Universidad de Granada

Department of Computer Architecture and Technology

Modelos de visión para tareas de videovigilancia en sistemas empotrados

Vision models for video surveillance tasks on embedded systems

PhD Thesis Dissertation

Enrique Jaime Fernández Sánchez

Granada, May 2013

Editor: Editorial de la Universidad de Granada
Autor: Enrique Jaime Fernández Sánchez
D.L.: GR 211-2014
ISBN: 978-84-9028-730-9

ugr Universidad de Granada

Vision models for video surveillance tasks on embedded systems

Modelos de visión para tareas de videovigilancia en sistemas empotrados

Presented By

Enrique Jaime Fernández Sánchez

To apply for the

International PhD Degree in Computer and Network Engineering

May 2013

ADVISORS

Eduardo Ros Vidal

Javier Díaz Alonso

D. Eduardo Ros Vidal y D. Javier Díaz Alonso, Catedrático y Titular de Universidad respectivamente del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN

Que la memoria titulada“Modelos de visión para tareas de videovigilancia en sistemas empotrados” ha sido realizada por D. Enrique Jaime Fernández Sánchez, 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 Mayo de 2013

Fdo. Eduardo Ros Vidal, Javier Díaz Alonso

Directores de la Tesis

Editor: Editorial de la Universidad de Granada
Autor: Enrique Jaime Fernández Sánchez
D.L: En trámite
ISBN:En trámite

El doctorando Enrique Jaime Fernández Sánchez y los directores de la tesis Eduardo Ros Vidal y Javier Díaz Alonso garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, a 13 de Mayo de 2013

Directores de la Tesis	Doctorando
Fdo. Eduardo Ros Vidal, Javier Díaz Alonso	Enrique Fernández Sánchez

Agradecimientos

Contents

Contents **xii**

List of Figures **xv**

List of Tables **xvii**

List of Abbreviations **xix**

Abstract **1**

Resumen **3**

1 Introducción **5**

1.1 Motivación 5

1.2 Objetivos científicos 9

1.3 Marco de proyectos 10

1.3.1 MULTIVISION 10

1.3.2 TOMSY 11

1.4 Material y métodos 12

1.5 Organización de los capítulos 12

2 Introduction **15**

2.1 Motivation 15

2.2 Scientific objectives 18

2.3 Projects 19

2.3.1 MULTIVISION 19

2.3.2 TOMSY 20

2.4 Methods and Tools 21
2.5 Organization of chapters 22

3 State of the art 23
3.1 Background subtraction 24
3.1.1 Background subtraction on embedded hardware 28
3.1.2 Background subtraction with sensor fusion 28
3.2 Object tracking 30
3.2.1 Feature selection 30
3.2.2 Tracking 30
3.2.3 Multi-camera approaches 31
3.2.3.1 Distributed and decentralized camera networks 31

4 Background subtraction on embedded hardware 33
4.1 Horprasert algorithm 34
4.1.1 Background Modeling 35
4.1.2 Subtraction Operation and Classification 36
4.1.3 Model simplifications 38
4.1.4 Hardware Architecture 39
4.2 Codebook algorithm 42
4.2.1 Background modeling 42
4.2.1.1 Maximum negative run-length 44
4.2.1.2 Color and brightness 44
4.2.2 Foreground detection 45
4.2.3 Background model update 45
4.2.4 Model simplifications 46
4.2.5 Hardware Architecture 49
4.3 Datasets and evaluation metrics 50
4.4 Results 54
4.4.1 Performance comparison with other approaches 54
4.4.2 Evaluation of the accuracy of the background model . 56
4.4.2.1 Background subtraction evaluation 57
4.4.2.2 Shadow detection behavior 60

4.5 Conclusions 62

5 Background subtraction based on color and depth cues 65

5.1 Depth estimation 66

5.1.1 Kinect sensor 68

5.1.2 Phase-based method 69

5.1.3 Variational approach 70

5.1.4 Semi-global block matching 71

5.2 Fusion model 72

5.2.1 4D Codebook: *CB4D* 72

5.2.2 Early Fusion: Depth-extended Codebook (*DECB*) . . 73

5.2.3 Late Fusion: *DECB-LF* 74

5.3 Datasets 77

5.3.1 Kinect-based dataset 77

5.3.2 Disparity-based dataset 78

5.4 Results 79

5.4.1 Evaluation of algorithms using Kinect 80

5.4.2 Evaluation of algorithms using disparity maps 88

5.5 Conclusions 97

6 Multi-camera tracking on embedded devices 101

6.1 Related work 102

6.1.1 Selection of the object of interest 103

6.1.2 Multi-camera tracking 104

6.1.3 Smart camera networks 104

6.2 Smart camera platform 106

6.2.1 Hardware architecture 106

6.2.2 Software architecture 107

6.3 Single-camera tracking: CAMSHIFT 109

6.4 Tracking initialization by an attention module 111

6.4.1 Automatic object segmentation 115

6.5 Multi-camera model 118

6.5.1 Projective transformations: homography 118

6.5.2 Communication network 120
6.5.3 Information exchange 120
6.5.4 Camera calibration 122
6.5.5 Multi-camera tracking 123
6.6 Experimental results 124
6.7 Conclusions 130

7 Conclusions 133
7.1 General discussion 133
7.2 Future work 136
7.3 Publications 136
7.3.1 International journals with scientific impact 136
7.3.2 National conference 137
7.4 Main contributions 137

8 Conclusiones 139
8.1 Discusión general 139
8.2 Trabajo futuro 142
8.3 Publicaciones 143
8.3.1 Revistas Internacionales con Índice de Impacto 143
8.3.2 Conferencias nacionales 143
8.4 Aportaciones principales 143

Bibliography 161

List of Figures

1.1 Estructura general de un sistema de videovigilancia automatizado. 7

2.1 General scheme of a video analytics system. 17

3.1 Example of scene with presence of foreground objects during the training stage 24

3.2 Example of scene with sudden illumination changes. 25

3.3 Example of scene with movements of objects that belong to the background. 25

3.4 Example of scene with presence of repetitive movements in background elements. 26

4.1 Graphic representation of the model used by Horprasert et al. to classify the pixels 37

4.2 Scheme of the complete architecture and connections between modules, peripherals, memory and processor. 41

4.3 Simplified datapath architecture for Horprasert background subtraction and blob detection core 42

4.4 Simplified datapath architecture for Codebook background subtraction and blob detection core 50

4.5 Evaluation frames of the Wallflower dataset 52

4.6 Statistical parameters for evaluation 53

4.7 Overall performance evaluated using F_1 and *Similarity* . . . 58

4.8 Performance in outdoor and indoor circumstances. 59

4.9 Wallflower evaluation frames, ground truth and resultant images from tested algorithms. 60

4.10 Shadow detection accuracy and discrimination accuracy of the original software model and the proposed approach, evaluated on *Intelligent Room* sequence. 61

4.11 Frames of the *"Intelligent Room"* sequence, frame 100 for **(a)** software and **(b)** hardware implementations, and frame 282 for **(c)** software and **(d)** hardware. 62

5.1 Disparity example 67

5.2 Components of Kinect sensor. 68

5.3 Example of complicated scenario for RGB-D methods: presence of shadows and flat foreground objects. 74

5.4 Scenario which offers different kinds of complications for disparity computation algorithms. 75

5.5 Complete scheme of masks fusion process. 76

5.6 Samples of frames from the proposed dataset based on Kinect. 78

5.7 Samples of frames from the proposed dataset based on disparity. 80

5.8 F_1 gain over standard (color-based) *CB* obtained from the test *ChairBox*. 81

5.9 Results obtained from the test *ChairBox*. 82

5.10 F_1 gain over *CB* obtained from the test *Wall* 83

5.11 Results obtained from the test *Wall*. Codebook-based approaches do not perform morphological opening. 84

5.12 F_1 gain over *CB* obtained from the test *Shelves* 85

5.13 Results obtained from the test *Shelves*. Codebook-based approaches do not perform a morphological opening stage. . . . 86

5.14 F_1 gain over *CB* obtained from the test *Hallway* 86

5.15 Results obtained from the test *Hallway*. Codebook-based approaches do not perform morphological opening. 87

5.16 Average F_1 and gain over *CB* obtained from the entire Kinect benchmark. 88

5.17 F_1 gain over *CB* obtained from the test *Suitcase*. 90

5.18 Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *Suitcase* sequence. 91

5.19 F_1 gain over *CB* obtained from the test *Crossing*. 92

5.20 Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *Crossing* sequence. 93

5.21 F_1 gain over *CB* obtained from the test *LCDScreen*. 94

5.22 Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *LCD-Screen* sequence. 95

5.23 F_1 gain over *CB* obtained from the test *LabDoor*. 96

5.24 Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *LabDoor* sequence. 97

5.25 Average F_1 and gain over *CB* obtained from the entire stereo benchmark. 98

6.1 Scheme of Android Architecture. 108

6.2 Architecture of the MultiCam framework. 110

6.3 Block diagram of CAMSHIFT tracker. 111

6.4 Example of use of CAMSHIFT 112

6.5 Attention module block diagram. 113

6.6 Example of usage of the Attention Module. 116

6.7 Example of usage of the skin detector included in our Attention Module. 116

6.8 Adaptation stage applied to the skin detector included in our Attention Module. 118

6.9 Scheme of homography 119

6.10 Scheme of multi-camera tracking via homography 124

6.11 Simplified example of tracking using view-to-view homography information. 125

6.12 Flexibility of attentional tracker. 127

6.13 Object leaving the field of view and entering it again. 128

6.14 Behavior of the tracking in the presence of occlusions. 129

6.15 Increase in robustness of single-camera tracker against lost tracks. 130

List of Tables

4.1 Bit-width of each variable taking part in the calculation of colordist and brightness in the Horprasert algorithm. 39

4.2 Bit-width of each variable involved in the Codebook algorithm. 48

4.3 Comparison with other previous approaches described in the literature. 56

4.4 Shadow detection and discrimination accuracy, tested on *"Intelligent Rooom"* sequence 61

5.1 Segmentation evaluation for sequence *ChairBox*. 81

5.2 Segmentation evaluation for sequence *Wall*. 83

5.3 Segmentation evaluation for sequence *Shelves*. 85

5.4 Segmentation evaluation for sequence *Hallway*. 85

5.5 Segmentation evaluation for sequence *Suitcase*. 89

5.6 Segmentation evaluation for sequence *Crossing*. 90

5.7 Segmentation evaluation for sequence *LCDScreen*. 92

5.8 Segmentation evaluation for sequence *LabDoor*. 94

6.1 Summary of related work on smart camera networks. 105

List of Abbreviations

API	Application Programming Interface
BG	Background
CAMSHIFT	Continuously Adaptive Mean Shift
DDR	Double Data Rate
DECB	Depth-Extended Codebook
DECB-LF	Depth-Extended Codebook with Late Fusion
DSP	Digital Signal Processor
FG	Foreground
FN	False Negatives
FP	False Positives
FPGA	Field Programmable Gate Array
fps	Frames per Second
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HSV	Hue, Saturation, Value
IR	Infrared
ISE	Integrated Software Environment
J2ME	Java 2 Micro Edition
JNI	Java Native Interface
KLT	Kanade-Lucas-Tomasi feature tracker

MCH	Memory Controller Hub
MNRL	Maximum Negative Run-Length
MOG	Mixture Of Gaussians
MPMC	Multi-Port Memory Controller
MPPS	MegaPixels per Second
NDK	Native Development Kit
NPI	Native Port Interface
PC	Personal Computer
PF	Particle Filtering
PLB	Processor Local Bus
PMHT	Probabilistic MultiHypothesis Tracking
RLE	Run-Length Encoding
RTL	Register Transfer Level
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SoC	System-on-Chip
STD	Standard Deviation
SURF	Speeded Up Robust Features
SVM	Support Vector Machines
TCP	Transmission Control Protocol
TN	True Negatives
ToF	Time-of-Flight
TP	True Positives
TSL	Tint, Saturation and Luminance
UDP	User Datagram Protocol
ZBT	Zero Bus Turnaround

Abstract

This dissertation presents our work with computer vision models applied to video surveillance tasks. This work is focused on key stages of a video analytics system, such as video segmentation and object tracking, and is specially centered on embedded devices.

This dissertation is structured in four parts.

In the first part, we review the state of the art, paying special attention to background subtraction and object tracking. After a general review of the state of the art, we analyze in more detail the works existing in the literature related to embedded hardware, sensor fusion, and multi-camera object tracking.

In the second part of this dissertation we focus on background subtraction algorithms on embedded hardware. We describe the two algorithms we have implemented on FPGA. We assess the two proposed architectures, performing a comparison with previous alternatives in the literature. We evaluate the performance concerning real-time constraints, hardware resources and energy consumption, as well as quality of the segmentation.

The third part consists of our work on sensor fusion applied to background subtraction. We study the integration of depth information in a background subtraction model, provided by multi-camera stereo vision systems and active depth sensors. Different fusion methods are studied and evaluated by means of new datasets. We assess the proposed approaches, achieving considerable improvement over previous algorithms.

The fourth part corresponds to multi-camera object tracking by using a smartphone network. We describe a prototype of an architecture to perform object tracking with two cameras, exchanging information in real time between devices. The information shared across the network is used to get basic calibration of the scene and to solve occlusions and recover from lost tracks, by gathering information from every device in a global manner. We also describe an attention model included in the architecture to detect and select objects of interest.

Resumen

Esta tesis doctoral presenta el trabajo realizado con modelos de visión artificial, aplicados a tareas de videovigilancia. Este trabajo se ha centrado en etapas claves para un sistema de videovigilancia activo, tales como la segmentación de vídeo y el seguimiento de objetos, estando especialmente orientado a dispositivos empotrados.

Esta tesis doctoral está dividida en cuatro partes.

En la primera parte revisamos el estado del arte, poniendo especial atención a la sustracción de fondo y el seguimiento de objetos. Tras una revisión general del estado del arte, analizamos de forma más detallada los trabajos existentes en la literatura en relación a hardware empotrado, fusión de información proveniente de distintos sensores, y seguimiento de objetos con múltiples cámaras.

En la segunda parte de esta tesis doctoral nos centramos en los algoritmos de sustracción de fondo en hardware empotrado. Se describen los dos algoritmos que hemos implementado en FPGA. Además, evaluamos las dos arquitecturas propuestas, llevando a cabo una comparación con otras alternativas presentes en la literatura. Realizamos una evaluación de prestaciones teniendo en cuenta restricciones de tiempo real, recursos hardware, consumo de energía, y la calidad de la segmentación.

La tercera parte contiene nuestro trabajo sobre fusión de información aplicada a sustracción de fondo. Estudiamos la integración de estimaciones de profundidad en un modelo de sustracción de fondo. Dicha información es proporcionada por sistemas de visión estéreo y sensores de profundidad alternativos. Se estudian diferentes métodos de fusión de información de profundidad y se evalúan por medio de nuevas secuencias de prueba. La evaluación de los métodos propuestos muestra una considerable mejora con respecto a algoritmos anteriores.

La cuarta y última parte se corresponde con el seguimiento de objetos con múltiples cámaras, utilizando para ello una red de *smartphones*. Describimos un prototipo de arquitectura que lleva a cabo seguimiento de objetos con dos cámaras, intercambiando información en tiempo real entre dispositivos. La información compartida a través de la red es utilizada para hacer una calibración básica de la escena, resolver oclusiones y recuperarse de pérdidas momentáneas en el seguimiento, reuniendo información de forma global desde cada dispositivo. Además describimos un modelo atencional incluido en la arquitectura, que lleva a cabo la detección y selección inicial de objetos de interés.

Chapter 1

Introducción

1.1 Motivación

Desde hace más de una década, ha habido un creciente interés por la aplicación de métodos de visión artificial para videovigilancia. Los sistemas de videovigilancia tradicional requieren un trabajo intensivo por parte de los vigilantes humanos, y a menudo se limitan a mostrar de forma conjunta una serie de imágenes provenientes de varias cámaras, con los problemas que eso plantea para el supervisor de dichas imágenes. La gran cantidad de información que producen estos sistemas, unida a la falta de sistemas automáticos de análisis y seguimiento, provoca que sólo tengan aplicación para labores forenses como prueba despúes de que un evento haya tenido lugar.

Por el contrario, la utilización de sistemas de videovigilancia automatizados permite facilitar la labor del personal humano, al advertir al mismo de eventos que pudieran ser de interés, así como permitir acciones preventivas cuando se detecta una situación anómala.

Los sistemas de videovigilancia analizan el contenido de los vídeos recibidos de cada cámara realizando la segmentación de la imagen en primer plano y fondo (*foreground* y *background* respectivamente), detectando objetos y haciendo un seguimiento (o *tracking*) de los mismos, y finalmente llevando a cabo diversos análisis de alto nivel. Dichos análisis desembocan en resultados que nos indican si el escenario es normal o anómalo, de modo que el operador humano pueda centrar su atención en los escenarios anómalos sin tener que mirar a un conjunto de pantallas tratando de encontrar algo de interés manualmente.

Es importante destacar que, si bien actualmente existen sistemas comerciales desarrollados para esta automatización de los sistemas de videovigilancia, tanto su escalabilidad como su robustez son aún aspectos críticos que requieren una mejora significativa y que motivan el desarrollo de este proyecto de tesis. La robustez es el requisito principal de los sistemas automáticos, y la causa principal de que aún no se haya extendido su implantación. Si un sistema genera falsos negativos, puede estar dejando pasar situaciones de alarma sin indicárselo al vigilante, lo que sería un fallo importante de los sistemas de videovigilancia. Por otro lado, si el sistema tiende a generar alarmas con frecuencia en situaciones que no tienen importancia, esto provocará una pérdida de atención por parte del vigilante. Por ello, el interés radica en mejorar el comportamiento de las técnicas de visión, ya sea utilizando algoritmos más sofisticados o utilizando diversas etapas de procesamiento que proporcionen resultados más fiables.

Una estructura general de un sistema de videovigilancia se muestra en la figura 1.1 [1], donde podemos ver las diferentes etapas de procesamiento de la información:

- Módulo de detección FG/BG: este módulo lleva a cabo la clasificación de cada pixel de la imagen en *foreground* o *background.*

- Módulo de detección de entrada de objetos: esta etapa utiliza la máscara resultante del módulo anterior para detectar cuando un nuevo objeto entra en la escena.

- Módulo de tracking: este módulo es inicializado por el *módulo de detección de entrada de objetos* (u otro método de detección de objetos), y realiza el seguimiento de una lista de objetos de interés en la secuencia.

- Módulo de generación de trayectorias: recolecta posiciones de objetos de interés y almacena la trayectoria de cada uno cuando el movimiento del objeto no está presente, como en el caso de oclusiones.

- Módulo de análisis de trayectorias: análisis de trayectorias y detección de trayectorias anómalas. Esta etapa está profundamente relacionada con la gestión de alarmas, ya que a partir de los datos proporcionados por las etapas previas de procesamiento tiene que decidir si la situación cumple las condiciones para la generación de una alerta o no.

La capacidad de extraer objetos en movimiento de una secuencia de vídeo es un problema crucial de muchos sistemas de visión, tales como videovigilancia [2, 3, 4], monitorización de tráfico [5], detección y tracking de personas para teleconferencias e interfaces avanzadas [6, 7, 8], entre otras aplicaciones que involucran visión artificial. La segmentación de vídeo, o lo que es lo mismo, separar el *foreground* del *background*, consiste en analizar un flujo de vídeo para detectar qué regiones de cada imagen pertenecen a objetos en movimiento y qué regiones son parte del fondo.

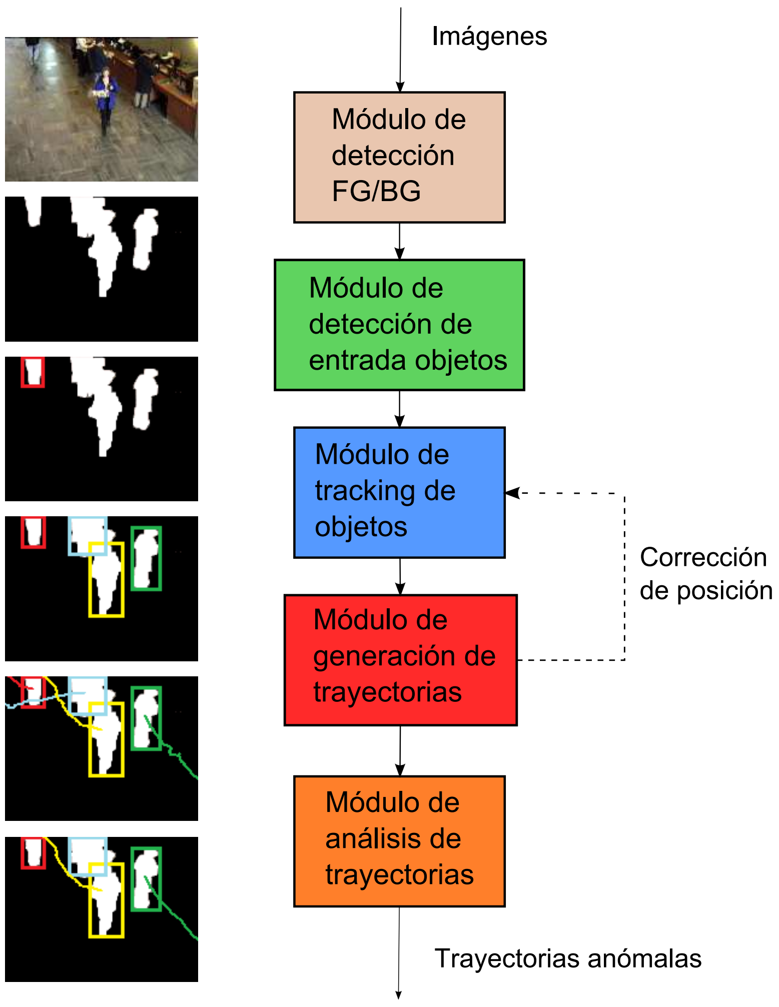

Imágenes
Módulo de detección FG/BG
Módulo de detección de entrada objetos
Módulo de tracking de objetos
Corrección de posición
Módulo de generación de trayectorias
Módulo de análisis de trayectorias
Trayectorias anómalas

Figure 1.1: Estructura general de un sistema de videovigilancia automatizado.

Si bien la segmentación de la escena es una tarea clave para llevar a cabo el procesamiento de vídeo desde una única cámara, cuando el sistema consta de múltiples cámaras se hace aún más necesaria. La presencia de múltiples cámaras y, por tanto, múltiples flujos de vídeo, provoca que el sistema de videovigilancia tenga que procesar una gran cantidad de información. Dado el creciente número de sistemas de videovigilancia, y el elevado número de cámaras que tienen que gestionar, la escalabilidad de estos sistemas es un problema clave. Para limitar la información que un sistema de visión procesa se recurre a distintos tipos de técnicas que centran la atención en regiones específicas de la escena.

El sistema visual humano es capaz de recibir y procesar las imágenes con increíble eficiencia a pesar de que el sustrato biológico (neuronas basadas en procesos electroquímicos) en el que se basa es considerablemente más lento que los sistemas electrónicos actuales [9]. El proceso de interpretación de la escena a partir de pares o secuencias de imágenes es un proceso jerárquico en el que la información fluye desde primitivas básicas (como información de color, contraste, movimiento, profundidad, etc) hasta niveles más altos de abstracción en los que se realiza la interpretación de la escena para la extracción de datos de más alto nivel [10, 11]. Nuestra intención es imitar este funcionamiento del cerebro, en el que las primeras etapas llevan a cabo tareas regulares que pueden ser implementadas en dispositivos empotrados. Posteriores etapas se basarán en la información obtenida sobre las primitivas básicas para llevar a cabo análisis más elaborados. En estos niveles se requieren descripciones más algorítmicas de los métodos utilizados y son difícilmente desarrollables en hardware de propósito específico [12].

Las mejoras en tecnología y técnicas de videoanálisis han permitido que el campo de investigación se centre en sistemas multicámara. Este tipo de sistemas tiene muchas ventajas, tales como la resolución de oclusiones en escenarios con cámaras con vistas solapadas o el seguimiento de objetos entre cámaras no solapadas [13, 14, 15]. No obstante, el creciente número de cámaras asociadas a estos sistemas conlleva ciertas complicaciones en relación con la configuración de la red de cámaras, así como la capacidad de procesamiento requerida para procesar múltiples flujos de vídeo. Debido a estas dificultades, la investigación en redes descentralizadas, en las que parte del procesamiento se lleva a cabo en cada nodo, ha suscitado gran interés [16]. Este tipo de solución permite reducir la cantidad de procesamiento a realizar en los nodos centrales o servidores. Además, la red requiere menor ancho de banda dado que no se transmite toda la información. Para habilitar el procesamiento en redes descentralizadas y distribuidas, se ha trabajado mucho en sistemas empotrados y *smart cameras* (cámaras con capacidades de procesamiento y comunicaciones).

Tradicionalmente, la investigación en *smart cameras* se ha orientado a arquitecturas de propósito específico tales como FPGAs y DSPs. Sin embargo, debido a las restricciones de recursos de estas arquitecturas, los algoritmos de visión artificial suelen requerir simplificaciones como el uso de aritmética en punto fijo, reducción de memoria y de complejidad de operaciones. Dichas simplificaciones conllevan una cierta pérdida de precisión. Además, los desarrollos hechos para una arquitectura específica raramente son compatibles con otras, siendo por ello dependientes de un cierto dispositivo o fabricante. En este contexto, resulta de interés el uso de nuevas plataformas, tales como los *smartphones*, como *smart cameras* para redes distribuidas. Los *smartphones* ofrecen varias ventajas, como por ejemplo disponer de cámaras razonablemente buenas, procesadores potentes para ser dispositivos móviles, diferentes tipos de conectividad y sensores. Además, al tratarse de dispositivos comerciales, tienen precios contenidos en comparación con otras arquitecturas hardware, así como bibliotecas y herramientas de desarrollo.

La aplicación de técnicas de detección y seguimiento de personas, así como modelos de inferencia de comportamiento, es un campo complejo y más cuando se utiliza una plataforma multicámara. El trabajo realizado en esta tesis comprende desde la segmentación de la imagen hasta la integración eficiente de estimaciones de distintas cámaras para realizar estas aplicaciones de forma robusta y global (en el marco del espacio cubierto por las múltiples cámaras).

1.2 Objetivos científicos

Los objetivos científicos establecidos para el desarrollo de esta tesis están relacionados con el análisis de secuencias de imágenes reales procedentes de sistemas de videovigilancia. Para ser más específicos, las metas que se establecieron son las siguientes:

- Desarrollo e implementación de modelos de extracción de fondo en tiempo real. Evaluación de métodos buscando compromiso entre calidad y eficiencia computacional.

- Integración de estimaciones de profundidad (obtenidas con visión estereoscópica multicámara y mediante sensores de profundidad alternativos) en modelos de extracción de fondo. Evaluación de calidad.

- Implementación de modelos de homografías para calibración del escenario objetivo de monitorización. Ampliación de modelos a plataformas multicámara.

- Utilización de modelos de atención en un sistema multivisión. Compromiso entre carga de computación y extracción de información global.

- Aplicaciones. Estudio de diversas aplicaciones en el campo de la videovigilancia con diferentes configuraciones de cámaras. Fusión de imágenes, seguimiento de objetos/personas, visión global sin espacios ocluidos utilizando múltiples cámaras, adaptación del sistema de visión a distintos escenarios, etc.

1.3 Marco de proyectos

El trabajo realizado en esta tesis se ha hecho en relación con un Proyecto de Excelencia de la Junta de Andalucía denominado MULTIVISION, y un proyecto europeo de investigación denominado TOMSY.

1.3.1 MULTIVISION

Sistema de visión en tiempo real multi-cámara para interpretación de escenas (MULTIVISION) (TIC-3873).

En este proyecto se abordará el desarrollo de un sistema de visión híbrido hardware/software en tiempo real basado en múltiples cámaras. En todo sistema de visión el objetivo es traducir imágenes a información concreta (datos extraídos de la "interpretación de la escena"). En este proyecto se estudiarán esquemas de visión que permitan el procesamiento eficiente de las imágenes extraídas de múltiples cámaras y el tratamiento de forma complementaria de las estimaciones de las diversas cámaras para realizar la tarea de "interpretación de la escena" de forma fiable y robusta. Este sistema tiene aplicación directa en plataformas de vigilancia y monitorización de espacios.

Los objetivos del proyecto se basan en resultados previos del grupo de investigación (y colaboraciones internacionales en el marco de proyectos Europeos y Nacionales).

El término de multivisión se refiere a la utilización de varias cámaras en diferentes configuraciones que permitirán estudiar conceptos distintos. La utilización de múltiples cámaras centradas en el mismo escenario (con solapamiento de campos de visión o sin solapamiento) permite la exploración de esquemas que permitan seguir objetos de un campo de visión a otro, estudiar trazas de movimientos entre distintos campos (cubiertos por distintas cámaras), etc. Además el proyecto estudiará también la configuración de varias cámaras con campo de visión solapados para cubrir el mismo escenario. En este caso estudiaremos esquemas de visión que permitan utilizar de forma complementaria la información extraída de cada una de las cámaras para realizar de forma fiable la "interpretación de la escena". Finalmente estudiaremos la configuración de varias cámaras con solapamiento parcial para estudiar esquemas de "campo visual global" compuesto por los campos visuales de cada una de las cámaras (esquemas de fusión de imágenes en tiempo real para la composición de un mosaico de imágenes captadas con diferentes cámaras).

Un escenario cubierto por múltiples cámaras genera una cantidad de datos difícil de procesar en tiempo real de forma centralizada. Por ello estudiaremos el diseño de esquemas de atención que permitan que sólo se procesen todas las imágenes a bajo nivel (movimiento, color, etc) de forma distribuida (en diversos dispositivos FPGA como núcleos de procesamiento), mientras que tareas de más alto nivel se realicen sólo de forma global (es decir, desde uno de los puntos de vista) y de forma centralizada. Para ello estudiaremos esquemas en los que cada una de las secuencias (tomada por cada una de las cámaras) se procesa de forma completa y lo compararemos con esquemas en los que un "modelo atencional" selecciona una cámara y extrae la información de más alto nivel de esta fuente. Compararemos los resultados de interpretación de la escena a los que se llega con estas dos configuraciones diferentes (análisis exhaustivo frente análisis selectivo). El objetivo último de esta línea de investigación (que se extenderá más allá de este proyecto) es diseñar "agentes virtuales" que puedan seleccionar una fuente de datos u otra en base a primitivas de bajo nivel y llegar a una interpretación parcial de la escena correcta.

El proyecto además incluye el estudio de esquemas de "aprendizaje". Debido a que la estructura y configuración de cámaras de monitorización de un espacio serán muy diversas en cada escenario, en el proyecto estudiaremos esquemas de aprendizaje que permitan que el sistema correlacione eventos visuales con acciones concretas de forma autónoma.

1.3.2 TOMSY

TOMSY (ingl. Topology Based Motion Synthesis for Dexterous Manipulation), proyecto europeo IST-FP7-Collaborative Project-270436. Descripción del proyecto. El objetivo de TOMSY es permitir un salto generacional en las técnicas y la escalabilidad de los algoritmos de síntesis de movimiento. Nos proponemos hacer esto por el aprendizaje y la explotación de la representación topológica adecuada y ponerlos a prueba en los dominios de manipulación de varios objetos, control de robots y la animación por ordenador. Los algoritmos tradicionales de planifcación de movimiento han tenido difcultades para enfrentar la dimensionalidad del espacio de estados y de la acción y la generalización de las soluciones en tales dominios. Esta propuesta se basa en las nociones geométricas de métricos topológicos y en métodos basados en datos para descubrir mapeos (asignaciones) multi-escala que capturan invarianzas más relevantes - una mezcla entre representaciones de espacios simbólicos, discretos y continuos.

Por primera vez, TOMSY aspira a lograr esto al darse cuenta de la flexibilidad en los tres niveles de percepción, representación y la generación de acciones mediante el desarrollo de nuevas representaciones de objeto-acción para percepción basado en 'manipulación variedad' (ingl. manipulation manifold) y el diseño de manipuladores metamórficos. Los métodos y hardware desarrollados se pondrán a prueba en retos del mundo real de manipulación que van desde mundos de bloque hasta doblar cartón u origami e interacciones de cuerpos humanoides con objetos flexibles.

Los resultados de este proyecto nos proporcionarían algunas respuestas a la pregunta de cual es la “correcta” representación en un control sensoriomotor y también proporcionaría una base para una futura generación de sistemas de visión robótica capaces de síntesis de movimiento en tiempo real que resultara en interactuación fluida con su entorno.

1.4 Material y métodos

Los algoritmos presentados en esta tesis doctoral han sido implementados, inicialmente, en el entorno de programación MATLAB. Hemos elegido MATLAB para la fase de prototipado porque permite una implementación rápida de los algoritmos y el posterior análisis de resultados. Además, MATLAB proporciona gran cantidad de funciones para visualización de imágenes y otros resultados, lo que facilita el desarrollo de distintas soluciones para los problemas planteados. No obstante, los modelos implementados en MATLAB adolecen de una considerable lentitud de ejecución, motivo por el que se ha procedido a implementar los algoritmos en C/C++ con soporte de la biblioteca OpenCV [17]. La implementación de estos modelos en C/C++ requiere un menor tiempo de ejecución, algo importante en algoritmos de procesamiento de vídeo, debido a la ingente cantidad de información que se tiene que procesar.

Con respecto a la implementación de los algoritmos para su síntesis e integración en hardware empotrado, se requiere el uso de lenguajes de descripción hardware. Se ha elegido una combinación de ImpulseC y VHDL, apoyados en los entornos EDK (Embedded Development Kit) e ISE Foundation de Xilinx Inc. La información de sensores de profundidad se ha obtenido utilizando la biblioteca OPENNI [18]. La implementación de algoritmos en dispositivos comerciales tales como smartphones Android ha sido realizada mediante la integración de Java y C++, utilizando JNI para la comunicación entre las aplicaciones Android y el código nativo.

1.5 Organización de los capítulos

La estructura de esta tesis doctoral continúa con una introducción al estado del arte de las técnicas utilizadas en videovigilancia. En el capítulo 3 se hace una revisión de las aportaciones existentes en la literatura al procesamiento de vídeo para tareas de videovigilancia en general y las técnicas estudiadas en el desarrollo de esta tesis. En el capítulo 4 estudiamos en más detalle las técnicas de sustracción de fondo orientadas a su implementación en hardware reconfigurable, los *datasets* (secuencias con información sobre el resultado “ideal”) y métricas disponibles para evaluarlas de forma exhaustiva, y analizaremos los resultados obtenidos por varias arquitecturas en FPGA. En el capítulo 5 nos centraremos en mejorar la robustez de los modelos de sustracción de fondo mediante la fusión de color y profundidad. Este capítulo incluye nuestra propuesta de incorporación de la profundidad en un algoritmo conocido, distintos métodos de obtención de la información de profundidad, así como resultados utilizando *datasets* grabados mediante cámaras estéreo y el sensor Kinect. En el capítulo 6 estudiamos la utilización de un sistema multicámara con *smartphones* para la detección y seguimiento de objetos de interés en una escena. La información obtenida por cada uno de los dispositivos se comparte con los otros para poder actuar en presencia de oclusiones y movimientos bruscos. Finalmente, el capítulo 7 resume las principales aportaciones de nuestra investigación y sugiere varias líneas de trabajo futuro.

Chapter 2

Introduction

2.1 Motivation

In recent years there has been an increasing interest for the application of computer vision methods to video surveillance tasks. Traditional video surveillance systems require intensive work by surveillance guards, and they usually show only a stream of images provided by several cameras, what causes different issues to the supervisors of those images. The big amount of information produced by these systems, as well as the shortage of automatic systems for analysis and tracking, usually limits their usage to forensics as evidence after an abnormal event has taken place.

On the contrary, the use of automatic video surveillance systems eases the task of human observers, since these systems inform them about events that might be of interest. In addition, they allow for preemptive actions when an abnormal situation is detected.

Video analytics systems analyze input video streams from each camera by performing image segmentation in foreground and background, detecting and tracking objects, and finally performing high-level analyses. These analyses lead to results which indicate whether the scenario is normal or not, so that the human observer can focus on abnormal scenarios.

It is important to remark that, although there currently are commercial video analytics systems, both scalability and robustness still are critical features which require significant improvement and motivate this thesis project. Robustness is the key requirement of automatic systems and the main reason why they are not more established. On the one hand, if a system gives false negatives, it may be overlooking alarm situations without warning about them. This would be a very important error in a video surveillance system. On the other hand, a system which often generates alarms in scenarios without importance will lead to lost of attention from human observers. For that reason, the interest lies in improving the performance of computer vision methods, either using more advanced algorithms or combining different processing stages in order to obtain more reliable results.

Figure 2.1 shows a general scheme of a video analytics system [1], in which the different processing stages can be seen:

- FG/BG detection module: this module performs foreground/background classification of each image pixel.
- Object entering detection module: this stage uses the resultant foreground mask from previous module to detect when a new object enters the scene.
- Tracking module: this module is initialized by the *object entering detection module* (or other object detection methods), and tracks a list of objects of interest in a video sequence.
- Trajectory generation module: it collects all objects positions and saves each trajectory when the motion of the object is no longer presented, for example in presence of occlusions.
- Trajectory analysis module: this module performs an object trajectory analysis and detection of abnormal trajectories. This stage is closely related with alarms management, since the module has to decide whether an alert must be generated or not, according to data provided by previous processing stages.

Extracting objects in movement in video sequences is a key issue for many vision-based tasks, such as video surveillance [2, 3, 4], traffic monitoring [5], person detection and tracking for teleconferencing and advanced interfaces [6, 7, 8], between other applications which involve computer vision.

Video segmentation consists of analyzing a video stream to detect which regions of each frame belong to objects in movement and which regions belong to the background.

Although the segmentation of the scene is a key task for mono-camera video surveillance, it is even more necessary when the system has multiple cameras. Working with multiple cameras, and thus with multiple video streams, leads to big amounts of information to be processed by the video surveillance system. Due to the increasing number of video analytics systems, which have to manage an also increasing number of cameras, scalability is an important requirement for these systems. In order to limit the information that a vision-based system processes, different kinds of techniques are used to focus the attention on specific regions of the scene.

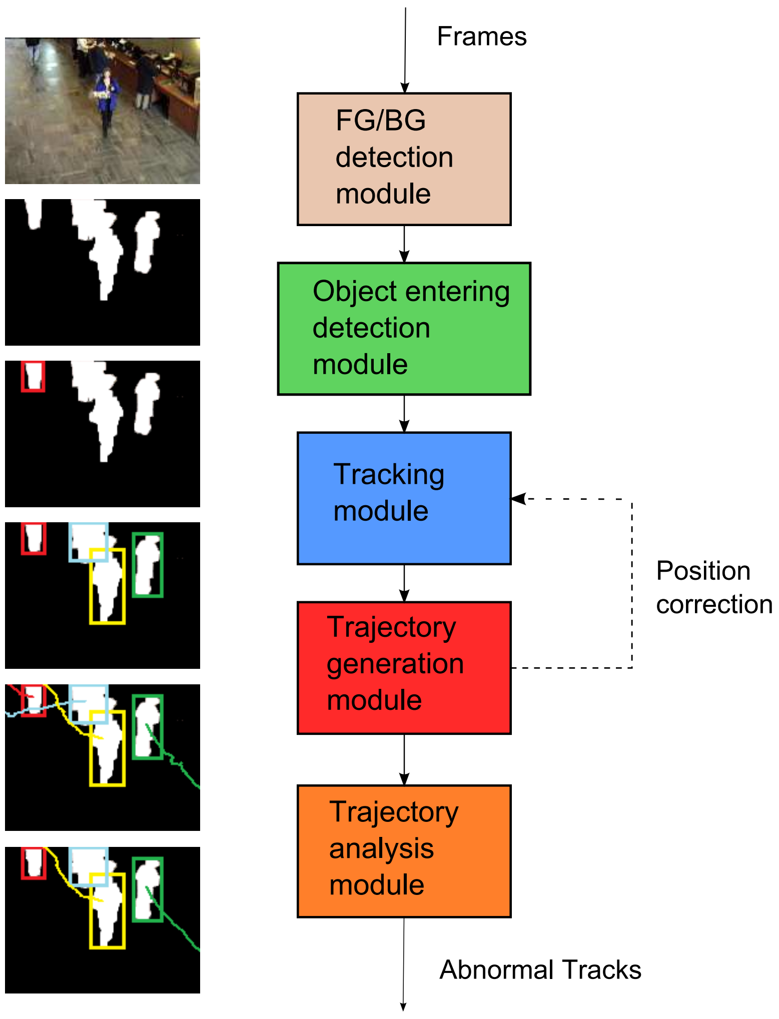

Frames
FG/BG detection module
Object entering detection module
Tracking module
Trajectory generation module
Trajectory analysis module
Position correction
Abnormal Tracks

Figure 2.1: General scheme of a video analytics system.

Human visual system is able to capture and process images with impressive efficiency despite the biological substrate (neurons based on electro-chemical processes) on which it is based is considerably slower than current electronic systems [9]. The scene understanding process from sequences of images is a hierarchical process in which information is gathered at different abstraction levels, from basic primitives (such as color, contrast, movement, depth, etc.) to higher abstraction levels which perform the interpretation of the scene [10, 11]. We intend to emulate this brain functioning, in which the first stages perform tasks which can be implemented on embedded devices. Further processing stages are based on information obtained from basic primitives to carry out more elaborate analyses. Tasks belonging to these abstraction levels require more algorithmic descriptions of the methods used and are not suitable for specific-purpose hardware development [12].

Improvements on technology and video analysis techniques have allowed the research field to focus on multi-camera systems. This kind of systems offers several advantages, such as occlusion solving in scenarios which have cameras with overlapping views or object tracking between non-overlapping cameras [13, 14, 15]. However, the increasing number of cameras in a system leads to several difficulties related to the camera network configuration, as well as the processing capabilities required to process multiple video streams. In order to deal with these difficulties, research on decentralized networks, where part of the processing is performed in each node, has aroused great interest [16]. This kind of solution allows for reducing the processing in central nodes or servers. In addition, the bandwidth required by the camera network can be more constrained, since not all information must be transmitted. Much work has been carried out on embedded systems and *smart cameras* (cameras with processing and communication capabilities) to enable the processing in decentralized and distributed networks.

Traditionally, research on smart cameras has been aimed to specific-purpose architectures such as FPGAs and DSPs. Nevertheless, computer vision algorithms usually have to be adjusted to work with constrained resources on these platforms, such as fixed-point arithmetics or memory constraints, which leads to certain loss of accuracy. In addition, development made for a specific architecture is seldom compatible with others, being thus dependent on a specific device or manufacturer. In this context, our interest lies in the usage of new platforms (i.e. smartphones) as smart cameras for distributed networks. Smartphones offer several advantages, such as having fairly good cameras, powerful processors in comparison with other embedded systems, and different kinds of connectivity and sensors.

The application of techniques of person detection and tracking is a complex field, specially on multi-camera platforms. The work performed in this thesis includes from image segmentation to efficient integration of estimations from different cameras to perform these tasks in a more robust and global way.

2.2 Scientific objectives

The scientific objectives established for the development of this thesis are related to the analysis of video streams from video surveillance systems. More specifically, our main goals are the following:

- Development and implementation of real-time background subtraction models. Method evaluation searching good trade-off between accuracy and computational efficiency.

- Integration of depth estimations (provided by multi-camera stereo vision and active depth sensors) in background subtraction models. Quality evaluation.

- Implementation of homography models to calibrate monitoring scenarios. Models aimed to multi-camera platforms.

