean squared error (MSE) produced by this interpolation process. Note that we are using real-world sequences in which the ground truth does not exist. This means that the error produced could very likely correspond to a regularization (see section 3.3.2. Figure B.3(a) shows the MSE error between original and interpolated disparity for a 20-frame sequence. On the other hand, Figure B.3(b) corresponds to the condensation ratio achieved by our algorithm, i.e. $condensed_size/original_size$. In fact, Fig. B.2 corresponds to one of the frames of this sequence.



(a) Mean Squared Error

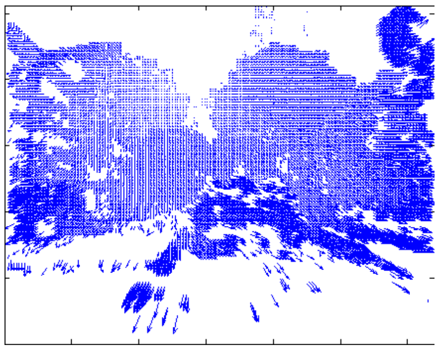


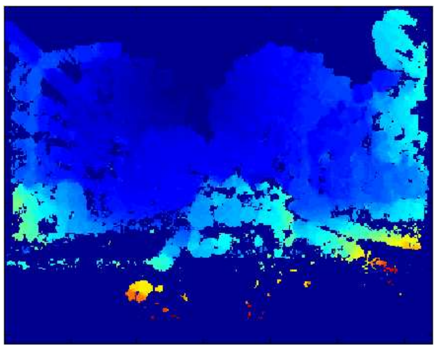
(b) Condensation Ratio

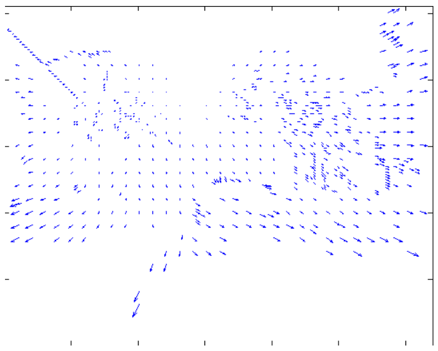
Figure B.3: (a) MSE between the original disparity and the interpolated one using linear approach. (b) Final condensation ratio for each frame of the sequence.

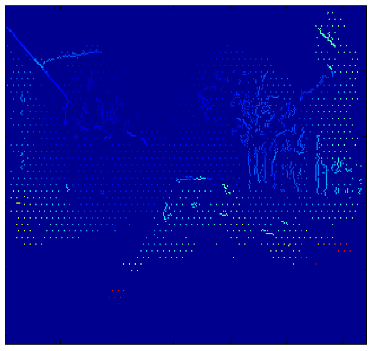
As we have already establish, ours is a general purpose condensation scheme so, in order to prove it, we have use it to condense and decondense optical flows as well. Figure B.4 shows an example of optical flow condensation and decondensation for the driving sequence using [85].

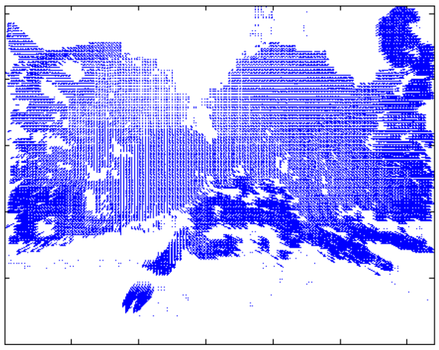
In Figure B.5 we obtain the MSE of each velocity component of the optical flow as well as the condensation ratio of the sequence. Not only the error produced is not high, but also it could mean regularization instead of actual error.


