UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesi Doctoral

3D PARTIAL SCANS MODELS FOR PEOPLE RECOGNITION WITH A RGB-D SENSOR

Karla Andrea Trejo Ramírez

Director: Cecilio Angulo Bahón

Juliol de 2019

Declaration of Authorship

I, Karla Andrea Trejo Ramírez, declare that this thesis titled, "3D Partial Scans Models for People Recognition with a RGB-D sensor" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"By knowing what exist, you can know that which does not exist. That is the void. People in this world look at things mistakenly, and think that what they do not understand must be the void. This is not the true void. It is confusion."

Miyamoto Musashi

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial
IDEAI-UPC. Intelligent Data Science and Artificial Intelligence Research Center

Doctoral dissertation

3D Partial Scans Models for People Recognition with a RGB-D sensor

by Karla Andrea Trejo Ramírez

Research developed in this thesis is motivated by a vision of a future where social activities and personalized services are delivered in intelligent environments, entering a new era of more natural and wholesome human-machine interactions. While people are moving freely within these environments, individuals should be recognized by an artificial agent through a wide range of diverse tasks: detection, identification, re-identification and tracking. In this context, facial features are not enough and thus a need appears about finding new sources of descriptive human information to support those actions.

Novel approaches in people recognition using human bodies as the main target are introduced throughout this dissertation. Our proposals intend to take full advantage of the powerful capabilities of one device only, regardless of whether the technology exploited is 2D (simple camera) or 3D (RGB-D sensor), favoring the effective use of minimum resources and combining less intensive measures. Applications start with the automatic landmarking of human shapes by learning a small dataset, upgraded by on-line 3D body shape contour tracking with an RGB-D sensor; next, groups of people are categorized and tracked in a public space and studied as a cognitive emulation of human behavior towards relationships generated through spatial and motion interactions; finally, a natural user interface is created with a RGB-D sensor for the identification and re-identification of individuals on the scene in real-time.

By employing practical computer vision and machine learning techniques, experimental evaluation will show fair and adequate performance of the introduced systems, comparable to other more complex ones. All this implementation is carried out with due respect for human perception and the quality of their interaction with the intelligent agents. Always bearing in mind that their application goes hand in hand with the idea of eventually incorporating them in smart environments and robotic platforms.

Resumen

La investigación desarrollada en esta tesis está motivada por una visión de futuro donde las actividades sociales y los servicios personalizados se realizan en entornos inteligentes, entrando en una nueva era de interacciones humano-máquina más naturales e íntegras. Cuando las personas se mueven libremente dentro de estos entornos, éstas deben ser reconocidas por un agente artificial a través de una amplia gama de tareas diversas: detección, identificación, re-identificación y seguimiento. En este contexto, el uso de características faciales no es suficiente y, por lo tanto, aparece la necesidad de encontrar nuevas fuentes de información humana descriptiva para respaldar esas acciones.

En esta memoria se presentan nuevos enfoques en el reconocimiento de personas que utilizan la forma del cuerpo humano como elemento principal de obtención de información. Nuestras propuestas pretenden explotar al máximo las capacidades que permiten el uso de un único dispositivo, independientemente de si la tecnología explotada es 2D (cámara simple) o 3D (sensor RGB-D). Este enfoque favorece el uso efectivo de recursos mínimos y combina formas de procesado menos intensivas. Las aplicaciones desarrolladas inicialmente refieren al etiquetado automático de la forma del cuerpo humano mediante el aprendizaje de un pequeño conjunto de datos, actualizado mediante seguimiento de contorno en forma de cuerpo 3D en línea con un sensor RGB-D; luego, se exponen los desarrollos realizados sobre categorización y rastreo de grupos de personas en un espacio público, los cuales son estudiados como una emulación cognitiva del comportamiento humano hacia las relaciones generadas a través de interacciones espaciales y de movimiento; finalmente, se crea una interfaz de usuario natural con un sensor RGB-D para la identificación y re-identificación de individuos en la escena en tiempo real.