- Usage of attention models in a multivision system. Trade-off between computational workload and extraction of global information.

- Applications. Study of diverse applications in the video analytics research field, using different camera setups. Image fusion, person and object tracking, global vision with occlusion solving by means of multiple cameras, adaptation of the vision system to different scenarios, etc.

2.3 Projects

The work developed in this thesis is related to a Project of Excellence from Junta de Andalucia named MULTIVISION and an european research project named TOMSY.

2.3.1 MULTIVISION

Real-time multi-camera vision system for scene understanding (MULTIVISION) (TIC-3873).

This project will address the development of an hybrid hardware/software vision system in real time, based on multiple cameras. The main goal of every vision system is extracting specific information from images (data extracted from “scene understanding”). In this project we will study vision schemes which allow for efficient processing of images provided by multiple cameras, and the joint management of estimations from the different cameras to carry out the “scene understanding” task in a more reliable and robust manner. This system has direct application in video surveillance and spaces monitoring platforms.

The objectives of this project are based on previous results from our research group (and international collaborations in the framework of European and national projects).

The term "multivision" is related to the usage of several cameras in different configurations which will allow us to study different notions. The use of multiple cameras focused in the same scenario (with or without overlapping views) enables exploring schemes to track objects between fields of view, studying traces of movements between different cameras, etc. In addition, the project will study configurations with several cameras with overlapping views. In this case we intend to study vision models which use complementary information from each camera to perform reliable "scene understanding". Finally we will study "global visual field" schemes, in which fields of view from several cameras with partial overlapping are combined.

A scenario covered by multiple cameras produces a considerable amount of information which is difficult to process in real time in a centralized way. For that reason we will study attention models which allow the system to perform low-level tasks in distributed nodes (for example, different FPGA devices), whilst high-level tasks are performed globally in central servers. We will study models in which all sequences are processed independently and we will compare them with models in which an "attentional model" selects a camera and extracts high-level information only from that source. We will compare the results obtained by both configurations (exhaustive analysis against selective analysis). The main goal is the design of "virtual agents" which can select a source of information based on low-level primitives to achieve correct partial understanding of the scene.

The project also includes the study of "learning" models. Since the structure and setup of surveillance cameras can vary in each scenario, the project will study learning models to correlate visual events with specific actions without supervision.

2.3.2 TOMSY

TOMSY, Topology Based Motion Synthesis for Dexterous Manipulation, EU project IST-FP7-Collaborative Project-270436.

Project description. The aim of TOMSY is to enable a generational leap in the techniques and scalability of motion synthesis algorithms. We propose to do this by learning and exploiting appropriate topological representations and testing them on challenging domains of flexible, multi-object manipulation and close contact robot control and computer animation. Traditional motion planning algorithms have struggled to cope with both the dimensionality of the state and action space and generalisability of solutions in such domains. This proposal builds on existing geometric notions of topological metrics and uses data driven methods to discover multi-scale mappings that capture key invariances - blending between symbolic, discrete and continuous latent space representations. We will develop methods for sensing, planning and control using such representations.

TOMSY, for the first time, aims to achieve this by realizing flexibility at all the three levels of sensing, representation and action generation by developing novel object-action representations for sensing based on manipulation manifolds and refining metamorphic manipulator design in a complete cycle. The methods and hardware developed will be tested on challenging real world robotic manipulation problems ranging from primarily "relational" block worlds, to articulated carton folding or origami and all the way to full body humanoid interactions with flexible objects.

The results of this project will go a long way towards providing some answers to the long standing question of the "right" representation in a sensorimotor control and provide a basis for a future generation of robotic and computer vision systems capable of real-time synthesis of motion that result in fluent interaction with their environment.

2.4 Methods and Tools

The algorithms presented in this thesis have been implemented, initially, in the development environment MATLAB. We have chosen MATLAB for the prototyping stage since it allows for fast algorithm development and result analysis. In addition, MATLAB provides big amount of functions to display images and other results, what eases the development of different solutions for the proposed problems. However, the models implemented in MATLAB are fast to develop but quite slow to run. For that reason, we have implemented the algorithms using C/C++ with support from the OpenCV library [17]. The implementation of these models in C/C++ requires less execution time, and this is important for video processing algorithms, due to the huge amount of information to be processed.

Regarding implementation of algorithms in embedded hardware, the use of hardware description language is required. A combination of ImpulseC [19] and VHDL has been chosen, using the environments EDK (Embedded Development Kit) and ISE Foundation by Xilinx Inc. [20]. Information from depth sensors (Microsoft Kinect [21]) has been obtained by means of OpenNI library [18]. Development of algorithms in commercial devices such as Android smartphones [22] has been performed by means of Java and C++, using JNI (Java Native Interface) to communicate the Android applications with native code.

2.5 Organization of chapters

The organization of this doctoral thesis continues with an introduction to the state of the art of the techniques used in video surveillance. In chapter 3 we revise the works existing in the literature for video analytics as well as the techniques studied in the development of this thesis. In chapter 4 we study with more detail background subtraction models suited to the implementation on reconfigurable hardware, the datasets (sequences with information about ideal results) and metrics used to evaluate them objectively, and we analyze the results obtained by several FPGA architectures. In chapter 5 we focus on improving the robustness of background subtraction models by fusing color and depth. This chapter includes our approach to integrate depth in a well-known algorithm, different methods to obtain depth information as well as results from datasets recorded by means of stereo cameras and Kinect sensor. In chapter 6 we study the use of smartphones in a multi-camera system to detect and track objects of interest in a scene. The information provided by each device is shared with the others to work in presence of occlusions and sudden movements. Finally, chapter 7 summarizes the main contributions of our research and suggests several lines of future work.

Chapter 3

State of the art

As we have mentioned in the introduction, there has been an increasing interest in the application of computer vision methods to video surveillance issues. Video surveillance addresses real-time observation of humans, vehicles and objects of interest in different environments, in order to perform "scene understanding", that is, leading to a description of the activities carried out by those objects. It is mostly used for security monitoring, traffic flow measuring, etc. [23, 24].

Video analytics is a field of research that is currently focusing on processing from multiple video streams. This is a logical evolution for the field due to its application to traditional video surveillance, in which every system consists of a camera network and a centralized monitoring terminal. However, single-camera video processing is already computationally expensive, and the increment on the number or sensors produces an even higher volume of information to process. This can make unfeasible the analysis in real-time of several video inputs.

The main interest of video analytics systems is the detection and tracking of people who appear in the scene to analyze trajectories and behaviors which could result in alarm situations. For that reason, extracting the background from a video sequence is a required feature for many applications related to video surveillance: vehicle traffic control, intruders' detection, suspicious objects, etc. The most usual approach to segment moving objects is known as background subtraction, and is considered a key first stage in video surveillance systems.

In the following sections we revise the state-of-the-art techniques for background subtraction in general, models and implementations oriented to embedded devices, and advanced methods which fuse information from different types of sensors. Furthermore, we revise the state of the art for multicamera video surveillance, including feature extraction, tracking, calibration and information exchange.

3.1 Background subtraction

Background subtraction algorithms get a sequence of frames as an input, and they build reference models which represent the static background of the scene during a certain period of time. These models are used to classify each pixel of a new frame as foreground (objects in movement which do not belong to the scene) or background. The segmentation obtained by the background subtraction model is provided to next stages of the video analytics system, allowing it to focus the attention in the objects of interest.

Multiple factors and events may affect the scene, causing the extraction of the background to be a non-trivial task. The main issues which background subtraction algorithms have to deal with are the following:

- Shadow detection. The shadow of an object produces a change in the lighting of certain region of the image, which can mislead the algorithm. Shadows and highlighted regions tend to be cause of false positives en those regions.
- Need of training. These algorithms build a reference model that represents the background. Nevertheless, there are scenes where the learning stage is complicated due to presence of foreground objects during the training.

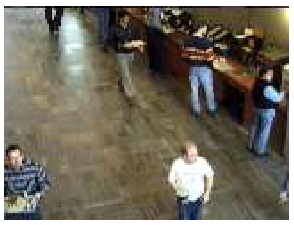


Figure 3.1: Example of scene with presence of foreground objects during the training stage. An algorithm must deal with foreground objects while building the initial model, since the presence of these can be unavoidable in certain scenes or camera setups, i.e. the hall of an airport.

- Illumination changes. Both gradual and sudden illumination changes are a challenge for this kind of algorithms, the formers related to progressive adaptation, and the latters related to the skill of the algorithm to detect global changes.

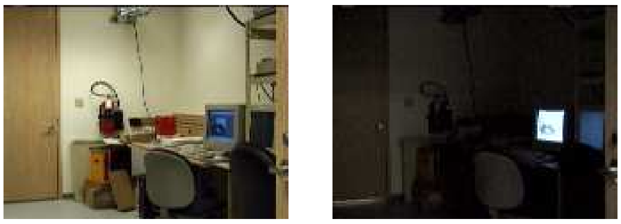


Figure 3.2: Example of scene with sudden illumination changes.

- Changes in the setup of the scene. In real-world problems, the background may change after the reference background model is built. Some examples of changes that would affect the segmentation are the addition of elements to the background, the movement of objects belonging to it or the removal of these objects from the camera field.

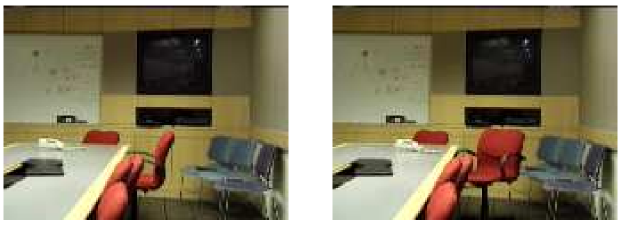


Figure 3.3: Example of scene with movements of objects that belong to the background.

- Repetitive movements. The presence of repetitive movements in elements that belong to the background is an important issue with which algorithms must deal. In order to deal with it, background models need to consider multiple color values for each pixel. Some examples that illustrate this problem are trees waving during a sequence, waves on the water surface or similar periodic movements. These should not be detected as foreground and require the generation of multivalued backgrounds.

There exist several methods of classification of background subtraction algorithms, according to how they deal with the mentioned problems. On the one hand, these techniques can be classified in static and dynamic algorithms. Static background subtraction methods assume that the background is fixed, and that differences are solely caused by foreground objects [25]. This kind of algorithms builds a reference background model during the training stage, but they do not perform any subsequent model maintenance. Dynamic algorithms, on the contrary, have an update stage which allows

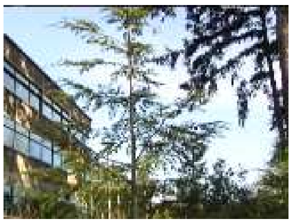


Figure 3.4: Example of scene with presence of repetitive movements in background elements.

them to adapt the model to changes in the setup of the scene as well as illumination changes.

On the other hand, background subtraction algorithms are also classified into unimodals or multimodals. This classification is related to the number of values that the algorithm can model for each pixel. Unimodal algorithms are the most basic ones, since they can model only one color value per pixel. For that reason, this kind of algorithms has considerable loss of accuracy with repetitive movements such as waving trees. In contrast, multimodal algorithms store in the model several possible values for each pixel, for example the green color of the trees and the color of behind them.

Regarding static background segmentation methods, there are different methods described in the literature which gather in this classification [25]. Haritaoglu et al. [3] propose a method which uses only grayscale information. Francois et al. [26] propose the usage of a Gaussian distribution to model mean and standard deviation in the RGB color space. Horprasert et al. [27] separate the luminance and chrominance in order to classify the pixel in foreground, background, shadow and highlighted background. Other works focus on color and edge detection [28, 29, 30, 31] or fuzzy classification [32]. The approach proposed by Horprasert el al. [27] is described in more detail in section 4.1.

Although these approaches allow for modeling shadows and small variations of color, they are limited to static backgrounds. In addition, most of the static models store only one value per pixel, being then unimodal algorithms. For that reason, they are affected by some of the problems mentioned before, what leads to loss of accuracy and large number of false alarms in real-world scenarios. The current state of the art of background segmentation algorithms is able to deal not only with static backgrounds, but also with moving ones. In order to model these complex scenarios, many models have been developed. One of the most widely used approaches for background modeling is the Mixture of Gaussians, MOG, by Stauffer and Grimson [33]. MOG is a pixel-wise technique which uses multiple Gaussians for each pixel in order to model multimodal backgrounds. Nevertheless, this algorithm suffers from slowness in the initial learning, as well as difficulties to differ between foreground objects and shadows, among other issues. For that reason, numerous improvements of the original method developed in [33] have been proposed over recent years. Bouwmans et al. [34] perform an exhaustive survey and of different kinds of improvements to MOG, classifying them into four different categories: intrinsic model improvements, extrinsic model improvements, reduction of computation time and fusion with other segmentation method.

In addition to models based on Mixture of Gaussians [33, 35], there are many others that use Bayesian decision rules [36, 37], Codebook-based models [38, 39, 40, 41], nonparametric kernel-based techniques [2], or Component Analysis (PCA and ICA) [42, 43]. The Codebook algorithm, proposed by Kim et al. [38] is explained in section 4.2.

Since there are many different types of algorithms, a proper comparison method is necessary. In [25] some relative metrics are proposed, based on True and False Positives and Negatives (TP,TN,FP,FN). Regarding test sequences and datasets which offer ground truth (ideal foreground) to obtain those quantitative measures, there exist several approaches in the literature [44, 45, 46, 47, 48]. Toyama et al. [44] proposed the dataset known as *Wallflower*, which has been used to test and compare background subtraction techniques during the last ten years. In [45], Prati et al. focus on algorithms with shadow detection capabilities. They present a comprehensive survey of moving shadow detection approaches, proposing both novel quantitative metrics and video sequences with associated ground truth to evaluate these approaches. Most recently, several new datasets have been proposed to test background subtraction methods [46, 47, 48]. The dataset proposed by Brutzer et al. [46] is an artificial dataset which is stated not to suffer from imperfect labels or small number of annotated frames.

As mentioned in the introduction, the work developed in this thesis project focuses on its application to smart cameras and distributed camera networks. In order to enable distributed and decentralized processing, much work has been performed on the deployment of video analytics techniques on embedded hardware such as FPGAs or DSPs. The usage of this kind of architecture allows the system to process the initial stages in each node, reducing bandwidth and required processing capabilities in the central servers. In addition, FPGAs require less power than commodity processor or GPUs, being more suitable for power constrained scenarios, between many others. In the following section, we revise the current state of the art of background subtraction methods regarding embedded platforms.

3.1.1 Background subtraction on embedded hardware

In spite of the differences between existing algorithms, background subtraction techniques are computationally expensive in general, especially when they are considered only the first stage in a multi-level video analytics system. For that reason, efficient implementation is key to the development of real-time video surveillance systems. In the framework of embedded systems implementations, characterized by power consumption and real-time constraints, several of these techniques have been implemented using FPGAs [43, 49, 50, 51, 52], or DSPs [53, 54]. There also are other real-time approaches using GPUs [55, 56], but these are not suitable yet for embedded devices. In the case of embedded systems, commodity processor implementations are not usually utilized although latest devices, such as Intel Atom, could soon address this market.

Oliveira et al. [51] introduce an FPGA implementation for the Horprasert algorithm, although the throughput reached by this approach is fairly low. Jiang et al. [50] present a compression scheme for Mixture of Gaussians model [33] which allows reaching a high frame rate. However, this approach is not explained in detail, and no results are shown about accuracy or power consumption. Appiah et al. [49] propose an implementation based on a simplified MOG, which offers fairly good throughput and acceptable accuracy. Bravo et al. [43] propose an FPGA implementation based on Principal Component Analysis (PCA), getting a good throughput and specifying the resource consumption. Nevertheless, there is no data about accuracy with a standardized dataset. Kryjak et al. [52] propose an FPGA architecture to perform background subtraction on high resolution images acquired through HDMI input, showing fairly good results both in accuracy and throughput by using high-end FPGAs. Carr [55] and Pham et al. [56] present GPU implementations based on MOG, with very different results in performance. Despite the approach described in [56] has higher frame rate than any of the other mentioned hardware implementations, being a GPU implementation is an impediment for embedded systems and low power constraints. Strictly speaking, there are new GPU families oriented to embedded devices that could solve that problem. However, the performance of these families is considerably lower than the ones of GPUs used on standard PCs or laptops.

3.1.2 Background subtraction with sensor fusion

Although state-of-the-art background subtraction algorithms are able to cope with many issues that affect this kind of technique, they rely specifically on color values. For that reason, they are prone to errors when foreground objects have color similar to the background. In addition, they suffer from the presence of shadows if these shadows are too dark, since the change of color in some image regions is greater than what the algorithm is able to adapt to. Since robustness is an important requirement of video analytics systems, there is a need for algorithms with higher accuracy than the studied ones.

Currently, there is a clear understanding that, in order to achieve this robustness, multimodal information fusion is required. For that reason, many works have proposed algorithms fusing intensity, edges and texture information [57, 58, 59, 60, 61, 36, 62, 29]. Even though these methods are a very valuable contribution for any robust surveillance application, the problems associated to the camera sensor still persist.

For that reason, the use of range information applied to foreground segmentation has been previously studied in the literature [63, 64, 65, 66, 67, 68, 69]. Cristani et al. [63] proposes a comprehensive review of background subtraction techniques, focusing on different sensor channels, including systems based on stereo cameras. Ivanov et al. [64] proposed an approach that uses the disparity to warp one image of the pair (principal) in the other one (auxiliary). If the color and brightness between corresponding points do not match, the pixels either belong to a foreground object or to an *occlusion shadow*. This method does not use background subtraction algorithms in order to perform the segmentation. In [65], Gordon et al. described a background estimation method based on range and color clustering at each image pixel. Disparity and color are used together to reduce the effect of classic segmentation problems in each data source when taken separately. This approach used an approximation of a Mixture of Gaussians to fit the background data. This approximation assumes only one of the clusters is associated to background information, thus being an unimodal background subtraction algorithm.

Kolmogorov et al. [66] described two algorithms for bi-layer segmentation fusing stereo and color/contrast information, focused on live background substitution for teleconferencing. In order to segment the foreground, this approach relies most on the stereo method, assuming that people participating in the teleconference are closer to the camera, and then color information is used to cope with stereo occlusion and low-texture regions.

Schiller et al. [67], Crabb et al. [68] and Zhu et al. [69] proposed combinations of color-based background subtraction and depth information obtained by low-resolution Time-Of-Flight camera (204×204 pixels, 160×120 and 176×144, respectively). Due to this low resolution, inaccuracies are produced at object boundaries. For that reason, Schiller et al. [67] proposed a reliability measure for depth information based on the amplitude image provided by the ToF camera. ToF cameras or the Kinect peripheral are not based on color, suffering thus from different issues than other color-based approaches such as binocular disparity. For that reason, these sensors are an interesting option that has also been studied in this project.

3.2 Object tracking

In addition to background subtraction, object tracking is considered an important task within the field of computer vision, and especially in video analytics. As it is mentioned in the chapter 2, the stages in a video surveillance system can be gathered in three steps: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior and detect abnormal situations.

In section 3.1 we have revised the first step, detecting interesting moving objects by means of video segmentation. The second step, object tracking, is an interesting application of computer vision that has aroused much interest in recent years. In [70], Yilmaz et al. propose a survey that categorizes many tracking methods according to several aspects: object representation and features selected to track, how the object detection is performed, and the methods used to generate robust trajectories.

In the following sections we revise the state of the art methods for modeling objects of interest and track them frame to frame. Furthermore, a revision of multi-camera segmentation and tracking techniques is performed in section 3.2.3.

3.2.1 Feature selection

Almost all tracking algorithms require detection of the objects either in the first frame or in every frame. This detection can be performed by means of point detectors, segmentation, background modeling or supervised classifiers. Some well-known works in the literature about point detectors include Harris interest point detector [71], KLT detector [72], SIFT detector [73], and SURF detector [74, 75]. The aim of image segmentation algorithms is to partition the image into similar regions. The most relevant image segmentation techniques for object tracking are Mean-Shift Clustering [76], Graph-Cut [77] and Active Contours [78]. Regarding supervised classifiers, the most widely used are SVM (Support Vector Machines) [79], Neural Networks [80] and Adaptive Boosting [81].

3.2.2 Tracking

The main aim of a tracker is the generation of object trajectories by locating the position of the object in every frame. There are many works which cover track generation, divided in different categories according to the representation of the object to track. Some examples of widely used point trackers are MGE tracker [82], Kalman filter [83], Particle filter [14, 84] and PMHT [85]. Regarding kernel tracking, Mean-Shift is the best known color-based tracker [86], while KLT [72] computes the translation of a region centered on an interest point, being thus a feature-based tracker.

3.2.3 Multi-camera approaches

In this thesis project we are interested on multi-camera networks for video surveillance. Despite the improvements on mono-camera trackers, the usage of multiple video streams improves the robustness in presence of occlusions or when tracking multiple objects. There exist many works in the literature which focus specifically on multi-camera segmentation and tracking [14, 87, 88, 13, 89, 90, 15, 62, 91, 92, 93, 16, 94, 95, 96, 97]. In [14], Nummiaro et al. use calibrated cameras in *smart rooms* in order to perform face tracking by means of particle filtering. The information from different cameras is integrated only when a camera loses the target. Möller et al. [87] use non-calibrated cameras by defining *transfer regions* to perform hand-over to other camera when the tracked object enters these regions. The first camera detects and stores the histogram of the object using [81], and objects are tracked by means of Mean-Shift [72]. Black et al. [88] require a full Tsai calibration [98] to obtain 3D geometry, computing homographies between each camera pair. After that, tracking is carried out by using Kalman filters, correlating the centroids by means of the homographies. Kim et al. [13] do not require full Tsai calibration [98], but homographies between each camera and the ground floor. They define a human appearance model assuming that people are standing and textures are thus vertical. Background subtraction [38] is combined with Bayes classifiers [99] and Particle Filtering to perform an iterative segmentation-tracking method. Berclaz et al. [97] use probability occupancy maps [92] to extract fragments of trajectories. This information is used to build behavioral maps in top-view. Khan et al. [15] do not build appearance models for each person, only segmentation. Instead of obtaining FG/BG masks, they compute foreground probability maps which are gathered in top-view by using homographies to the ground floor. Tracking is performed only in reference view and propagated to the others.

3.2.3.1 Distributed and decentralized camera networks

About distributed and decentralized camera networks, [16] presents a survey covering calibration, synchronization and fusion of information from different cameras, as well as a comparison between decentralized trackers. In

[100], four kinds of two-node architectures to perform particle filtering are analyzed regarding bandwidth utilization and energy consumption. These architectures differ in the information sent across the network, and the steps of the particle filtering (PF) algorithm executed in each node.

Regarding smart camera networks, [101] propose a self-organizing network of mobile phones based on Java J2ME which performs communication between devices through Bluetooth connections. Background differencing is used in order to detect objects entering or leaving the field of view. Experiments were performed on a two-node setup. In [102], a hardware architecture on DSPs to track people in a two-camera network is proposed. Tracking is combined with predefined transfer regions to perform handover between cameras. [103] present an architecture using MIDs (Mobile Internet Devices) that detects people using the Android built-in face detector, and obtains the optical flow field in order to perform frame synchronization.

Chapter 4

Background subtraction on embedded hardware

Extracting background from a video sequence is a required feature for many applications related to video surveillance: vehicle traffic control, intruders' detection, suspicious objects,etc. The most usual approach to segment moving objects is known as background subtraction, as has been explained in section 3.1. This technique consists of building a reference model which represents the static background of the scene during a certain period of time. Multiple factors and events may affect the scene, making background subtraction a non-trivial task: sudden and gradual illumination changes, presence of shadows, or background repetitive movements (such as waving trees), among many others.

Despite there exist many different algorithms to perform background subtraction, in general it is a computationally expensive task. Whilst this could be solved by using powerful processing units in a centralized network, in multi-camera video analytics systems that would not be a scalable solution. When the number of cameras increases, having a centralized surveillance network leads to issues related to the network topology, due to high bandwidth requirements. For that reason, our interest in this part of the thesis project lies in the deployment of background subtraction techniques on embedded hardware, which allows for distributed and decentralized camera networks. Thus, the use of FPGAs [43, 49, 50, 51] and DSPs [53, 54] is justified by requirements of scalability, size and low power consumption which are key features that other technologies are not able to achieve.

The state of the art of background subtraction on embedded hardware has been summarized on section 3.1.1. In this thesis project, we have proposed two FPGA architectures based on the method described by Horprasert

[27, 25], with the extension that allows for shadow detection [45], and the Codebook method described by Kim et al. [38].

The first method [27] has been selected since it requires less memory to store the model while keeping fairly good accuracy, hence being more suitable for implementation in low cost FPGAs [51]. This algorithm builds a static background model, which means that the model is obtained at an initial training phase. There are other methods which build dynamic background models [33, 38], which can adapt themselves to changes in the scene. The main difference between these models, as far as required hardware resources are concerned, is that the latter have much higher memory consumption requiring external memory with an important bandwidth. Furthermore, the shadow detection capabilities increase the accuracy of the object shape detection, which helps to achieve a better object classification and reduces the errors due to shadows artifacts.

The second implementation is based on the algorithm proposed by Kim et al. [38]. This algorithm has been classified by several authors [40, 41] as a good trade-off between accuracy and efficiency. This has motivated our choice as a target model for implementation even though it is a much more complex model than the previous hardware implementations available in the literature. This model has much higher memory comsumption than the one proposed by Horpraser et al. [27], requiring thus both optimization and simplifications as well as external memory with an important bandwidth.

The rest of this chapter is organized as follows. In sections 4.1 and 4.2 we explain the two background algorithms studied for the deployment on reconfigurable hardware. In section 4.3 we describe the datasets and the metrics used to evaluate and compare our architectures with previous works in the literature. Section 4.4 shows the experimental results obtained by the proposed architectures. Finally, we summarize in section 4.5 the conclusions of this work.

4.1 Horprasert algorithm

As previously mentioned, our first implementation is based on the algorithm proposed by Horprasert et al. [27]. This algorithm basically obtains a reference image to model the background of the scene so that it can perform automatic threshold selection, subtraction operation and, finally, pixel-wise classification.

Horprasert et al. [27] propose a color model in the three-dimensional RGB color space. The main idea is that chromaticity and brightness are perceived separately, since humans tend to assign a constant color to an object despite illumination changes.

In this section we describe the original model, explaining the background modeling and the subtraction operation and classification steps. We also give an overview of the architecture we have developed on FPGA and the modifications made to the model in order to aim for hardware-friendly implementation.

4.1.1 Background Modeling

In order to build a reference image which represents the background, a number N of images will be used, whose color space is given in RGB. The algorithm is pixel-wise, being the background modeled statistically on a pixel by pixel basis. Each pixel $< i >$ from the image is modeled by a 4-tuple $< E_i, S_i, a_i, b_i >$, where each element is defined as follows:

- E_i the expected color value, defined as

$$E_i = [\mu_R(i), \mu_G(i), \mu_B(i)] \tag{4.1}$$

  being $\mu_R(i), \mu_G(i), \mu_B(i)$ the arithmetic means of each color channel for pixel i.

- S_i the value of the color standard deviation for each channel, defined as

$$S_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)] \tag{4.2}$$

- a_i the variation of the brightness distortion α_i, given by Equation (4.3).

$$\begin{aligned}\alpha_i = min &\left[\left(\tfrac{I_R(i)-\alpha_i\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\tfrac{I_G(i)-\alpha_i\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\tfrac{I_B(i)-\alpha_i\mu_B(i)}{\sigma_B(i)}\right)^2\right] \\ &= \frac{\frac{I_R(i)\mu_R(i)}{\sigma_R^2(i)} + \frac{I_G(i)\mu_G(i)}{\sigma_G^2(i)} + \frac{I_B(i)\mu_B(i)}{\sigma_B^2(i)}}{\left(\frac{\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\frac{\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\frac{\mu_B(i)}{\sigma_B(i)}\right)^2}\end{aligned} \tag{4.3}$$

- b_i the variation of chromaticity distortion CD_i, which is described in Equation (4.4).

$$CD_i = \sqrt{\left(\frac{I_R(i)-\alpha_i\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\frac{I_G(i)-\alpha_i\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\frac{I_B(i)-\alpha_i\mu_B(i)}{\sigma_B(i)}\right)^2} \tag{4.4}$$

Since α and CD follow different distributions for different pixels, the model requires the use of the variation of these values as a normalization factor to use the same thresholds for all pixels. These variations are included in the background model as a_i and b_i in the 4-tuple that models the background for each pixel.

a_i represents the variation of the brightness distortion of the i_{th} pixel, given by

$$a_i = RMS(\alpha_i) = \sqrt{\frac{\sum_{i=0}^{N} (\alpha_i - 1)^2}{N}} \tag{4.5}$$

b_i represents the variation of the chromaticity distortion of the i_{th} pixel, given by

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{i=0}^{N} CD_i^2}{N}} \tag{4.6}$$

More detailed information about how the background model is built can be found in [27].

4.1.2 Subtraction Operation and Classification

In this stage, the difference between the background model and the current image is evaluated. This difference consists of two components: brightness distortion α_i and chromaticity distortion CD_i. In order to use a single threshold for all pixels, it is necessary to normalize α_i and CD_i as follows:

$$\hat{\alpha_i} = \frac{\alpha_i - 1}{a_i} \tag{4.7}$$

$$\hat{CD_i} = \frac{CD_i}{b_i} \tag{4.8}$$

After the normalization of brightness and chromaticity distortions, the given pixel can be classified into one of the four categories:

- *Background.* If both brightness and chromaticity of the current pixel are similar to those of the background model.
- *Shadowed background.* If the chromaticity is similar to the background model but the brightness is lower in the current pixel. This assumes that the chromaticity of a shadow is similar to the one of the object under normal lighting.
- *Highlighted background.* If the chromaticity is similar to the background model but the brightness is higher.

- *Foreground.* If the pixel has different chromaticity from the expected values.

Based on these definitions, a pixel is classified into one category according to Equation (4.9), that is the analytical representation derived from the model presented in Figure 4.1.

$$C(i) = \begin{cases} \text{Foreground:} & \hat{CD}_i > \tau_{CD}, \text{ or } \hat{\alpha}_i < \tau_{\alpha lo} \text{ else} \\ \text{Background:} & \hat{\alpha}_i > \tau_{\alpha_1}, \text{ and } \hat{\alpha}_i \geq \tau_{\alpha 2} \text{ else} \\ \text{Shadowed background:} & \hat{\alpha}_i < 0, \text{ else} \\ \text{Highlighted background:} & \text{otherwise} \end{cases} \tag{4.9}$$

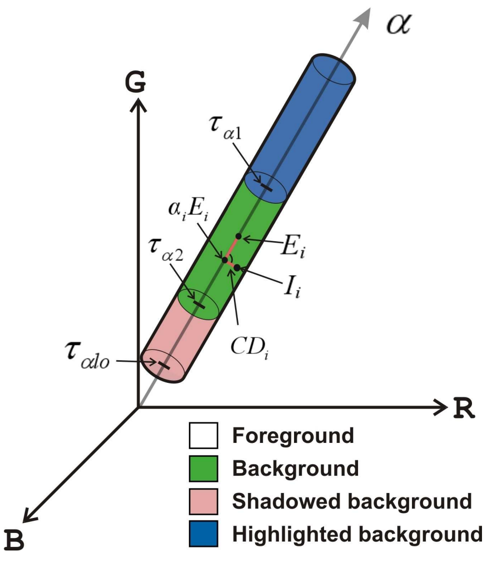


Figure 4.1: Graphic representation of the model used to classify the pixels in the categories. This model is oriented to shadow and highlights detection, taking into account chromaticity lines as well as brightness changes.

The thresholds τ_{CD}, $\tau_{\alpha 1}$, $\tau_{\alpha 2}$ are automatically selected from the information obtained during the training stage, as explained in [27]. $\tau_{\alpha lo}$ is a lower bound used to avoid misclassification of dark pixels.

This approach to shadow detection is considered a Statistical non-parametric (SNP) method [45], what means that the approach uses probabilistic functions to describe the class membership, and it is non-parametric since the thresholds are automatically determined by means of a statistical learning procedure.

This model has been selected for its implementation on reconfigurable hardware. For that reason, several modifications have been made in order to reduce the hardware complexity of the architecture. These simplifications towards a hardware friendly model generate some degradation on the original model's quality that will be evaluated in subsequent sections. These modifications are described in Section 4.1.3.

4.1.3 Model simplifications

The foreground/background segmentation is executed by a hardware module with an independent access to the memory, where the current image and the background model are stored. Considerable reduction of the hardware complexity of the architecture is achieved through precalculating and storing several constants during the training stage and avoiding division operations by substituting them for multiplications, which require less hardware resources. In the case of brightness distortion α_i, these constants are computed according to Equation (4.10):

$$\begin{aligned}
A_i &= \left(\frac{\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\frac{\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\frac{\mu_B(i)}{\sigma_B(i)}\right)^2 \\
B_i &= \left(\frac{\mu_R(i)}{A_i\sigma_R^2(i)}\right) \\
C_i &= \left(\frac{\mu_G(i)}{A_i\sigma_G^2(i)}\right) \\
D_i &= \left(\frac{\mu_B(i)}{A_i\sigma_B^2(i)}\right)
\end{aligned} \tag{4.10}$$

The brightness distortion α_i will remain as in Equation (4.11), making use of the constants B_i, C_i, D_i.

$$\alpha_i = B_i I_R(i) + C_i I_G(i) + D_i I_B(i) \tag{4.11}$$

In order to remove the divisions in the computation of the chromaticity distortion CD_i, we store $(S_i)^{-1}$, $(a_i)^{-1}$ and $(b_i)^{-1}$ instead of S_i, a_i and b_i. Besides, the training stage is done with $N = 128$ images to facilitate the computation of the mean, standard deviation and root mean square, avoiding divisions. Previously, the model had a 4-tuple for each pixel, composed by $< E_i$, S_i, a_i, $b_i >$, whereas now a 7-tuple will have to be stored$< E_i$, B_i, C_i, D_i, $(S_i)^{-1}$, $(a_i)^{-1}$, $(b_i)^{-1} >$. The hardware complexity has been reduced considerably, but at the cost of increasing memory consumption, since now we also have to store the constants B_i, C_i and D_i.

The software implementation has been developed using double floating-point representation. This allows reaching a higher degree of accuracy at the expense of a worse performance on embedded devices. In order to develop a hardware implementation on FPGA with constrained resources, a fixed-point representation is usually employed since it adjusts itself better to the type of available resources, although a detailed study is required in order to optimize the trade-off between accuracy and hardware consumption. It is important to take into account that an insufficient number of bits may lead to inaccurate results with high quantification noise. On the contrary, the use of too many bits can increase the hardware resources consumption, making the system implementation on a moderate cost FPGA unfeasible. In order to determine the appropriate number of bits for the fractional part of the variables, we have measured the error between the results obtained with different bit-width configurations and the floating-point representation. This comparison has been performed by using the Wallflower dataset [44], which we describe in section 4.3. The selected bit-width configuration is shown in Table 4.1

Table 4.1: Bit-width of each variable taking part in the calculation of colordist and brightness. The first value represents the integer part and the second value represents the fractional part. The bit-width values have been determined as the minimum values of the fractional part for which the quantization error is approximately stable.

Variable	**Bits**
$< B_i, C_i, D_i >$	[18 8]
$< E_i, (S_i)^{-1} >$	[8 8]
$< (a_i)^{-1}, (b_i)^{-1} >$	[8 10]
α_i	[28 8]
CD_i	[18 8]
$\hat{\alpha_i}$	[36 10]
$\hat{CD_i}$	[26 10]

4.1.4 Hardware Architecture

An optimized hardware architecture has been developed using novel ideas that allow for a high degree of algorithm tuning for optimized digital hardware implementation. They can be summarized as follows:

1. Hardware/software co-design. The use of a mixed hardware/software architecture allows us to share its resources to solve many algorithm stages as the ones related with communication, system initialization, basic control, system debugging,etc... It is not necessary to develop custom datapaths for these stages because no critical real-time restrictions are imposed to them. This permits to reduce hardware resources, to extend the system flexibility and to significantly reduce development time.

2. Superscalar and pipelined architecture. Multiple functional units run in parallel to adapt the intrinsic algorithm parallelism. The whole implementation has been carefully pipelined in order to increase the throughput. These strategies allow us to keep the pixel-rate very high and to achieve a significant performance.

3. Adaptable fixed-point arithmetics. The bit-width of the different processing stages has been tuned according to the accuracy requirements of each processing element. This approach is very different from the one used on many DSPs or digital hardware implementations that has a basic bit-width for all the processing stages. Our approach allows us to keep resources always tuned to the required accuracy at the cost of increasing the complexity of the system design. Hopefully, the use of high-level description languages helps to reduce the development time and make this option feasible with acceptable design time.

4. Proper utilization of the right level of abstraction for description of the different algorithm modules. The processing stages mainly require a DSP-based design flow which is well described using high-level description languages (as provided by ImpulseC [19]) whilst basic controllers such as the ones required for memory interfaces or low level communications are better described in RTL (for instance using VHDL or Verilog). In addition, sequential operations such as the ones required to build communication packages are well described by software code. Our implementation uses different descriptions based on the previous considerations. This enables to get the maximum output out of each description level in terms of performance or development time.

As it could be understood from the previous sentences, the advantage of our implementation relies on the combination of the latest design methodologies, seldom addressed together in the same design. The drawback of this novel approach is that it requires a high degree of competences at many different design levels, languages and tools. Nevertheless, the advantage is that it allows highly optimized designs that completely fit the target application.

In order to address this implementation, we have used EDK (Embedded Developer's Kit) of Xilinx Inc. [20]. The EDK environment facilitates the design of complex and completely modular SoC architectures able to support embedded microprocessors (MicroBlaze, PowerPC, ...), peripheral and memory controllers (Ethernet, DDR2, ZBT, ...), and interconnecting buses (PLB, NPI, MCH ...), whilst IP cores for specific processing can be designed using HDL languages through the ISE tool. As board we use the ViSmart4 video processing board from Seven Solutions [104].

This architecture consists of several modules and interconnect buses, as shown in Figure 4.2. Processing modules, peripherals and a Microblaze pro-

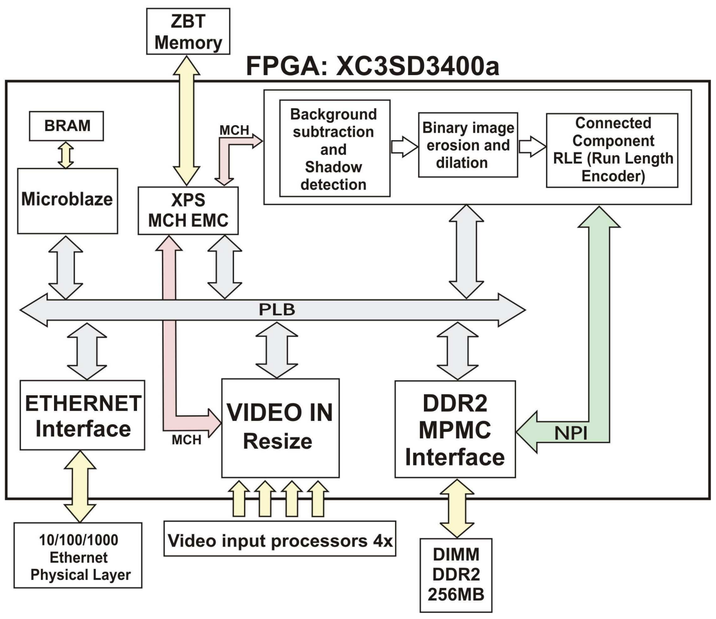

ZBT Memory
FPGA: XC3SD3400a
BRAM
Microblaze
MCH
XPS MCH EMC
Background subtraction and Shadow detection
Binary image erosion and dilation
Connected Component RLE (Run Length Encoder)
PLB
ETHERNET Interface
MCH
VIDEO IN Resize
DDR2 MPMC Interface
NPI
10/100/1000 Ethernet Physical Layer
Video input processors 4x
DIMM DDR2 256MB

Figure 4.2: Scheme of the complete architecture and connections between modules, peripherals, memory and processor.

cessor are connected to a PLB bus. The VIDEOIN module captures images from four independent analog inputs and stores them in a ZBT SSRAM external memory, through the MCH port. Through the PLB bus, Microblaze has access to: memory regions (ZBT or DDR2), configuration registers of the peripherals and the ethernet interface for data and image sending/receiving. The Background subtraction, shadow detection and blob detection (erosion, dilation and RLE) module performs an intensive processing on the pixels of each image in order to separate foreground and background, and then proceeds to the blob extraction of the different objects. This module uses ImpulseC [19] in order to develop a DSP-based design flow system. The Microblaze processor is programmed in C/C++ for initialization and communications tasks, and the rest of peripherals are described in VHDL. The MPMC module (DDR2 memory controller) offers an easy access to the external DDR2 memory, which stores the background model. This memory offers efficient high bandwidth access, thus providing a feasible use for applications requiring real-time processing.

Figure 4.3 shows a basic scheme of the proposed architecture for the IP core that performs the background subtraction, pixel classification and blob detection processing stages [105, 106]. This architecture consists of a pipelined structure divided into several basic stages which work in parallel. In addition, it is controlled by means of the embedded Microblaze processor. It is important to note that memory has a key role in the system performance

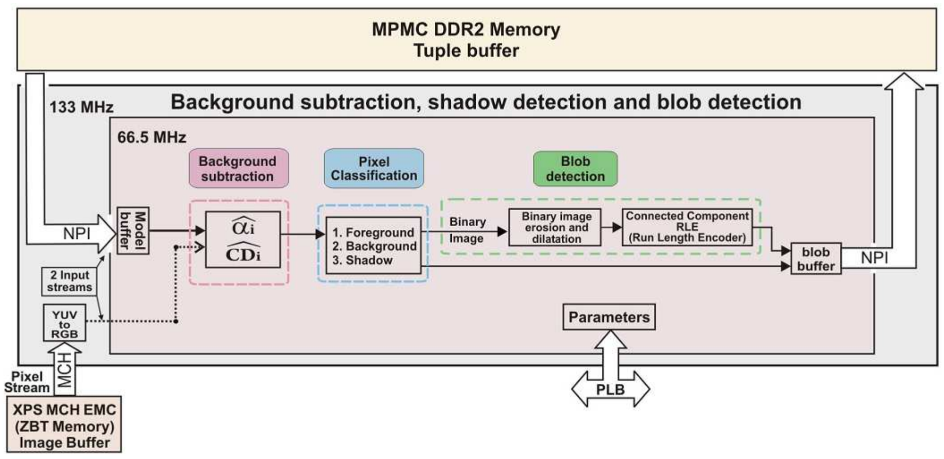


Figure 4.3: Simplified datapath architecture for background subtraction and blob detection core. The IP core can process streams from up to four cameras.

and requires an efficient memory accessing scheme. This has motivated the use of high performance multiport memory controllers (Xilinx MPMC for DDR2) as well as very specific and optimized memory ports (NPI).

4.2 Codebook algorithm

The Codebook algorithm, as proposed by Kim et al. [38], is based on the construction of a background model adopting a quantization/clustering technique described by Kohonen [39] and Ripley [107]. The above-mentioned work shows that the background model for each pixel is given by a codebook consisting of one or more codewords. Codewords are data structures that contain information about pixel colors, color variances and information about how frequently each codeword is updated or accessed.