(a) Original Optical Flow


(b) Original Optical Flow Module


(c) Condensed Optical Flow


(d) Condensed Optical Flow Module


(e) Interpolated Optical Flow

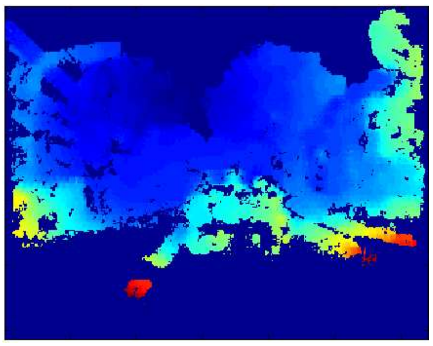
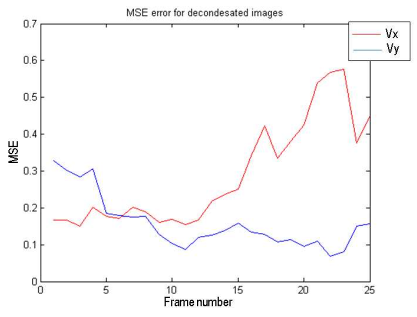
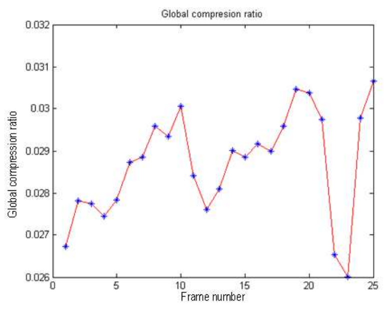

(f) Interpolated Optical Flow Module

Figure B.4: Comparison between different interpolation methods to recover a dense map from a semidense one. The semidense map can be found in Figure B.1.



(a) Mean Squared Error



(b) Condensation Ratio

Figure B.5: (a) MSE of each component between the original optical flow and the interpolated one using linear approach. (b) Final condensation ratio for each frame of the sequence.

Bibliography

[1] M. Anguita, J. Diaz, E. Ros, and F.J. Fernandez-Baldomero. Optimization strategies for high-performance computing of optical-flow in general-purpose processors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(10):1475 –1488, oct. 2009.

[2] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M.J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *Int. J. Comput. Vision*, 92(1):1–31, March 2011.

[3] D. Barash. Bilateral filtering and anisotropic difusion: Towards a unified viewpoint. In Michael Kerckhove, editor, *Scale-Space and Morphology in Computer Vision*, volume 2106 of *Lecture Notes in Computer Science*, pages 273–280. Springer Berlin / Heidelberg, 2006.

[4] F. Barranco, J. Díaz, E. Ros, and B. Del Pino. Visual system based on artificial retina for motion detection. *Trans. Sys. Man Cyber. Part B*, 39:752–762, June 2009.

[5] F. Barranco, M. Tomasi, J. Díaz, S. Granados, and E. Ros. Hierarchical architecture for motion and depth estimations based on color cues. *Journal on Real-Time Image Processing*, 2012.

[6] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros. Parallel architecture for hierarchical optical flow estimation based on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1 –10, 2011.

[7] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros. Fpga-based low-cost architecture for the extraction of visual primitives on real time. *submitted to Digital Signal Processing (in peer review)*, 2012.

[8] E. Baseski, L. Baunegaard With Jensen, N. Pugeault, F. Pilz, K. Pauwels, M. M. Van Hulle, F. Wörgötter, and N. Krüger. Road interpretation for driver assistance based on an early cognitive vision system. In *VISAPP (1)*, pages 496–505, 2009.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[10] A. Benoit, A. Caplier, B. Durette, and J. Herault. Using human visual system modeling for bio-inspired low level image processing. *Computer Vision and Image Understanding*, 114(7):758 – 773, 2010.

[11] D.C.M. Bilsby, R.L. Walke, and R.W.M. Smith. Comparison of a programmable dsp and a fpga for real-time multiscale convolution. *IEE Seminar Digests*, 1998(197):4–4, 1998.

[12] K. Boahen. A retinomorphic chip with parallel pathways: Encoding on, off, increasing, and decreasing visual signals. *Journal of Analog Integrated Circuits and Signal Processing*, 30(2):121–135, 2002.

[13] K. Boahen. A burst-mode word-serial address-event link-i: Transmitter design. *IEEE Trans. Circuits Systems I, Reg. Papers*, pages 1269–1280, 2004.

[14] K.A. Boahen. A burst-mode word-serial address-event link-ii: receiver design. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 51(7):1281 – 1291, july 2004.