The different stages of the Codebook algorithm are described below:

4.2.1 Background modeling

Given a set of N time steps (frames), a training sequence S is used for each pixel consisting of N RGB vectors. Each pixel has a different codebook, represented as $C = \{c_1, c_2, c_3, ...c_L\}$, consisting of L codewords, where L can be different for each pixel. Each codeword $c_i, i = 1...L$; consists of a RGB vector $v_i = \left(\bar{R}_i, \bar{G}_i, \bar{B}_i\right)$ and a 6-tuple $aux_i = \left\langle \check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i \right\rangle$, described as follows:

- $v_i = \left(\bar{R_i}, \bar{G_i}, \bar{B_i}\right)$, average value of each color component.

- $\check{I}_i, \hat{I}_i$, min and max brightness, respectively, of all pixels assigned to codeword c_i.

- f_i, the frequency (number of frames) with which codeword c_i has been updated.

- λ_i, the *maximum negative run-length* (MNRL), defined as the longest interval of time during which codeword c_i has not been updated.

- p, q, the first and last updating access times of codeword c_i.

The detailed algorithm for codebook construction is given in Alg. 1.

Algorithm 1 Algorithm for Codebook construction

```
$C = \phi, i = 0$
**for** $t = 1 \rightarrow N$ **do**
  $x_t = (R, G, B), I \leftarrow \sqrt{R^2 + G^2 + B^2}$
  Find the codeword $c_m$ in $C$ matching to $x_t$ based on two conditions:
   (a) $colordist(x_t, v_m) \leq \epsilon_1$
   (b) $brightness\left(I, \left\langle \check{I}_m, \hat{I}_m \right\rangle\right) =$ **true**
  **if** $C = \phi$ **or** there is no match **then**
    {Create a new codeword}
    $v_i = (R, G, B)$
    $aux_i = \langle I, I, 1, t-1, t, t \rangle$
  **else**
    Update matched codeword $c_m$ consisting of $v_m = \left(\bar{R}_m, \bar{G}_m, \bar{B}_m\right)$ and
    $aux_m = \left\langle \check{I}_m, \hat{I}_m, f_m, \lambda_m, p_m, q_m \right\rangle$ by setting
    $v_m = \left(\frac{f_m\bar{R}_m+R}{f_m+1}, \frac{f_m\bar{G}_m+G}{f_m+1}, \frac{f_m\bar{B}_m+B}{f_m+1}\right)$
    $aux_m = \left\langle min(I, \check{I}_m), max(I, \hat{I}_m), f_m + 1, max(\lambda_m, t - q_m), p_m, t \right\rangle$
  **end if**
**end for**
```

Conditions (a) and (b) in step 2 must be evaluated in order to determine if a pixel $x_t = (R, G, B)$ matches the codeword c_m. $colordist(x_t, v_m) \leq \epsilon_1$ is satisfied when the pure colors of x_t are similar to those of v_m, whilst $brightness\left(I, \left\langle \check{I}_m, \hat{I}_m \right\rangle\right)$ is true when the brightness of x_t lies between the acceptable bounds of c_m. These conditions are explained in more detail in section 4.2.1.2

4.2.1.1 Maximum negative run-length

The set of codebooks obtained from the previous step may include some moving foreground objects as well as noise. In order to obtain the true background model, it is necessary to separate the codewords containing foreground objects from the true background codewords. This true background includes both static pixels and background pixels with quasi-periodic movements (for instance waving trees in outdoor scenarios). This motivates the temporal criterion of MNRL (λ), which is defined as the maximum interval of time that the codeword has not appeared during the training period. A codeword with large λ is associated to an object that does not appear in that location except for specific moments, being probably a foreground object.

The background model M obtained from the initial set of codebooks C is given in (4.12). λ_m is the MNRL of codeword c_m. T_M is the time threshold set equal to half the number of training frames, $N/2$. Thus, codewords having a large λ_m (larger than T_M) will be eliminated from the corresponding codebook.

$$M \leftarrow \{c_m | c_m \in C \wedge \lambda_m \leq T_M\} \tag{4.12}$$

4.2.1.2 Color and brightness

In order to deal with illumination changes, the original algorithm [38] develop a color model to perform a separate evaluation of color and brightness distortions. This separation was already addressed in the other studied algorithm [27].

Summarizing, we can say that the evaluation of color distorsion basically consists of determining the distance between the color of an input pixel $x_t = (R, G, B)$ and $v_i = \left(\bar{R}_i, \bar{G}_i, \bar{B}_i\right)$ of codeword c_i, as indicated in (4.13).

$$\begin{aligned} \|x_t\|^2 &= R^2 + G^2 + B^2 \\ \|v_i\|^2 &= \bar{R}_i^2 + \bar{G}_i^2 + \bar{B}_i^2 \\ \langle x_t, v_i \rangle^2 &= \left(\bar{R}_i R + \bar{G}_i G + \bar{B}_i B\right)^2 \end{aligned} \tag{4.13}$$

Color distortion δ is calculated as indicated in (4.14).

$$\begin{aligned} p^2 &= \|x_t\|^2 cos^2\theta = \frac{\langle x_t, v_i \rangle^2}{\|v_i\|^2} \\ colordist\left(x_t, v_i\right) &= \delta = \sqrt{\|x_t\|^2 - p^2} \end{aligned} \tag{4.14}$$

Now, condition (b) evaluates how brightness change of $x_t = (R, G, B)$ lies within $[I_{low}, I_{hi}]$ range for codeword c_i. In this way, we allow the brightness change to vary in a certain range, as defined in (4.15).

$$\begin{aligned} I_{low} &= \alpha \hat{I} \\ I_{hi} &= min\left\{\beta \hat{I}, \frac{\check{I}}{\alpha}\right\} \end{aligned} \tag{4.15}$$

Typically, α is in the interval [0.4, 0.8], and β is in the interval [1.1, 1.5]. The brightness function is defined in (4.16).

$$brightness\left(I, \left\langle \check{I}, \hat{I} \right\rangle\right) = \begin{cases} \text{true} & if I_{low} \leq \|x_t\| \leq I_{hi} \\ \text{false} & otherwise \end{cases} \tag{4.16}$$

4.2.2 Foreground detection

Extracting the foreground from the current image is straightforward once we have obtained the background model M. The algorithm performing this task is detailed in Alg. 2.

Algorithm 2 Algorithm for Background subtraction

```
$x_t = (R, G, B), I \leftarrow \sqrt{R^2 + G^2 + B^2}$
For all codewords in $M$, Find the codeword $c_m$ matching to $x_t$ based on two conditions:
(a) $colordist(x_t, v_m) \leq \epsilon_2$
(b) $brightness\left(I, \left\langle \check{I}_m, \hat{I}_m \right\rangle\right) = \textbf{true}$
Update matched codeword as in algorithm 1
$BGS(X) = \begin{cases} \text{foreground} & \text{if there is no match} \\ \text{background} & \text{otherwise} \end{cases}$
```

4.2.3 Background model update

The original model proposed by Kohonen [39] assumes that the background obtained during the initial background modeling is permanent. In order to improve the model, making it more useful in a surveillance system, Kim et al. [38] have proposed a layered modeling and detection scheme. The initial scene can change after training. Therefore, these changes should be used to update the background model M. This can be done by defining an additional model H, called *cache*, where the new codewords are stored. Three new parameters $(T_H, T_{add}, T_{delete})$ are also defined. The periodicity of a codeword h_i stored in cache H is filtered by T_H, as we did previously with T_M in the background model M in eq. (4.12). The codewords h_i remaining in cache H for a time interval larger than T_{add} are added to the background model M. Codewords of M not accessed foa a period of time (T_{delete}) will be deleted from the background model. The detail procedure is given below:

Algorithm 3 Algorithm for Background model update

```
After training, the background model $M$ is obtained. Create a new model $H$ as cache.
For an incoming pixel $x$, find a matching codeword in $M$. If found, update the codeword.
Otherwise, find a matching codeword in $H$ and update it. If not found, create a new codeword $h$ and add it to $H$.
Filter out the cache codewords based on $T_H$:
  $H \leftarrow H - \{h_i | h_i \in H \wedge \lambda_i > T_H\}$
Move the cache codewords staying for enough time to $M$:
  $M \leftarrow M \cup \{h_i | h_i \in H \wedge h_i$ stays longer than $T_{add}\}$
Delete the codewords not accessed for a long time from $M$:
  $M \leftarrow M - \{c_i | c_i \in M \wedge \lambda_i > T_{delete}\}$
Repeat process from the step 2.
```

4.2.4 Model simplifications

The original model by Kim et al. [38] needs to be simplified in order to arrive at an affordable high performance hardware system. The main stages suitable for simplification can be summarized as follows:

- Storage and memory management. The amount of memory required to store the codebooks may change because the total number of codewords may increase or decrease dynamically. Our system uses a DDR2 memory, which provides high bandwidth, but it needs a regular access to reach the maximum performance, which complicates the dynamic management of the set of codebooks.
- Color distortion and brightness distortion computation. Equation (4.14) requires a square root and division operations, which are expensive on resources-constrained hardware devices.
- Model accuracy degradation due to fixed point arithmetic. Customized hardware systems normally use fixed-point data representation to reduce hardware resources utilization. However, this strategy requires careful analysis to avoid any degradation in accuracy.

In order to adapt the algorithm to the hardware implementation, allowing an easier use of the external memory containing the codewords, we have limited the number of codewords in each pixel. To establish the maximum number of codewords, we have carried out a detailed study using the

Wallflower dataset [44], which tests different algorithms in several problematic situations from which de ground truth is known. If we limit the number of codewords to 2 or 3 in sequences with non-static backgrounds (i.e. waving trees), our architecture performs similarly to unimodal models (Horprasert et al. [27]). After an in-depth analysis of accuracy and resources consumption, our choice of optimal number of codewords for a wide range of possible scenes is set to 5.

As stated before, the number of codewords assigned to a pixel will be limited in order to facilitate the hardware implementation, without significantly compromising the accuracy of the system. With this simplification, it may happen that the memory space of certain codebooks is already full (maximum limit reached) when new codewords are created. Then, it is necessary to replace an existing codeword. If this is the case, the replaced codeword will be the one not having been accessed for the longest time period. This condition will be added in Alg. 1, when new codewords are created.

Regarding the computation of color and brigthness distortions, all the codewords c_m are compared with the incoming pixel $x_t = (R, G, B)$ in parallel, using a $colordist(x_t, v_i)$ and $brightness\left(I, \langle \check{I}, hatI\rangle\right)$ block for each codeword c_m. Both division and square root operations implemented in these blocks represent a considerable consumption of hardware resources. Therefore, it is desirable to avoid these calculations in the FPGA implementation without significantly affecting the accuracy of the algorithm, using in some cases multipliers which are optimized with the embedded resources of the FPGA DSP48 for a Xilinx Spartan-3A DSP [20].

With respect to color distortion δ, we have implemented the modifications indicated in eq. (4.17).

$$\begin{aligned} \|v_i\|^2 p^2 &= \langle x_t, v_i\rangle^2 \\ \|v_i\|^2 \delta^2 &= \|v_i\|^2\|x_t\|^2 - \|v_i\|^2 p^2 = \|v_i\|^2\|x_t\|^2 - \langle x_t, v_i\rangle^2 \end{aligned} \tag{4.17}$$

Condition (a) in Alg. 2 will remain as eq. (4.18).

$$\|v_i\|^2\|x_t\|^2 - \langle x_t, v_i\rangle^2 \leq \epsilon^2 \|v_i\|^2 \tag{4.18}$$

With respect to brightness, we have established values $\alpha = 0.5$ and $\beta = 1.25$, so that calculations in eq. (4.15) can be easily computed by means of bit shifts, as follows in eq. (4.19).

$$\begin{aligned} I_{low} &= \alpha\hat{I} = \hat{I} >> 1 \\ I_{hi} &= min\left\{\beta\hat{I}, \frac{\check{I}}{\alpha}\right\} = min\left\{\hat{I} + (\hat{I} >> 2), \check{I} << 1\right\} \end{aligned} \tag{4.19}$$

In this way, we have reduced the consumption of hardware resources by avoiding the use of two multipliers and one division.

Table 4.2: Bit-width of each variable involved in the Codebook algorithm. The first value represents the integer part and the second value, the fractional part.

	Variable		Bits
	R_i, G_i, B_i		[8 0]
	$\bar{R_i}, \bar{G_i}, \bar{B_i}$		[8 5]
$\check{I}, \hat{I}$	$\lVert x_t \rVert$		[9 3]
$\langle x_t, v_i \rangle^2$	$\lVert v_i \rVert^2 \lVert x_t \rVert^2 - \langle x_t, v_i \rangle^2$	$\lVert v_i \rVert^2 \lVert x_t \rVert^2$	[36 5]
$\lVert v_i \rVert^2$	$\lVert x_t \rVert^2$		[18 5]
	$\epsilon^2 \lVert v_i \rVert^2$		[26 5]

During the updating process of v_m, there is a division for each color component. In order to reduce the consumption of resources in the FPGA, we have approximated f to its nearest power of 2, so that shift operations can be used instead of divisions. The implementation details of this modification can be seen in Alg. 4:

Algorithm 4 Simplified algorithm for updating v_m

```
if $1 \leq f \leq 3$ then sf = 2
else if $4 \leq f \leq 6$ then sf = 4
else if $7 \leq f \leq 12$ then sf = 8
else if $13 \leq f \leq 24$ then sf = 16
else sf = 32
$v_m \leftarrow \left\{ \frac{sf \cdot \bar{R_m} - \bar{R_m}}{sf} + \frac{R}{sf}, \frac{sf \cdot \bar{G_m} - \bar{G_m}}{sf} + \frac{G}{sf}, \frac{sf \cdot \bar{B_m} - \bar{B_m}}{sf} + \frac{B}{sf} \right\}$
```

The software implementation has been developed using double floating-point representation, which enables a high accuracy (at the expense of using high cost computing units with high resources consumption). For FPGA hardware implementation with constrained resources, a fixed-point data representation is usually adopted, as it is more suitable for the type of resources in FPGA devices. In addition, specific purpose architectures cannot afford floating point arithmetic when implementing long-datapath pipelined computing architectures that may have a large number of processing elements. In this case, a study has been performed using an analog method to the one used with Horprasert algorithm. We have measured the error between the results obtained with different bit-width configurations and the original floating-point approach. In order to evaluate the accuracy of each configuration, the Wallflower dataset [44] has been used.

Table 4.2 shows the main algorithm data structures (system registers) and the associated bit-width choices, fixed after this study.

Once we have established the bit-width of the fractional part of $v_i = \left(\bar{R}_i, \bar{G}_i, \bar{B}_i\right)$ and $\left(\check{I}, \hat{I}\right)$, we have also evaluated the degradation of our design combining all the modifications required to obtain a hardware-friendly model. These results are shown in section 4.4.

4.2.5 Hardware Architecture

An optimized hardware architecture for the Codebook algorithm is described in this section, following the same ideas that motivated the development of the Horprasert architecture, summarized in section 4.1.4.

The proposed architecture has been developed using EDK (Embedded Developer's Kit) and ISE Foundation of Xilinx Inc [20]. The EDK environment facilitates the design of complex and completely modular SoC architectures able to support embedded microprocessors (MicroBlaze, PowerPC. . .), peripheral and memory controllers (Ethernet, DDR2, ZBT...), interconnecting buses (PLB, NPI, MCH, FSL. . .) whilst IP cores for specific processing can be designed using HDL languages in the ISE tool. For a better understanding of the designed architecture using EDK, it is important to highlight the use of a ViSmart video processing board from Seven Solutions [104], including: two Xilinx XC3SD3400aFG676 FPGAs, two 256MB DDR2 DIMM memory modules, four independent analog video inputs, two gigabit ethernet connections, 485 connection, a 3G connection module, a 64 MB Flash memory, and two 1MBx36 bits ZBT memories. In our case, we have only used one of the FPGAs included in the ViSmart board. This architecture has already been described in Figure reffig:HorprasertEDK

The hardware description language that we have used to implement this IP core is ImpulseC [19], which allows us to work at a high level of abstraction, enabling the construction of a multi-stage pipelined architecture running in parallel. The parallel execution of these stages is the key point for the high performance obtained in our system. Figure 4.4 shows the proposed architecture for this IP core. Blob detection is performed in an analog way to the architecture designed for the Horprasert algorithm, which is explained in [105, 106]. It is important to remark that memory has a key role in the performance of the system and requires an efficient memory accessing scheme. The codebook model requires an intensive utilization of memory resources and poor system memory architectures drastically reduce the system performance. This has motivated the utilization of high performance multiport memory controllers (Xilinx MPMC for DDR2 and XPS_EMC_MCH for SSRAM) as well as very specific and optimized memory ports (NPI for DDR2 and MCH for SSRAM).

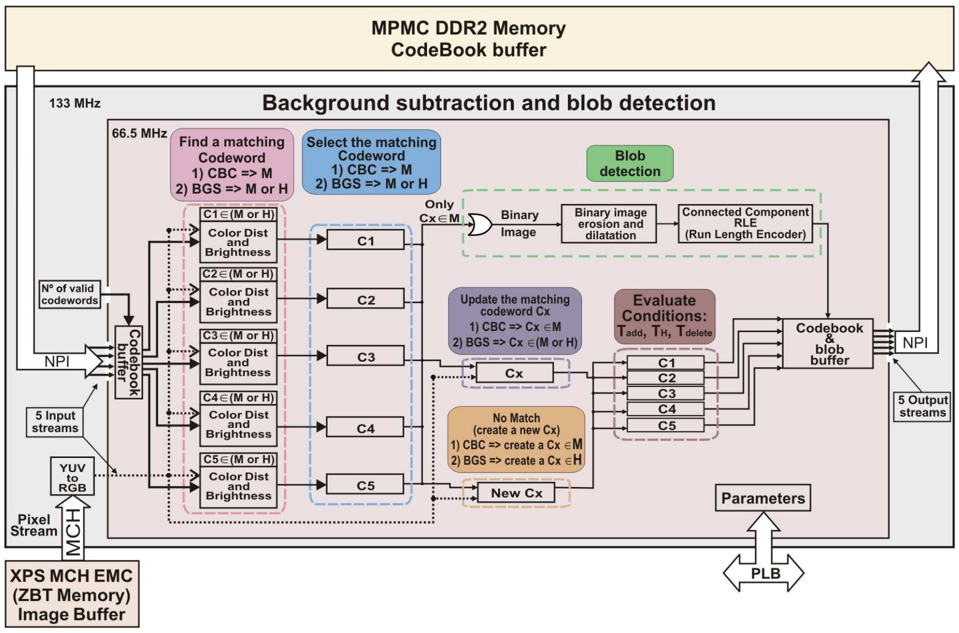

MPMC DDR2 Memory
CodeBook buffer
133 MHz
Background subtraction and blob detection
66.5 MHz
Find a matching Codeword 1) CBC => M 2) BGS => M or H
Select the matching Codeword 1) CBC => M 2) BGS => M or H
Blob detection
Only Cx∈M
Binary Image
Binary image erosion and dilatation
Connected Component RLE (Run Length Encoder)
Nº of valid codewords
Codebook buffer
NPI
5 Input streams
YUV to RGB
Pixel Stream
MCH
XPS MCH EMC (ZBT Memory) Image Buffer
C1∈(M or H)
Color Dist and Brightness
C1
C2
C3
C4
C5
Update the matching codeword Cx 1) CBC => Cx ∈M 2) BGS => Cx ∈(M or H)
Cx
Evaluate Conditions: Tadd, TH, Tdelete
No Match (create a new Cx) 1) CBC => create a Cx ∈M 2) BGS => create a Cx ∈H
New Cx
Codebook & blob buffer
5 Output streams
Parameters
PLB

Figure 4.4: Simplified datapath architecture for background subtraction and blob detection core with 5 scalar units (C1, C2, C3, C4, C5).

4.3 Datasets and evaluation metrics

As it has been mentioned in the chapter 3, there are many different types of algorithms in the literature. For that reason, it is important to evaluate the quality of the segmentation obtained by a new algorithm and to carry out a comparison with other background subtraction algorithms.

Basically, in order to evaluate the accuracy of a background subtraction algorithm by means of a quantitative analysis, it is required the use of a dataset with ground truth information, that is, the ideal segmentation that an algorithm should obtain. Regarding this kind of test sequences and datasets, there exist several approaches in the literature [44, 45, 46, 47, 48]. Toyama et al. [44] proposed the dataset known as *Wallflower*, which has been used to test and compare background subtraction techniques during the last ten years. In [45], Prati et al. focus on algorithms with shadow detection capabilities. They present a comprehensive survey of moving shadow detection approaches, proposing both novel quantitative metrics and video sequences with associated ground truth to evaluate these approaches. Most recently, several new datasets have been proposed to test background subtraction methods [46, 47, 48]. The dataset proposed by Brutzer et al. [46]

is an artificial dataset which is stated not to suffer from imperfect labels or small number of annotated frames.

In the work developed in this section we have used the sequences which belong to the dataset *Wallflower* [44]. This dataset has been widely used in the field of background subtraction techniques [34, 35, 37, 44, 52], and consists of a set of sequences in which each sequence presents a different issue. The performance of the algorithm is evaluated against a hand-segmented frame. The sequences are the following:

- Moved Object: a person enters a room, makes a phone call, and gets out of the room leaving the telephone and a chair in a different position. This sequence allows researchers to test the capability of the algorithm to adapt to movement in background objects.
- Time Of Day: this sequence shows an initially darkened room that gradually becomes more illuminated. While this happens, a person walks into the room and sits on a couch. This sequence aims to test the adaptation to gradual illumination changes.
- Light Switch: the scene begins with a room in which the lights are on during a period of time longer than the training stage. Later, a person enters the room and turns the light off for a long time. Finally, a person enters the room and turns the light on again. The evaluation frame takes place shortly after turning the light on, thus leaving the algorithm inadequate time to adapt.
- Waving Trees: a person walks in front of a tree which keeps moving, testing the capability of the algorithm to detect foreground over a non-static background.
- Camouflage: a person walks in front of a monitor which has interference bars on the screen. These bars have similar color to the person's clothes.
- Bootstrapping: the sequence consists of several minutes of a view of a cafeteria, with people moving in every frame. This sequence offers some complications for the training stage of the algorithm, since the training has to be performed in presence of foreground objects.
- Foreground Aperture: a person with uniformly colored clothes, who had been absorbed in the background model, wakes up and slowly begins to move.

Figure 4.5 shows the evaluation frames of the mentioned sequences, as well as the ideal foreground segmentation.

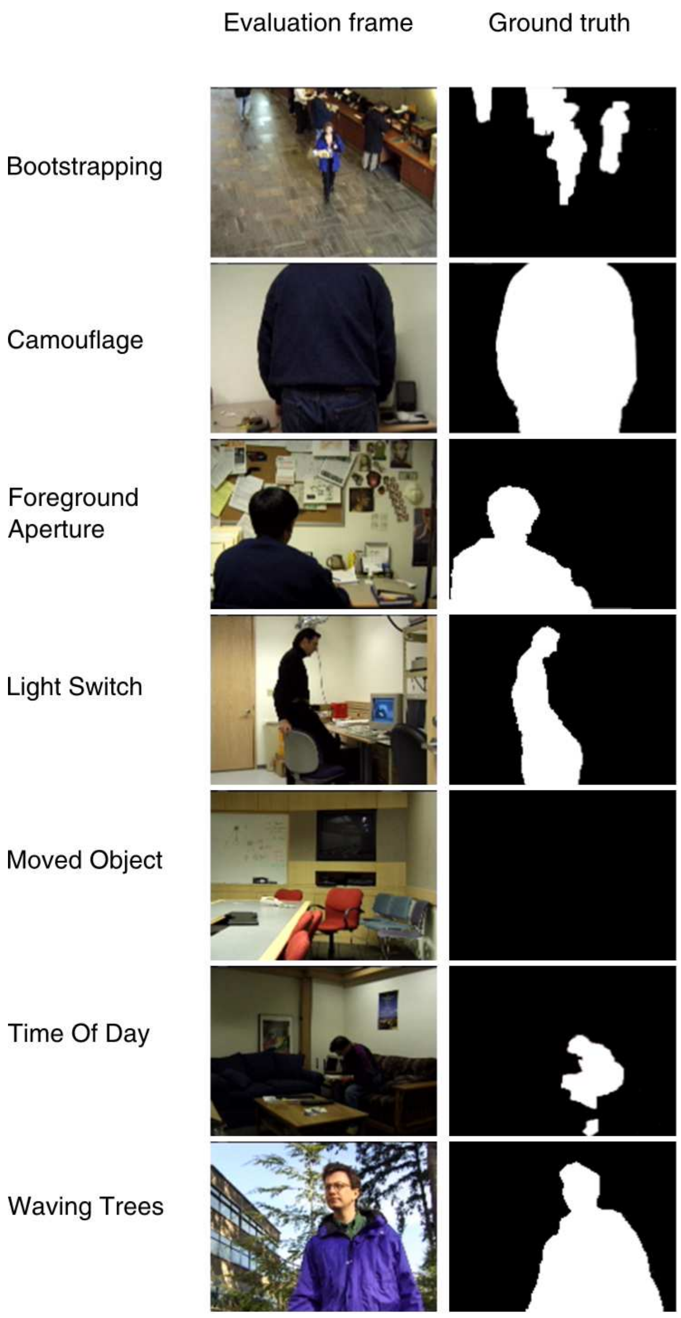


Figure 4.5: Evaluation frames of the Wallflower dataset. There is an evaluation frame with hand-segmented ground truth in each video sequence.

Since foreground/background segmentation is a two-fold classification problem, the quantitative results are based on measures related to True and False Positives and Negatives (TP, FP, TN and FN).

		Ground Truth	
		Positive	Negative
Segmentation	Positive	**TP**	**FP**
	Negative	**FN**	**TN**

Figure 4.6: Statistical parameters for evaluation

Although the amount of errors is a measure which allows evaluating the quality of an algorithm, it is an absolute measure and thus depends entirely on the sequence. In addition, it does not represent a reliable measure of the quality of the algorithm, since there are cases in which a few errors can be located in important areas of the image, leading to false alarms or misdetecting potential alarm situations.

For that reason, relative measures have been used to compare algorithms in different test sequences maintaining similar ranges of values. These measures include *Recall*, *Precision*, F_1 and *Similarity* [25, 108].

The use of *Recall* offers the ratio of correctly classified positives, defined as the number of true positives against the total number of positive pixels in the ground truth, following eq. (4.20):

$$R = \frac{TP}{TP + FN} \tag{4.20}$$

Precision is the ratio of correctly classified foreground pixels among all pixels labelled as foreground, as defined in eq. (4.21)

$$P = \frac{TP}{TP + FP} \tag{4.21}$$

These metrics, despite offering objective evaluation regarding the sensitivity of the algorithm to true positives and false positives, respectively, are not reliable separately. For example, an algorithm classifying every pixel as foreground would have maximum *Recall*, although with many false positives. For that reason, there are two accuracy metrics, F_1 and *Similarity*, which combine *Precision* and *Recall* to evaluate an overall quality of the segmentation.

$$F_1 = 2\frac{P \cdot R}{P + R} \tag{4.22}$$

$$Similarity = \frac{TP}{TP + FP + FN} \tag{4.23}$$

These measures offer a balance between the ability of an algorithm to detect relevant and non-relevant pixels and have been widely used in the literature [25, 109, 110, 108].

Although the presented metrics allow for evaluating the overall quality of the segmentation computed by a background subtraction algorithm, in order to evaluate the performance in presence of shadows two metrics have been proposed by Prati et al. [45], based on the work presented in [111]: the *shadow detection accuracy* η and the *shadow discrimination accuracy* ξ. These two metrics are defined as follows:

$$\eta = \frac{TP_S}{TP_S + FN_S} \tag{4.24}$$

$$\xi = \frac{\bar{TP}_F}{TP_F + FN_F} \tag{4.25}$$

where the subscript S stands for shadow and F for foreground. $\bar{TP}_F$ is the number of ground truth points of the foreground minus the number of points detected as shadows belonging to foreground objects. The first measure, the *shadow detection accuracy*, shows the capability of the algorithm to detect shadow points, or the low probability to misclassify a shadow point. The second measure shows the discrimination capability, that is, the low probability to classify a non-shadow foreground pixel as shadow.

In order to obtain the values for these metrics, a dataset needs ground truth information which classifies each pixel not only in foreground or background, but also shadows. We have tested the Horprasert approach using the *Intelligent Room* sequence provided by Prati et al. [45], which offers hand-segmented ground truth for most of the frames of the sequence, allowing researchers to get more meaningful results than a single value. Some frames of this sequence are shown in section 4.4.

4.4 Results

This section shows the results obtained by the proposed architectures. We have analyzed the performance of the algorithm in comparison with other hardware-oriented approaches, as well as an objective accuracy evaluation based on the Wallflower dataset [44].

4.4.1 Performance comparison with other approaches

It is important to compare the current implementation with other approaches described in the literature (shown in Table 4.3). In order to evaluate the processing speed, we use the MegaPixels per Second measure (MPPS), multiplication of image size by frame rate. The background subtraction algorithm by Horprasert [27] has been implemented by other authors [51], reaching 30 fps with resolution 240x120, i.e. 0.824 MPPS. Our architecture for Horprasert presents a large improvement over this performance (32.8 fps, 1024x1024, i.e. 32.8 MPPS), and we have implemented other features such as morphological filters, shadow detection, and a mechanism to send results through Ethernet gigabit. On the other hand, the architecture developed to perform background subtraction according to the Codebook approach [38] reaches 24 fps with resolution 1024x1024, obtaining much higher accuracy, as we show in section 4.4.2.

Other authors have proposed different approaches, as in [50] and [49] based on MOG (Mixture of Gaussians). Jiang et al. [50] reach 38 fps with resolution 1024x1024 by applying a compression scheme, but with a considerable loss of accuracy. The system proposed by Appiah et al. [49] performs 145 fps for 768x576 frames, but obtaining worse results in terms of accuracy than our presented approach (Section 4.4.2). Bravo et al. [43] implement PCA algorithm on FPGA, which performs at maximum between 190 and 250 fps for 256x256 frames depending on the number of significant eigenvectors, i.e. between 11.875 and 15.625 MPPS. However, due to the lack of accuracy information, a deeper comparison is not possible.

For standard GPU platforms, the approaches described in Carr [55] and Pham et al. [56] achieve high accuracy. Furthermore, Pham et al. [56] presents a high frame rate (980 fps, 400x300, i.e. 112.15 MPPS). The main limitation of our approaches with respect to other contributions based on MOG (Mixture of Gaussians) such as [56] is the accuracy of results, but on the other hand, GPU platforms have the problem of implementation for embedded systems especially in terms of portability, size and power consumption.

Other implementations have been proposed using TI DM642 DSP platform, as in [53]. This contribution is based on MOG (Mixture of Gaussians) and it has been implemented using fixed-point arithmetics. According to datasheet [112] and using the spreadsheet spra962f [113], we have calculated the power consumption of DM642 DSP, obtaining 2.5 W. We have assumed that this DSP run at 720 MHZ and a 80% CPU utilization. According to Table 4.3, this DSP is able to compute 2.03 MPPS and if we take into account its low power consumption (2.5 W), it has 0.8 MPPS per Watt. If we do the calculations for our systems (5.76 W for Horprasert and 5.13 W for Codebook), we achieve 5.7 MPPS per Watt and 4.67 MPPS per Watt respectively, therefore our FPGA-based systems have a better performance.

For some applications, these DSPs offer all the performance we need. In addition, DSPs enable rapid development of complex algorithms and are bet-

Table 4.3: Comparison with other previous approaches described in the literature.

Approach	Method	Resolution	Frame Rate	MPPS	Processor Type
Oliveira et al. [51]	Horprasert	240x120	30	0.824	FPGA
Jiang et al. [50]	MOG	1024x1024	38	38	FPGA
Appiah et al. [49]	MOC	768x576	145	61.18	FPGA
Bravo et al. [43]	PCA	256x256	190-250	11.875-15.625	FPGA
Carr, P. [55]	MOG	704x576	16.7	6.46	GPU
Pham et al. [56]	Zivkovic's Extended MOG	400x300	980	112.15	GPU
Ierodiaconou et al. [53]	MOG	352x288	21	2.03	DSP
Presented work	Horprasert	1024x1024	32.8	32.8	FPGA
Presented work	Codebook	1024x1024	24	24	FPGA

ter suited for low power applications, although they can only run up to four calculations at a time. On the other hand, when we need more performance for other applications (e.g background subtraction, image stabilization...), FPGAs are a good option, since they can perform mathematical operations in parallel at one time. Furthermore, FPGAs are excellent for glue logic, connecting multiple processing chips, peripherals and memories together. Therefore it is often better to use the FPGA as a coprocessor (video pre-processing functions) for a DSP. The integration of these two devices onto a single development platform can offer the best of both architectures by increasing performance and reducing overall cost [114].

Finally, note that the processing performance is directly determined by the running clock frequency, and we are using low cost FPGAs with a reduced performance compared to other FPGAs on the market. Therefore, migration to faster technologies as Virtex-6 or Virtex-7 FPGAs could directly represent an improvement of the system performance. An easy way to increase this performance in case of need could be simply replicate the processing cores and split the input image into a number of parts equal to the replication of cores, although the system would have a significant increment on price and power consumption.

4.4.2 Evaluation of the accuracy of the background model

Apart from the evaluation of system performance performed in the previous subsection, it is important to evaluate the quality of the segmentation obtained by the proposed architectures and to carry out a comparison with other background subtraction algorithms found in the literature. The algorithms which have been used for this comparison are MoG (Mixture of Gaussians) [33], a segmentation method based on Bayes decision rules [37], a simplification of MoG for FPGAs [49], and the original algorithms described in this chapter, that is, Horprasert [27] and Codebook [38]. These models have been selected since they represent different kinds of algorithms and they are among the most frequently used. The implementations of MoG and the Bayesian algorithm that have been used are versions from the OpenCV library, while the other approaches have been developed by ourselves from the information shown in their respective papers. In this section, the methodology used to compare the different approaches is presented. We have evaluated the general performance as a background subtraction algorithm of the two proposed approaches and the behavior in presence of shadows of the Horprasert implementation. The former has been performed by means of the dataset Wallflower [44], described in section 4.3, while the latter has been studied using the sequences presented in [45].

4.4.2.1 Background subtraction evaluation

The metrics and dataset used in order to evaluate and compare different background subtraction algorithms have been described in section 4.3. However, the test *Moved Object* cannot be evaluated using these metrics, since the ground truth does not have any foreground pixel. As a result, there are not True Positives (TP) or False Negatives (FN), being impossible to compute *Precision* or to get useful results from *Similarity*. For that reason, the performance in this test is only studied in a qualitative manner, by observing the resultant images (Figure 4.9).

Figure 4.7 shows the F_1 and *Similarity* values obtained by each algorithm in the dataset. From this figure, it can be seen that our proposed implementation for the algorithm by Horpraser [27] offers acceptable results, especially in comparison with the other hardware-oriented implementation [49]. Regarding the Codebook implementation, we can see that it gets very good results, being the best algorithm in two of the tests, and obtaining very high marks in the others. Concerning hardware approaches studied in this work, the accuracy decreases minimally so that they get slightly worse results than the original ones. However, the degradation caused by fixed-point limitations and additional modifications is fairly acceptable, obtaining our architectures very good results, specially the one based on the Codebook algorithm.

Besides the general comparison, it is interesting to see the accuracy of these algorithms in outdoor and indoor situations. Instead of constructing figures for all the tests of the benchmark, we have grouped the sequences

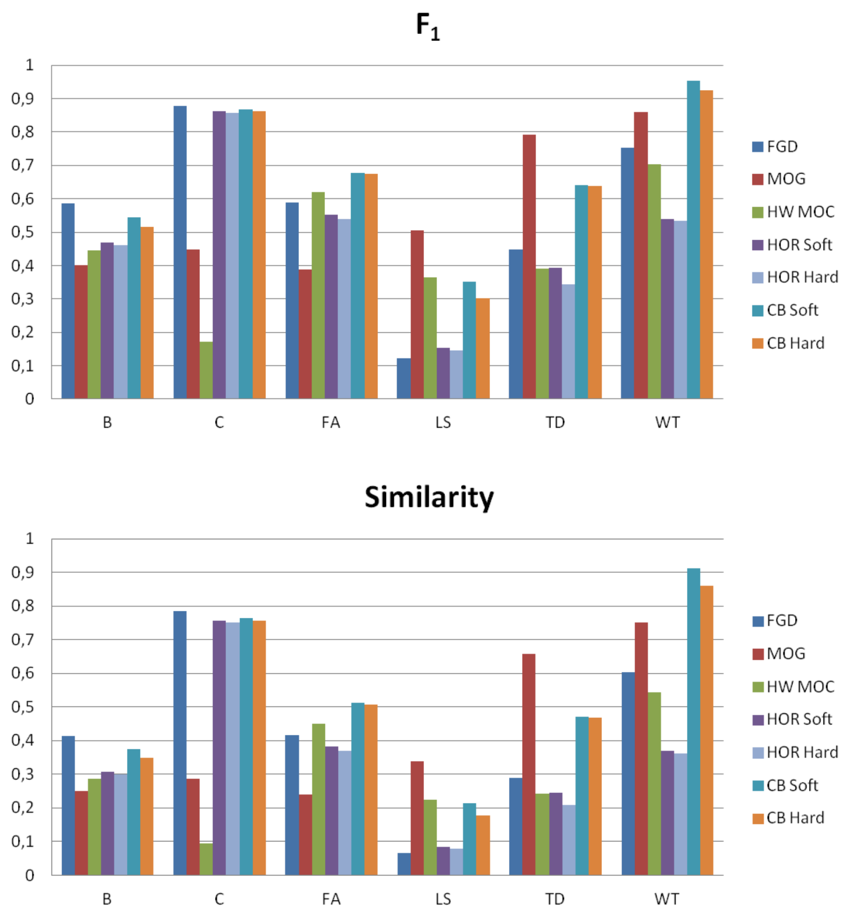
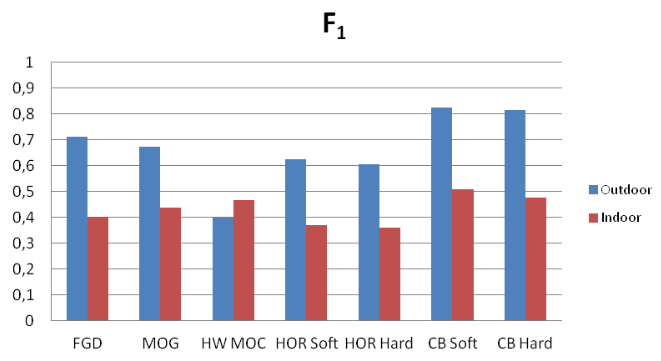


Figure 4.7: Overall performance evaluated using F_1 and *Similarity*. FGD is the Bayesian algorithm [37], MOG Mixture of Gaussians [33], HWMOC the FPGA implementation [49] for MOG, HOR Soft the approach by Horprasert [27], CB Soft [38] the original implementation of Codebook, and HOR Hard and CB Hard the proposed approaches.

in two groups according to the characteristics of each one, and the results are obtained as a weighted averages of the results of each test. In the indoor group, we have taken into account the sequences *Bootstrap*, *Foreground Aperture* and *Light Switch*. In the outdoor group, the sequences are *Camouflage*, *Time of Day* and *Waving Trees*.

Figure 4.8 shows that the quality of the segmentations provided by the proposed implementations are similar to the ones provided by the original algorithms. In the case of Horprasert implementation (*HOR Hard*), it performs better in outdoor situations than the previous hardware-oriented



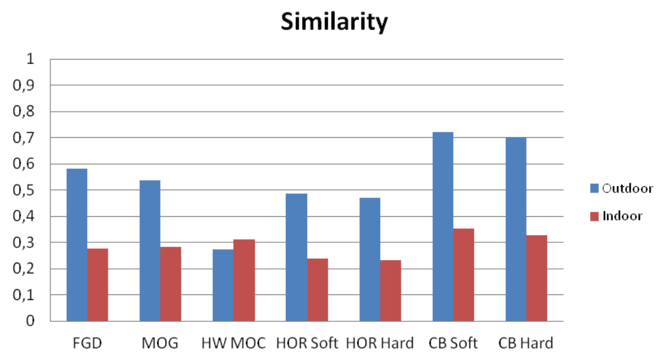


Figure 4.8: Performance in outdoor and indoor circumstances.

approach, while maintaining accuracy in indoor situations in an acceptable level. However, the proposed Codebook architecture (*CB Hard*) gets results similar to the obtained by Bayesian [36] and MOG [33] models, and it offers a great improvement over other hardware solutions.

In Figure 4.9, the evaluation images resultant from each algorithm on the Wallflower dataset [44] are displayed, as well as the original frame and the ground truth to evaluate the quality of the segmentation. This comparison allows us to see in a qualitative manner the foreground/background segmentation quality of the different approaches. This subjective evaluation supports the conclusions of the quantitative analysis. Regarding the *Moved Object* test, the Horprasert algorithm, due to its static nature, commits more mistakes than MOG [33] or HW MOC [49]. In the Codebook implementation, only a small regions of less than 10 pixels has been misclassified. In both cases, most of the misclassified pixels are sparse errors which could be removed by subsequent morphological filtering.

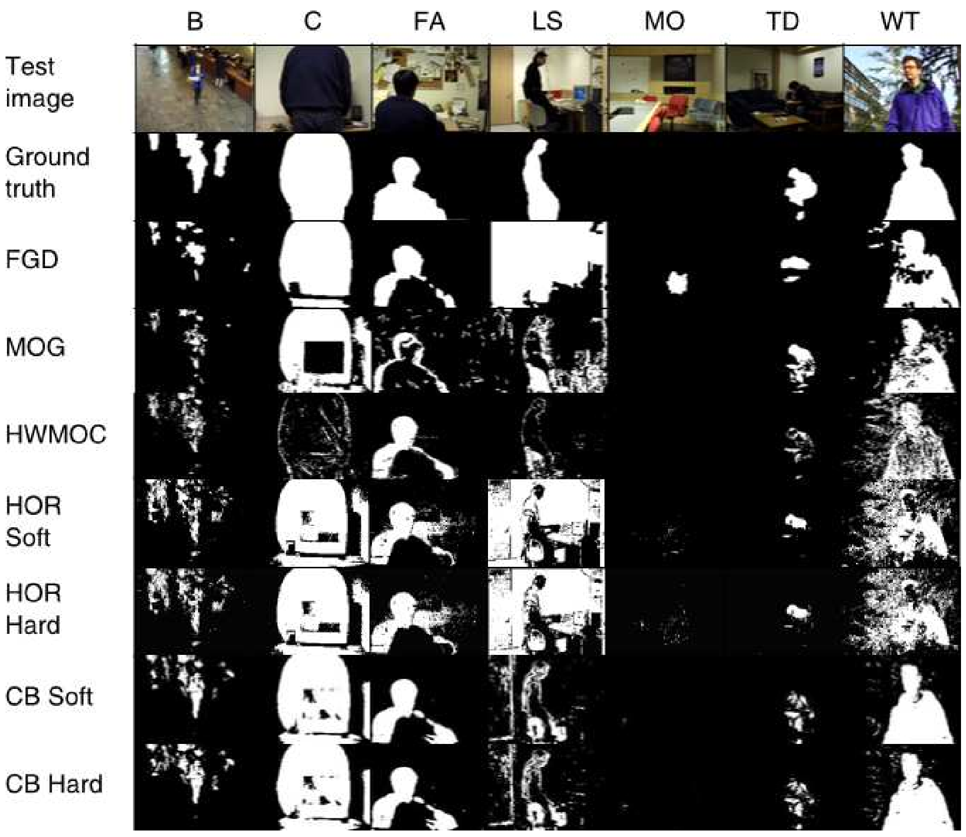


Figure 4.9: Wallflower evaluation frames, ground truth and resultant images from tested algorithms.

4.4.2.2 Shadow detection behavior

Despite the implementation proposed for the Codebook algorithm offers much higher accuracy than the Horprasert architecture, one of the benefits of the latter is that it is able to compute not only the background information of the scene but also information about the visible shadows. In order to evaluate these capabilities, we use the metrics proposed by Prati et al. [45], the *shadow detection accuracy* η and the *shadow discrimination accuracy* ξ.

Table 4.4 shows the results obtained by the proposed architecture and the original software implementation in the *Intelligent Room* sequence as well as the results from other approaches found analyzed in [45]. Despite the degradation suffered by the hardware implementation (mainly due to the utilization of fixed-point arithmetics), it offers acceptable results, considering the greater complexity of the other approaches that makes them unsuitable for FPGAs with limited resources.

Table 4.4: Shadow detection and discrimination accuracy, tested on *"Intelligent Rooom"* sequence. SNP (Statistical Non Parametric, Horprasert, our approach), SP (Statistical Parametric) [115], DNM1 [116] and DNM2 [117] (Deterministic Non-Model-based approaches).

Approach	**Intelligent Room**	
	η (%)	ξ (%)
SNP Soft	74.54	91.76
SNP Hard	71.14	88.13
SP	76.27	90.74
DNM1	78.61	90.29
DNM2	62.00	93.89

About the degradation between the software implementation and the proposed one, figure 4.10 shows the results for the *Intelligent Room* sequence during a series of evaluation frames. In the worst of the scenarios, the loss of accuracy due to the restrictions of the hardware implementations is limited to 5%, offering fairly good results in both detection and discrimination metrics.

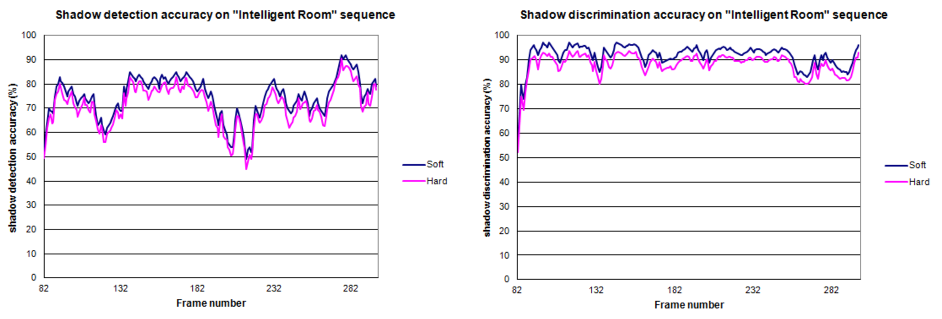


Figure 4.10: Shadow detection accuracy and discrimination accuracy of the original software model and the proposed approach, evaluated on *Intelligent Room* sequence.

This loss of accuracy can be easily seen in figure 4.11. Images a) and b) show the segmentation obtained by the software and hardware implementation respectively. The degradation is noticeable in the higher dispersion of the shadow points in the hardware detection, whilst the shadow regions resultants from the software implementation are denser. The same effect is shown in images c) and d), as well as some noise detected as shadows instead of be classified as foreground. However, the results are fairly accurate and the noise can be removed during the connected component stage, which was not included here in order to facilitate comparison with other approaches.

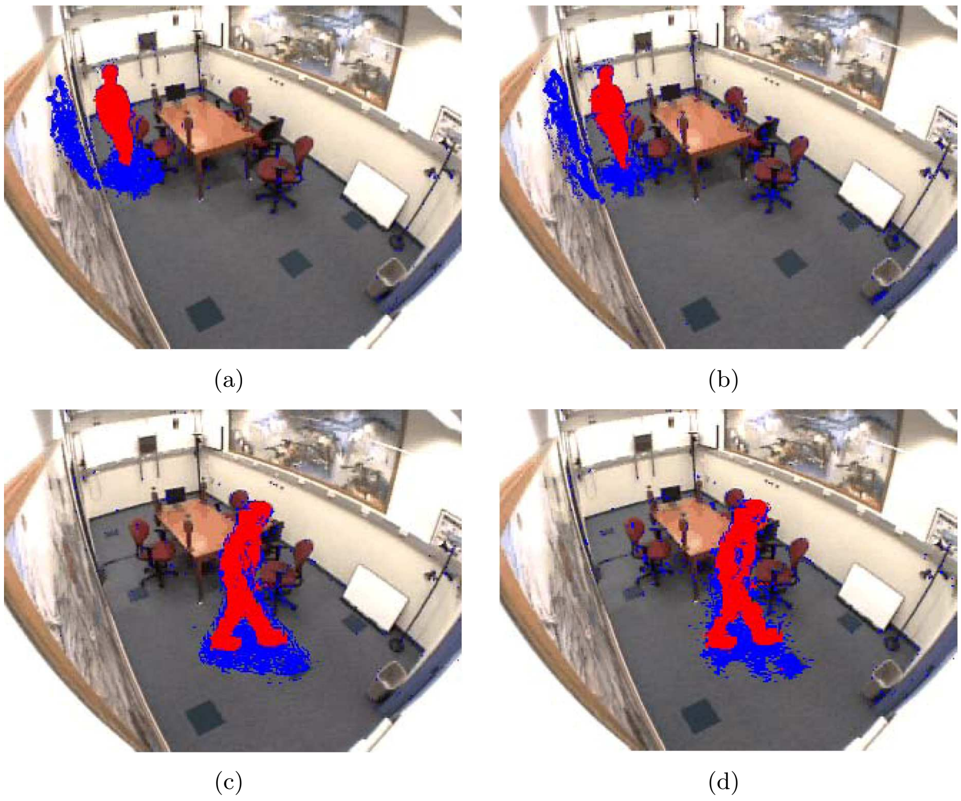


Figure 4.11: Frames of the *"Intelligent Room"* sequence, frame 100 for **(a)** software and **(b)** hardware implementations, and frame 282 for **(c)** software and **(d)** hardware.

4.5 Conclusions

We have designed and analyzed two architectures to perform background subtraction in video sequences. Our first approach is based on the algorithm by Horprasert [27], a static model whose simplicity allows for a low cost FPGA implementation and which has been extended to perform also shadow detection. The second one is based on the Codebook algorithm by Kim et al. [38], which allows us to model dynamic and multimodal backgrounds and is well known because of its robustness and good balance between accuracy and efficiency.

Two FPGA implementations of background subtraction algorithms have been studied which offer low degradation in comparison with the original ones. Since the hardware environment has limited resources, we have optimized memory access, low-level interfaces with external memory and storage of the background models. For the first time, an FPGA implementation of a background model includes shadow detection logic. This allows us to increase the model robustness as well as to improve object localization on the scene. This is a valuable contribution that significantly enhances the applicability of the proposed approach. The second approach, despite not having shadow detection capabilities, uses a much more accurate algorithm which can be used in a wider range of scenarios.

We have evaluated the approaches with the benchmark Wallflower [44] to test the quality of the segmentation. The first architecture offers good results (in terms of accuracy) in comparison with other hardware implementations found in the literature [49]. Furthermore, shadow detection behavior has been analyzed by means of manually segmented video sequences [45]. The implementation is able to segment objects in complex sequences with resolution 1,024 $\times$ 1,024 at 32.8 fps (therefore 32.8 MPPS, Megapixels per Second) or from up to four cameras with less resolution. This represents a speed up over 35$\times$ with respect to the other approach [51] based on Horprasert. Concerning the cost of the system, the architecture has been designed for low cost FPGAs Spartan-3 by Xilinx, and it offers low power consumption (5.76 W). Therefore we achieve 5.7 MPPS per Watt. Our approach can be included in embedded systems, where parameters such as size and power are key elements that are not achievable by other approaches, such as commodity processors or GPU-based systems.

Regarding the second architecture, the results are excellent in comparison with other hardware-oriented approaches found in the literature, being similar to the ones offered by advanced software algorithms such as Bayesian and MoG. The implementation is able to segment objects in complex sequences with resolution 768x576 at 50 fps (therefore 21.1 MPPS) or at a higher speed with less resolution sources. Concerning the cost of the system, the architecture has been designed for low cost FPGAs Spartan-3 by Xilinx, with an estimated power consumption of 5.13 W (4.11 MPPS per Watt).

The work performed in this chapter aimed to the first objective presented in Section 2.2, which is the development of background subtraction models in real-time and the evaluation of methods which offer good trade-off between accuracy and computational efficiency. The methodology we have used to validate our architectures and to compare them with the state-of-the-art methods has been followed by recent works related to background subtraction on embedded hardware [52].

Chapter 5

Background subtraction based on color and depth cues

Background subtraction, as mentioned in chapter 4, is a well-known technique which has aroused much interest as a research field. However, despite current state-of-the-art algorithms are able to cope with classic issues (such as sudden and gradual illumination changes, moving background objects, repetitive movements...), robustness is a critical requirement for video analytics. Most of the studied techniques rely specifically on color values [33, 38, 37, 2]. For that reason, they are prone to errors and misclassifications when foreground objects have colors similar to those of the background. In addition, although some approaches are specially suited to work in presence of shadows [27, 45], they suffer from it if these shadows are too dark, since the change of color in some image regions is greater than what the algorithms are capable to adapt to.

Currently, there is a clear understanding that, in order to achieve higher robustness, multimodal information fusion is required. For that reason, many works have proposed algorithms fusing intensity, edges and texture information [57, 58, 59, 60, 61, 36, 62, 29]. Even though these methods are a very valuable contribution for any robust surveillance application, the problems associated to the camera sensor still persist.

In order to reduce the impact of issues related to camera sensors, we focus on the combined use of depth and color. Depth is an interesting signal for segmentation that is less affected by the classic color segmentation issues, such as shadows or highlighted regions. Depth information can be obtained in real-time by different methods or technologies: stereo-camera setups with disparity estimation algorithms [118], Time-of-Flight (TOF) cameras [119] or the Kinect peripheral from Microsoft [21]. In this thesis project we have studied the usage of depth information provided by the Kinect sensor [21] as well as stereo cameras with disparity computation. The combination of depth and visual (RGB) sensing allows for more robust and accurate object detection.

The state of the art of background subtraction with sensor fusion has been summarized on section 3.1.2. In this thesis project, we have proposed a model which fuses color and depth, based on the Codebook method described by Kim et al. [38]. Depth is included in the background model, allowing for the combined use of depth and RGB.

The rest of this chapter is organized as follows. In section 5.1 we describe the different methods used to compute depth information. In Section 5.2 the fusion techniques used to integrate background and depth are explained. In Section 5.3 we describe the datasets proposed for the evaluation and comparison of the algorithms based on color and depth. Section 5.4 shows the experimental results obtained by the proposed methods. Finally, conclusions of this work are presented in Section 5.5.

5.1 Depth estimation

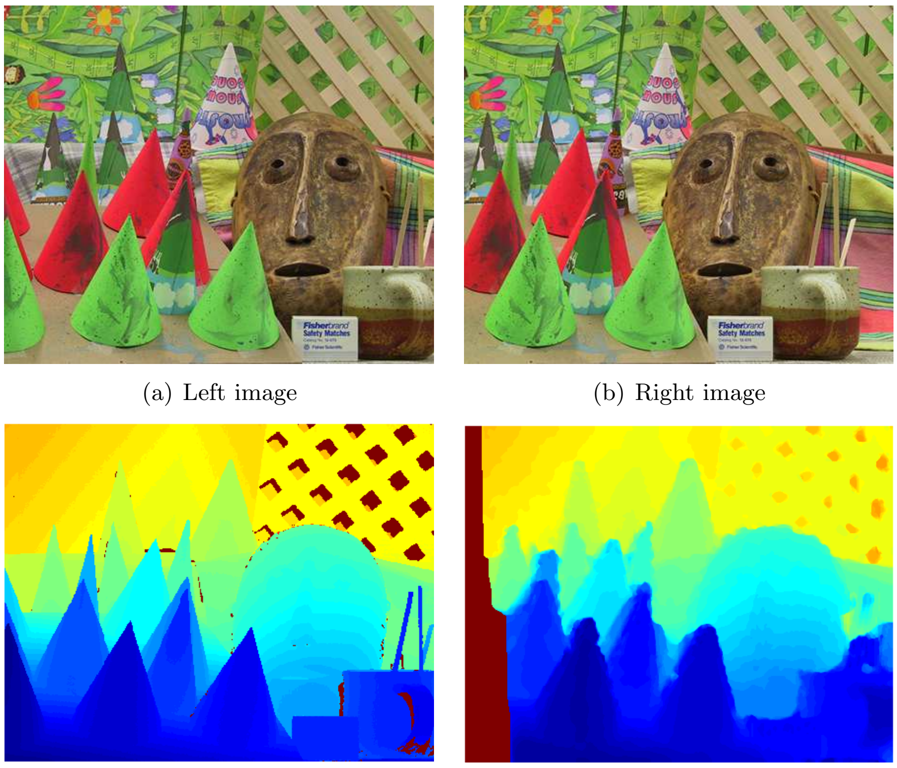
Depth information can be obtained by means of stereo camera rigs or active sensors such as Kinect [21] or Time-of-Flight cameras [120, 121, 119].

Depth cameras offer several advantages over traditional intensity sensors, such as performance under low light, being color and texture invariant and resolving silhoutte ambiguities in pose [122]. However, since they use infrared light to compute depth, they do not work outdoors. For that reason, we have studied the use of both depth cameras and stereo camera rigs for foreground segmentation.

Regarding stereo camera rigs, depth perception derives from the differences in the positions of points in correspondence between stereo images captured by a pair of binocular cameras. As a first approximation, these positions are related by a 1-D horizontal shift, taking into account that the pair of cameras differ only in their horizontal position. The disparity is related to the direction of the epipolar lines [123, 124, 125].

Accurate stereo matching is a required feature for many applications, like 3D reconstruction. Matching is a challenging task due to occlusions, object boundaries, lack of structure or repetitive textures. Therefore, many algorithms and improvements have been studied in the literature.

In the next subsections, we describe the different methods used in this work, including a brief explanation of the Kinect sensor [21] and different stereo algorithms. These algorithms are representative models of the most common approaches used for disparity estimation, i.e. coarse-to-fine local methods, global ones here represented by a variational technique and finally an intermediate local-global solution based on pixels correspondences. They have been chosen because each of them suffers from different constraints and limitations and this allows us to generalize the results to other algorithms. Of course, the quality of the final stereo algorithm has an impact on the final segmentation accuracy, but the motivation of the current methods selection is to provide an overview of the different possibilities and trade-offs. The choice of disparity algorithm should be made based on accuracy as well as additional considerations as real-time constraints or robustness to image artifacts.



(a) Left image (b) Right image

(c) Ground truth (d) Disparity

Figure 5.1: Disparity example: 5.1(a) and 5.1(b) are the left and right original images from Middlebury's *cones* sequence. 5.1(d) is the disparity calculated using [126] where warm colors codify farther objects and cold colors, closer objects.

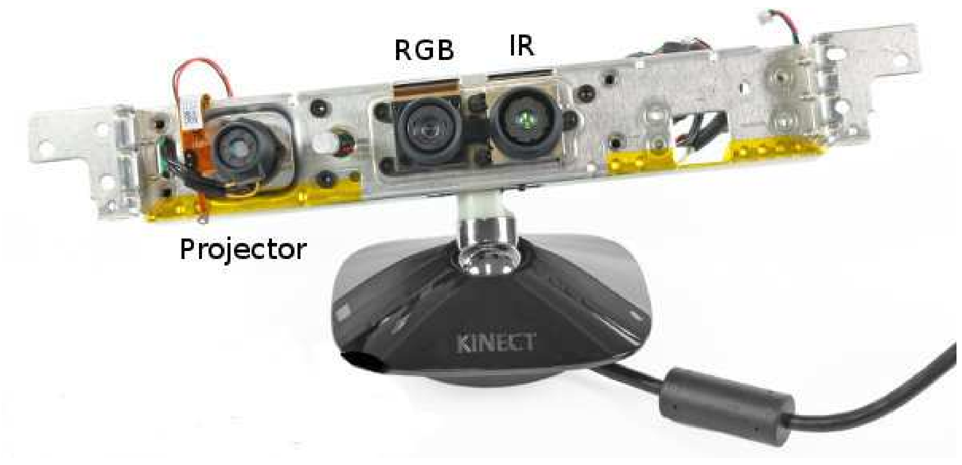


Figure 5.2: Kinect consists of infrared emitter, infrared sensor and RGB camera, as well as microphones and tilt motor. (Illustration from [129]).

5.1.1 Kinect sensor

The Kinect peripheral from Microsoft [21] is a composite device which has three openings on its front, each housing a core piece of technology [127]. On the left, there is an IR emitter which projects a factory calibrated pattern of dots. The second opening is an RGB camera. Finally, the third is the IR sensor which reads the dot pattern to triangulate points in space. A scheme of Kinect is shown in Figure 5.2. As a measuring device, Kinect delivers three outputs: IR image, RGB image, and Depth image [128].

Data from Kinect have been obtained by using OpenCV [17] and OpenNI drivers [18]. Since we are interested on the segmentation in the RGB video stream, all sensors in Kinect must be calibrated so that depth information can be directly applied to RGB images. In this work, we have used the default calibration offered by OpenNI [18] driver, which allows for a fairly good mapping. Both depth maps and RGB images are obtained at the default 640x480 resolution, working at 30 fps. The sequences recorded by using Kinect are explained in detail in Section 5.3.1.

Since Kinect depth estimation is performed by using infrared sensors, it does not work properly in several situations. For example, black objects do not reflect infrared light, being thus undetectable to the IR depth sensor. In addition, Kinect and ToF-cameras are not suitable for outdoor scenarios. For that reason, we have studied stereo disparity algorithms as well as active sensors.

5.1.2 Phase-based method

This method is based on a local technique with a coarse-to-fine refinement to extend the depth range. It measures the disparity in terms of phase differences in the output of local, band-pass, linear filters applied to the stereo image pair [130]. One important advantage of phase-based approaches is that the estimation of disparity is obtained with sub-pixel accuracy. For that reason, these techniques do not require an explicit sub-pixel signal reconstruction of feature detection and localization. In addition, phase-based approaches are more robust to image deformations typically present between left and right views. In particular, phase information is stable under small scale changes and contrast variations. Despite the general stability of phase information, there are regions which are prone to errors near phase singularities where disparity estimations should not be trusted. However, these regions can be detected through simple constraints and thresholds in order to discard unreliable measurements at the cost of producing a sparse disparity map.

The algorithm used in this work is based on the computing model proposed by Solari and Sabatini [131] and on its multi-scale extension proposed in [132]. Phase measurements can be obtained by filtering operations with Gabor filters.

The basic steps of the mono-scale computation can be summarized as follows:

- Even and odd (C, S) filtering with quadrature filter image pairs.
- Disparity computation at eight orientations and threshold assuming $k(x) \approx k_0$.
- Final disparity estimation chosen as the median value of the different orientations.

This mono-scale version is extended by using a coarse-to-fine approach as proposed in [132]. To perform this strategy efficiently, an image pyramid is used where the resolution is halved at each level. Specifically, a coarse-to-fine Gaussian pyramid is constructed [133], [134] in which each layer is separated by an octave scale. The image is blurred with a Gaussian kernel and sub-sampled to build each level. At each pyramid level k, resolution is halved with respect to level $k-1$, reducing the disparity range and enabling the filters to tune their response. The control strategy starts at the lowest resolution and uses the mono-scale method at each level. The stereo estimation obtained is used to warp the images at the next level in order to remove the estimated disparity, so that the residual disparity is within the range of the filters for the next level. The stereo algorithm manages strictly local information, what makes it particularly suitable for this warping strategy. After the disparity has been computed in every level of the pyramid, disparity values are transformed from the low resolution levels and propagated to the next ones. This procedure is repeated until the original resolution is obtained.

The multi-scale extension enables the method to cope with higher disparity range. In addition, the multiple orientation scheme allows it to optimize the phase estimations, what leads to accuracy increase. The main limitations of the approach are caused by its local nature, that frequently leads to wrong values specially at object boundaries, therefore reducing the overall method accuracy.

5.1.3 Variational approach

An example of a global method is presented here based on a variational approach. Variational methods are robust and relatively well performing, due to the incorporation of a regularization term. In addition, these methods have a solid mathematical base with existing efficient solvers, what allows researchers to straightforwardly extend the algorithms [135], [136], [137]. In addition, these kinds of methods enable the inclusion of both spatial and temporal constraints in the framework [126]. These constraints bound the solution based on a priori information, that is, based on what is known of a possible solution. For example, in real image sequences, there are some assumptions that can be made about the solution: the sky is far from the cameras (i.e. disparity near zero), the ground is a relatively flat surface, and in the case of automatic video surveillance, background structures tend to be flat and static. In the variational disparity calculation, the model minimizes the energy functionals given in (5.1):

$$\begin{aligned} E(d) &= \int_\Omega \left(D\left(I_L, I_R, d\right) + \alpha S\left(\nabla I_L, \nabla d\right)\right) dx + \\ &+ \gamma_S \int_\Omega \left(C_S\left(d_{SC}, d\right)\right) dx \end{aligned} \tag{5.1}$$

where the data term is $D\left(I_{L,1}, I_{R,1}, d\right)$, while $S\left(\nabla I_{L,1}, \nabla d\right)$ is the regularization term, $\alpha > 0$ is the weight of the smoothness term, and d is the disparity. The spatial and temporal constraint is $C_S\left(d_{SC}, d\right)$, where d_{SC} is the constraining value and γ_S is the spatial and temporal constraint weight. More information about how data terms, regularization terms and spatial and temporal constraints are obtained can be found in [138].

The energy functional (5.1) is minimized using the corresponding Euler-Lagrange equation: a necessary (but not sufficient) condition for a minimum is for the Euler-Lagrange equation to be zero. Because of the late linearization of the data term the model is able to cope with large displacements. However, as a consequence of this, the energy functional is not convex. Due to the non-convexity, searching for a suitable minimizer becomes more difficult. A suitable minimizer is searched for using a coarse-to-fine strategy, while non-linearities are dealt with using a lagged diffusivity fixed point method [137]. The solver used for the linearized version of the equation is ALR (Alternating Line Relaxation) which a Gauss-Seidel type block solver [135].

5.1.4 Semi-global block matching

The Semi-Global Matching (SGM) method [139] combines a local matching based on mutual information with a semi-global optimization that is computed using a cost aggregation function from, typically, multiple directions through the image.

Mutual information has the advantage to be insensitive to radiometric problems and due this it has been used as matching cost. This cost function relies on the idea that, for well registered images, the joint entropy is low, since one image can be predicted by the other, which corresponds to low information and translates on a higher Mutual Information. Therefore, estimation of the disparity values consists of finding the function that, after warping one image towards the other, maximizes their mutual information.

In order to achieve a good matching without using any additional method, the authors use a coarse-to-fine strategy. This allows researchers to hierarchically estimate the best matching pixel for each image scale and to refine the process at the next level. In addition, pixel-wise cost calculation is generally ambiguous and wrong matches can easily have lower cost than correct ones, due to noise, etc.. For that reason, an additional constraint is added that supports smoothness by penalizing changes of neighboring disparities. As important differences with the variational method, this approach uses a new matching approach based on the idea of aggregating 1D directions equally instead of performing a global optimization. The aggregated (smoothed) cost for a pixel and disparity is calculated by summing the costs of all 1D minimum-cost paths that end in pixel at given disparity. This technique leads to efficient implementations with almost no accuracy degradation.

Moreover, the method uses some additional disparity refinements as peak removal, intensity-consistent disparity selection or discontinuity preserving interpolation. The final approach is among the most accurate approaches for stereo disparity computation and it is very efficient, which has motivated some real-time implementations as [140, 141].

5.2 Fusion model

The fusion of background subtraction models with stereo models for disparity computation has been previously studied by Gordon et al. [65], improving the performance obtained by each separate technique. In [65] a 4-channel background subtraction algorithm based on an unimodal mixture of Gaussians [33] is proposed.

In our contribution we have studied three fusion methods which can be used cumulatively and differ on the integration of depth information with RGB values. Our approach consists of an update of the model proposed by Gordon et al. [65], although a 4-channel (R, G, B, Z) Codebook has been used. The first fusion method simply considers depth as the fourth channel of the Codebook, which has a completely independent mechanism than color and brightness. The second one is based on this approach, but the distance in chromaticity associated to a pixel is biased by the depth distance. The third approach has been applied as a post-processing stage after the second one, and consists of performing morphological reconstruction of the foreground mask obtained by the previous method, using the output from the color-based Codebook algorithm. These approaches are further elaborated in following subsections.

5.2.1 4D Codebook: *CB4D*

Our first approach to RGB-D background subtraction is the generalization of the Codebook model proposed by Kim et al. [38], described in Section 4.2, to work with depth values as a fourth channel. The *4D Codebook* works enhancing the matching conditions between an input pixel value and a codeword. In the original algorithm, the pixel value matches the codeword if both color and brightness distortions are below a threshold ((4.14) and (4.16)). Our approach includes a third condition based on depth. Since depth information is 1-dimensional, we have considered the evaluation of matching between the pixel value and the background model using a method similar to the brightness condition.

$$\begin{aligned} D_{low} &= \alpha\hat{D} \\ D_{hi} &= min\left\{\beta\hat{D}, \frac{\check{D}}{\alpha}\right\} \end{aligned} \tag{5.2}$$

In (5.3), we obtain a range of values $[D_{low}, D_{hi}]$ which represents the depth change allowed for input values. D_{low} and D_{hi} are computed from $\check{D}$ and $\hat{D}$, which are the min and max depth values for a codeword. These two values are added to the 6-tuple described in the original model (Section

4.2.1). α and β define the threshold in the depth distortion, being typically α between 0.4 and 0.7, and β between 1.1 and 1.5. The logical disparity function is defined as follows:

$$disparity\left(D, \left\langle \check{D}, \hat{D} \right\rangle\right) = \begin{cases} \text{true} & \text{if } \neg Valid(D) \vee \\ & \vee\, (D_{low} \leq D \wedge \\ & \wedge\, D \leq D_{hi}) \\ \text{false} & otherwise \end{cases} \tag{5.3}$$

When color, brightness and disparity distortions have been computed, the algorithm matches the current pixel value with the appropriate codeword based on these conditions. A pixel is then classified as foreground or background as in (5.4):

$$BGS(x) = \begin{cases} \text{BG} & \text{if } (colordist\,(x_t, v_i) < \epsilon) \wedge \\ & \wedge\ brightness\left(I, \left\langle \check{I}, \hat{I} \right\rangle\right) \wedge \\ & \wedge\ disparity\left(D, \left\langle \check{D}, \hat{D} \right\rangle\right) \\ \text{FG} & otherwise \end{cases} \tag{5.4}$$

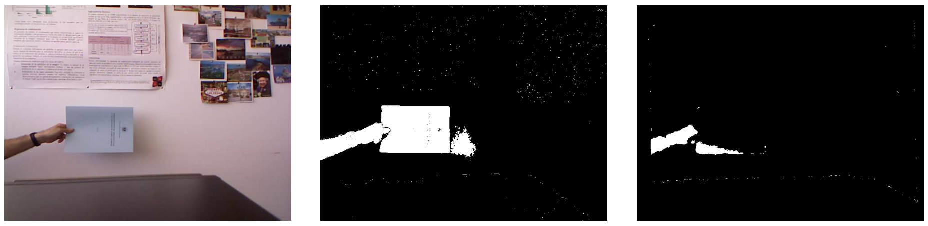
According to (5.3), the condition $disparity\left(D, \left\langle \check{D}, \hat{D} \right\rangle\right)$ is always true if the depth value of the pixel obtained by the Kinect sensor is *invalid*. Therefore, when the depth value is *invalid*, the condition required in (5.4) depends entirely on $colordist\,(x_t, v_i)$ and $brightness\left(I, \left\langle \check{I}, \hat{I} \right\rangle\right)$, relying the foreground/background classification on the color-based background model.

5.2.2 Early Fusion: Depth-extended Codebook (*DECB*)

As mentioned in the introduction of Section 5.2, a second approach has been tested, based on the *4D Codebook* described in the previous subsection.

The main motivation of this approach is improving the robustness of the *4D Codebook* to shadows, highlighted regions and sudden illumination changes. Depth computation sensors are more robust to lighting artifacts and shadows than passive sensors such as cameras, since they work at the infrared range without interferences with visible light. For that reason, instead of simply considering depth as an independent fourth channel, deeper dependence between RGB and depth has been studied.

The most straightforward method to remove shadows and highlighted regions will be not considering color distortion if the pixel is background according to depth information. However, Fig. 5.3 shows a scenario where



(a) Original frame (b) RGB Detection (c) Depth Detection

Figure 5.3: Example of complicated scenario for RGB-D methods: presence of shadows and flat foreground objects. Foreground objects are correctly detected by color-based algorithms while are misdetected by depth-based ones, since the objects are too close to the wall to be discernible.

this approach would produce misdetections due to the presence of foreground objects with similar depth to the background.

Our approach consists of biasing the threshold for color distortion depending on the depth information. When the condition $disparity\left(D, \left\langle \check{D}, \hat{D} \right\rangle\right)$ is fulfilled, a second threshold ϵ_2 for color distortion is used. This threshold has a fixed value depending on the original one, $\epsilon_2 = 1.8\epsilon_1$. Thus, the matching condition regarding color distance remains as follows in (5.5):

$$\begin{aligned} color(x) = \quad & colordist(x, c_m) \leq \epsilon_1 \vee \\ & \vee\ (\epsilon_1 < colordist(x, c_m) \leq \epsilon_2 \wedge \\ & \wedge\ disparity\left(D, \left\langle \check{D}, \hat{D} \right\rangle\right)) \end{aligned} \tag{5.5}$$

Equation (5.5) can be interpreted in the following way: if an input pixel is considered to be foreground (based on color), but it is close enough to the threshold, the classification will take into account the knowledge about the depth value for that pixel.

This modification will produce less foreground pixels than the *4D Codebook*, being most of the removed pixels false positives in the original model. Section 5.4 shows the experiments performed and the results obtained with both RGB-D algorithms as well as the color-based Codebook.

5.2.3 Late Fusion: *DECB-LF*

The third approach consists of refining the foreground mask obtained by the *Depth-Extended Codebook* algorithm (DECB) using the output of the color-based algorithm. This approach has been specially designed for the usage of depth information from stereo-cameras and disparity estimation algorithms.

Since disparity images tend to have more noise than color ones, we have evaluated a fusion method that reduces the impact of that noise in the resultant segmentation without having to perform aggresive erosion or small region suppression.

Our assumption is that the Codebook model over color and brightness is less affected by noise in isolated pixels thanks to the uniformity of image regions. In this approach, the methods used to obtain the range information are based on color correspondences between cameras. Possible disruptions and noise in the original images will affect both the color-based Codebook model and the disparity algorithms. In addition, the errors obtained in the disparity algorithm will be propagated to the Codebook model over disparity. This two-stage error propagation leads us to consider the color-based algorithm alone more reliable than a method based only on the disparity. Nevertheless, depth cues still provide very relevant information, as we show in Section 5.4.

An example of complicated scenario for disparity computation algorithms is shown in Figure 5.4. Since disparity is computed from color correspondences, it is affected by events such as flickering lights, overexposed and uniform regions, or the presence of many levels of depth. Figure 5.4 shows that this type of scenario requires the use of more advanced and accurate disparity computation algorithms such as the variational [126], leading to higher computational costs of the total system.

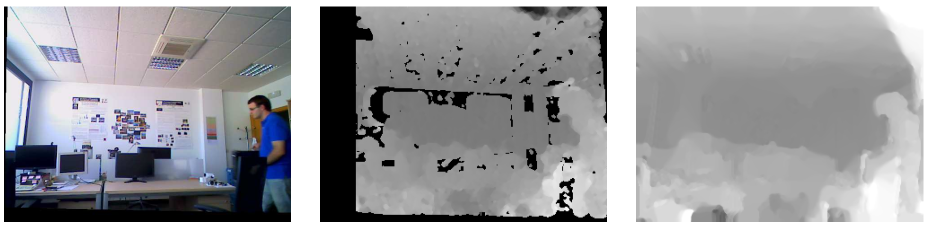

(a) Original frame (b) Phase disparity (c) Variational disparity

Figure 5.4: Scenario which offers different kinds of complications for disparity computation algorithms: flickering lights, overexposed regions and different range levels. These difficulties result in some local errors in 5.4(b), with a fair amount of invalid pixel values, as well as noise in planar surfaces. This scenario requires more accurate methods such as Variational [126], shown in 5.4(c).

In order to remove foreground pixels caused by noise from disparity from the mask obtained by the Early Fusion method, we perform a morphological reconstruction [142] of this mask taking the output of the color-based Codebook algorithm as the marker.

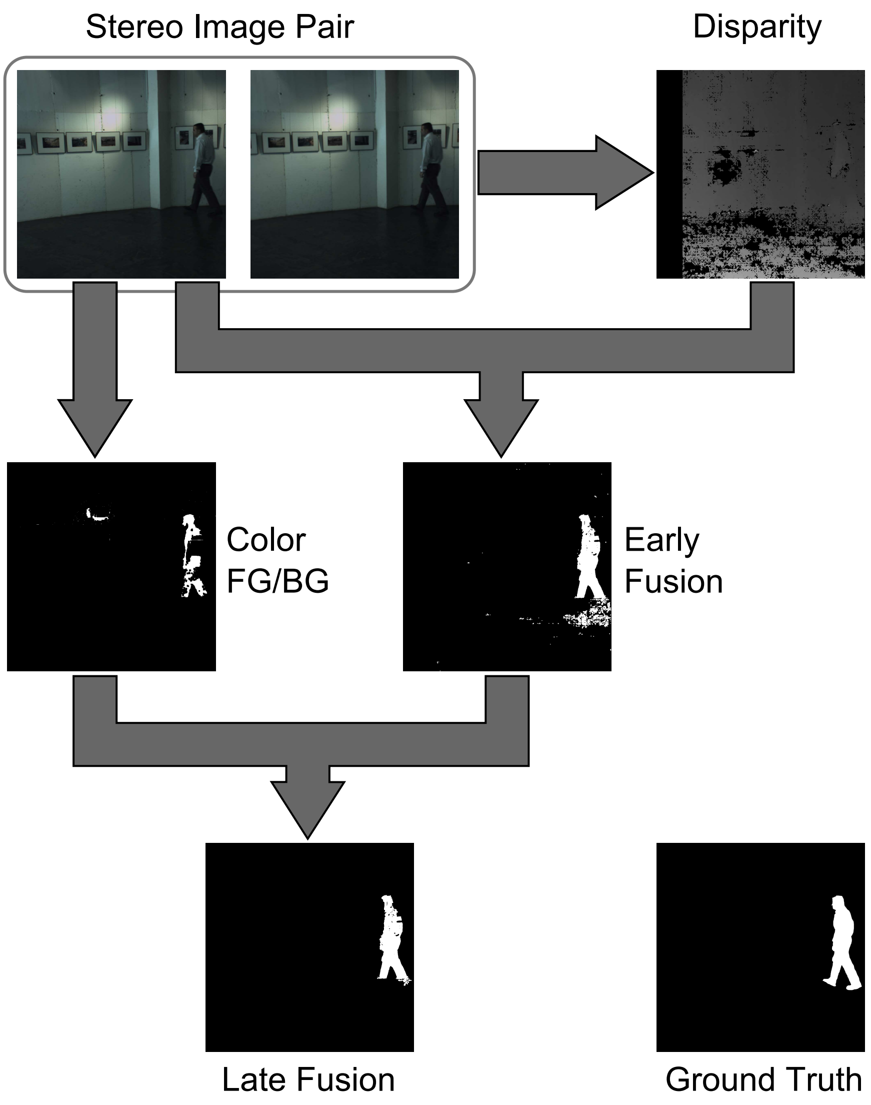


Figure 5.5: Complete scheme of masks fusion process.

Fig. 5.5 shows the complete process from stereo image pairs until the resultant *Late Fusion mask*. Foreground masks are obtained by the classic color-based algorithm and the proposed *Depth-Extended Codebook* approach. Then, morphological reconstruction is performed over the *Early Fusion mask* using *Color FG/BG* mask as a marker. This reconstruction produces a binary image with the connected components in the *Early Fusion mask* which partially overlap with foreground regions in the marker. As a result, most of the false foreground regions due to noise in disparity data are removed from the final segmentation.

This approach is specially aimed to the combination of background subtraction techniques with faster and less accurate disparity computation algorithms, such as the Phase-based approach used in this work [143]. The use of this kind of algorithms is justified by the need of real-time video surveillance systems, where the computational costs associated to each stage of the final system are quite constrained.

5.3 Datasets

As explained in the chapter 4, in order to evaluate objectively background subtraction algorithms by means of a quantitative analysis, we require the use of a dataset with ground truth segmentation. There are different benchmarks used for evaluation of background models [44, 45], but they do no have available information about depth. Thus, since we are focused on the use of Kinect [21] and stereo-cameras with disparity estimation algorithms, we have recorded and manually segmented some sequences by using these sensors. In the following subsections, we describe the two datasets recorded.

5.3.1 Kinect-based dataset

Data from Kinect have been obtained by using OpenCV [17] and OpenNI drivers [18]. The recorded sequences have been made publically available at [144]. The sequences are the following:

- *ChairBox*: a person enters the field of view and leaves a box on a chair. There are flickering lights as well as areas where depth cannot be obtained by the Kinect.
- *Wall*: a flat object (paper sheet) appears close to a wall, creating shadows and highlighted regions. The main difficulties are the similarity of depth between foreground and background and the changes of lighting.
- *Shelves*: a person enters the scene and puts two objects on shelves. There are changes of exposure as well as difficult depth estimation.
- *Hallway*: sequence recorded aiming to a hallway. There are reflections, complicated lighting, objects similar to background, and sudden illumination changes.

Figure 5.6 shows some frames from the four described sequences. We have hand-segmented the ground truth for several frames in each of these

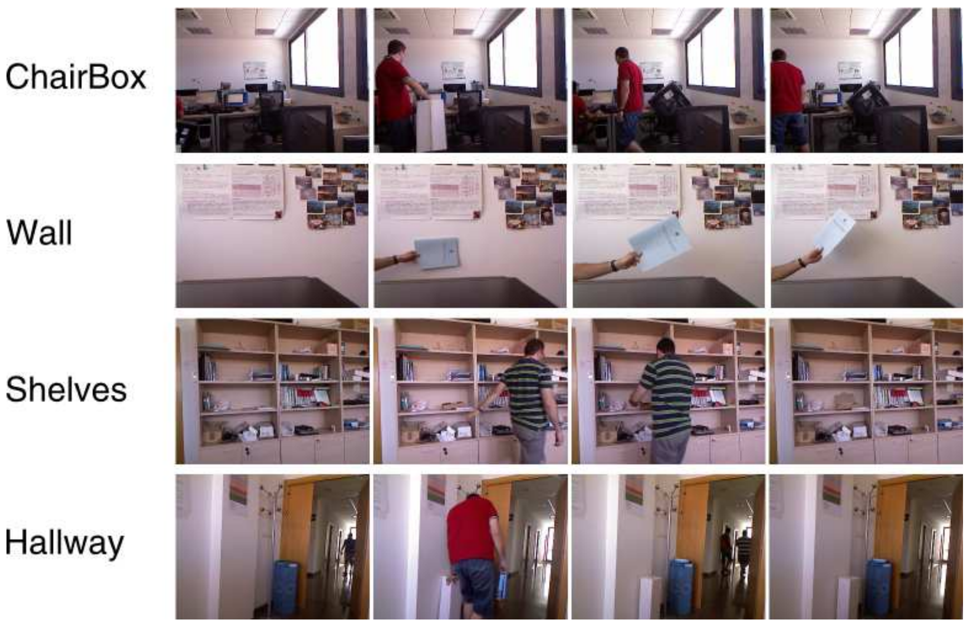


Figure 5.6: Samples of frames from the proposed dataset based on Kinect.

sequences. More specifically, there are four hand-segmented frames in *Chair-Box* sequence, five in *Wall* sequence, five in *Shelves* and seven in *Hallway* sequence. Depth sensor is calibrated so that the information can be applied to the corresponding pixels in the RGB image.

5.3.2 Disparity-based dataset

In addition to the usage of Kinect [21] peripheral, depth information can be obtained by means of stereo cameras. Regarding contributions that aim to fuse the color and range information, in [66] a set of stereo sequences has been proposed in order to evaluate the quality of the segmentation. However, these sequences are not suitable for video surveillance applications, since they are aimed to videoconferencing. Therefore, there are people present in the foreground during the entire sequences, what makes these sequences unsuitable for background subtraction algorithms, since the background is not visible in most of the frames.

To the best of our knowledge, there are no video sequences available in the literature that offer the required information. For that reason, we have recorded and manually segmented some sequences which allow us to numerically evaluate the different approaches. The goal is to show that color information is not enough to achieve a good foreground/background estimation but it could be significantly improved by the utilization of the approaches here proposed.

The dataset has been recorded by using two Philips SPC1300NC cameras working at 30 fps. The stereo-camera rig has been calibrated to rectify the distortion produced by the camera pair and to enable proper stereo disparity estimation. These sequences are publically available at [144]. They are the following:

- *Suitcase.* In this sequence, a person enters the scene with a suitcase and leaves it on the floor. The main difficulty of the sequence is the low lighting and color saturation, as well as the similar color between the suitcase and the floor.
- *Crossing.* Two people walk in and out of the camera field. The dark floor complicates the detection based on color when they get close to the camera, while the range is less useful when they get close to the wall.
- *LCDScreen.* A person walks into a lab and deposits a black box in front of a black LCD screen. In addition, there are flickering lights in the ceiling.
- *LabDoor.* A person walks in and out of the camera field, projecting shadows on background objects. In addition, there are occlusions due to background objects, flickering lights in the ceiling and sudden illumination changes.

As shown in Figure 5.7, each sequence presents important problems that are typically challenging for background subtraction algorithms. Since the quantitative evaluation is really important in order to compare models objectively, we have hand-segmented the ground truth for several frames in each of these sequences. More specifically, there are four hand-segmented frames in *Suitcase* sequence, six in *Crossing* sequence, three in *LCDScreen* and seven in *LabDoor* sequence. All sequences have been recorded with stereo cameras to compute depth information. Cameras have been calibrated and each pair of images has been rectified as a required stage prior to accurate disparity computation. The ground truth has been segmented in the left images of the corresponding frames.