[15] T. Camus. Calculating time-to-contact using real-time quantized optical flow. In *National Institute of Standards and Technology NISTIR 5609*, 1995.

[16] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698, nov. 1986.

[17] M. Chessa, F. Solari, S. P. Sabatini, and G. M. Bisio. Motion interpretation using adjustable linear models. In *BMVC*, 2008.

[18] N. Chumerin. mylabel: Matlab graphical tool. http://sites.google.com/site/chumerin/projects/mylabel, 2008.

[19] N. Chumerin. *From Multi-Channel Vision Towards Active Exploration.* PhD thesis, K.U.Leuven, 2011.

[20] Open Cores: JPEG Hardware core. http://opencores.org/project,jpeg. online, 2012.

[21] E. Culurciello, R. Etienne-Cummings, and K.A. Boahen. A biomorphic digital image sensor. *Solid-State Circuits, IEEE Journal of*, 38(2):281 – 294, feb 2003.

[22] E.R. Davies. *Machine Vision: Theory, Algorithms and Practicalities, Third Edition.* Morgan Kaufmann Publishers, 2004.

[23] R. Detry, N. Pugeault, and J.H. Piater. A probabilistic framework for 3d visual object representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1790 –1803, oct. 2009.

[24] J. Diaz, E. Ros, S. Mota, and R. Carrillo. Image processing architecture for local features computation. In *Reconfigurable Computing: Architectures, Tools and Applications*, volume 4419 of *Lecture Notes in Computer Science*, pages 259–270. Springer Berlin / Heidelberg, 2007.

[25] J. Díaz Alonso. *Multimodal bio-inspired vision system. High performance motion and stereo processing architecture.* PhD thesis, University of Granada, 2006.

[26] D.L. Donoho. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on*, 41(3):613 –627, may 1995.

[27] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics Automation Magazine, IEEE*, 13(2):99 –110, june 2006.

[28] A. Ess, B. Leibe, and L. Van Gool. Depth and appearance for mobile scene analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[29] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[30] D. Gabor. Theory of communication. *Electrical Engineers - Part I: General, Journal of the Institution of*, 94(73), january 1947.

[31] J. Gai and R.L. Stevenson. Optical flow estimation with p-harmonic regularization. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1969 –1972, 2010.

[32] S. Granados, N. Chumerin, S. Mota, and J. Diaz. Obstacle detection using semidense representation maps. *[SUBMITTED TO] Journal of Visual Communication and Image Representation*, 2012.

[33] S. Granados, F.Barranco, S. Mota, and J. Diaz. On-chip semidense representation map for dense visual features driven by attention processes. *[SUBMITTED TO] Journal on Real-Time Image Processing. Special Issue on Real-Time Image Processing in Embedded Systems*, 2012.

[34] S. Granados, E. Ros, R. Rodríguez, and Javier Díaz. Visual processing platform based on artificial retinas. In *Proceedings of the 9th international work conference on Artificial neural networks*, IWANN'07, pages 506–513, Berlin, Heidelberg, 2007. Springer-Verlag.

[35] Mentor Graphics. Handel-c synthesis methodology. http://www.mentor.com/products/fpga/handel-c/, 2011.

[36] P.W. Holland and R.E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977.

[37] B.K.P. Horn and B.G. Schunck. Determining optical flow. Technical report, Cambridge, MA, USA, 1980.

[38] L. Itti, N. Dhavale, and F. Pighin. Realistic avatar eye and head animation using a neurobiological model of visual attention. In B. Bosacchi, D. B. Fogel, and J. C. Bezdek, editors, *Proc. SPIE 48th Annual International Symposium on Optical Science and Technology*, volume 5200, pages 64–78, Bellingham, WA, Aug 2003. SPIE Press.

[39] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001.

[40] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10-12):1489–1506, 2000.

[41] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, 1995.

[42] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[43] M. Jourlin and J.C. Pinoli. A model for logarithmic image processing. *Journal of Microscopy*, 149(1):21–35, 1988.

[44] H. Kolb and L.E. Lipetz. The anatomical basis for colour vision in the vertebrate retina. *Vision and Visual Dysfunction. The Perception of colour*, 6:128–145, 1991.