5.4 Results

In this section, we analyze the differents studied fusion methods by evaluating them with the proposed datasets. In order to evaluate these models,

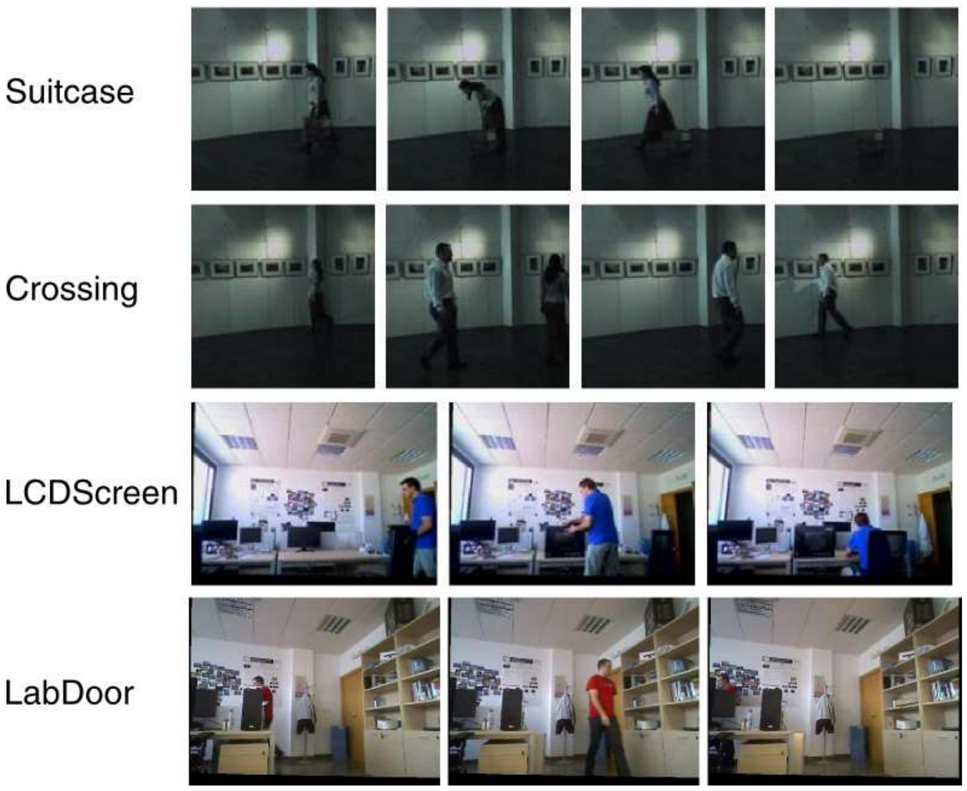


Figure 5.7: Samples of frames from the proposed dataset based on disparity.

we have used the same metrics proposed for the comparison of background subtraction algorithms in previous chapter, that is, *Recall*, *Precision* and F_1 measures.

Each of the evaluations we have performed compares two of the proposed fusion models. The tests run over the Kinect-based dataset involve the *4D-Codebook* (*CB4D*) and the *Depth-extended Codebook* (*DECB*). On the other hand, based on the results obtained with this dataset, our evaluation of fusion methods using disparity maps has been carried out on *DECB* and its *Late Fusion* variation, *DECB-LF*.

5.4.1 Evaluation of algorithms using Kinect

By using the sequences described in Section 5.3.1, five different approaches have been studied and evaluated. These approaches are the following ones: a 4D version of MOG based on the implementation proposed by Schiller et al. [67] (*MOG4D*), the original color-based Codebook (*CB*), the Codebook based only on depth (*CB1D*), the *4D Codebook* (*CB4D*), and the *depth-*

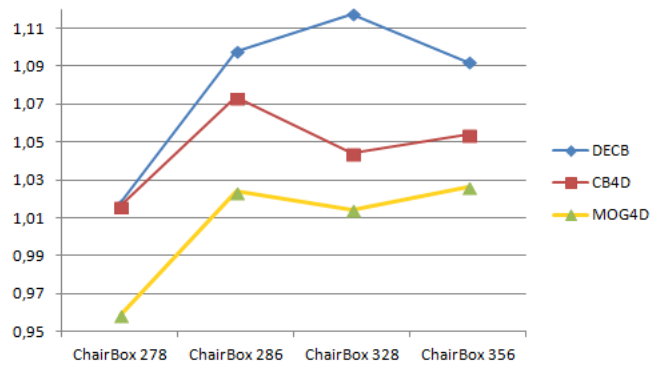


Figure 5.8: F_1 gain over standard (color-based) *CB* obtained from the test *ChairBox*. The gain using *CB1D* is not displayed in this dataset because it is very unstable and it reduces the readability of the comparisons.

extended Codebook (*DECB*). The tested version of *MOG4D* differs slightly from the proposed by Schiller et al. [67], since we cannot use the amplitude image provided by the ToF-camera. For that reason, the fusion of color and range has been performed according to Gordon et al. [65], as a disjunction of the previous results.

The experiments performed on Codebook-based approaches involve only the segmentation stage, without morphological filtering. We have decided to avoid any postprocessing stage to evaluate the capabilities of the algorithms by themselves, although raw results can be easily improved by these simple operators. In addition, morphological filtering can be applied after segmentation in any moment. Nevertheless, the *MOG4D* approach includes morphological filtering, as in the approach proposed by Gordon et al. [65], in order to remove small isolated foreground points caused by noise.

Table 5.1: Segmentation evaluation for sequence *ChairBox*. The table shows F_1 results for the five studied approaches on four different evaluation frames, the mean and standard deviation on the entire sequence.

ChairBox	**Evaluation Frame**				**Global**	
Approach	**278**	**286**	**328**	**356**	**Mean**	**STD**
DECB	**0,937**	**0,928**	**0,876**	**0,914**	**0,914**	**0,027**
CB4D	0,936	0,907	0,819	0,882	0,886	0,050
CB	0,921	0,845	0,784	0,837	0,847	0,057
CB1D	0,904	0,904	0,800	0,808	0,854	0,058
MOG4D	0,883	0,865	0,795	0,859	0,851	0,038

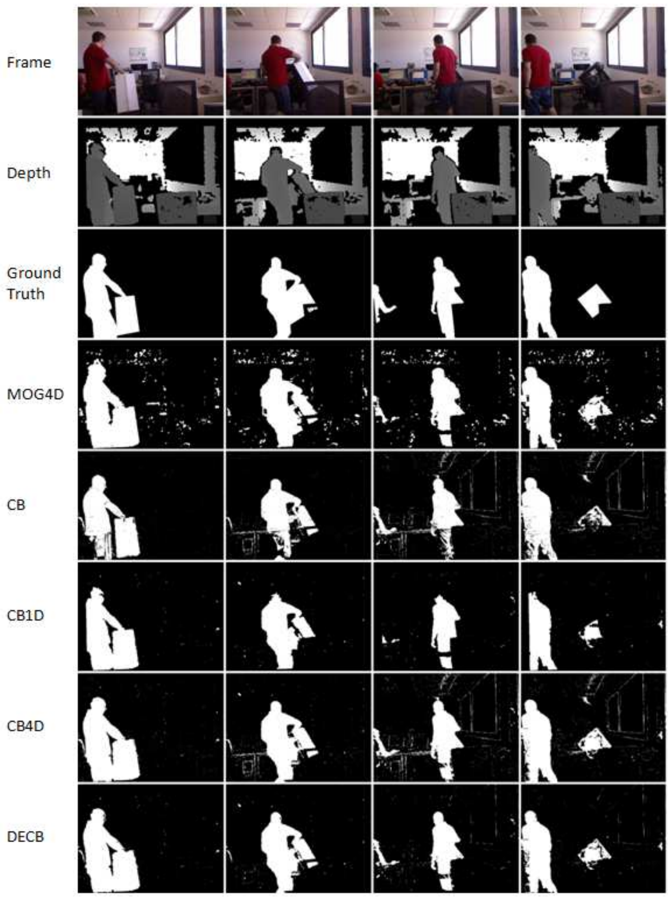


Figure 5.9: Results obtained from the test *ChairBox*. *MOG4D* includes a morphological opening stage, whilst Codebook-based approaches do not perform it, producing then more noise due to isolated pixels. Most of this noise is filtered by *DECB* by means of the fusion method.

Figure 5.8 and Table 5.1 show the quantitative results obtained in the *ChairBox* sequence. Table 5.1 shows F_1 values resultant from the five approaches on the evaluation frames, the mean and standard deviation. Figure 5.8 shows the *gain* on F_1 obtained by the three RGB-D algorithms (*MOG4D*, *CB4D* and *DECB*) over the color-based one (*CB*). All RGB-D approaches get improvements against *CB*, obtaining higher F_1 values despite the good performance of the color-based method. This good performance explains why the *gain* is moderate, since the *gain* is limited by $1/F_1^{CB}$, where F_1^{CB} is the F_1 value obtained by the *CB* algorithm (for example, when $F_1^{CB} = 0.845$, $gain \leq 1.183$). In any case, the graph shows that *depth-extended Codebook* obtains the best results in all tests, whilst *MOG4D* gets more moderated results than the Codebook-based approaches.

Figure 5.9 shows the segmentation produced by the five approaches. In general, the *CB4D* algorithm improves over *CB* and *CB1D* by using depth and color, but *DECB* reduces the amount of noise generated by both algorithms (specially noticeable on last two frames).

The second sequence, *Wall*, is especially complicated for the depth-based algorithm, due to similar depth between foreground objects and background. This is shown in Figure 5.10, where *MOG4D* obtains worse results than *CB* in all tests, whilst *depth-extended Codebook* obtains slightly worse results than *CB* in one frame. This can be explained by checking Figure 5.11, where in the first frame the *CB1D* approach is unable to detect the object, thus misleading the *4D Codebook*. However, despite being based on useless data, *DECB* gets $F_1 = 0.9$ (Table 5.2), showing that it is fairly robust to difficult situations.

Table 5.2: Segmentation evaluation for sequence *Wall*. The table shows F_1 results for the five studied approaches on five different evaluation frames, the mean and standard deviation on the entire sequence.

Wall	**Evaluation Frame**					**Global**	
Approach	**74**	**93**	**134**	**168**	**199**	**Mean**	**STD**
DECB	0,900	**0,966**	**0,912**	**0,957**	**0,952**	**0,938**	**0,029**
CB4D	0,939	0,843	0,901	0,857	0,800	0,868	0,054
CB	**0,942**	0,850	0,910	0,851	0,664	0,843	0,108
CB1D	0,006	0,927	0,314	0,919	0,806	0,595	0,414
MOG4D	0,860	0,406	0,699	0,734	0,435	0,627	0,198

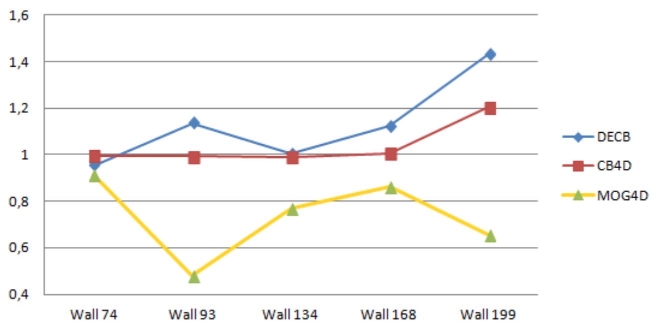


Figure 5.10: F_1 gain over *CB* obtained from the test *Wall*

In addition, it gets much better results on every other frame, reaching *gain* values over 40%. Last two frames in Figure 5.11 show the reasons of this

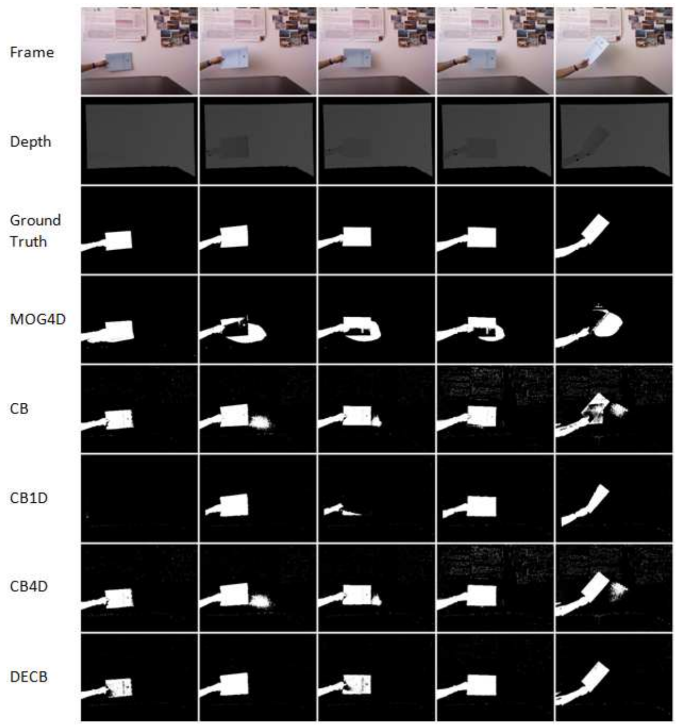


Figure 5.11: Results obtained from the test *Wall*. Codebook-based approaches do not perform morphological opening.

gain, that are proficient noise reduction and a complete shadow suppression by using depth values.

In the third sequence, *Shelves*, the main difficulty is related to changes of lighting and exposure that produce many false positives on the entire image. This can be seen in Table 5.3 with the decrease of F_1 obtained by the *CB* approach, as well as in Figure 5.13 with fair amount of noise on the furniture. Depth is a more stable cue, although there are regions too close to the sensor to be estimated as well as foreground objects too close to the background. Figure 5.12 shows that the *DECB* algorithm obtains much better results by using depth and color combined, since each different input can overcome the weakness of the other. *MOG4D* gets very good results in four frames, although it is prone to errors due to noise in frame 299. In this graph, *gain* values between 10% and more than 60% are obtained by *DECB* in all tests of the sequence, proving that the proposed method is much more robust than the original one based only on color cues.

Figure 5.14 and Table 5.4 show the results for the last sequence, *Hallway*. Being this sequence specially complicated due to the amount of difficulties, F_1 values for the *CB* algorithm are quite low, which allows for higher possible

Table 5.3: Segmentation evaluation for sequence *Shelves*. The table shows F_1 results for the five studied approaches on five different evaluation frames, the mean and standard deviation on the entire sequence.

Shelves	**Evaluation Frame**					**Global**	
Approach	**197**	**212**	**299**	**364**	**418**	**Mean**	**STD**
DECB	0,926	0,909	**0,622**	**0,876**	0,909	**0,848**	**0,128**
CB4D	0,855	0,681	0,365	0,819	0,837	0,711	0,205
CB	0,818	0,655	0,380	0,804	0,838	0,699	0,192
CB1D	0,897	**0,942**	0,595	0,863	0,876	0,835	0,137
MOG4D	**0,927**	0,892	0,154	0,862	**0,937**	0,754	0,337

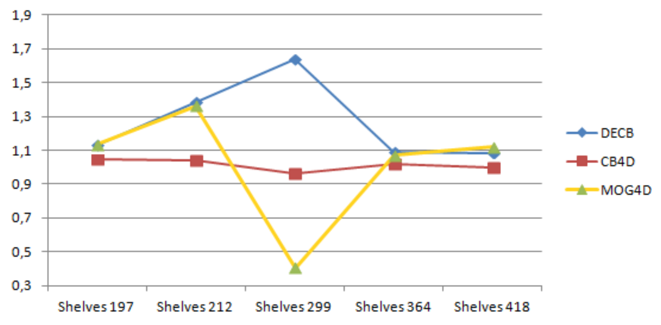


Figure 5.12: F_1 gain over *CB* obtained from the test *Shelves*

gain values (higher improvement), as seen in Figure 5.14. According to this graph, both *CB4D* and *DECB* approaches offer improvement over the original algorithm, but the latter gets much greater *gain* values (up to 120% in one test). *MOG4D* shows good results in four of the frames, but performs worse than the others in presence of sudden illumination changes.

Table 5.4: Segmentation evaluation for sequence *Hallway*. The table shows F_1 results for the five studied approaches on seven different evaluation frames, the mean and standard deviation on the entire sequence.

Hallway	**Evaluation Frame**							**Global**	
Approach	**120**	**258**	**308**	**363**	**435**	**524**	**565**	**Mean**	**STD**
DECB	0,782	0,888	**0,930**	**0,844**	**0,905**	0,385	**0,745**	**0,783**	0,187
CB4D	0,606	0,701	0,835	0,629	0,675	0,222	0,653	0,617	0,190
CB	0,598	0,625	0,802	0,565	0,593	0,174	0,529	0,555	0,189
CB1D	**0,791**	**0,939**	0,791	0,630	0,801	**0,693**	0,744	0,770	**0,097**
MOG4D	0,424	0,875	0,875	0,744	0,732	0,128	0,221	0,571	0,311

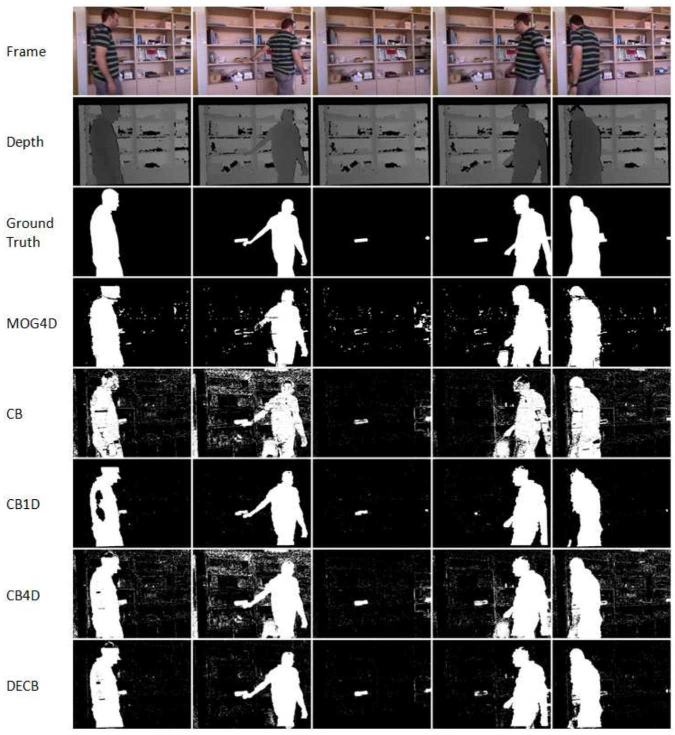


Figure 5.13: Results obtained from the test *Shelves*. Codebook-based approaches do not perform a morphological opening stage.

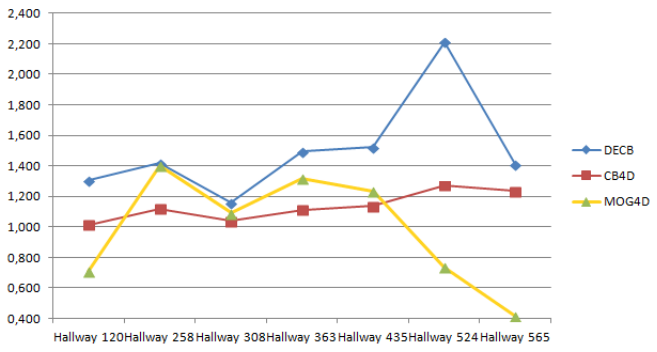


Figure 5.14: F_1 gain over *CB* obtained from the test *Hallway*

By checking Figure 5.15, more detailed qualitative analysis can be performed. In general, it is shown that *DECB* algorithm gets an important noise reduction as well as almost total shadow suppression. In addition, the presence of objects with similar color to the background is complicated for the *CB* approach, but solved with the usage of depth information. This also happens on the fifth frame, with reflections on the floor that are detected

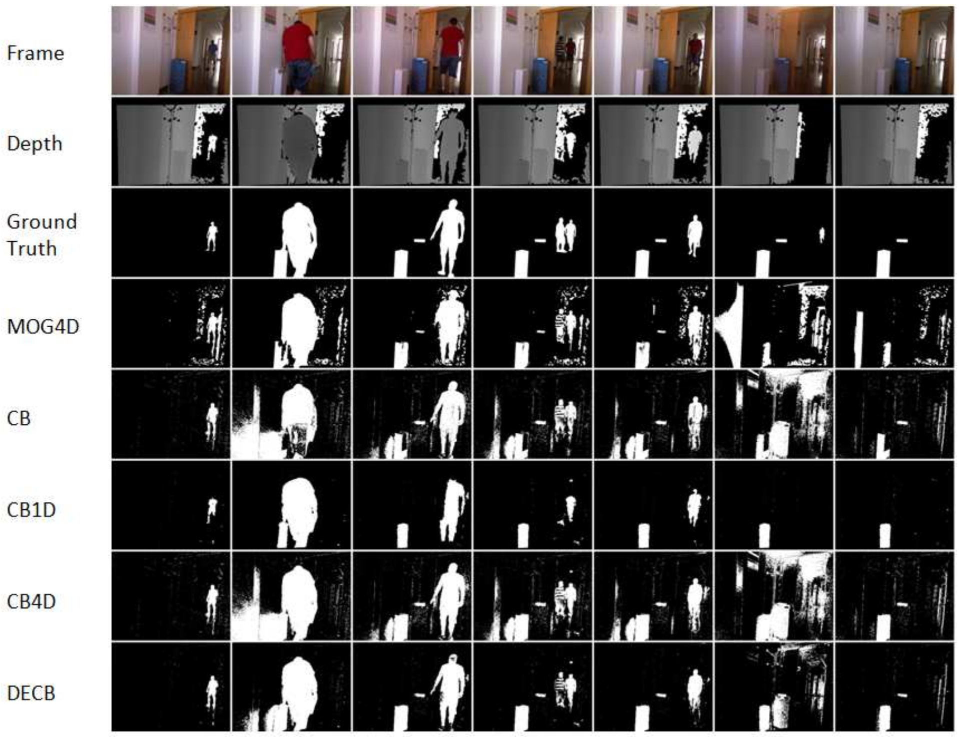


Figure 5.15: Results obtained from the test *Hallway*. Codebook-based approaches do not perform morphological opening.

correctly by the *CB1D* and *DECB* approaches. The most complicated frame in this sequence, that is the sixth evaluation frame, includes sudden illumination changes. A directional light is turned on, producing changes in a big region of the image. Since the *CB1D* approach is based only on depth obtained by infrared sensors from Kinect, it does not suffer from this lighting change. For that reason, despite the *CB* and *CB4D* approaches have a considerable amount of false positives, *DECB* minimizes this amount, thus being more robust than the other methods.

Finally, Figure 5.16 shows the average F_1 and gain over *CB* obtained by each approach in each sequence of the entire benchmark, while error bars show the standard deviation. According to this figure, the *depth-extended Codebook* (*DECB*) shows the best results on every sequence, and the standard deviation associated to this approach is lower than any other, what is a sign of its robustness. Only in one case the *CB1D* algorithm has lower standard deviation, because of the change of illumination in *Hallway* sequence, but even in this case the *depth-extended Codebook* outperforms the other algorithm.

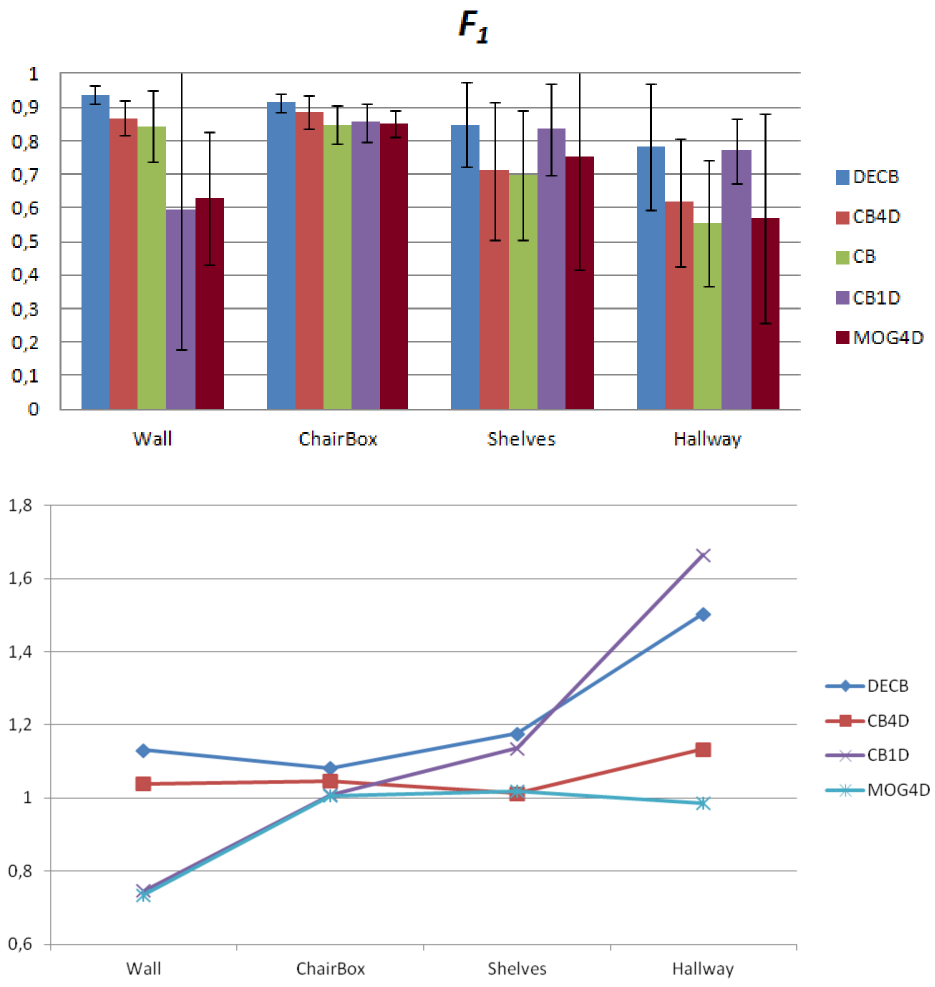


Figure 5.16: Average F_1 obtained from the entire benchmark, including error bars showing the standard deviation, and average F_1 gain over *CB* (along each benchmark sequence).

5.4.2 Evaluation of algorithms using disparity maps

After having performed the experiments with depth information provided by the Kinect peripheral, we conclude that the *DECB* approach obtains more accurate foreground masks than *CB4D* or the 4-channel version of MOG (*MOG4D)*. For that reason, in the following experiments, based on disparity estimation, we have focused on improvements over *CB* and *DECB*, instead of comparison against previous methods.

Seven different approaches have been studied and evaluated with the benchmark described in Section 5.3.2. These approaches are the following ones: the original Codebook model [38], the *Early Fusion* (*DECB*) using three different disparity stereo models as source of depth information, and the model including *Late Fusion* (*DECB-LF*) also using the three different disparity models.

Table 5.5: Segmentation evaluation for sequence *Suitcase*. The table shows F_1 results for the seven studied approaches on four different evaluation frames, the mean and standard deviation on the entire sequence.

Suitcase	**Evaluation Frame**				**Global**	
Approach	**143**	**171**	**190**	**265**	**Mean**	**STD**
CB	0,608	0,677	0,665	0,131	0,520	0,261
DECB-LF-Var	0,827	0,782	**0,814**	0,578	0,750	0,116
DECB-Var	0,836	0,795	0,798	0,635	0,766	**0,089**
DECB-LF-Phase	0,786	0,768	0,792	0,551	0,724	0,116
DECB-Phase	0,594	0,585	0,533	0,282	0,499	0,147
DECB-LF-SGBM	0,878	0,822	0,805	0,555	0,765	0,143
DECB-SGBM	**0,884**	**0,835**	0,799	**0,642**	**0,790**	0,105

Figure 5.17 and Table 5.5 show the quantitative results obtained by each approach in the *Suitcase* sequence. There are four different evaluation frames in order to test the algorithms in a more reliable manner in comparison with the results from only one frame. These results show a fairly good improvement when using range information against only color and intensity. In this sequence the Semi-Global and Variational approaches get the best results, although *DECB-LF* with Phase information gets similar results. The *Late Fusion* improves the average results from the *DECB* when using Phase disparity in more than 0.2. The Semi-Global and Variational models obtain slightly better results when used in the *DECB* without the *Late Fusion* stage. This is due to the high accuracy of the depth information from these models in this video sequence. For that reason, the suppression performed during *Late Fusion* removes less false positives caused by noise than true positives.

Figure 5.18 shows the foreground/background segmentation produced by the studied approaches, as well as the original left frame, the disparity obtained by each image pair and the hand-made segmentation (ground truth). The images for *Suitcase* correspond with the obtained results, being Variational and Semi-Global approaches the ones with best performances. The Phase-based model performs much better with the masks fusion method (i.e. Late Fusion, *DECB-LF*), since it suppresses most of the noise generated by disparity.

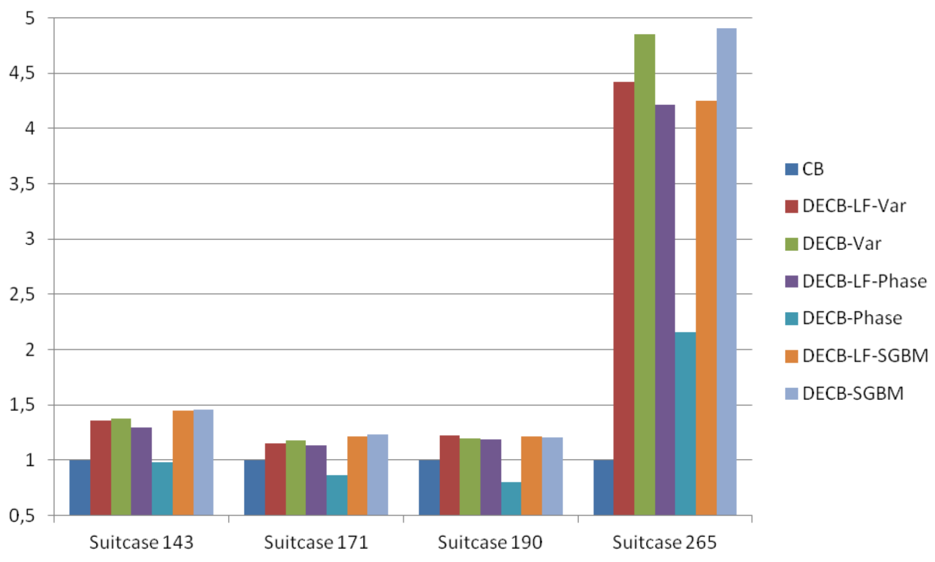


Figure 5.17: F_1 gain over *CB* obtained from the test *Suitcase*. *CB* is the original Codebook algorithm [38]; *DECB-Var*, *DECB-Phase* and *DECB-SGBM* are the early fusion methods with each one of the disparity models; *DECB-LF-Var*, *DECB-LF-Phase* and *DECB-LF-SGBM* are the methods using early and late fusion combined.

Table 5.6: Segmentation evaluation for sequence *Crossing*. The table shows F_1 results for the seven studied approaches on six different evaluation frames, the mean and standard deviation on the entire sequence.

Crossing	**Evaluation Frame**						**Global**	
Approach	**100**	**163**	**334**	**381**	**483**	**565**	**Mean**	**STD**
CB	0,706	0,860	0,592	0,583	0,587	0,592	0,653	0,112
DECB-LF-Var	0,824	0,900	0,731	0,664	0,816	**0,810**	0,791	0,082
DECB-Var	0,816	0,898	0,734	0,667	0,826	0,740	0,780	0,082
DECB-LF-Phase	0,823	0,915	0,631	0,640	0,774	0,808	0,765	0,111
DECB-Phase	0,569	0,723	0,626	0,611	0,646	0,643	0,636	0,051
DECB-LF-SGBM	**0,830**	**0,901**	**0,847**	**0,843**	**0,888**	0,797	**0,851**	**0,038**
DECB-SGBM	0,796	0,834	**0,847**	0,839	0,788	0,718	0,804	0,048

In the second sequence, *Crossing*, two different kinds of evaluation frames have been selected. The first one corresponds to people walking next to the wall, thus complicating the disparity-based detection. During the second one, people walk next to the camera. For that reason, results in Fig. 5.19 and Tab. 5.6 show several frames where the improvement over the color-based

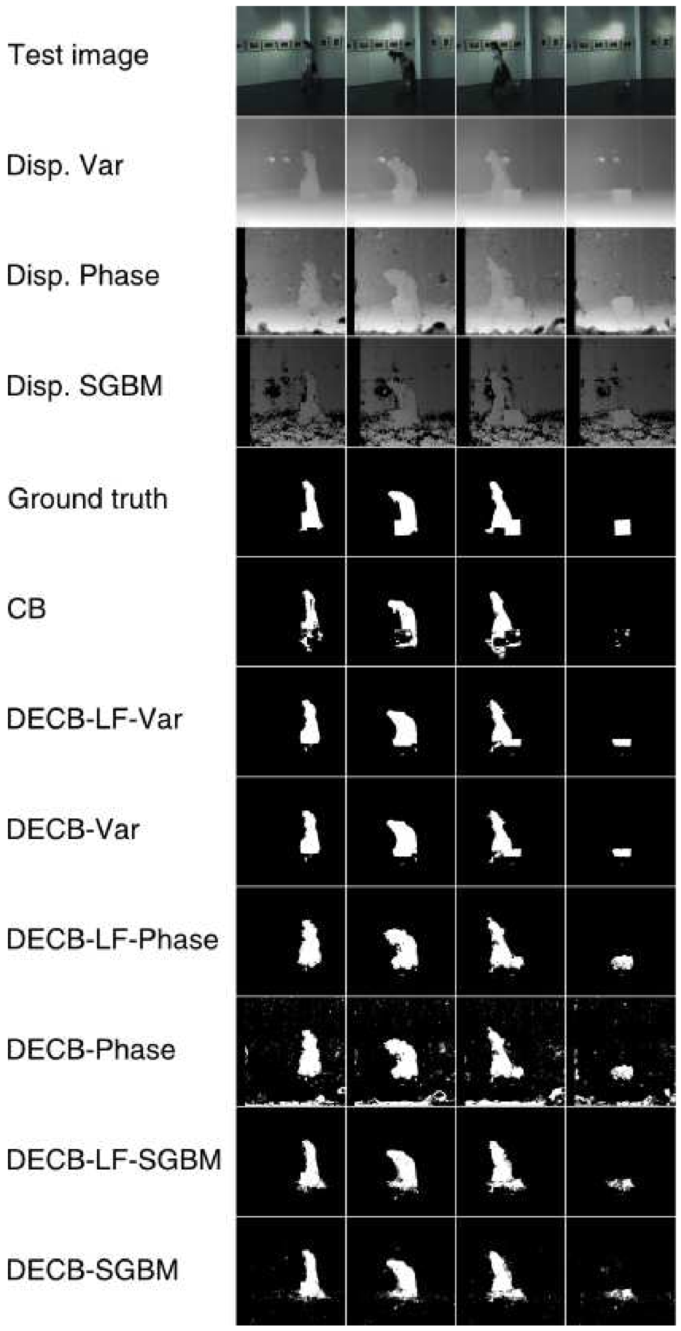


Figure 5.18: Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *Suitcase* sequence.

Codebook is more moderate, although the studied approaches get better results in every test. The only exception to this statement is the *DECB* with

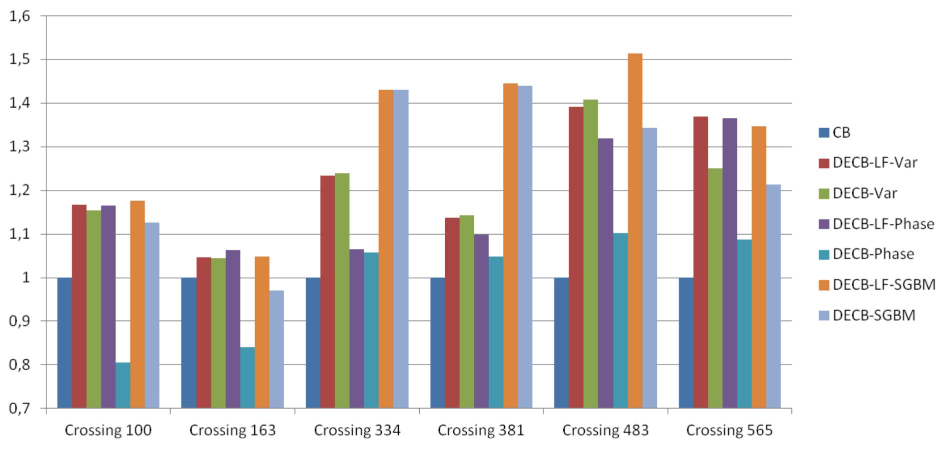


Figure 5.19: F_1 gain over *CB* obtained from the test *Crossing*.

Phase disparity, approach that benefits significantly from the *Late Fusion* post-processing stage. In this test, the approaches with Late mask fusion (*DECB-LF*) perform better than their equivalent without it. Regarding the comparison with the color-based algorithm, the average improvement of the usage of depth information varies between 0.11 and 0.2 (17% and 30% respectively). Segmentation frames obtained with this test are shown in Figure 5.20.

Table 5.7: Segmentation evaluation for sequence *LCDScreen*. The table shows F_1 results for the seven studied approaches on three different evaluation frames, the mean and standard deviation on the entire sequence.

LCDScreen	**Evaluation Frame**			**Global**	
Approach	**303**	**335**	**435**	**Mean**	**STD**
CB	0,875	0,749	0,612	0,745	0,132
DECB-LF-Var	0,884	0,784	0,740	0,803	0,074
DECB-Var	0,880	0,746	0,725	0,784	0,084
DECB-LF-Phase	0,746	0,717	0,611	0,691	0,071
DECB-Phase	0,671	0,637	0,607	0,639	**0,032**
DECB-LF-SGBM	**0,889**	**0,859**	0,748	**0,832**	0,075
DECB-SGBM	0,858	0,851	**0,750**	0,820	0,061

Fig. 5.21 and Tab. 5.7 show the results obtained over the *LCDScreen* sequence. Taking into account these results, the approaches that perform better are the Semi-Global and Variational ones. These methods get slightly

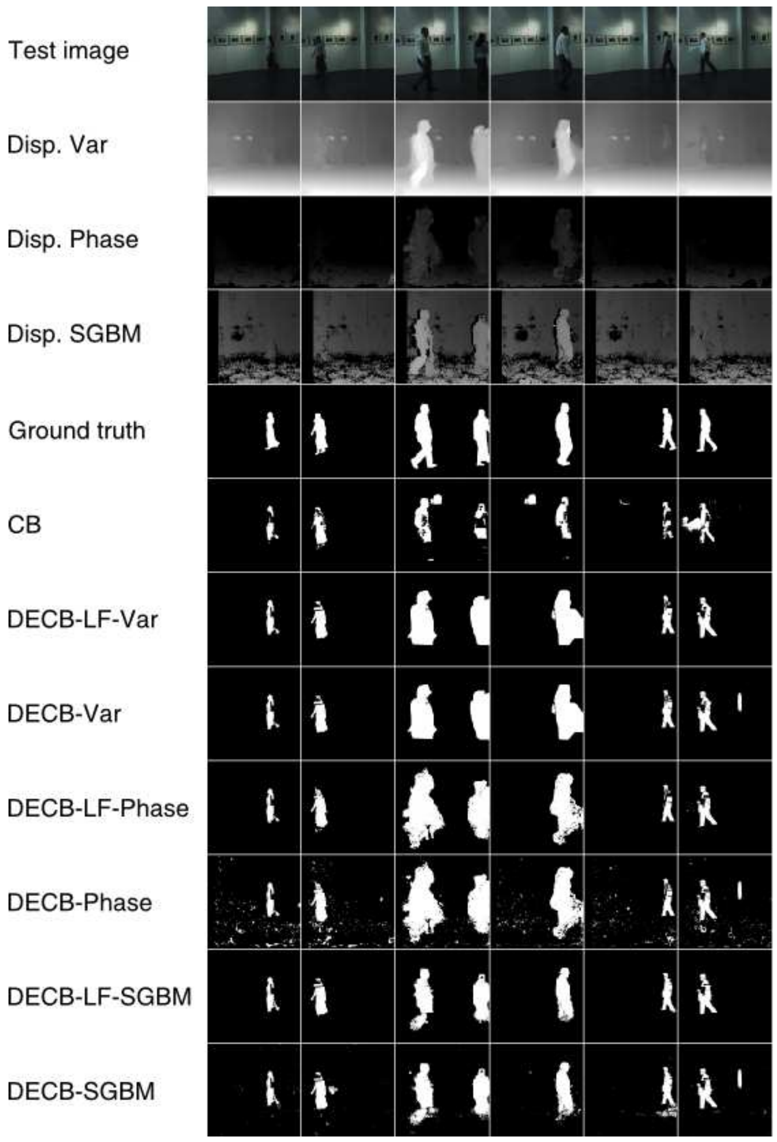


Figure 5.20: Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *Crossing* sequence.

higher marks with the *DECB-LF* approach, and both obtain better results than the color-based approach in every frame. In this test, lighting changes due to fluorescent lights produce a loss of performance in the Phase disparity estimation (phase information is robust against illumination changes, but the reliability threshold and corresponding density of the disparity map still have dependencies on the illumination values). For that reason, the output obtained by the approaches based on this depth information does not repre-

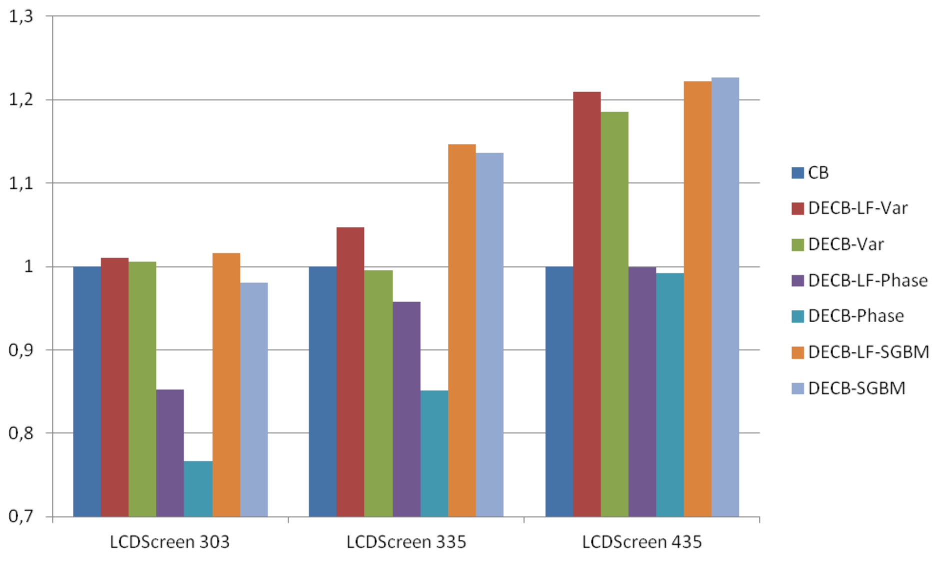


Figure 5.21: F_1 gain over *CB* obtained from the test *LCDScreen*.

sent a real improvement over the original background subtraction method. Qualitative results obtained by the studied approaches on this sequence are shown in Fig. 5.22.

As mentioned in Section 5.2.3, the scenario shown in the *LCDScreen* sequence includes several issues that affect the performance of disparity computation algorithms based on color. Therefore, since the main problem is the influence of the scenario over the computed disparity, this scenario would benefit from the use of active sensors such as Kinect [21] or Time-Of-Flight cameras.

Table 5.8: Segmentation evaluation for sequence *LabDoor*. The table shows F_1 results for the seven studied approaches on seven different evaluation frames, the mean and standard deviation on the entire sequence.

LabDoor	**Evaluation Frame**							**Global**	
Approach	**157**	**197**	**216**	**299**	**792**	**854**	**1003**	**Mean**	**STD**
CB	0,806	0,738	0,360	0,703	0,368	0,591	0,642	0,601	0,176
DECB-LF-Var	0,792	**0,880**	0,411	**0,762**	0,465	0,643	0,763	0,674	0,176
DECB-Var	0,748	0,675	0,331	0,662	0,375	0,334	0,708	0,548	0,190
DECB-LF-Phase	0,755	0,857	0,445	0,729	**0,552**	0,621	0,748	0,673	0,140
DECB-Phase	0,754	0,822	0,429	0,700	0,547	0,606	0,745	0,658	**0,137**
DECB-LF-SGBM	**0,857**	0,813	**0,493**	0,707	0,510	**0,663**	**0,794**	**0,691**	0,145
DECB-SGBM	0,839	0,804	0,468	0,693	0,470	0,580	0,775	0,661	0,156

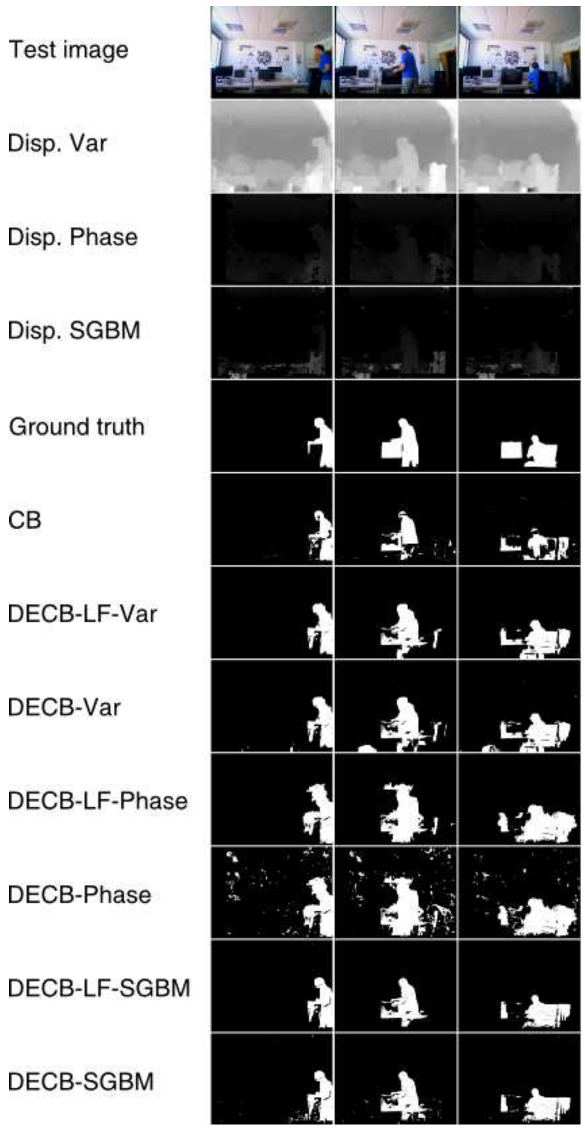


Figure 5.22: Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *LCDScreen* sequence.

The results obtained from the *LabDoor* test sequence are shown in Fig. 5.23 and Tab. 5.8. As seen in Tab. 5.8, the *DECB-LF* based on SGBM disparity obtains the most accurate results, while Variational and Phase ones get similar results, differing only slightly (in the range of 10^{-3}). Average improvement of depth and color approaches over the original Codebook is more constrained in this test, between 0.07 and 0.09 (12% and 15% respectively). This is so thanks to acceptable performance from the color-based algorithm

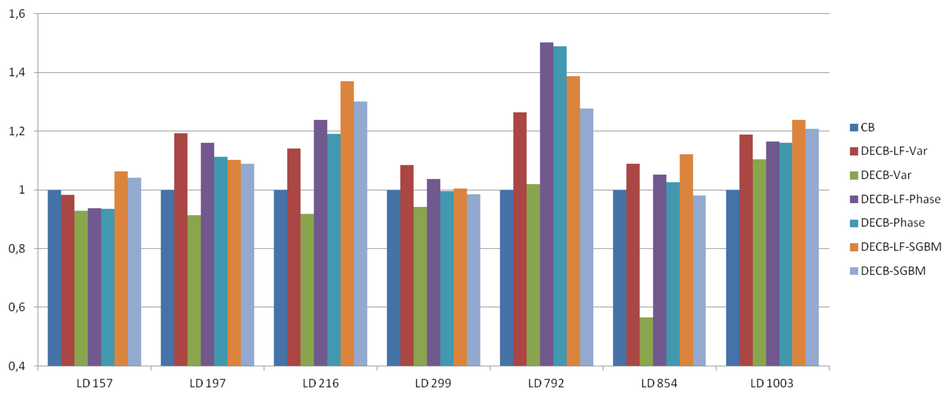


Figure 5.23: F_1 gain over *CB* obtained from the test *LabDoor*.

and to the difficulty of the scenario for disparity estimation. Qualitative results from this sequence are shown in Fig. 5.24.

As conclusion, in all the experiments we have done we see that the inclusion of depth information significantly improves the foreground segmentation process. Fig. 5.25 shows the overall results obtained by all approaches on each sequence, averaging the F_1 values from every evaluation frame, and showing the standard deviation of these values as error bars in the plot.

In general, the *Depth-Extended Codebook* gets better results with the *Late Fusion* post-processing stage (average improvement of 7.5% over *DECB*, taking into account all disparity estimation approaches). The version without mask fusion only got higher marks on the *Suitcase* sequence, with the most accurate disparity estimation methods.

For faster disparity computation approaches such as the phase-based one, the *DECB-LF* method obtains much better results than the other approach (average improvement of 17.46%, being able to achieve up to 30% of improvement). This is justified by the use of morphological reconstruction using the color-based output as marker, which enables the combined algorithm to accurately remove false positives without the use of erosion filters or the definition of minimal area size.

It must be highlighted that depth information has been computed by visual disparity algorithms. For that reason, illumination changes and issues that affect the codebook algorithm will affect resultant range information as well. Nonetheless, the fusion models offer very good results in every test sequence.

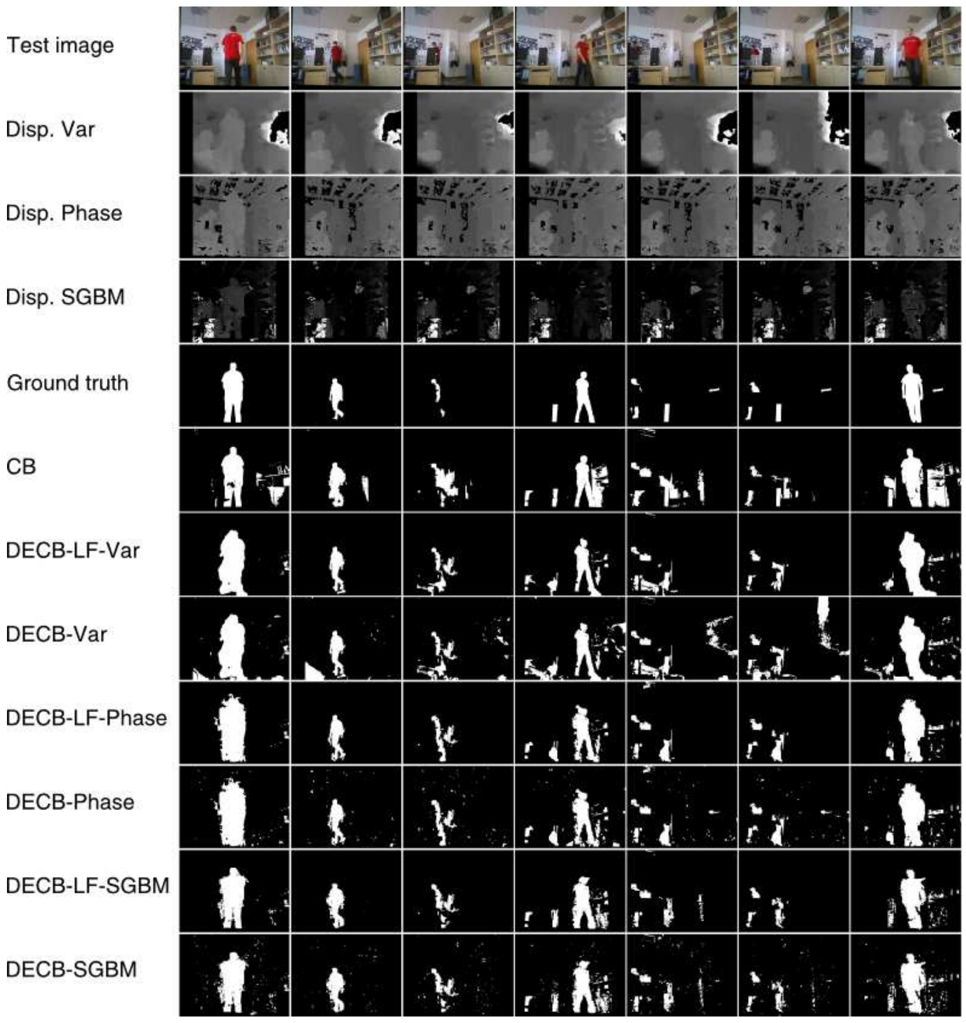
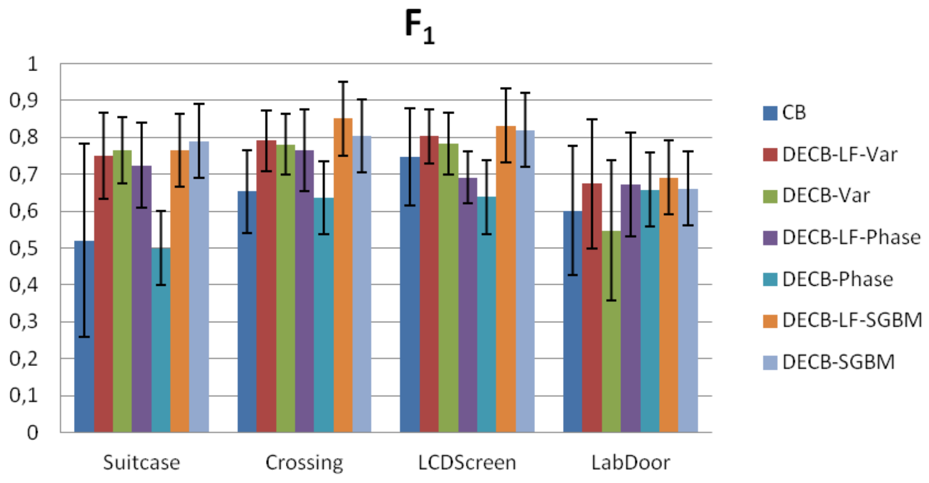


Figure 5.24: Original frames, disparity computed by each stereo algorithm, and FG/BG masks obtained by all approaches in the *LabDoor* sequence.

5.5 Conclusions

As stated in the introduction of this chapter, the problem of foreground segmentation is a well-known problem but still far from being solved. Therefore, in this chapter we have analyzed the inclusion of depth information in an advanced background subtraction algorithm such as the Codebook proposed by Kim et al. [38]. Having range information in addition to color and intensity can improve the robustness of the algorithms used in the segmentation stage of a video surveillance system.



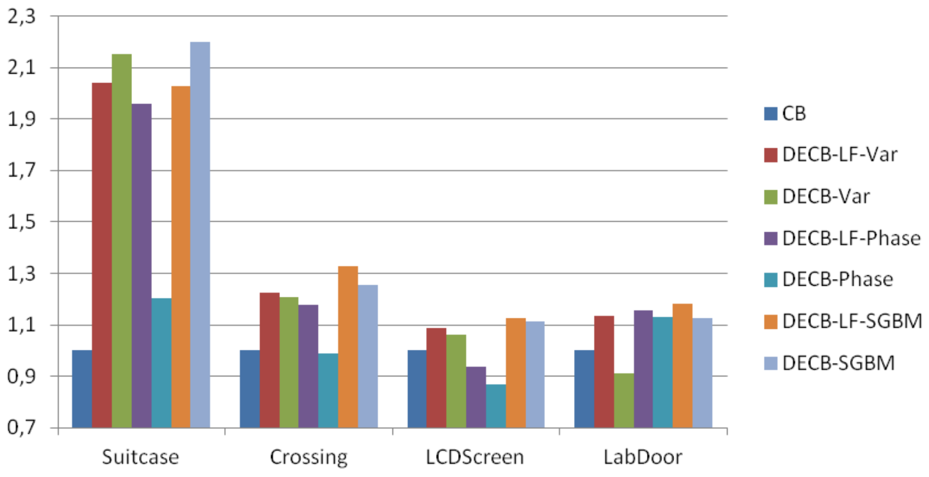


Figure 5.25: Average F_1 obtained from the entire benchmark, including error bars showing the standard deviation, and average F_1 gain over *CB* (along each benchmark sequence)

Depth information can be obtained by means of different computer vision methods and range sensors. We have studied the usage of depth information provided by both an active RGB-D sensor such as Microsoft Kinect [21] and stereo disparity methods. On the one hand, Kinect sensor allows for high-resolution depth maps at a lower cost than Time-of-Flight cameras. Since depth is obtained by using infrared structured light instead of image processing, both signals are complementary and can be used to tackle classical issues of background subtraction algorithms.

On the other hand, Kinect cannot be used in outdoor scenarios and has a more constrained range than stereo camera rigs. For that reason, we have studied the fusion of RGB and depth information provided by disparity estimation algorithms. The only preliminary stage to be done is to calibrate and rectify the two camera views. This is a rather standard method that needs to be done only when installing the system. Three different stereo models for disparity computation have been studied: Variational [126], Phase [143] and Semi-global block matching [139].

We propose an adaptation of the Codebook algorithm [38] to use depth as well as color. The Codebook algorithm is an advanced multimodal method which offers good trade-off between accuracy and efficiency, which makes it a very appropriate approach for implementation on embedded systems and smart cameras. Furthermore, it is robust to dynamic background and gradual scene changes. The use of depth enables proficient shadow suppression as well as reduction of noise due to sudden illumination changes. In addition, it minimizes the impact of *camouflage* (foreground objects with color similar to background).

We have studied three models that differ in the integration method. The first one (*CB4D*) simply considers depth as a fourth channel of the background model. The second one, named *Depth-Extended Codebook* (*DECB*), considers the range information an additional channel of the model and biases the thresholds on color-based detection depending on depth information. Results show a considerable improvement on accuracy and robustness when using depth and color combined, since the proposed approach outperforms the other methods in almost every test performed by using Kinect [144].

However, this technique has been proved to highly depend on the quality of the depth information. For that reason, we have developed a third approach, *Late Fusion* (*DECB-LF*), which consists of a post-processing stage which fuses the resultant FG/BG masks from the color-based Codebook and the *Depth-Extended Codebook*. We propose a mask fusion method based on morphological reconstruction of connected components. This reconstruction uses the color mask as a marker to suppress false foreground regions due to disparity noise, thus being specially suited for faster and less accurate disparity estimation algorithms.

Qualitative and quantitative analysis have been performed by using two complete datasets recorded with Kinect and stereo cameras, which are made publically available at [144].

In the dataset recorded by using Kinect, results show a considerable improvement on accuracy and robustness when using depth and color combined, since the proposed approach outperforms the other methods in almost every test.

When using disparity methods, the inclusion of depth cues represents an average improvement of around 10.8% with respect to the standard color-based Codebook, whilst the average improvement when using our Late Fusion scheme is approximately 19%. According to the experimental results, the disparity quality (among the three methods evaluated) can represent an impact of 10% in the final segmentation accuracy.

Regarding computational costs, the color-based algorithm has been previously implemented in real-time on FPGA, as seen in Section 4.2, being the *depth-extended Codebook* suitable for embedded systems and smart cameras.

The work performed in this chapter focuses on the second objective presented in Section 2.2, the integration of depth estimations in background subtraction models and quality evaluation.

Chapter 6

Multi-camera tracking on embedded devices

Whilst the previous work has been focused on background subtraction as a key task for video analytics systems, we are also interested on further processing stages of these systems, such as object tracking.

As seen in Chapter 3, object tracking is an interesting application of computer vision that has caused much interest in recent years. There are many works in the literature regarding object representation and selection of features to track [71, 72, 73, 74, 75], object detection and segmentation [76, 77, 78], and tracking of objects of interest frame to frame [82, 83, 14, 84, 85].

With the improvements in computer technology, real-time techniques are becoming more feasible in terms of computing power and time constraints. These advances have enabled researchers to focus on multi-camera systems. This kind of systems has many advantages, such as occlusion solving with overlapping views or object tracking between non-overlapping views. This significantly helps to reduce the impact of problems that affect mono-camera systems, such as false alarm generation produced by problematic object views or illumination changes. Multi-camera systems can add robustness to the tracking mechanisms providing a more stable solutions [95, 13, 15, 88, 92]. However, the increasing number of cameras associated to these systems includes some issues related to the physical setup of the surveillance network, as well as to the computing power required to process multiple video streams.

Due to these difficulties of centralized processing, research on distributed networks, where part of the processing is made in each node, has aroused great interest. This kind of solution can reduce the amount of computation to be made by the central servers. In addition, the network requires less bandwidth since not all information has to be sent from sensors to servers.

The information that is shared and transferred between nodes is of higher level of abstraction than raw images. This helps to optimize the communication bandwidth that otherwise would become unaffordable when the number of cameras increases. The idea is to keep the first low-level pixel-wise processing stages on the nodes so that traffic can be minimized. At the same time, if complex decision-making processes are required they can be done on a centralized way so that simple devices can be used at each node. In order to enable decentralized processing, much research has been done on embedded systems and smart cameras.

Traditionally, work on smart cameras has been oriented to dedicated computing architectures (specific purpose hardware) such as FPGAs (Field Programmable Gate Arrays) and DSPs (Digital Signal Processors), as mentioned in Chapter 4. In this context, we focus on the use of embedded devices such as smartphones as smart cameras for distributed and decentralized camera networks. Smartphones have several advantages, such as having reasonably good cameras, fast-evolving powerful processors despite being mobile systems, different types of connectivity and sensors. In addition, they are much more common in the market (in comparison with DSPs and FPGAs), what leads to low prices due to high sales, active communities and availability of development tools that also allow researchers to easily migrate the models among different types of smartphones. We take advantage of these properties to propose a distributed video surveillance network based on smartphones.

The rest of this chapter is organized as follows. We analyze, in Section 6.1, the related work in the literature. In Section 6.2, we describe the architecture of our system, considering both hardware structure and software implementation. Section 6.3 shows an introduction to single-camera tracking and the explanation of the single-camera tracker used in each camera. We explain, in Section 6.5, the information sharing between devices, and the approach to multi-camera setup. In Section 6.6, experimental results obtained with the architecture are shown and analyzed. Finally, conclusions and discussion are presented in Section 6.7.

6.1 Related work

Object tracking is a research field that has aroused much interest in recent years. For that reason, there are many works in the literature regarding tracking, selection of the object of interest and usage of multi-camera networks. These works have already been revised in Section 3.2. In this section, we revise the literature related to our contribution, focusing on the selection of the object of interest, multi-camera tracking and smart camera networks.

6.1.1 Selection of the object of interest

The selection of the object of interest to be tracked is the first step of any tracking application. This is usually performed by manually selecting the target, using blob-based motion detectors or detecting objects by means of supervised classifiers, among many others (as seen in Section 3.2.1). We propose the use of human visual attention to automatically capture the most salient object in the scene [145]. Visual saliency is computed in a bottom-up way and encodes the relevance of each pixel to visual attention.

The visual information is processed along two parallel visual paths after entering the primary visual cortex: the “where pathway” or dorsal stream that directs the gaze towards the most interesting objects in the scene, and the “what pathway” or ventral stream that participates in object recognition [146]. Attention is part of the “where pathway” and selects the most relevant areas, according to their inherent relevance (bottom-up selection) and the task that is being performed (top-down modulation). Its main objective is the optimization of the bandwidth by efficiently reducing the information that has to be processed by the visual system. Moreover, the selection can also be used to increase the robustness of the final multi-tracking system, helping to recover the track after a sudden large movement or a total occlusion.

Itti and Koch [145] considered a bottom-up model with spatial competition for saliency maps. Based on center-surround local contrast of intensity, orientation, and color differences at different spatial resolutions, it computes the saliency map. Although there exist several approaches based for example on edges [147], image torque [148], or segments in the polar coordinate system [149], we use Itti's approach because it is the metric against which most attention models are compared [150] due to its accuracy predicting human gazing.

Additionally, the bottom-up selection of the most relevant features may be modulated by a conscious top-down task-dependent mechanism [146]. In our case, this top-down modulation is deployed using a skin detector operator based on TSL-colored (Tint, Saturation, and Luminance) Gaussian model [151]. This feature was selected due to its feasibility for video-surveillance tasks for which people actions are usually the target, such as for example wandering, bag dropping, or loitering. Nevertheless, other features can be used to drive this task-dependent mechanism if they are relevant for any specific surveillance application. The use of this top-down task-dependent mechanism allows the system to be easily reconfigurable or adapted to diverse surveillance applications.

6.1.2 Multi-camera tracking

There are several works in the literature regarding multi-camera tracking with calibrated cameras for overlapping views. Kim et al. [13] propose multi-camera person tracking assuming planar homography constraint and that the homography between the image plane and floor is known. Models of appearance are defined assuming people are standing, and tracking is performed with particle filtering. In [15], objects are tracked in the reference view and results are propagated to other views by means of the image-to-floor homographies. Tracking is performed based only on blob detection, without building color models. In [95], people tracking is done in a top-view perspective after fusion of foreground mask from every view in the floor plane. Multiple homography layers are computed at different heights. In [14], epipolar geometry between cameras is used in order to integrate tracking information when the track is lost in one camera. These works require a full Tsai calibration [98] or several homographies computed previously, whilst our approach aims for uncalibrated cameras.

Regarding calibration-free camera networks, [87] and [102] use multi-camera tracking in order to perform camera hand-over based on transfer regions. People are tracked in one camera at a time, and the histogram information is sent to other camera when the tracked object enters the defined transfer region. In both works, Mean-Shift tracking [72, 76] based on hue color histograms is performed.

6.1.3 Smart camera networks

There are many works in the literature about video analytics on embedded systems as FPGAs and DSPs, as has been mentioned in Chapters 3 and 4. The use of FPGAs and DSPs allows for the implementation and optimization of computer vision techniques on low power devices, and peripherals can be used in order to exchange information with servers or other embedded systems. However, reconfigurable hardware systems are difficult to develop, requiring deep knowledge about architecture. Because of this, time to market for this kind of platforms is long, usually not comparable with other software solutions. Furthermore, migration of this specific purpose hardware to other devices or platforms can be troublesome. This has motivated our focus on smartphones as development platforms.

About distributed and decentralized camera networks, [16] presents a survey covering calibration, synchronization and fusion of information from different cameras, as well as a comparison between decentralized trackers. In [100], four kinds of two-node architectures to perform particle filtering are analyzed regarding bandwidth utilization and energy consumption. These architectures differ in the information sent across the network, and the steps of the particle filtering (PF) algorithm executed in each node.

Regarding smart camera networks, Bolliger et al. [101] propose a self-organizing network of mobile phones based on Java J2ME which performs communication between devices through Bluetooth connections. Background differencing is used in order to detect objects entering or leaving the field of view. Experiments were performed on a two-node setup. In [102], a hardware architecture on DSPs to track people in a two-camera network is proposed. Tracking is combined with predefined transfer regions to perform handover between cameras. Wittke et al. [103] present an architecture using MIDs (Mobile Internet Devices) that detects people using the Android built-in face detector, and obtains the optical flow field in order to perform frame synchronization.

Table 6.1: Summary of related work on smart camera networks.

	Quaritsch (2007)[102]	Bolliger (2007)[101]	Wittke (2008)[103]
Platform	DSP	J2ME	Android
Camera number	2	2	3
Network	Ethernet	Bluetooth	Wi-Fi/P2P
Task	Tracking	Background subtraction	Optical flow
Objective	Handover	Synchronize	Synchronize
Resolution	352x288	160x120	352x288
Frame rate	20	12.8	10

Table 6.1 summarizes the main characteristics of the different architectures proposed in the literature for distributed and decentralized smart camera networks. Despite a direct comparison between them would not be completely fair, since they operate on different platforms, we consider that it is a relevant starting point. The architecture proposed in [102] offers the best frame rate with greater or equal resolution in comparison with the other approaches. However, this approach is an implementation on specific purpose hardware, whilst [101] and [103] focus on commercial mobile phones. These two architectures can be more easily compared, since both aim to synchronize cameras based on visual cues and operate on more related platforms. [103] outperforms [101] by using Android devices instead of Java 2 MicroEdition (J2ME). Nevertheless, the latter uses commercial *Nokia 6630* mobile phones (released in November 2004), while the former uses small computers with Android OS [152] installed.

In this chapter, we describe an architecture that performs multi-camera color-based tracking on a fully distributed and decentralized camera network with overlapping views. Tracked object trajectories are used in order to obtain the homography between cameras [90]. With this basic calibration, single-camera tracking is guided by means of the information from other cameras, allowing us to improve the tracker, easily recovering from lost tracks whilst solving occlusions.

6.2 Smart camera platform

Smart cameras are key components in video analytics systems on distributed an decentralized networks. In this type of network, each node must not only capture images but also make some kind of image processing independently of the rest of the network. With hybrid networks, where there are servers in addition to smart cameras, the processing made in the cameras allows for the reduction of the information that the servers need to analyze. The tasks that can be performed in the nodes include motion detection, object recognition and tracking.

In order to be suitable for this work, smart cameras require sufficient computing power for image analysis, as well as the presence of development tools and libraries for video and image processing. Furthermore, since our aim is focused on camera networks, connectivity capabilities are an important requirement.

In this section, we describe the hardware architecture in which our approach has been developed and tested. We also explain the software architecture and give implementation details.

6.2.1 Hardware architecture

The hardware platform has to provide the computer power required to perform video analytics tasks, as well as to enable communications between different devices.

Our approach is oriented to commercial smartphones with Android OS [152]. The research and implementation of this work has been done using high-end devices such as Samsung Galaxy S II and Samsung Galaxy Tab 10.1. The key components of the architecture of these devices, from the point of view of smart cameras, are the sensors, the processor and the connectivity unit.

The core of the sensing unit of the smartphone is an 8-megapixel back-illuminated sensor that can record videos in full high-definition 1080p at 30 frames per second. Nevertheless, the resolution of the images has been reduced in order to improve the throughput, as we mention in Section 6.6. In addition to the camera sensor, the hardware offers different categories of sensors [22]:

- Motion sensors. These sensors measure acceleration and rotational forces of the devices along three axes. This category includes accelerometer, gravity sensor, gyroscope and rotational vector sensors.
- Environmental sensors, which measure several parameters such as temperature, illumination and humidity.
- Position sensors. These sensors derive the physical position of the device from the earth's magnetic field in combination with the accelerometer and gyroscope.

The Galaxy S II includes a 1.2 GHz dual core ARM Cortex-A9 processor that uses Samsung's own Exynos SoC (System on a chip). This SoC also includes an ARM Mali-400 MP GPU. In addition to the processor, the hardware has 1 GB of dedicated RAM. The Samsung Galaxy Tab 10.1 includes a 1.0 GHz dual core ARM Cortex-A9 processor with Nvidia Tegra 2 chipset and 1GB of RAM.

The connectivity unit integrates 802.11n Wi-Fi and Bluetooth 3.0, between others kinds of communication protocols. The communication chip also supports Wi-Fi Direct that communicates directly with other device without having to interact with an access point. However, in order to avoid aiming to such a specific variety of commercial devices (for the sake of portability), our architecture works using only Wi-Fi and Bluetooth.

6.2.2 Software architecture

The software architecture is supported by Android OS [152] for all the hardware architecture components: sensing, image processing and communication. Android is a freely downloadable open source software stack that includes an operating system specially suited for ARM architecture, middleware and key applications based on Linux and a Java API as programming interface [103].

Figure 6.1 [152] shows a complete scheme of the architecture of Android system. Android is built on top of a Linux kernel, which acts as a hardware abstraction layer. This layer contains the drivers for all key hardware components, such as the display, the camera, Wi-Fi and Bluetooth, and power management and memory access.

On the next abstraction level, Android provides a set of C/C++ libraries used by various components of the Android system, as well as the *Android Runtime*. These libraries are available not only to internal components of Android, but also to developers through the *Android Application Framework*. From the many different libraries provided by the system, our architecture focuses on *Surface Manager*, *Media Framework*, *OpenGL* and

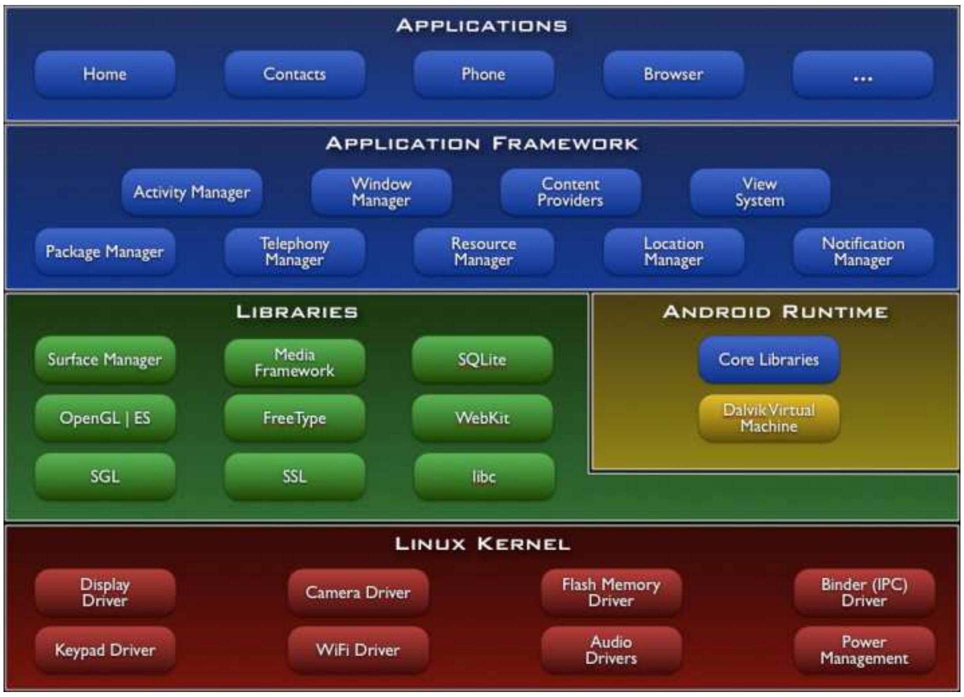


Figure 6.1: Scheme of Android Architecture.

libc. *Surface Manager* enables access to the display subsystem and allows for the management of graphics; *Media Framework* consists of libraries for audio and video playback and recording, as well as support for static image files; *OpenGL* is used for visualization issues; finally, *libc* is the implementation of standard C system library for embedded Linux-based devices, being thus the basis for native code development. *Android Runtime* runs over the previous described libraries, offering additional core libraries that provide most on the funcionality available in the Java programming language. These core libraries support Android's own Java Virtual Machine, called *Dalvik Virtual Machine*, which is specifically designed for mobile devices regarding embedded environment constraints such as limited battery, memory and CPU power.

The *Android Application Framework* layer offers full access to the APIs provided by the core libraries, while allowing developers to access additional non-code resources such as security permissions associated to the application, graphics and layout files, or data from other applications.

Additionally to the possibilities offered by the Android platform, our architecture also uses the recent OpenCV port for Android [17] for image processing and computer vision methods. The OpenCV port works over Android NDK (Native Development Kit), since it is developed in C/C++, thus being Android NDK more suitable for OpenCV than the Dalvik Virtual Machine over which Android SDK (Software Development Kit) works. However, it is not advisable to implement an entire Android application with NDK due to features present in the SDK. For that reason, Java wrappers are implemented in OpenCV to call functions of the NDK port through JNI (Java Native Interface). JNI allows for communication between the application running over the Virtual Machine and the code developed using native code.

Fig. 6.2 shows a scheme of our architecture. This scheme includes the components provided by Android and the native and managed versions of OpenCV, as well as the different modules of our implementation. Some modules are implemented directly over the Dalvik Virtual Machine in order to benefit from its features. These modules are: the *Graphical User Interface* (GUI) that allows the user to interact with the devices; the *Networking* module, which is responsible of the connection between devices in the network and the exchange of information between them; the *Homography Computation* module obtains the homography between two cameras from the information given by single-camera trackers. On the other hand, the *Attention Module*, the *Histogram Manager* and the *CAMSHIFT Tracker* run over the NDK in order to improve the efficiency of the application by reducing unnecessary JNI function calls.

6.3 Single-camera tracking: CAMSHIFT

Visual tracking involves the detection and extraction of objects from video streams, and tracking them over time in order to form trajectories. The main requirements for a tracking algorithm are robutsness and stability, while keeping good throughput in terms of frames per second (fps).

There are many different approaches in the literature to perform object tracking. Between the most widely used, we can find tracking algorithms based on features and color-based ones. Probably the best known feature tracker is KLT [72] which is based on tracking point features, such as corners, using a correlation measure. Regarding color-based trackers, the mean-shift algoritm [76] uses color distributions in order to track objects.

In this work, we use the Continuously Adaptive Mean Shift algorithm (CAMSHIFT), presented by Bradski [153], which is a generalization of mean-shift [76]. It combines the basic mean-shift algorithm with an adaptive region-sizing step. As well as mean-shift, CAMSHIFT operates on a color probability distribution image obtained from histogram back-projection. The model of the tracked object is a histogram of Hue from HSV color model.

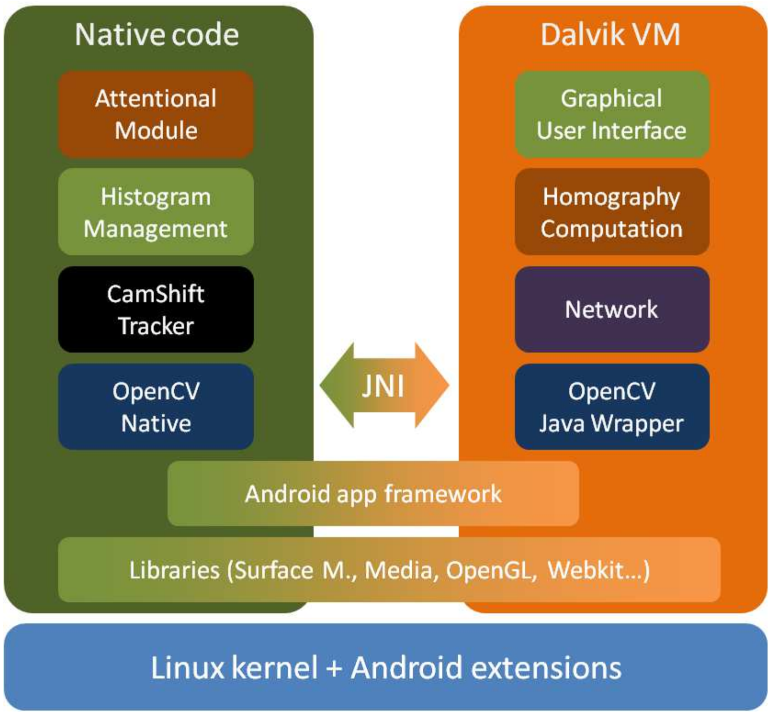


Figure 6.2: Architecture of the MultiCam framework, which makes use of the OpenCV port for Android. We have implemented the native modules for attention, histogram management and CAMSHIFT tracking as well as the modules to share information through the network and to perform calibration by means of homography.

Figure 6.3 summarizes the algorithm for CAMSHIFT tracker. For each video frame, the raw image is converted to a color probability distribution image via a color histogram model of the color being tracked. The center of the color object is found via mean-shift, and the size is adjusted according to the area under the search window. The 2D orientation is then computed using second moments. Finally, the current size and location of the tracked object are reported and used to set the search window in the next video image.

We have considered appropriate the use of this tracker in our architecture due to its low computing requirements in comparison with more advanced trackers. Furthermore, CAMSHIFT deals fairly well with image problems such as irregular object motion due to perspective, image noise, distractors and partial occlusion [153].

Figure 6.4 shows an example of use of CAMSHIFT tracker. This example consists of non-consecutive video frames in which the size and rotation of

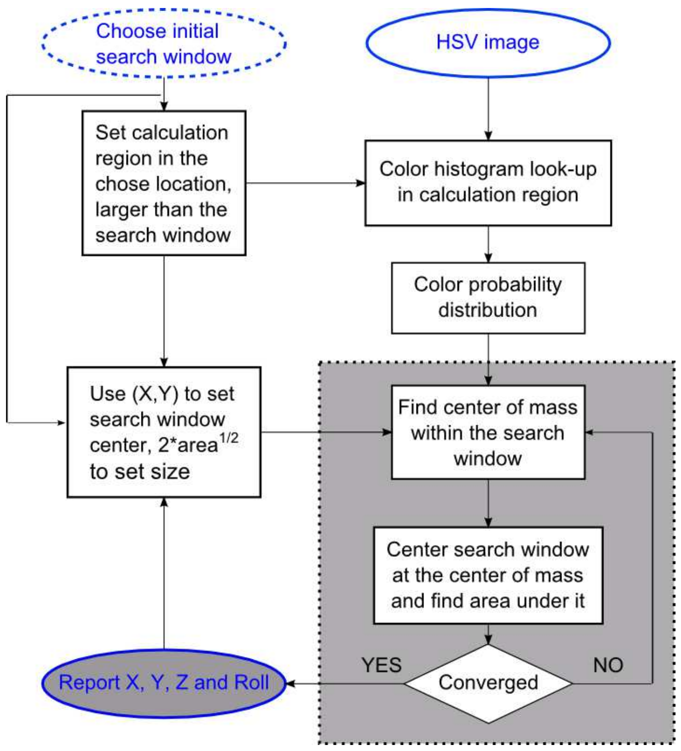

Choose initial search window
HSV image
Set calculation region in the chose location, larger than the search window
Color histogram look-up in calculation region
Color probability distribution
Use (X,Y) to set search window center, 2*area$^{1/2}$ to set size
Find center of mass within the search window
Center search window at the center of mass and find area under it
YES
NO
Converged
Report X, Y, Z and Roll

Figure 6.3: Block diagram of CAMSHIFT tracker.

the tracked object are adjusted by means of the probability distribution and second moments. Calculation of 2D rotation is explained in more detail in Section 6.4.

6.4 Tracking initialization by an attention module

As mentioned in Section 6.1, the object of interest is selected according to Itti's saliency model [145]. The main interest of this automatic selection resides in the flexibility and power that it confers to the multi-camera tracking system. On the one hand, the bio-inspired selection of the object is performed without a previous segmentation, mimicking the human visual system to enhance the most significant cues. On the other hand, this flexible approach differs from the traditional ones which need the object of interest main features. In our case, the attentional module can be easily modulated through a new top-down mechanism. In our application, the skin detector modulates the response to enhance people detection. Nevertheless, the addition of for example, a simple motion detector may modulate the system to detect any animated intruder in a video surveillance system. Thus, the a

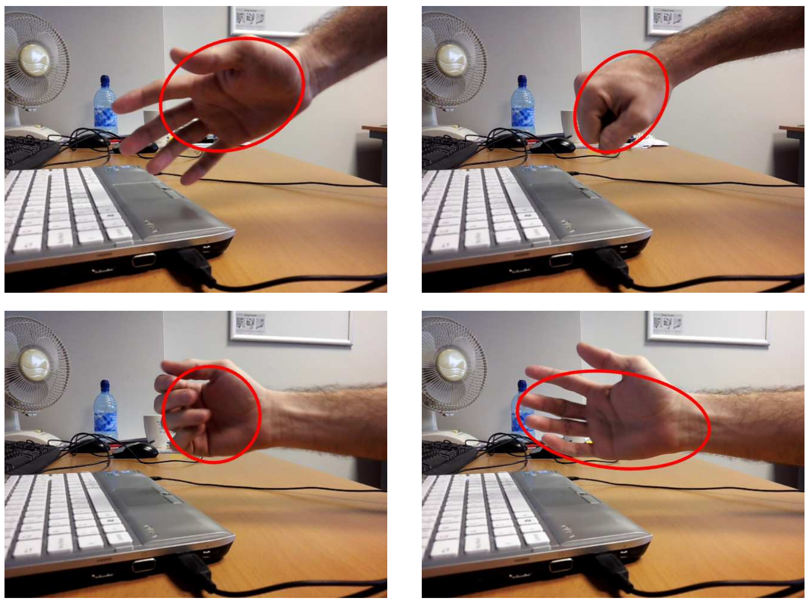


Figure 6.4: Example of use of CAMSHIFT. Object size and 2D rotation are adjusted for every new frame.

priori knowledge about the object of interest converts this module, and for extension the complete tracking, in a very powerful tool.

The block diagram in Fig. 6.5 illustrates the main stages of our attention module that integrates the bottom-up inherent stream and the top-down task-dependent modulation.

Regarding the bottom-up saliency model, the initial assumption is that visual information is centralized in a topographical map that represents the relevance of the different regions in the scene. The saliency map computation is summarized as follows:

1. Compute the input image pyramid for different spatial resolutions.
2. Extract the bottom-up features: color differences (red-green and blue-yellow), intensity, orientation.
3. Compute the center-surround subtraction for different spatial scales within the same feature (local contrast).
4. Calculate the weighted summation of the previous maps into a map per each feature ("conspicuity maps").
5. Combine the "conspicuity maps" and the top-down modulation maps into a unique saliency map.

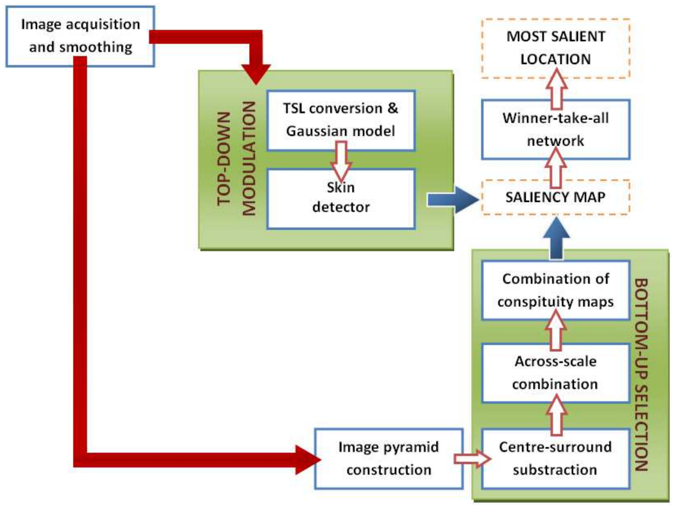


Image acquisition and smoothing
TOP-DOWN MODULATION
TSL conversion & Gaussian model
Skin detector
MOST SALIENT LOCATION
Winner-take-all network
SALIENCY MAP
Combination of conspituity maps
Across-scale combination
Centre-surround substraction
BOTTOM-UP SELECTION
Image pyramid construction

Figure 6.5: Attention module block diagram. It shows the main stages of the bottom-up saliency model and the top-down modulation that is deployed in our case using a skin detector.

Regarding bottom-up feature extraction, Itti's model [145] focuses on three different sets of feature maps. The first set is concerned with intensity contrast, which is detected in dark centers on bright surrounds or bright centers on dark surrounds. The second set is constructed for color channels, enhancing spatial and chromatic opponency, such as red/green and blue/yellow. The third set of feature maps consists of local orientation information between the center and surround scales.