[45] N. Kruger and M. Felsberg. A continuous formulation of intrinsic dimension. In *Proceedings of the British Machine Vision Conference*, 2003.

[46] N. Krüger, M. Lappe, and F. Wörgötter. Biologically motivated multimodal processing of visual primitives. *The Interdisciplinary Journal OF Artificial Intelligence and the Simulation of Behaviour*, 1(4):2004, 2003.

[47] N. Logothetis. *Vision: A Window into Consciousness*, volume 16. Scientific American, 2006.

[48] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[49] M.R. Luettgen, W. Clem Karl, and A.S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *Image Processing, IEEE Transactions on*, 3(1):41 –64, jan 1994.

[50] N.R. Luque, J.A. Garrido, R.R. Carrillo, O.J.M.D. Coenen, and E. Ros. Cerebellarlike corrective model inference engine for manipulation tasks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 41(5):1299–1312, 2011.

[51] I. Markelic, A. Kjaer-Nielsen, K. Pauwels, L.B.W. Jensen, N. Chumerin, A. Vidugiriene, M. Tamosiunaite, A. Rotter, M. Van Hulle, N. Kruger, and F. Worgotter. The driving school system: Learning basic driving skills from a teacher in a real car. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1135 –1146, dec. 2011.

[52] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[53] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1792 –1799 Vol. 2, oct. 2005.

[54] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615 –1630, oct. 2005.

[55] C.A. Morillas, S.F. Romero, A. Martínez, F.J. Pelayo, E. Ros, and E. Fernández. A design framework to model retinas. *Biosystems*, 87(2-3):156–163, 2007.

[56] Sonia Mota Fernández. *Circuitos bio-inspirados para la evaluación de movimiento en tiempo real y sus aplicaciones*. PhD thesis, University of Granada, 2007.

[57] M. Nagao and T. Matsuyama. Edge preserving smoothing. *Computer Graphics and Image Processing*, 9(4):394 – 407, 1979.

[58] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:2006, 2006.

[59] NVIDIA. Webpage about cuda computing platform http://www.nvidia.com/object/cuda_home_new.html. online, 2012.

[60] R. Oktem. Regularization-based error concealment in jpeg 2000 coding scheme. *Signal Processing Letters, IEEE*, 14(12):956 –959, dec. 2007.

[61] E.M. Ortigosa, A. Cañas, E. Ros, P. Martínez-Ortigosa, S. Mota, and J. Díaz. Hardware description of multi-layer perceptrons with different abstraction levels. *Microprocessors and Microsystems*, 30(7):435–444, 2006.

[62] K. Pauwels, N. Kruger, M. Lappe, F. Worgotter, and M.M. Van Hulle. A cortical architecture on parallel hardware for motion processing in real time. *Journal of Vision*, 10(10), 2010.

[63] K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros, and M. Van Hulle. A comparison of fpga and gpu for real-time phase-based optical flow, stereo, and local image features. *Computers, IEEE Transactions on*, PP(99):1, 2011.

[64] F.J. Pelayo, A. Martinez, S. Romero, C.A. Morillas, E. Ros, and E. Fernandez. Cortical visual neuro-prosthesis for the blind: retina-like software/hardware preprocessor. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 150 – 153, march 2003.

[65] F.J. Pelayo, S. Romero, C.A. Morillas, A. Martínez, E. Ros, and E. Fernández. Translating image sequences into spike patterns for cortical neuro-stimulation. *Neurocomputing*, 58-60(0):885 – 892, 2004. Computational Neuroscience: Trends in Research 2004.

[66] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629 –639, jul 1990.

[67] R. J. Peters, A. Iyer, C. Koch, and L. Itti. Components of bottom-up gaze allocation in natural scenes. In *Proc. Vision Science Society Annual Meeting (VSS05)*, May 2005.

[68] M. Popovic, D. Kraft, L. Bodenhagen, E. Baseski, N. Pugeault, D. Kragic, T. Asfour, and N. Krüger. A strategy for grasping unknown objects based on co-planarity and colour information. *Robotics and Autonomous Systems*, 58(5):551 – 565, 2010.