According to the original [146] and to the modifications performed in [154], the first step consists in sub-sampling the input image I into a Gaussian pyramid of factor two (with 6 levels) in the same way that [133]. Assuming an RGB-color input image, we firstly extract the color differences for all the scale resolutions: red-green and blue-yellow (6.1).

$$\begin{aligned} M_{RG} &= \frac{r-g}{max(r,g,b)} \\ M_{BY} &= \frac{b-min(r,g)}{max(r,g,b)} \end{aligned} \tag{6.1}$$

Gabor filters $G_\theta(\mathbf{x})$ oriented for $\theta = \{0°,\ 45°,\ 90°,\ 135°\}$ are used for the computations of the local orientation maps (see (6.2)). In (6.2), $\mathbf{x} = (x,y)^T$ represents the pixel, w_0 the peak frequency, σ the standard deviation, and $\bullet$ stands for the convolution. ρ_θ and ϕ_θ are the amplitude and the phase components of the complex response, and C_θ and S_θ the real and imaginary responses.

$$\begin{aligned} G_\theta(\mathbf{x}) &= e^{\frac{x^2+y^2}{2\delta^2}} e^{jw_0(xcos\theta+ysin\theta)} \\ Q_\theta(\mathbf{x}) &= (I \bullet G_\theta)(\mathbf{x}) = \rho_\theta(\mathbf{x})e^{j\phi_\theta(\mathbf{x})} = C_\theta(\mathbf{x}) + jS_\theta(\mathbf{x}) \end{aligned} \tag{6.2}$$

The orientation maps M_θ are computed as the amplitude response for the different orientations. The intensity map M_I is calculated as the square root of the average of the amplitude response (as defined in (6.3)).

$$\begin{aligned} M_\theta &= C_\theta^2(\mathbf{x}) + S_\theta^2(\mathbf{x}) \\ M_I &= \sqrt{\frac{\sum_N M_\theta}{N}} \end{aligned} \tag{6.3}$$

After the feature extraction, the model performs the center-surround subtraction $\ominus$ between a center c and a surround s scale levels for couples of maps of the same feature M_f, as shown in (6.4). Next, the across-scale combination ($\oplus_i^j$) sums the maps for the different spatial resolutions into a single map F_f with the resolution of the third scale. Furthermore, we are using a normalization operator N that simulates local competition. Between the proposals in [155], we have selected the iterative localized interaction, that convolves iteratively the feature maps with a difference of Gaussians. After the normalization, it obtains the "conspicuity maps" for each feature map C_F as in (6.5).

$$\overline{F}_f = N\left(\oplus_{c=2}^{4} \oplus_{s=c+3}^{c+4} \left(N(|M_f(c) \ominus M_f(s)|)\right)\right) \tag{6.4}$$

$$\begin{aligned} \overline{C}_I &= \overline{F}_I \\ \overline{C}_C &= N\left(\sum_{C\in\{RG,\ BY\}} \overline{F}_C\right) \\ \overline{C}_O &= N\left(\sum_{O\in\{0,\ 45,\ 90,\ 135\}} \overline{F}_O\right) \end{aligned} \tag{6.5}$$

Finally, the method generates the bottom-up saliency map S as shown in (6.6). After the generation of the saliency map, its maximum peak would correspond to the location to be attended.

$$S = \sum_{F\in\{I,C,O\}} \overline{C}_F \tag{6.6}$$

Regarding the top-down modulation, the skin operator computes a map of skin likelihood. The implementation is based on a Gaussian chrominance distribution model [151]. Firstly, the original RGB input images are converted to the TSL (Tint, Saturation, and Luminance) color space. In the mentioned work, the normalized TSL-color image yields the best fit to the model. The normalized TSL color space is defined from the RGB as in (6.7)

$$\begin{aligned} r &= \frac{R}{R+G+B} & g &= \frac{G}{R+G+B} \\ r' &= (r-1/3) & g' &= (g-1/3) \\ S &= \sqrt{9/5(r'^2+g'^2)} & L &= 0.2999R+0.587G+0.114B \end{aligned}$$

$$T = \begin{cases} arctan(r'/g')/(2\pi)+1/4 & \text{if } g'>0 \\ arctan(r'/g')/(2\pi)+3/4 & \text{if } g'<0 \\ 0 & \text{if } g'=0 \end{cases} \tag{6.7}$$

The skin chrominance model assumes that the skin distribution may be modeled by an elliptical Gaussian joint probability density function that is defined as in (6.8)

$$P\left[\frac{\mathbf{x}(i,j)}{C_s}\right] = \frac{1}{2\pi\sqrt{|\Sigma_s|}} e^{\frac{\lambda_s^2(i,j)}{2}} \tag{6.8}$$

where the vector $\mathbf{x}(i,j)$ stores the T and S values of the pixel (i,j), C_s represents the skin class, and Σ_s is the covariance matrix for TS skin value. λ_s is the Mahalanobis distance as defined in (6.9)

$$\lambda_s^2(i,j) = (\mathbf{x}(i,j)-\mu_s)^T \Sigma_s^{-1} (\mathbf{x}(i,j)-\mu_s) \tag{6.9}$$

where μ_s is the mean vector for the TS skin chrominance. Both, μ_s and Σ_s are estimated for the TS values using a set of images belonging to the IBTD database available at [156].

6.4.1 Automatic object segmentation

The proposed Attention Module that integrates the bottom-up saliency and the top-down modulation mechanism generates a saliency map. Whilst taking the most salient pixel of that map and setting a fixed area region around it would be straightforward, this approach is not suitable for systems where the objects of interests may differ in size or distance to the camera. For that reason, we have performed an adaptation stage obtained from the CAMSHIFT algorithm.

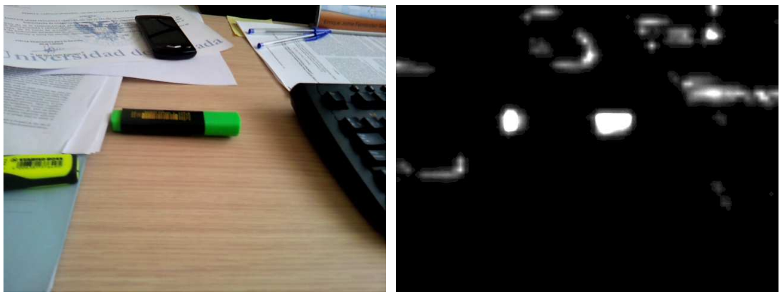


Figure 6.6: Example of usage of the Attention Module. Attention can be focused on different kinds of characteristics, such as color, texture or contrast.

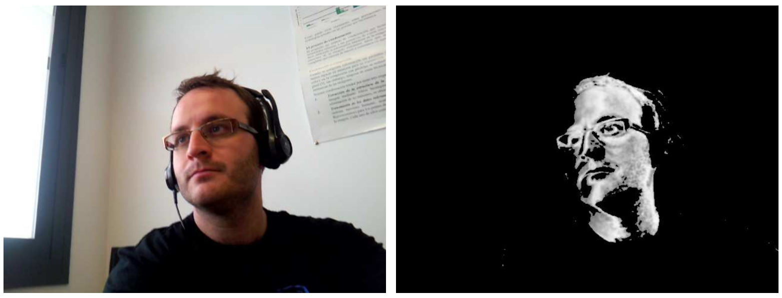


Figure 6.7: Example of usage of the skin detector included in our Attention Module.

The goal is to infer information about position, size and orientation of the object detected by means of the Attention model. The adaptation method we use was presented by Freeman et al. [157], and uses image moments to calculate an equivalent rectangle for the information presented in the current image.

According to [157], if $I(x, y)$ is the image intensity at position (x, y), then the image moments, up to second order, are computed as in (6.10):

$$\begin{aligned} M_{00} &= \sum_x \sum_y I(x,y) & M_{11} &= \sum_x \sum_y xyI(x,y) \\ M_{10} &= \sum_x \sum_y xI(x,y) & M_{01} &= \sum_x \sum_y yI(x,y) \\ M_{20} &= \sum_x \sum_y x^2 I(x,y) & M_{02} &= \sum_x \sum_y y^2 I(x,y) \end{aligned} \tag{6.10}$$

The method consists of finding the position (x_c, y_c), orientation θ and dimensions l_1 and l_2 of an equivalent rectangle which has the same moments as those measured in the region of interest. The position (x_c, y_c) is obtained as follows:

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}} \tag{6.11}$$

The intermediate variables a, b and c are defined as:

$$\begin{aligned} a &= \frac{M_{20}}{M00} - x_c^2 \\ b &= 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \\ c &= \frac{M_{02}}{M_{00}} - y_c^2 \end{aligned} \tag{6.12}$$

The orientation θ is computed as follows:

$$\theta = \frac{arctan(b,(a-c))}{2} \tag{6.13}$$

and the dimensions of the rectangle are

$$\begin{aligned} l_1 &= \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \\ l_2 &= \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \end{aligned} \tag{6.14}$$

The extracted parameters are independent of the overall image intensity. Therefore, this method is suitable for both attention and skin detection algorithms, despite they give results with different ranges of saliency.

Moments are obtained from a region of interest of the saliency map. For that reason, in order to use this method we set an initial window around the most salient pixel with dimensions 10% the ones of the image. The adaptation performed by this stage allows the algorithm to select an appropriate

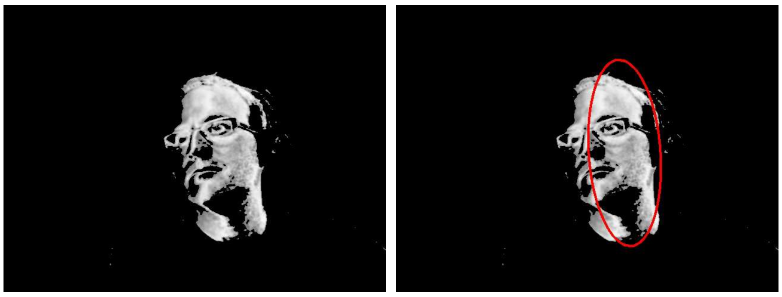


Figure 6.8: Adaptation stage applied to the skin detector included in our Attention Module.

size and orientation according to the saliency map. Fig. 6.8 shows the result of applying this method on a saliency map obtained by the skin detection module. The area selected by means of the described method is used to compute the histogram for the CAMSHIFT tracker.

6.5 Multi-camera model

In this section, we explain the multi-camera model used in our approach. We start by explaining briefly what projective transformations or homographies are. Later, the importance of multi-camera setups and communication between devices is highlighted. Furthermore, we describe the sharing of information through the network and the calibration performed by means of this information. Finally, we explain the incorporation of information exchanged across the network and its usage to improve the single-camera tracker.

6.5.1 Projective transformations: homography

A *projective transformation* or *homography* is a linear transformation on homogeneous 3-vectors represented by a non-singular 3×3 matrix [123]:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{6.15}$$

or more briefly, $x' = Hx$, where $x = (x_1, x_2, x_3)^T$ is a point on the image plane when the camera is at position C and $x' = (x'_1, x'_2, x'_3)$ the corresponding point of x when viewed from C'. A scheme of this relation is shown in Figure 6.9.

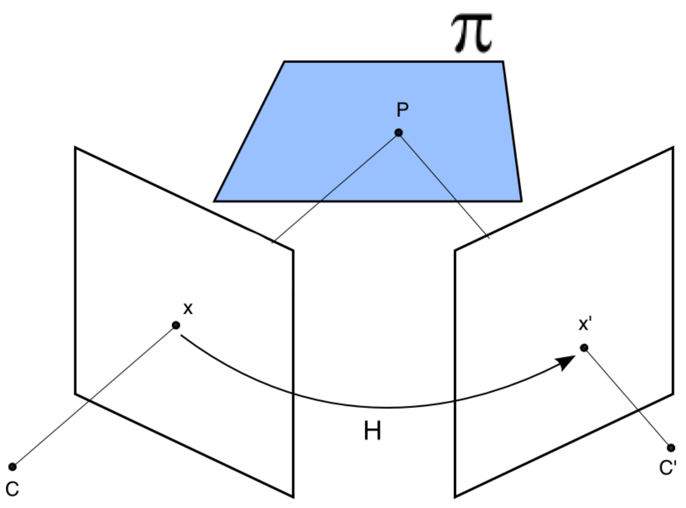


Figure 6.9: Two images of the same planar object are related by a linear transformation in homogeneous coordinates called projective transformation or homography

Homographies have been widely used as a mean to locate objects in multi-camera setups, as seen in Section 6.1. The idea relies on having the homographies of the ground plane or planes parallel to it between pairs of cameras, so that successful tracks of an object in a camera view can be propagated to other views.

The problem of estimation of a 2D homography is defined as follows [123]: given a set of points x_i in $\mathbb{P}^2$ and a corresponding set of points x'_i likewise in $\mathbb{P}^2$, compute the projective transformation that takes each x_i to x'_i.

We consider a set of point correspondences $x_i \leftrightarrow x'_i$ between two images. Our problem is to compute a 3×3 matrix H such that $Hx_i = x'_i$ for each i. Since the matrix H contains 9 entries but is defined up to scale, the total number of degrees of freedom is 8. On the other hand, each correspondence between points specifies two degrees of freedom, since both x_i and x'_i are homogeneous 3-vectors. As a consequence, in order to estimate H at least four point correspondences are required. If more than four correspondences are given, instead of having an exact solution, the homography is estimated by minimizing some cost function. The estimation of 2D homographies is explained with more detail in [123].

6.5.2 Communication network

Exchanging information between cameras is important in order to increase the robustness of the tracking. Such information exchange can be used for many applications in different forms. For instance, multiple camera setups can be used for obtaining different views of the same object and determine which view is the best one. In addition, in a scenario with overlapping views, an object can be tracked by all cameras at the same time, refining the tracker information and allowing the network to handle occlusions.

In our architecture, we consider a network architecture where nearby phones form an ad-hoc network using short range communication such as Bluetooth. Communication of this ad-hoc network with other networks or central servers can be performed through a Wi-Fi connection. The usage of Bluetooth connections allows us to deploy the device network in places without access to a Wi-Fi network. In addition, devices can be paired directly, instead of requiring knowledge about the IP addresses associated to each smartphone.

As the bandwidth of wireless communication among devices is rather constrained, collaboration in our framework is based on distributed events instead of exchanging raw image streams. The information exchanged between devices will be described in Section 6.5.3.

Regarding the communication network implementation, device discovery is performed by using Bluetooth API, requiring this process at least 15 seconds to complete. For that reason, the use of this function has been reduced to on-demand operation. After that, the peer-to-peer connection remains established, allowing for full-duplex communication.

Unfortunately, the Bluetooth API JSR-82 [158] has the fundamental limitation that only a single Bluetooth connection can be open at a time. For that reason, broadcasting to a network or simply exchanging information between more than two devices requires either the use of Wi-Fi connections or closing Bluetooth connections before opening new ones. In our architecture, we plan to make the extension by using Wi-Fi in order to perform UDP broadcasting and use TCP for peer-to-peer connections as future work.

6.5.3 Information exchange

As mentioned in Section 6.5.2, the sharing of information in a smart camera network is important to increase the tracking robustness. In our architecture, cameras are able to exchange many types of information, from raw image streams to keypoints or descriptors. Nevertheless, in order to perform multi-camera tracking, the information that is sent through the network is the following:

- Histogram: the detection of the tracked object is performed by one device of the network, which extracts the Hue histogram and sends the histogram bins to the other devices. Tracking in the other devices starts automatically when the histogram is received.

- Trajectories: every camera computes CAMSHIFT obtaining the track window (center, size and angle). This information is thus sent to the other device.

- Homography: parameters for the 3x3 view-to-view homography are exchanged between cameras when the homography is recomputed.

In the current state of our approach, we have implemented a very simple form of exchange of information, based on event distribution. When a node generates an event, this event is broadcast to the other devices using the open Bluetooth connections. Each other device stores the information received as corresponding to the most recent state.

Histogram information, composed by the Hue histogram bins obtained from the object to track, is sent to the other devices right after its computation. In our experiments, this information consists of 16 bins of 1-channel Hue values, requiring 72 bytes to store and send (64 of values plus 8 of the header information). However, this information is sent only once during the tracking process, being initiated by user interaction. The other devices start the tracking automatically after the reception of the histogram.

Regarding trajectory information, the exchange is continuous. During the tracking process, if the current device is connected to others, data about tracked object is sent as soon as they are available. Thus, this exchange of information is performed as many times per second as the throughput of the tracking algorithm. Since the requirements on synchronization accuracy in our architecture are not very strong, we assume that the latest information received is paired with the current local information at a given time, being the inaccuracy more relevant when the frame rate is lower. The trajectory information for every state of the tracking process consists of center, size and angle, being then 4 integers and 1 float.

Homography information consists of 9 parameters, which compose the 3x3 homography matrix H. Homography is computed from at least four plane-to-plane correspondences. This process is explained in more detail in Section 6.5.4. The homography matrix between two image views has to be shared every time it is recomputed, that is, when a new point in correspondence is selected, being thus four or more of them.

6.5.4 Camera calibration

Our approach focuses on cameras with overlapping views. In order to use information from the cameras to refine the single-camera tracking, cameras have to be aware of each other and to recognize where and at what abstraction level the information is fused.

Multicamera fusion thus requires at least basic calibration, although full Tsai calibration provides more information. While it would be possible to perform calibration by placing each camera in a predefined location, this is not a practical solution. The interest for our architecture relies on an automated way to calibrate the camera network. In general, calibration methods are based either on visual evidence, such as markers in overlapping fields of view, or based o motion of objects.

Our architecture calibrates a pair of smart cameras by computing the image-to-image homography. Automatic homography computation could be done from feature matching performed by using descriptors such as SIFT [73] or SURF [74]. However, in order to obtain valid correspondences, we require matched features to be coplanar, being non-coplanar features outliers for the homography computation algorithm. In indoor scenarios, for instance, many features can be detected on walls or pieces of furniture, which makes homography computation a difficult task due to the high number of outliers.

For that reason, we have decided to compute the homography using object tracks in a single plane. In video surveillance scenarios, most objects lie on or near a single ground plane [90]. Thus, this paper assumes tracked objects' centroids are roughly coplanar. Under this assumption, the position of an object that is visible in two different camera views a and b is related by a 3x3 homography H_{ab}.

$$H_{ab} \cdot p_a(t) = p_b(t) \tag{6.16}$$

where $p_a(t) = (x, y, 1)$ is the location of the object in camera a in homogeneous coordinates at time t, $p_b(t)$ the location of the same object in camera b, and H_{ab} the homography.

This approach of plane-to-plane homography computation requires that the objects involved in the calibration process are correctly tracked. If the tracking is not correct the centroid of the tracked object will be shifted from its real position, leading to inaccuracies in the homography computation process or outliers.

In the current state of our architecture, object tracking starts on demand in one camera, then the color histogram is sent to the other camera to start the tracking automatically. When both cameras are successfully tracking the object, the track information (centroid, size and angle) is exchanged at each time t, using remote and local information as correspondences for the homography estimation process.

6.5.5 Multi-camera tracking

After the homography between a pair of cameras is correctly estimated, we incorporate the information exchanged across the network to the single-camera tracker. When the tracker is unable to provide a correct track, the last information received from the other camera is then used.

In order to improve the robustness of the single-camera tracking to sudden movements or occlusions, the main idea is using the information resultant from tracking algorithm running in the other cameras as the search window in which to resume the tracker when it fails to successfully track the object. This allows the tracker to easily recover from lost tracks due to sudden movements whilst one of the cameras in the network has tracked the object correctly.

Regarding occlusion handling, the search window of the tracker is updated continuously according to the information received from cameras with fields of view where this occlusion is not taking place. Thus, the search window will be set to the real location of the object and the tracker will recover the track as soon as the object becomes visible again.

We assume that camera a has lost the tracked object. The remote information obtained from camera b consists of the centroid p_b, the area size and the 2D rotation θ_b. Let $rp_b(t,i) = (x,y,1)$ with $i = 1..4$ be the points which determine the bounding box for the ellipse tracked by camera b in time t. Then, warped points $wp_a(t,i)$ are computed as follows

$$wp_a(t,i) = H_{ab}^{-1} rp_b(t,i) \tag{6.17}$$

where H_{ab}^{-1} is the inverse of the homography H_{ab} computed in Section 6.5.4. These warped points correspond to the region in camera a where the object successfully tracked by camera b is located. However, the warped points $wp_a(t,i) i = 1..4$ define a non-rectangular area due to the homographic transformation, which is not suitable as search window for the tracker. For that reason, instead of using these points directly, we compute the rectangular bounding box of $wp_a t, i$ and use it as search window for the next iteration of CAMSHIFT tracker.

Figures 6.10 and 6.11 shows a scheme that illustrates the usage of remote information and an example of information sharing while tracking a simple object.

The detailed multi-camera tracking process is given in Algorithm 5.

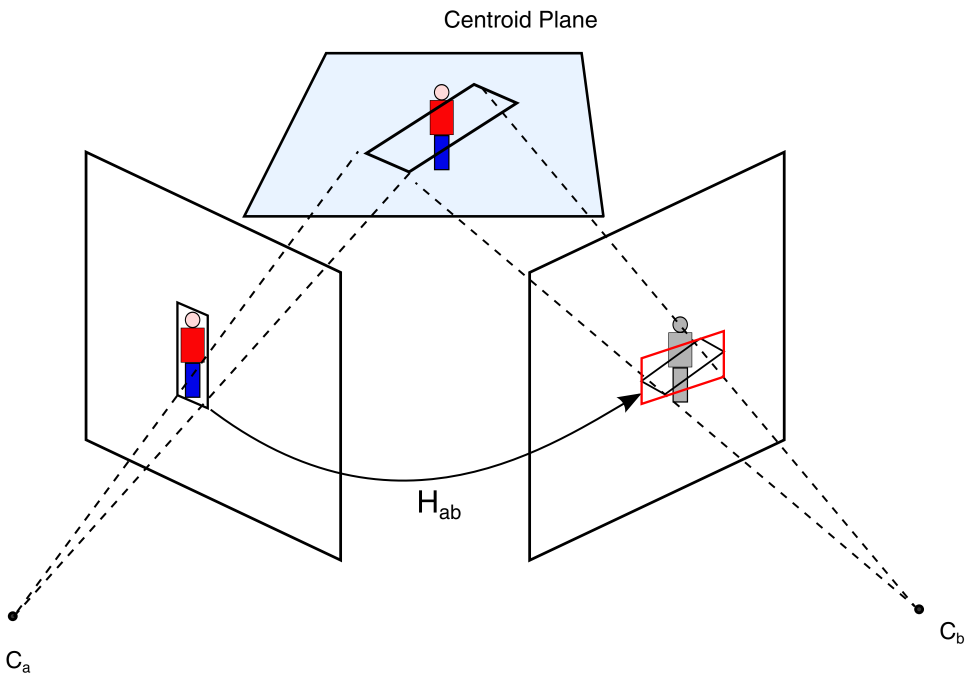


Figure 6.10: Scheme of multi-camera tracking via homography. The object in camera b is not correctly tracked, whilst tracking is successful in camera a. The bounding box in camera a is warped into a quadrilateral in camera b by means of H_{ab}, giving the tracker associated to camera b an approximate location of the object of interest. The track window in b is set to the bounding box of the quadrilateral, which is highlighted in red.

6.6 Experimental results

Our focus for application scenarios is mostly on indoor video surveillance, where smartphones can be easily deployed. In addition, video analytics applications require longer lifetime than what a smartphone battery is able to provide, thus needing access to a permanent power supply.

For the evaluation of the architecture we performed experiments with a commercial Samsung Galaxy S II smartphone and a Samsung Galaxy Tab 10.1 tablet. Our implementation platform bases upon Android SDK and NDK running over Android 4.0.1 (Ice Cream Sandwich). The frame rate is betweem 8 and 9 frames per second with images of resolution 640x480. The native code has been optimized, although the OpenCV library is still in development for the Android platform [17], and the model implemented is computationally expensive despite being a more basic tracker than particle

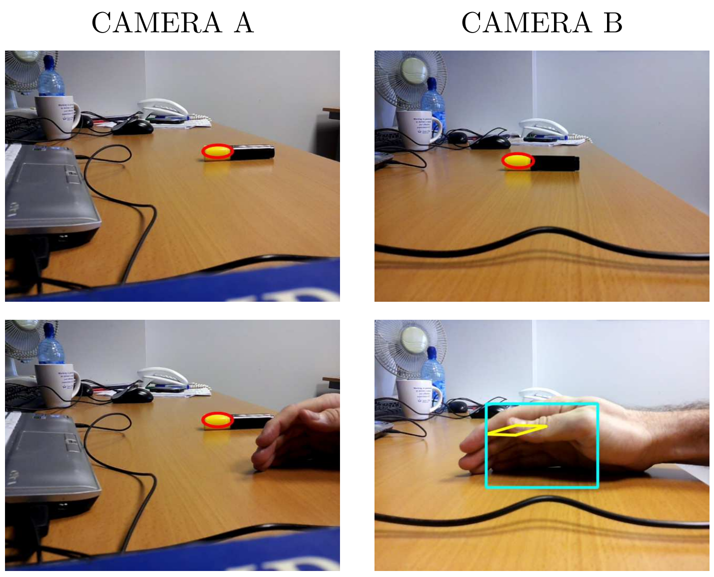


Figure 6.11: Simplified example of tracking using view-to-view homography information. The object location and bounding box in camera a are sent to camera b to incorporate the warped bounding box information (yellow quadrilateral) in the tracker. The blue rectangle indicates that the tracker hast lost the object, and the track window needs to be reset.

filtering or KLT. The architecture has been developed using Eclipse [159] for the Java elements, and g++ over cygwin for the native code.

As described in Section 6.5.2, we use Bluetooth to establish the communication between the two nodes. Since latency in Bluetooth communications is mainly dominated by opening and closing connections [101], we have left the connection established during the tracking tests.

In order to evaluate the performance of the architecture with the multi-camera tracker approach described in Section 6.5, we have performed qualitative tests checking the adaptability of the tracker to change of target, objects leaving the field of view, long occlusions and sudden large movements. Our system shows on-screen information in three different ways: when the tracking is successful, a red ellipse is shown around the object; a blue rectangle means the tracker has completely lost the tracked object (it resets the search window); finally, the yellow quadrilateral corresponds to the warped points obtained from the bounding box of the other camera, using the homography H_{ab} between both cameras. This quadrilateral is thus the location of the occluded object in the view.

Algorithm 5 Multi-camera tracking algorithm, cameras a and b

```
loop
  $frame \leftarrow CaptureFrame(a)$
  $(p_a, size_a, \theta_a) \leftarrow CAMSHIFT(frame, window)$
  if $size > 0$ then
    {Tracking successful}
    $Send(p_a, size_a, \theta_a)$
    $window \leftarrow BoundingBox(p_a, size_a, \theta_a)$
  else
    {Tracking failed}
    $Receive(p_b, size_b, \theta_b)$
    $rp_b(i) \leftarrow BoundingBox(p_b, size_b, \theta_b)$
    for $i = 1 \rightarrow 4$ do
      $wp_a(i) = H_{ab}^{-1} \cdot rp_b(i)$
    end for
    $window \leftarrow wp_a(i); i = 1..4$
  end if
end loop
```

We show, in Figure 6.12, an example of the flexibility that the usage of an attentional model offers. At the beginning the tracker was monitoring the luggage. However, when the tracked object stopped moving, we detected a new object of interest using the top-down attention model with the skin detector, following the person to check how far from the luggage he goes.

Figure 6.13 shows the behavior of the tracker when the tracked object leaves the field of view of one of the cameras, whilst is correctly tracked by the other. Since the actual location of the object is exchanged through the network, our tracker updates the search window according to it. Due to this, the object is relocated as soon as it appears in the camera field of view.

Regarding occlusion handling, Fig. 6.14 shows a different experiment where the object is completely occluded in one of the views (camera A, on the left side). In this test the images are not of consecutive frames, but on frames that show interesting changes. In the first image, we can see the entire scene, with the two devices in the left border and the top-right corner. The second image (first frame) shows the object being successfully tracked by both cameras. The next two frames show how the actual location of the occluded object is updated by means of remote information, since the yellow quadrilateral changes to reflect the movements of the object. In the last frames, the object appears again in a different region than the one where it has disappeared. As soon as the object becomes visible, remote information allows camera A to locate it in that precise instant, refining the tracking when the whole object is visible.

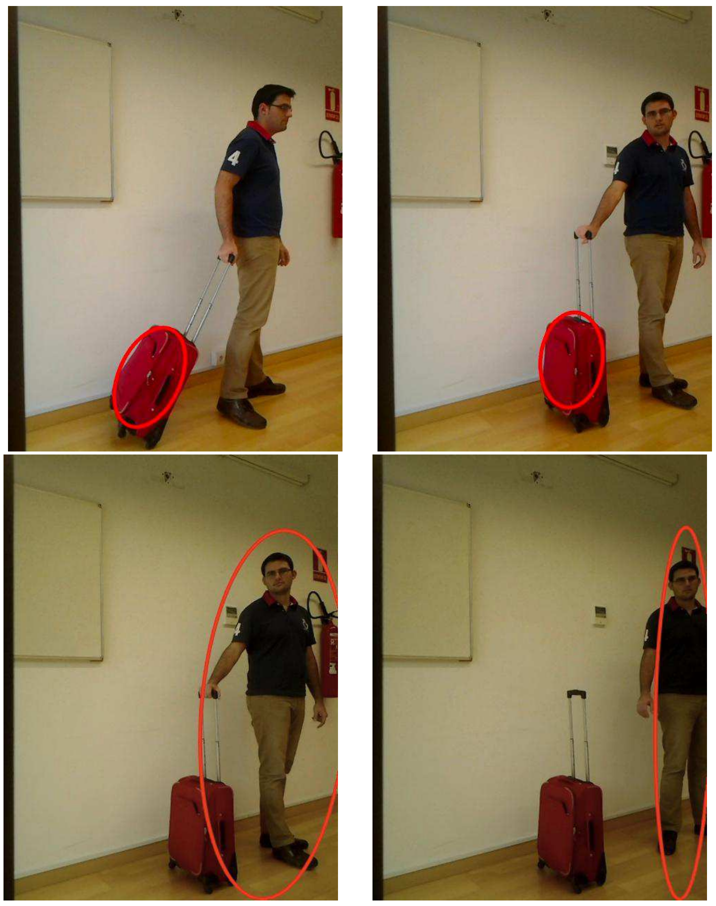


Figure 6.12: Flexibility of attentional tracker. The use of an attentional model to detect objects of interest allows for adapting the tracker, choosing a new object to track if we consider it more interesting.

Fig. 6.15 shows four consecutive frames of one test with the two devices. In the first frame, camera A is correctly tracking the object, which is out of the field of view of camera B. In the second frame, camera B finds the object by means of remote information (yellow shape in previous frame), while camera A loses the object due to sudden movement and occlusion. The third frame shows the information provided by camera B to camera A, which allows the latter to locate the object again in the last frame.

According to the experiments, the multi-camera tracking algorithm is able to recover from occlusions and lost tracks whilst one of the cameras keeps tracking the object successfully. For that reason, the main requirement

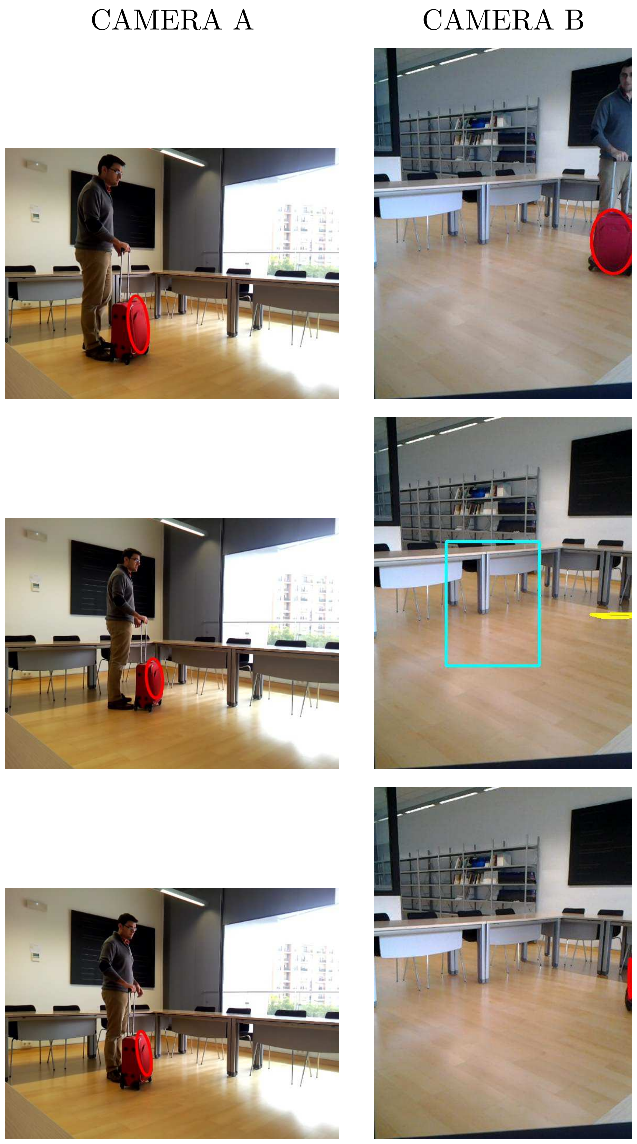


Figure 6.13: Object leaving the field of view and entering it again. Camera *B* loses track of the object when it leaves the field of view, but it keeps information about its true location sent by camera *A* (the yellow quadrilateral is the closest the tracker can get to the location of the object). When the object reappears in the field of view of camera *B*, the tracker recovers despite the object is barely visible. It must be remarked that the fields of view are fairly different, affecting not only to scene geometry but also to lighting conditions.

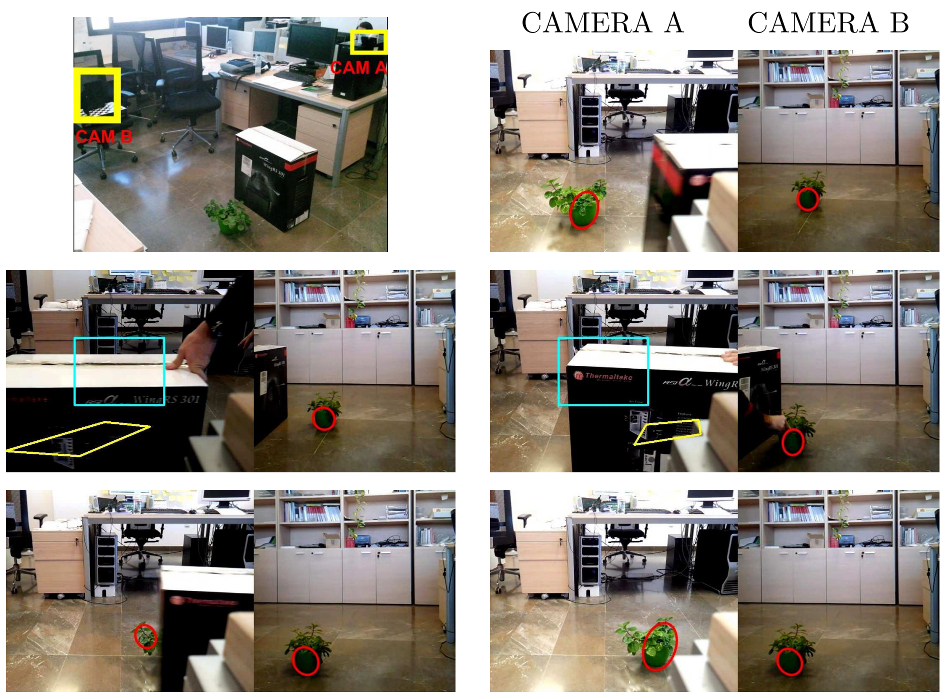


Figure 6.14: Behavior of the tracking in the presence of occlusions. The object is completely occluded in camera A, depending the tracker on remote information provided by camera B. Movements of the object are tracked by camera B and forwarded to camera A, allowing it to update the object's location and the search window of the CAMSHIFT tracker. Thus, the object is located and tracked in the first moment that it becomes visible again.

of this architecture is having a reliable tracker that minimizes the possibility of losing the objects in all cameras of the network at the same time.

One issue that we can find in the current version of the architecture involves coping with false positive and misdetections. If the cameras are tracking an object, and one of the trackers is incorrect, the information that the camera will broadcast to the network will be incorrect too. Then, if any other camera of the network has a lost track or occlusion, it will use remote information to update the location of the object, leading to greater errors. Apart from the use of better single-camera trackers that would reduce these errors, we are interested on evaluating the reliability of the information that every camera is providing before using it (in multi-camera setups with more than two cameras).

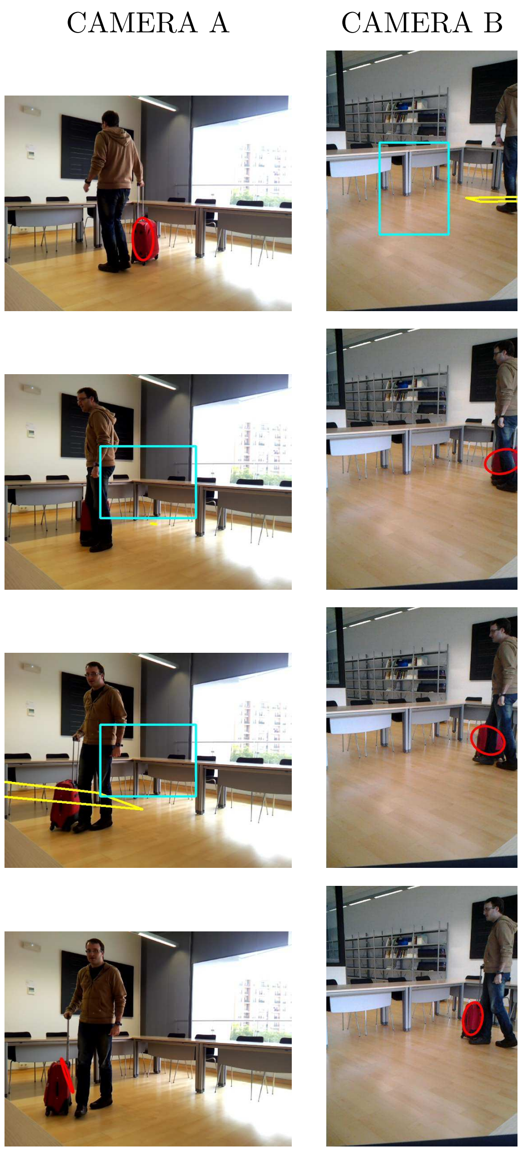


Figure 6.15: Increase in robustness of single-camera tracker against lost tracks. While any of the cameras tracks the object, the other recovers instantly from lost tracks by setting the search window to the actual location of the object.

6.7 Conclusions

In this chapter, we have presented our approach to multi-camera tracking on smartphone architectures, using them as embedded smart cameras. In comparison with other approaches in the literature, we focus on the use of commercial smartphones using Android as operating system. Smartphones have several advantages, such as having reasonably good cameras, powerful ARM processors despite being embedded systems, different types of connectivity and sensors. Furthermore, the high sales of these devices lead to lower prices than custom-built hardware.

We have implemented a prototype of our architecture and used it to track an object simultaneously with two cameras, with real-time exchange of information between the devices. Instead of computing a full Tsai calibration, homography between two cameras is estimated by using tracked object trajectories. We illustrate with an example how the tracking information from one device can be shared with the second device, allowing it to handle occlusions and recover from lost tracks.

Moreover, we have also included an automatic bio-inspired method for the selection of objects of interest. Our proposal is based on a visual attention method that conjugates bottom-up inherent saliency for the selection of regions of interest and top-down task-dependent conscious modulation. Additionally, this module is also used to increase the robustness of the method. It helps the tracking application to recover the target after being lost due to total occlusions or sudden large movements, in the same way that homographies between the two cameras are used. Finally, as our target application is related with video surveillance, we have included a skin detector operator as top-down modulation since the possible aims of the work usually involve people actions.

The work that has been presented in this chapter focuses on multi-camera platforms and calibration of the camera setup by means of homographies, as well as the usage of attention models on a real application field such as video surveillance.

Chapter 7

Conclusions

This doctoral thesis shows our contributions to the areas of computer vision and video surveillance. This chapter is structured as follows. Firstly, we present a general discussion of the problems that have motivated the development of this thesis and the proposed solutions. Secondly, several suggestions for future work are included. Finally, we enumerate the publications derived from our work and highlight the main contributions achieved.

7.1 General discussion

Active video surveillance systems analyze the contents of video streams provided by each camera, segmenting the image in foreground and background, detecting and tracking objects. Besides these basic tasks, diverse high-level analyses are performed which indicate whether the situation is normal or not. Thus, human observers can focus their attention on potential alarm situations. Accuracy and robustness are key to this kind of systems, since errors can lead to false alarms or undetected alarm situations.

The first stage of video surveillance systems usually is the extraction of background in a video sequence. A large part of this thesis has been focused on background subtraction methods, presenting real-time implementations on embedded hardware or developing methods which fuse image and depth information.

We have designed and validated two architectures to carry out video segmentation on FPGA, based on well-known background subtraction algorithms. The first architecture implements an unimodal static algorithm [27] which is less complex than the methods that define the state of the art. This lack of complexity enables its implantation on low-cost FPGA platforms with constrained energy consumption. In addition, due to the simplicity of the algorithm, both morphological filtering, connected components encoding and a shadow detection and discrimination stage have been incorporated.

The second architecture implements a multimodal dynamic algorithm [38] which has greater complexity than all previous hardware implementations of background subtraction methods.

We have evaluated both solutions by means of a dataset which has been widely used in the literature [44]. The two architectures have been validated considering accuracy and quality of the segmentation as well as processing speed and energy consumption. The comparison with previous embedded hardware implementations is focused on the image resolution, throughput and energetic requirements. In this comparison, the first solution represents a considerable improvement over previous works, processing 32.8 fps with resolution 1024×1024, and a consumption of 5.76 Watt. The second solution processes 24 images per second, with a consumption of 5.13 Watt.

The evaluation of the accuracy shows that the first architecture performs better than previous solutions on reconfigurable hardware in scenarios with shadows and objects with colors similar to those of the background, and it gets acceptable results in the other sequences. Besides the evaluation and comparison by means of objective metrics, we have validated the shadow detection and discrimination capability of the architecture, which offers good results despite the degradation due to hardware constraints. The second architecture obtains very good results in every test, better than the other hardware-oriented algorithms in most of the cases, comparable to more advanced algorithms oriented to software platforms.

Besides the study and evaluation of models suitable for embedded hardware, we have researched the integration of depth estimations in background subtraction models. This study is motivated by the fact that video segmentation algorithms continue dealing with classic issues derived from image capture process. These issues can be more effectively solved by having depth information. We have integrated depth information obtained with multi-camera stereo vision and active depth sensors such as Kinect.

We have developed several versions of background subtraction models which combine color and depth, based on the Codebook algorithm [38]. We have chosen this algorithm because we have worked with it previously in architectures on FPGA. In addition, it offers an appropriate balance between accuracy and requirements related to computational cost. We have studied three models which differ in the degree of information fusion: the first one includes depth as an independent channel; the second one takes into account the detection by depth to bias the classification based on color; the last one suppresses error regions produced by noise in depth information using the detection by color.

In order to carry out an objective and quantitative evaluation, we have built two datasets which include video sequences, depth estimations and information about ideal segmentation. These datasets have been recorded by means of Kinect in the first case, and two stereo cameras combined with disparity estimation algorithms in the second one. Both datasets are publically available for the research community at [144].

We have evaluated the proposed approaches by means of these datasets. The results show considerable improvement over previous algorithms both with depth information provided by Kinect and with the one provided by stereo disparity algorithms. In the dataset recorded with Kinect, the proposed method (*DECB*) obtains the best average values for the metric F_1 in every sequence. In the sequences recorded with stereo cameras, the third proposed approach, which includes a post-processing fusion stage (*DECB-LF*), gets better results than the previous method without post-processing (average improvement 7.5% over *DECB*), because of the elimination of noise produced by disparity estimation.

Continuing with vision models applied to video surveillance, we have studied the use of multi-camera system with smartphones to detect and track objects of interest in a scene. We have implemented a prototype of our architecture to perform object tracking simultaneously with two cameras, with information sharing in real time between devices. Multi-camera setups require some sort of calibration to exchange information processed by each one and work with it globally. Thus, we have used the trajectories of the tracked objects to compute correspondences between cameras and calculate the homography from a camera to the other.

We have studied the exchange of different types of information, such as appearance of the object of interest (color histogram, keypoints and descriptors, etc.) and tracking information (location, size and orientation of the object). The usage of information shared between devices allows the system to solve occlusions, when the object of interest is not visible in the field of view of one camera but it is in the other one, as well as to continue the tracking after a lost track. We have illustrated the functioning of the system with several examples under controlled conditions.