[69] N. Pugeault and N. Krüger. Temporal accumulation of oriented visual features. *J. Visual Communication and Image Representation*, 22(2):153–163, 2011.

[70] N. Pugeault, F. Wörgötter, and N. Krüger. Visual primitives: Local, condensed, semantically rich visual descriptors and their applications in robotics. *I. J. Humanoid Robotics*, 7(3):379–405, 2010.

[71] Nicolas Pugeault, Karl Pauwels, Marc M. Van Hulle, Florian Pilz, and Norbert Krüger. A three-level architecture for model-free detection and tracking of independently moving objects. In *VISAPP (1)*, pages 237–244, 2010.

[72] J. Ralli, J. Díaz, S. Kalkan, N. Krüger, and E. Ros. Disparity disambiguation by fusion of signal- and symbolic-level information. *Machine Vision and Applications*, 23:65–77, 2012.

[73] J. Ralli, J. Díaz, and E. Ros. A method for sparse disparity densification using voting mask propagation. *Journal of Visual Communication and Image Representation*, 21(1):67 – 74, 2010.

[74] P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):pp. 871–880, 1984.

[75] S.P. Sabatini, G. Gastaldi, F. Solari, K. Pauwels, M.M. Van Hulle, J. Diaz, E. Ros, N. Pugeault, and N. Krüger. A compact harmonic code for early vision based on anisotropic frequency channels. *Comput. Vis. Image Underst.*, 114:681–699, June 2010.

[76] T. Sanger. Stereo disparity computation using gabor filters. *Biological Cybernetics*, 59:405–418, 1988. 10.1007/BF00336114.

[77] Pedro Santana, Magno Guedes, Luís Correia, and José Barata. Saliency-based obstacle detection and ground-plane estimation for off-road vehicles. In *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ICVS '09, pages 275–284, Berlin, Heidelberg, 2009. Springer-Verlag.

[78] K. Sayood. *Introduction to data compression.* Morgan Kaufmann Publishers Inc., 2000.

[79] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, April 2002.

[80] M.A. Shubin. *Encyclopedia of Mathematics*, chapter Laplace operator. Springer, 2001.

[81] Seven Solutions. http://www.sevensols.com. online, 2012.

[82] L Stryer. Visual excitation and recovery. *Journal of Biological Chemistry*, 266(17):10711–10714, 1991.

[83] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839 –846, jan 1998.

[84] M. Tomasi. *Pyramidal architecture for stereo vision and motion estimation in real-time FPGA-based devices.* PhD thesis, University of Granada, 2010.

[85] M. Tomasi, M. Vanegas, F. Barranco, J. Diaz, and E. Ros. Massive parallel-hardware architecture for multi-scale stereo, optical flow and image-structure computation. *Circuits and Systems for Video Technology, IEEE Transactions on*, PP(99):1, 2011.

[86] M. Tomasi, M. Vanegas, F. Barranco, J. Diaz, and E. Ros. Real-time architecture for a robust multi-scale stereo engine on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1 –12, 2011.

[87] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall PTR, 1998.

[88] Mauricio Vanegas, Matteo Tomasi, Javier Díaz, and Eduardo Ros Vidal. Multi-port abstraction layer for fpga intensive memory exploitation applications. *Journal of Systems Architecture - Embedded Systems Design*, 56(9):442–451, 2010.

[89] Dirk Walther, Ueli Rutishauser, Christof Koch, and Pietro Perona. Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. *Computer Vision and Image Understanding*, 100(1-2):41–63, 2005.

[90] Drivsco Webpage. DRIVSCO European Project. http://www.pspc.dibe.unige.it/∼drivsco/.

[91] Ben Weiss. Fast median and bilateral filtering. *ACM Trans. Graph.*, 25:519–526, July 2006.

[92] E.J. Wharton, K. Panetta, and S.S. Agaian. Logarithmic edge detection with applications. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 3346 –3351, oct. 2007.

[93] Xilinx. http://www.xilinx.com. online, 2012.

[94] M. H. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39(4):561–577, 1993.