In addition, we have included an automatic bio-inspired method to detect and select the object of interest. Since our main goal is the application to video surveillance, a skin detector has been included as *top-down* modulator of attention, because the tasks are usually centered on actions performed by people. The inclusion of attention dependent of the task to carry out (*top-down*) allows the multi-camera tracker to acquire new objectives which may be of interest at a given moment. It also can help the tracker to recover from lost tracks by relocating the object.

To sum up, we have worked in different computer vision models applied to video analytics, focusing on quality evaluation methodologies, comparison with previous works, and real-time implementation on several kinds of embedded systems.

7.2 Future work

As future work, we intend to continue with the line of multi-camera systems for video surveillance.

On the one hand, we intend to improve the process of mono-camera tracking, by using more advanced models than the color-based one used in chapter 6, such as KLT or particle filtering. In addition, we want to combine the tracker with the foreground/background segmentation stage.

On the other hand, we will work on multi-camera setups with multiple nodes as well as central processing servers, combining different types of connectivity (such as Bluetooth and Wi-Fi) for the communications through the network. Our intent with these multi-camera configurations is not only the exchange of information through the network, but also to provide a reliability measure of that information in order to avoid using incorrect data. Besides, we intend to study the balance between distributed and centralized processing. We want to study network topologies which include cameras with overlapping views, as we have seen in chapter 6, and non-overlapping views in which the goal is tracking an object from a view to another (problem known as *person re-identification*).

Finally, we would like to continue with further processing stages of video analytics systems, performing high-level analysis of the information provided by segmentation and tracking to automatically detect situations of interest, and direct the system to solve specific problems.

7.3 Publications

The published or submitted works related to this doctoral thesis are the following:

7.3.1 International journals with scientific impact

- Rodriguez-Gomez R., Fernandez-Sanchez E.J., Diaz J., Ros E. FPGA Implementation for Real-Time Background Subtraction Based on Horprasert Model. Sensors. 2012; 12(1):585-611.

- Rodriguez-Gomez R., Fernandez-Sanchez E.J., Diaz J., Ros E. Codebook hardware implementation on FPGA for background subtraction. Real-Time Image Processing, Journal of. 1-15.

- Fernandez-Sanchez E.J., Diaz J., Ros E. Background subtraction model based on color and depth cues. Submitted to Special Issue on "Background Modeling for Foreground Detection in real-world dynamic scenes". Journal of Machine Vision and Applications

- Fernandez-Sanchez E.J., Diaz J., Ros E. Video segmentation based on color and depth using Kinect. Submitted to Journal of Visual Communication and Image Representation.

7.3.2 National conference

- Rodríguez-Gómez R., Fernández-Sánchez E.J., Rat B., Agis R., Diseño de una arquitectura Hw/Sw en FPGA para la sustracción del fondo en secuencias de video. III Congreso Español de Informática (CEDI 2010), X Jornadas de Computación Reconfigurable y Aplicaciones (JCRA2010), 7-10 Septiembre 2010. Valencia (Spain). Pp.135-142. ISBN 978-84-92812-56-1

7.4 Main contributions

In this section, we include a summary of the main contributions achieved in this Ph.D work:

- We have modified models and evaluated two architectures on FPGA to perform background subtraction on video sequences. The first one, based on a static algorithm, includes a shadow detection and discrimination stage, which is a novelty regarding hardware implementations. The second one is based on a complex, multimodal and dynamic algorithm, better suited for the wide variety of scenarios.

- We have validated both architectures, performing a comparison with previous approaches described in the literature. We have evaluated the performance concerning real-time constraints, hardware resources and energy consumption. We have also evaluated the quality of the segmentation by means of widely used datasets, obtaining satisfactory results.

- We have studied the integration of depth estimations in a background subtraction model. The depth information is provided both by multi-camera stereo vision systems and active depth sensors such as Kinect.

- We have studied three depth fusion methods in a specific background subtraction model. These methods differ in the degree of integration of depth and color, and they aim to solve certain issues which affect color-based algorithms.

- We have built two new datasets to validate the studied methods, since there was no dataset suitable for depth-based algorithms in video surveillance scenarios. We have created two sets of sequences, the first one recorded by Kinect, and the second one recorded by stereo cameras and disparity estimation algorithms. Both datasets have been made available for the research community.

- We have evaluated the proposed approaches using the previously mentioned datasets. Our methods outperform previous algorithms in both sets of sequences, being specially noticeable when using information provided by Kinect.

- We have studied the usage of a multi-camera system with smartphones to detect and track objects of interest. The feasibility of these devices has been studied both regarding hardware and connectivity and available development tools. We have implemented a prototype of an architecture to perform object tracking simultaneously with two cameras, with real-time information exchange between different devices.

- We have used information of tracked object trajectories to perform a basic scene calibration. This calibration allows for globally using the information obtained by each camera independently.

- We have studied the exchange of different types of information across the network, such as information about the appearance of the object of interest, tracking information or parameters related to the calibration of the scene. We have used the information shared between devices to solve occlusions when the object of interest is visible in only one camera, and to increase the robustness of the system to momentary lost tracks.

- We have included an automatic bio-inspired method to detect and select the object of interest. This enables modulating the attention of the system depending on the task to perform or the kind of objects which needs to be detected. More specifically, we have integrated a skin detector to focus on detecting people in the scene.

Chapter 8

Conclusiones

Esta tesis doctoral muestra nuestras aportaciones a las áreas de visión artificial y videovigilancia. Este capítulo está estructurado de la siguiente manera. En la primera sección, presentamos una discusión general de los problemas que han motivado el desarrollo de esta tesis y las soluciones propuestas. A continuación se incluyen algunas sugerencias para trabajo futuro. Después enumeramos las publicaciones derivadas de nuestro trabajo y, para finalizar, destacamos las principales aportaciones realizadas.

8.1 Discusión general

Los sistemas de videovigilancia activa analizan el contenido del vídeo procedente de cada cámara de vigilancia segmentando la imagen en primer plano y fondo, detectando objetos y haciendo un seguimiento de los mismos. Además de estas tareas más básicas, se llevan a cabo diversos análisis de más alto nivel cuyos resultados indican si la situación que está teniendo lugar es normal o anómala, de modo que el vigilante humano pueda centrar su atención en situaciones de alarma. La precisión y robustez es clave en este tipo de sistemas, ya que los fallos pueden dar lugar a falsas alarmas o situaciones de alarma no detectadas.

La primera etapa de un sistema de videovigilancia suele ser la extracción de fondo en una secuencia de vídeo. Gran parte de esta tesis doctoral se ha centrado en los métodos de extracción de fondo, presentando tanto implementaciones en tiempo real sobre hardware empotrado como desarrollando métodos que integran imagen e información de profundidad.

Hemos diseñado y validado dos arquitecturas para llevar a cabo la segmentación de vídeo en FPGA, basadas en algoritmos de sustracción de fondo bien conocidos. La primera arquitectura implementa un algoritmo estático unimodal [27], de menor complejidad a los métodos que conforman el estado del arte. Esto permite su implantación en plataformas FPGA de bajo coste y con un consumo de energía contenido. Además, debido a la mayor sencillez del algoritmo, se han podido incorporar filtros morfológicos, codificación de componentes conexas, y una etapa de detección y discriminación de sombras.

La segunda arquitectura implementa un algoritmo dinámico multimodal [38], cuya complejidad es mayor a la de todas las implementaciones hardware de métodos de sustracción de fondo realizadas con anterioridad.

Hemos evaluado ambas soluciones por medio de secuencias de test ampliamente utilizadas en la literatura [44]. Se han validado las dos arquitecturas teniendo en cuenta tanto la precisión y calidad de la segmentación como la velocidad de procesamiento y el consumo energético. La comparación con anteriores implementaciones en hardware empotrado se ha centrado en la resolución a la que trabajan, el número de imágenes por segundo, y el consumo energético. En esta comparativa, la primera solución representa una mejora considerable con respecto a trabajos anteriores, procesando 32.8 imágenes por segundo con resolución 1024×1024, y un consumo de 5.76 vatios. La segunda solución procesa 24 imágenes por segundo, con un consumo de 5.13 vatios.

La evaluación de la calidad de la segmentación muestra que la primera arquitectura se comporta mejor que soluciones anteriores en hardware reconfigurable en situaciones con sombras y objetos similares al fondo, y obtiene resultados aceptables en las demás secuencias. Aparte de la evaluación y comparación utilizando métricas objetivas, hemos validado la capacidad de detección y discriminación de sombras de la arquitectura, que ofrece buenos resultados a pesar de la degradación producida por las limitaciones del hardware. La segunda arquitectura obtiene muy buenos resultados en todos los tests, superiores en la mayoría de los casos a las otras alternativas hardware, y comparables a algoritmos complejos orientados a plataformas software.

Además del estudio y la evaluación de modelos para implementaciones en hardware empotrado, hemos investigado la integración de estimaciones de profundidad en modelos de extracción de fondo. La motivación para este estudio es que los algoritmos de segmentación de vídeo siguen enfrentándose a problemas clásicos derivados de la captura de imagen, que teniendo información de profundidad pueden ser solventados de forma más eficaz. Hemos integrado información de profundidad obtenida con visión estereoscópica multicámara y mediante sensores de profundidad activos como Kinect.

Hemos desarrollado varias versiones de modelos de sustracción de fondo que combinan color y profundidad, basadas en el algoritmo Codebook [38]. Hemos seleccionado este algoritmo por haber trabajado con él anteriormente en las arquitecturas para FPGA, y porque ofrece un equilibrio adecuado entre precisión y requisitos relacionados con el coste computacional. Hemos estudiado tres modelos que difieren en su grado de fusión de información: el primero incorpora la profundidad como un canal independiente; el segundo tiene en cuenta la detección por profundidad para modificar la clasificación basada en color; el último suprime regiones de error producidas por ruido en la información de profundidad, utilizando la detección por color.

Para poder llevar a cabo una evaluación objetiva y cuantitativa, hemos construido dos conjuntos de secuencias que incluyen tanto los vídeos como las estimaciones de profundidad e información de la segmentación ideal. Dichos conjuntos de secuencias han sido grabados utilizando Kinect en un caso, y dos cámaras estéreo junto a algoritmos de estimación de disparidad en el otro. Ambos conjuntos de prueba han sido puestos a disponibilidad de la comunidad investigadora en [144].

Hemos evaluado los métodos propuestos utilizando estos bancos de prueba, mostrando los resultados una mejora considerable con respecto a algoritmos anteriores tanto con información de profundidad proporcionada por Kinect como con la obtenida mediante algoritmos de disparidad estéreo. En el conjunto de pruebas grabado utilizando Kinect, el método propuesto (*DECB*) obtiene los mejores valores medios de la métrica F_1 para todas las secuencias. En las secuencias grabadas con cámaras estéreo y algoritmos de disparidad, el último método propuesto, que incorpora una etapa de fusión en post-procesado (*DECB-LF*), obtiene mejores resultados que el método anterior sin post-procesado (mejora media de 7.5% sobre *DECB*), gracias a la supresión de ruido provocado por la disparidad.

Continuando con modelos aplicados a videovigilancia, hemos estudiado la utilización de un sistema multicámara con smartphones para la detección y seguimiento de objetos de interés en una escena. Hemos implementado un prototipo de nuestra arquitectura para llevar a cabo el *tracking* de un objeto simultáneamente con dos cámaras, con intercambio de información en tiempo real entre los distintos dispositivos. Las configuraciones con varias cámaras requieren cierta calibración para poder poner la información procesada por cada una en común. Así, hemos utilizando las trayectorias de objetos bajo seguimiento para calcular correspondencias entre cámaras y calcular la homografía de una cámara a otra.

Hemos estudiado la compartición de diversos tipos de información, tales como información de la apariencia del objeto de interés (histograma de color, puntos característicos y descriptores, etc.) o información de *tracking* (posición, tamaño y orientación del objeto). La utilización de información compartida entre dispositivos permite al sistema la resolución de oclusiones, cuando el objeto de interés no es visible en el campo de visión de una cámara pero sí lo es en la otra, así como continuar el *tracking* tras haber perdido el objeto momentáneamente. Hemos ilustrado el funcionamiento del sistema con una serie de ejemplos bajo condiciones controladas.

Además, hemos incorporado un método automático bio-inspirado para la detección y selección inicial del objeto de interés. En nuestro caso, dado que nuestro principal objetivo es la aplicación a videovigilancia, se ha incluído un detector de piel como modulación *top-down*, ya que las tareas suelen centrarse en acciones llevadas a cabo por personas. Esta incorporación de atención dependiente de la tarea a realizar (*top-down*) permite a la aplicación de *tracking* multicámara adquirir nuevos objetivos que resulten de interés en un momento dado, así como recuperar el objeto de interés en casos en que el *tracker* se haya perdido completamente.

En resumen, hemos trabajado en distintos modelos de visión artificial aplicados a videovigilancia, centrándonos en metodologías de evaluación de calidad de los mismos, comparación con trabajos previos, y su implementación en tiempo real en distintos tipos de sistemas empotrados.

8.2 Trabajo futuro

Como trabajo futuro, nos planteamos continuar con la línea de sistemas multicámara para videovigilancia.

Por un lado, pretendemos mejorar el proceso de *tracking* monocámara, utilizando modelos más avanzados que el modelo de color utilizado en el capítulo 6, tales como KLT o filtros de partículas, además de combinar el *tracking* en sí con la etapa de segmentación en primer plano y fondo.

Por otro lado, trabajaremos en configuraciones multi-cámara con múltiples nodos así como servidores centrales de procesamiento, combinando distintos tipos de conexiones (como Bluetooth y Wi-Fi) para las comunicaciones en la red. Con estas configuraciones multi-cámara, nuestra intención es no sólo el intercambio de información a través de la red, sino también proporcionar una medida de fiabilidad de dicha información para evitar utilizar información incorrecta, además de llevar a cabo estudios de equilibrio entre procesamiento distribuido y centralizado. Dentro de las distintas configuraciones posibles de la red, queremos estudiar topologías que incluyan cámaras con vistas solapadas, como en el capítulo 6, y cámaras con vistas no solapadas en las que se trata de seguir a un objeto de una vista a otra (problema conocido como *re-identificación de personas*).

Finalmente, nos gustaría continuar con posteriores etapas de procesamiento del sistema de videovigilancia, realizando análisis de alto nivel de la información proporcionada por la segmentación y *tracking* para detectar situaciones de interés de forma automatizada, y orientar el sistema a resolver problemas específicos.

8.3 Publicaciones

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

8.3.1 Revistas Internacionales con Índice de Impacto

- Rodriguez-Gomez R., Fernandez-Sanchez E.J., Diaz J., Ros E. FPGA Implementation for Real-Time Background Subtraction Based on Horprasert Model. Sensors. 2012; 12(1):585-611.
- Rodriguez-Gomez R., Fernandez-Sanchez E.J., Diaz J., Ros E. Codebook hardware implementation on FPGA for background subtraction. Real-Time Image Processing, Journal of. 1-15.
- Fernandez-Sanchez E.J., Diaz J., Ros E. Background subtraction model based on color and depth cues. Submitted to Special Issue on "Background Modeling for Foreground Detection in real-world dynamic scenes". Journal of Machine Vision and Applications
- Fernandez-Sanchez E.J., Diaz J., Ros E. Video segmentation based on color and depth using Kinect. Submitted to Journal of Visual Communication and Image Representation.

8.3.2 Conferencias nacionales

- Rodríguez-Gómez R., Fernández-Sánchez E.J., Rat B., Agis R., Diseño de una arquitectura Hw/Sw en FPGA para la sustracción del fondo en secuencias de video. III Congreso Español de Informática (CEDI 2010), X Jornadas de Computación Reconfigurable y Aplicaciones (JCRA2010), 7-10 Septiembre 2010. Valencia (Spain). Pp.135-142. ISBN 978-84-92812-56-1

8.4 Aportaciones principales

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

- Hemos modificado modelos y evaluado dos arquitecturas en FPGA para llevar a cabo la extracción del fondo en secuencias de vídeo. La primera, basada en un algoritmo estático, incluye una etapa de detección y discriminación de sombras, que es algo novedoso en lo que a implementaciones hardware se refiere. La segunda está basada en un algoritmo complejo, dinámico y multimodal, más apropiado para la gran variedad de escenarios que se pueden plantear.

- Hemos validado ambas arquitecturas, comparándolas con otras alternativas presentes en la literatura. Hemos evaluado su rendimiento en lo que concierne a restricciones de tiempo real, recursos hardware y consumo energético. También hemos evaluado la calidad de la segmentación por medio de bancos de prueba ampliamente utilizados en la literatura, obteniéndose resultados satisfactorios.

- Hemos estudiado la integración de estimaciones de profundidad en un modelo de sustracción de fondo procedente de sistemas de visión estereoscópica multicámara y sensores de profundidad activos como Kinect.

- Hemos estudiado tres métodos de fusión de profundidad en un modelo concreto de sustracción de fondo. Dichos métodos difieren en el grado de integración de la información de color y profundidad, y se orientan a resolver determinados problemas asociados a modelos que usan únicamente color.

- Hemos construido nuevos bancos de pruebas para la validación de los métodos estudiados, ya que no existía ninguno apropiado para comprobar el comportamiento de estos algoritmos en escenarios de videovigilancia. Hemos creado dos conjuntos de secuencias, el primero grabado con el sensor Kinect, y el segundo con cámaras estéreo utilizando algoritmos de estimación de disparidad. Ambos conjuntos de prueba han sido puestos a disponibilidad de la comunidad investigadora.

- Hemos evaluado los métodos propuestos utilizando los bancos de prueba mencionados anteriormente. Nuestros métodos obtienen una mejora considerable con respecto a algoritmos anteriores en ambos conjuntos de secuencias, siendo más significativo con información de profundidad proporcionada por Kinect.

- Hemos estudiado la utilización de un sistema multicámara con *smartphones* para la detección y seguimiento de objetos de interés. Se ha estudiado tanto la viabilidad de estos dispositivos desde el punto de vista de hardware y conectividad como en lo referente a herramientas disponibles. Hemos implementado un prototipo de una arquitectura para llevar a cabo el *tracking* de un objeto simultáneamente con dos cámaras, con intercambio de información en tiempo real entre los distintos dispositivos.

- Hemos utilizado información relativa a trayectorias de objetos bajo seguimiento para hacer una calibración básica de la escena. Esto permite utilizar de forma conjunta la información obtenida por cada cámara independientemente.

- Hemos estudiado la compartición de información de diversos tipos a través de la red, como información de la apariencia del objeto de interés, información de *tracking* o información de calibración de la escena. Hemos utilizado la información compartida entre dispositivos para resolver oclusiones cuando el objeto de interés no es visible en una cámara pero sí en otra, así como aumentar la robustez del sistema a pérdidas momentáneas del *tracker*.

- Hemos incorporado un método automático bio-inspirado para la detección y selección inicial del objeto de interés. Esto permite modular la atención del sistema dependiendo de la tarea a realizar o del tipo de objetos que se desea detectar. Concretamente, hemos integrado un detector de piel para la detección de personas en la escena.

Bibliography

[1] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov, "Computer vision workload analysis: Case study of video surveillance systems.," *Intel Technology Journal - Compute-intensive, highly parallel applications and uses*, vol. 9, pp. 109–118, 2005.

[2] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, pp. 1151 – 1163, jul 2002.

[3] I. Haritaoglu, D. Harwood, and L. Davis, "W4s: A real-time system for detecting and tracking people in 2 1/2d," in *Computer Vision ECCV'98* (H. Burkhardt and B. Neumann, eds.), vol. 1406 of *Lecture Notes in Computer Science*, pp. 877–892, Springer Berlin Heidelberg, 1998.

[4] P. Rosin, "Thresholding for change detection," in *Computer Vision, 1998. Sixth International Conference on*, pp. 274–279, IEEE, 1998.

[5] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pp. 175–181, Morgan Kaufmann Publishers Inc., 1997.

[6] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 780–785, 1997.

[7] J. Ohya, J. Kurumisawa, R. Nakatsu, K. Ebihara, S. Iwasawa, D. Harwood, and T. Horprasert, "Virtual metamorphosis," *MultiMedia, IEEE*, vol. 6, no. 2, pp. 29–39, 1999.

[8] J. Davis and A. Bobick, "The representation and recognition of human movement using temporal templates," in *Computer Vision and*

Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on, pp. 928–934, IEEE, 1997.

[9] J. Diaz, *Multimodal bio-inspired vision system. High performance motion and stereo processing architecture. PhD Dissertation.* University of Granada, July 1996.

[10] C. Weems, "Architectural requirements of image understanding with respect to parallel processing," *Proceedings of the IEEE*, vol. 79, pp. 537 –547, apr 1991.

[11] A. Rares, M. Reinders, and E. Hendriks, "Image interpretation systems," tech. rep., MCCWS project Technical Report (MCCWS 2.1.1.3.C), 1999.

[12] J. Díaz, E. Ros, A. Prieto, and F. J. Pelayo, "Fine grain pipeline systems for real-time motion and stereo-vision computation," *International Journal of High Performance Systems Architecture*, vol. 1, pp. 60–68, 2007.

[13] K. Kim and L. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *Computer Vision ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3953 of *Lecture Notes in Computer Science*, pp. 98–109, Springer Berlin / Heidelberg, 2006.

[14] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. Gool, "Color-based object tracking in multi-camera environments," in *Pattern Recognition* (B. Michaelis and G. Krell, eds.), vol. 2781 of *Lecture Notes in Computer Science*, pp. 591–599, Springer Berlin Heidelberg, 2003.

[15] S. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Computer Vision ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3954 of *Lecture Notes in Computer Science*, pp. 133–146, Springer Berlin / Heidelberg, 2006.

[16] M. Taj and A. Cavallaro, "Distributed and decentralized multicamera tracking," *Signal Processing Magazine, IEEE*, vol. 28, pp. 46 –58, may 2011.

[17] OpenCV. `http://opencv.willowgarage.com/wiki/`, 2012.

[18] OpenNI. `http://www.openni.org/`, 2012.

[19] Impulse accelerated technologies. `http://www.impulseaccelerated.com/`, 2011.

[20] Xilinx. `http://www.xilinx.com`, 2013.

[21] Microsoft Corporation. `http://www.microsoft.com/en-us/kinectforwindows/`, 2012.

[22] Google, "Android Developers." http://developer.android.com, 2012.

[23] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti, "Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking," *Signal Processing Magazine, IEEE*, vol. 22, pp. 38 – 51, march 2005.

[24] R. Collins, A. Lipton, and T. Kanade, "Special section on video surveillance," *PAMI, IEEE Transactions on*, 2000.

[25] M. Karaman, L. Goldmann, D. Yu, and T. Sikora, "Comparison of static background segmentation methods," in *Proc. SPIE 5960, 596069 (2005);*, vol. 5960, 2005.

[26] A. François and G. Medioni, "Adaptive color background moeling for realtime segmentation of video streams," in *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, pp. 227–232, 1999.

[27] D. L. Horprasert T, Harwood D, "A statistical approach for real-time robust background subtraction and shadow detection," in *IEEE Frame-Rate Applications Workshop, Kerkyra, Greece*, 1999.

[28] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42 – 56, 2000.

[29] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, "Detection and location of people in video images using adaptive fusion of color and edge information," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 4, pp. 627 –630 vol.4, 2000.

[30] A. Cavallaro and T. Ebrahimi, "Accurate video object segmentation through change detection," in *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, vol. 1, pp. 445 – 448 vol.1, 2002.

[31] D. Hong and W. Woo, "A background subtraction for a vision-based user interface," in *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, vol. 1, pp. 263 – 267 Vol.1, dec. 2003.

[32] J. Shen, “Motion detection in color image sequence and shadow elimination,” pp. 731–740, 2004.

[33] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, pp. 2 vol. (xxiii+637+663), 1999.

[34] T. Bouwmans, F. El Baf, and B. Vachon, “Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey,” *Recent Patents on Computer Science*, vol. 1, pp. 219–237, Nov. 2008.

[35] P. KaewTraKulPong and B. R., “An improved adaptive background mixture model for real-time tracking with shadow detection.,” in *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems (AVBS01)*, 2001.

[36] L. Li and M. Leung, “Integrating intensity and texture differences for robust change detection,” *Image Processing, IEEE Transactions on*, vol. 11, pp. 105 –112, feb 2002.

[37] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, “Foreground object detection from videos containing complex background,” in *Proceedings of the eleventh ACM international conference on Multimedia*, MULTIMEDIA ’03, (New York, NY, USA), pp. 2–10, ACM, 2003.

[38] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, “Real-time foregroundbackground segmentation using codebook model,” *Real-Time Imaging*, vol. 11, no. 3, pp. 172 – 185, 2005. Special Issue on Video Object Processing.

[39] T. Kohonen, “Learning vector quantization,” *Neural Networks*, vol. 1, pp. 3 – 16, 1988.

[40] A. Ilyas, M. Scuturici, and S. Miguet, “Real time foreground-background segmentation using a modified codebook model,” in *Advanced Video and Signal Based Surveillance, 2009. AVSS ’09. Sixth IEEE International Conference on*, pp. 454 –459, sept. 2009.

[41] M. Wu and X. Peng, “Spatio-temporal context for codebook-based dynamic background subtraction,” *AEU - International Journal of Electronics and Communications*, vol. 64, no. 8, pp. 739 – 747, 2010.

[42] H. Jiménez-Hernández, “Background subtraction approach based on independent component analysis,” *Sensors*, vol. 10, no. 6, pp. 6092–6114, 2010.

[43] I. Bravo, M. Mazo, J. L. Lázaro, A. Gardel, P. Jiménez, and D. Pizarro, "An intelligent architecture based on field programmable gate arrays designed to detect moving objects by using principal component analysis," *Sensors*, vol. 10, no. 10, pp. 9232–9251, 2010.

[44] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Computer Vision, IEEE International Conference on*, vol. 1, (Los Alamitos, CA, USA), p. 255, IEEE Computer Society, 1999.

[45] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: Algorithms and evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 918–923, 2003.

[46] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Computer Vision and Pattern Recognition (CVPR)*, pp. 1937–1944, IEEE, 2011.

[47] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pp. 1 –8, june 2012.

[48] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre, "A benchmark dataset for foreground/background extraction.," in *In ACCV 2012, Workshop: Background Models Challenge.*, 2012.

[49] K. Appiah and A. Hunter, "A single-chip fpga implementation of real-time adaptive background model," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, pp. 95 – 102, dec. 2005.

[50] H. Jiang, H. Ardo, and V. Owall, "Hardware accelerator design for video segmentation with multi-modal background modelling," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 1142 – 1145 Vol. 2, may 2005.

[51] J. Oliveira, A. Printes, R. C. S. Freire, E. Melcher, and I. S. S. Silva, "Fpga architecture for static background subtraction in real time," in *Proceedings of the 19th annual symposium on Integrated circuits and systems design*, SBCCI '06, (New York, NY, USA), pp. 26–31, ACM, 2006.

[52] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Real-time background generation and foreground object segmentation for high-definition colour video stream in fpga device," *Journal of Real-Time Image Processing*, pp. 1–17, 2012.

[53] S. Ierodiaconou, N. Dahnoun, and L. Xu, “Implementation and optimisation of a video object segmentation algorithm on an embedded dsp platform,” in *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pp. 432 –437, june 2006.

[54] C. Peng, “Video background/foreground detection implementation on tms320c64/64x+ dsp,” tech. rep., Texas Instruments, 2007.

[55] P. Carr, “Gpu accelerated multimodal background subtraction,” in *Computing: Techniques and Applications, 2008. DICTA ’08.Digital Image*, pp. 279 –286, dec. 2008.

[56] V. Pham, P. Vo, V. T. Hung, and L. H. Bac, “Gpu implementation of extended gaussian mixture model for background subtraction,” in *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on*, pp. 1 –4, nov. 2010.

[57] O. Javed, K. Shafique, and M. Shah, “A hierarchical approach to robust background subtraction using color and gradient information,” *Motion and Video Computing, IEEE Workshop on*, vol. 0, p. 22, 2002.

[58] B. Zhang, B. Zhong, and Y. Cao, “Complex background modeling based on texture pattern flow with adaptive threshold propagation,” *Journal of Visual Communication and Image Representation*, vol. 22, no. 6, pp. 516 – 521, 2011.

[59] M. Heikkila and M. Pietikainen, “A texture-based method for modeling the background and detecting moving objects,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 657 – 662, april 2006.

[60] Y. Tian, F. Duan, M. Zhou, and Z. Wu, “Active contour model combining region and edge information,” *Machine Vision and Applications*, pp. 1–15, 2011. 10.1007/s00138-011-0363-7.

[61] S. S. C. X. G. W. Zhong B, Yao H, “Texture and motion pattern fusion for background subtraction,” in *Proceedings 11th Joint Conference on Information Sciences*, 2008.

[62] Q. Zhang and K. N. Ngan, “Multiview video based multiple objects segmentation using graph cut and spatiotemporal projections,” *Journal of Visual Communication and Image Representation*, vol. 21, no. 5-6, pp. 453–461, 2010. Special issue on Multicamera Imaging, Coding and Innovative Display.

[63] M. Cristani, M. Farenzena, D. Bloisi, and V. Murino, "Background subtraction for automated multisensor surveillance: a comprehensive review," *EURASIP Journal on Advances in signal Processing*, vol. 2010, p. 43, 2010.

[64] Y. Ivanov, A. Bobick, and J. Liu, "Fast lighting independent background subtraction," *International Journal of Computer Vision*, vol. 37, pp. 199–207, 2000. 10.1023/A:1008107805263.

[65] G. Gordon, T. Darrell, M. Harville, and J. Woodfill, "Background estimation and removal based on range and color," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, pp. 2 vol. (xxiii+637+663), 1999.

[66] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer segmentation of binocular stereo video," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 407 – 414 vol. 2, june 2005.

[67] I. Schiller and R. Koch, "Improved video segmentation by adaptive combination of depth keying and mixture-of-gaussians," in *Image Analysis* (A. Heyden and F. Kahl, eds.), vol. 6688 of *Lecture Notes in Computer Science*, pp. 59–68, Springer Berlin / Heidelberg, 2011.

[68] R. Crabb, C. Tracey, A. Puranik, and J. Davis, "Real-time foreground segmentation via range and color imaging," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1 –5, june 2008.

[69] J. Zhu, M. Liao, R. Yang, and Z. Pan, "Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 453 –460, june 2009.

[70] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, Dec. 2006.

[71] C. Harris and M. Stephens, "A combined corner and edge detector.," in *4th Alvey Vision Conference*, pp. 147–151, 1988.

[72] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593 –600, jun 1994.

[73] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[74] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. ¡ce:title¿Similarity Matching in Computer Vision and Multimedia¡/ce:title¿.

[75] N. Cornelis and L. van Gool, “Fast scale invariant feature detection and matching on programmable graphics hardware,” in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW ’08. IEEE Computer Society Conference on*, pp. 1 –8, june 2008.

[76] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, pp. 142 –149 vol.2, 2000.

[77] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 888 –905, aug 2000.

[78] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic active contours,” *International Journal of Computer Vision*, vol. 22, pp. 61–79, 1997.

[79] C. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Computer Vision, 1998. Sixth International Conference on*, pp. 555 –562, jan 1998.

[80] H. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 23 –38, jan 1998.

[81] P. Viola, M. Jones, and D. Snow, “Detecting pedestrians using patterns of motion and appearance,” *International Journal of Computer Vision*, vol. 63, pp. 153–161, 2005.

[82] V. Salari and I. Sethi, “Feature point correspondence in the presence of occlusion,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 87 –91, jan 1990.

[83] T. J. Broida and R. Chellappa, “Estimation of object motion parameters from noisy images,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 90 –99, jan. 1986.

[84] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking,” *Signal Processing, IEEE Transactions on*, vol. 50, pp. 174 –188, feb 2002.

[85] R. L. Streit and T. E. Luginbuhl, “Maximum likelihood method for probabilistic multihypothesis tracking,” pp. 394–405, 1994.

[86] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 603 –619, may 2002.

[87] B. Moller, T. Plotz, and G. Fink, “Calibration-free camera hand-over for fast and reliable person tracking in multi-camera setups,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1 –4, dec. 2008.

[88] J. Black and T. Ellis, “Multi camera image tracking,” *Image and Vision Computing*, vol. 24, no. 11, pp. 1256 – 1267, 2006. Performance Evaluation of Tracking and Surveillance.

[89] L. Guan, J.-S. Franco, and M. Pollefeys, “Multi-object shape estimation and tracking from silhouette cues,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1 –8, june 2008.

[90] C. Stauffer and K. Tieu, “Automated multi-camera planar tracking correspondence modeling,” in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–259 – I–266 vol.1, june 2003.

[91] G. Kayumbi and A. Cavallaro, “Multiview trajectory mapping using homography with lens distortion correction,” *J. Image Video Process.*, vol. 2008, pp. 1:1–1:11, Jan. 2008.

[92] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multicamera people tracking with a probabilistic occupancy map,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 267 –282, feb. 2008.

[93] F. Daniyal, M. Taj, and A. Cavallaro, “Content and task-based view selection from multiple video streams,” *Multimedia Tools and Applications*, vol. 46, pp. 235–258, 2010.

[94] M. Bauml and R. Stiefelhagen, “Evaluation of local features for person re-identification in image sequences,” in *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pp. 291 –296, 30 2011-sept. 2 2011.

[95] D. Arsic, A. Lyutskanov, G. Rigoll, and B. Kwolek, “Multi camera person tracking applying a graph-cuts based foreground segmentation in a homography framework,” in *Performance Evaluation of Tracking*

and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on, pp. 1 –8, dec. 2009.

[96] E. Maggio and A. Cavallaro, "Learning scene context for multiple object tracking," *Image Processing, IEEE Transactions on*, vol. 18, pp. 1873 –1884, aug. 2009.

[97] J. Berclaz, F. Fleuret, and P. Fua, "Multicamera tracking and atypical motion detection with behavioral maps," in *Computer Vision ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5304 of *Lecture Notes in Computer Science*, pp. 112–125, Springer Berlin Heidelberg, 2008.

[98] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *Robotics and Automation, IEEE Journal of*, vol. 3, pp. 323 –344, august 1987.

[99] A. Mittal and L. Davis, "M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene," *International Journal of Computer Vision*, vol. 51, pp. 189–203, 2003.

[100] D. Forte and A. Srivastava, "Adaptable architectures for distributed visual target tracking," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pp. 339 –345, oct. 2011.

[101] P. Bolliger, M. Köhler, and K. Römer, "Facet: towards a smart camera network of mobile phones," in *Proceedings of the 1st international conference on Autonomic computing and communication systems*, Autonomics '07, (ICST, Brussels, Belgium, Belgium), pp. 17:1–17:10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[102] M. Quaritsch, M. Kreuzthaler, B. Rinner, H. Bischof, and B. Strobl, "Autonomous multicamera tracking on embedded smart cameras," *EURASIP J. Embedded Syst.*, vol. 2007, pp. 35–35, January 2007.

[103] M. Wittke, M. Hoffmann, J. Hahner, and C. Muller Schloer, "Midsca: Towards a smart camera architecture of mobile internet devices," in *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pp. 1 –10, sept. 2008.

[104] Seven Solutions S.L. `http://www.sevensols.com`, 2013.

[105] H. Hedberg, F. Kristensen, P. Nilsson, and V. Owall, "A low complexity architecture for binary image erosion and dilation using structuring element decomposition," in *Circuits and Systems, IEEE International Symposium on (ISCAS)*, vol. 4, pp. 3431–3434, may 2005.

[106] K. Appiah, A. Hunter, P. Dickinson, and J. Owens, “A run-length based connected component algorithm for fpga implementation,” in *ICECE Technology. International Conference on*, pp. 177 –184, dec. 2008.

[107] B. D. Ripley, *Pattern recognition and neural networks.* Cambridge: Cambridge University Press;, 1996.

[108] G. Gualdi, A. Prati, and R. Cucchiara, “Video streaming for mobile video surveillance,” *Multimedia, IEEE Transactions on*, vol. 10, no. 6, pp. 1142–1154, 2008.

[109] S. Huang, “An advanced motion detection algorithm with video quality analysis for video surveillance systems,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 1, pp. 1–14, 2011.

[110] L. Maddalena and A. Petrosino, “A self-organizing approach to background subtraction for visual surveillance applications,” *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1168–1177, 2008.

[111] I. Cohen and G. Medioni, “Detecting and tracking moving objects for video surveillance,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[112] I. Garcia, “Tms320dm64x power consumption summary. texas instruments, application report spra962f.,” tech. rep., Texas Instruments, February 2005.

[113] Texas Instruments. `http://www-s.ti.com/sc/psheets/spra962f/spra962f.zip`, 2005.

[114] W. B. Liu Xi-Ling and Z. Zhi-Hui, “Design of airport video aided surveillance system based on dsp+fpga,” in *30th Chinese Control Conference (CCC)*, pp. 3214 – 3217, July 2011.

[115] I. Mikic, P. C. Cosman, G. T. Kogut, and M. M. Trivedi, “Moving shadow and object detection in traffic scenes,” *Pattern Recognition, International Conference on*, vol. 1, p. 1321, 2000.

[116] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting objects, shadows and ghosts in video streams by exploiting color and motion information,” in *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pp. 360 –365, sep 2001.

[117] J. Stander, R. Mech, and J. Ostermann, “Detection of moving cast shadows for object segmentation,” *Multimedia, IEEE Transactions on*, vol. 1, pp. 65 –76, mar 1999.

[118] M. Tomasi, M. Vanegas, F. Barranco, J. Diaz, and E. Ros, "A novel architecture for a massively parallel low level vision processing engine on chip," in *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pp. 3033 –3039, july 2010.

[119] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 755 –762, june 2010.

[120] D. Grest, J. Woetzel, and R. Koch, "Nonlinear body pose estimation from depth images," in *Pattern Recognition*, pp. 285–292, Springer, 2005.

[121] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3d human body tracking with an articulated 3d body model," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1686–1691, IEEE, 2006.

[122] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[123] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[124] O. Faugeras, *Three dimensional computer vision, a geometric viewpoint.* MIT Press, 1993.

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

[126] J. Ralli, J. Diaz, and E. Ros, "Spatial and temporal constraints in variational correspondence methods," *Machine Vision and Applications*, pp. 1–13, 2011. 10.1007/s00138-011-0360-x.

[127] K. for Windows Team, "Inside the newest kinect for windows sdk - infrared control." `http://blogs.msdn.com/b/kinectforwindows/archive/2012/12/07/inside-the-newest-kinect-for-windows-sdk-infrared-control.aspx`, December 2012.

[128] J. Smisek, M. Jancosek, and T. Pajdla, "3d with kinect," in *Consumer Depth Cameras for Computer Vision*, pp. 3–25, Springer, 2013.

[129] K. Konolige and P. Mihelich, "Technical description of kinect calibration." `http://www.ros.org/wiki/kinect_calibration/technical`, 2011.

[130] D. J. Fleet, A. D. Jepson, and M. R. Jenkin, "Phase-based disparity measurement," *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 198 – 210, 1991.

[131] F. Solari, S. Sabatini, and G. Bisio, "Fast technique for phase-based disparity estimation with no explicit calculation of phase," *Electronics Letters*, vol. 37, pp. 1382 –1383, nov 2001.

[132] S. P. Sabatini, G. Gastaldi, F. Solari, K. Pauwels, M. M. V. Hulle, J. Diaz, E. Ros, N. Pugeault, and N. Krüger, "A compact harmonic code for early vision based on anisotropic frequency channels," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 681 – 699, 2010. Special Issue on Multi-Camera and Multi-Modal Sensor Fusion.

[133] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *Communications, IEEE Transactions on*, vol. 31, pp. 532 – 540, apr 1983.

[134] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33 – 41, 1984.

[135] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr, "Variational optical flow computation in real time," *Image Processing, IEEE Transactions on*, vol. 14, pp. 608 –615, may 2005.

[136] J. Weickert and C. Schnörr, "A theoretical framework for convex regularizers in pde-based computation of image motion," *International Journal of Computer Vision*, vol. 45, pp. 245–264, 2001. 10.1023/A:1013614317973.

[137] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Computer Vision - ECCV 2004* (T. Pajdla and J. Matas, eds.), vol. 3024 of *Lecture Notes in Computer Science*, pp. 25–36, Springer Berlin / Heidelberg, 2004.

[138] J. Ralli, J. Diaz, and E. Ros, "Complementary image representation spaces in variational disparity calculation," *EURASIP Journal on Advances in Signal Processing*, 2011.

[139] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 328 –341, feb. 2008.

[140] I. Ernst and H. Hirschmüller, “Mutual information based semi-global stereo matching on the gpu,” in *Advances in Visual Computing*, vol. 5358 of *Lecture Notes in Computer Science*, pp. 228–239, Springer Berlin / Heidelberg, 2008.

[141] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, “Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation,” in *Embedded Computer Systems (SAMOS), 2010 International Conference on*, pp. 93 –101, july 2010.

[142] L. Vincent, “Morphological grayscale reconstruction in image analysis: applications and efficient algorithms,” *Image Processing, IEEE Transactions on*, vol. 2, pp. 176 –201, apr 1993.

[143] M. Tomasi, M. Vanegas, F. Barranco, J. Diaz, and E. Ros, “Massive parallel-hardware architecture for multiscale stereo, optical flow and image-structure computation,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 282 –294, feb. 2012.

[144] E. J. Fernandez-Sanchez. `http://atcproyectos.ugr.es/mvision/`, 2012.

[145] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 11, pp. 1254 – 1259, 1998.

[146] L. Itti and C. Koch, “Computational modelling of visual attention,” *Nature Review Neuroscience*, vol. 2, no. 3, pp. 194 – 203, 2001.

[147] T. Alter and R. Basri, “Extracting salient curves from images: an analysis of the saliency network,” in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR ’96, 1996 IEEE Computer Society Conference on*, pp. 13 –20, 1996.

[148] M. Nishigaki, C. Fermuller, and D. Dementhon, “The image torque operator: A new tool for mid-level vision,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 502 – 509, june 2012.

[149] A. Mishra, Y. Aloimonos, and C. Fermuller, “Active segmentation for robotics,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 3133 –3139, 2009.

[150] N. Bruce, *Saliency, attention and visual search: an information theoretic approach.* PhD thesis, York University, 2008.

[151] J.-C. Terrillon, M. Shirazi, H. Fukamachi, and S. Akamatsu, "Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pp. 54 –61, 2000.

[152] Google, "Android." http://www.android.com/about/, 2012.

[153] G. R. Bradski, S. Clara, and I. Corporation, "Computer vision face tracking for use in a perceptual user interface," *Interface*, vol. 2, no. 2, pp. 12–21, 1998.

[154] F. Barranco, J. Diaz, A. Gibaldi, S. P. Sabatini, and E. Ros, "Vector disparity sensor with vergence control for active vision systems," *Sensors*, vol. 12, no. 2, pp. 1771–1799, 2012.

[155] L. Itti and C. Koch, "Feature combination strategies for saliency-based visual attention systems," *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 161–169, 2001.

[156] ICBS Learning-Based Multimedia: Eclipse IDE. `http://lbmedia.ece.ucsb.edu/resources/dataset/ibtd.zip`, 2013.

[157] W. Freeman, K. Tanaka, J. Ohta, and K. Kyuma, "Computer vision for computer games," in *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pp. 100 –105, oct 1996.

[158] JSR 82: Java APIs for Bluetooth. `http://jcp.org/en/jsr/detail?id=82`, 2012.

[159] The Eclipse Foundation: Eclipse IDE. `http://www.eclipse.org/`, 2012.