Mediante el empleo de técnicas prácticas de visión por computador y aprendizaje automático, la evaluación experimental mostrará un rendimiento equilibrado y adecuado de los sistemas presentados, comparable a otros más complejos. Toda esta implementación se lleva a cabo con el debido respeto por la percepción humana y la calidad de su interacción con los agentes inteligentes, siempre teniendo en cuenta que su aplicación va de la mano con la idea de eventualmente incorporarlos en entornos inteligentes y plataformas robóticas.

Resum

La recerca desenvolupada en aquesta tesi està motivada per una visió de futur on les activitats socials i els serveis personalitzats es realitzen en entorns intel·ligents, entrant en una nova era d'interaccions humà-màquina més naturals i íntegres. Quan les persones es mouen lliurement dins d'aquests entorns, els individus han de ser reconeguts per un agent artificial a través d'una àmplia gamma de tasques diverses: detecció, identificació, re-identificació i seguiment. En aquest context, l'ús de característiques facials no és suficient i, per tant, sorgeix la necessitat de trobar noves fonts d'informació humana descriptiva per recolzar aquestes accions.

En aquesta memòria es presenten nous enfocaments en el reconeixement de persones que utilitzen la forma del cos humà com a element principal d'obtenció d'informació. Les nostres propostes pretenen explotar al màxim les capacitats que permeten l'ús d'un únic dispositiu, independentment de si la tecnologia explotada és 2D (càmera simple) o 3D (sensor RGB-D). Aquest enfocament afavoreix l'ús efectiu de recursos mínims i combina formes de processament menys intensives. Les aplicacions desenvolupades inicialment refereixen a l'etiquetatge automàtic de la forma del cos humà mitjançant l'aprenentatge d'un petit conjunt de dades, actualitzat mitjançant seguiment de contorn en forma de cos 3D en línia amb un sensor RGB-D; després, s'exposen els desenvolupaments realitzats sobre categorització i rastreig de grups de persones en un espai públic, els quals són estudiats com una emulació cognitiva del comportament humà cap a les relacions generades a través d'interaccions espacials i de moviment; finalment, es crea una interfície d'usuari natural amb un sensor RGB-D per a la identificació i re-identificació d'individus en l'escena en temps real.

Mitjançant l'ús de tècniques pràctiques de visió per ordinador i aprenentatge automàtic, l'avaluació experimental mostrarà un rendiment equilibrat i adequat dels sistemes presentats, comparable a altres més complexos. Tota aquesta implementació es porta a terme amb el degut respecte per la percepció humana i la qualitat de la seva interacció amb els agents intel·ligents, sempre tenint en compte que la seva aplicació va de la mà amb la idea d'eventualment incorporar-los en entorns intel·ligents i plataformes robòtiques.

Acknowledgements

I am deeply thankful to my thesis director Dr. Cecilio Angulo for giving me the opportunity to perform this research under his supervision, for being there since day one. I will never forget how supportive you were since my remote application to the master program seven years ago. You are the most cheerful and positive advisor ever.

I would like to thank my colleagues and friends from the Knowledge Engineering Research Group (IDEAI-GREC) –namely Núria Agell, Mónica Sánchez and Francisco Ruiz, for the kind atmosphere, good chat and discussions, the support and the great fun.

My thanks must also go to Juan Acevedo for making this adventure easier to bear. I was fortunate in having not only a Mexican classmate, but a close friend I can relate to and work side by side with, helping each other in the good times, the bad times and the taco times.

I also want to thank Jennifer Nguyen, partner in crime, for awaking my business side, for the laughs to maintain our sanity and all the culinary rides we had and we will continue to have!

Living abroad can be harsh sometimes, it is thus a great blessing I have been able to meet my amazing Mexican friends and PhD fellows –Ana and Sergio, Vero and Isra. Our special bond started with mezcal and fermented into friendships I dearly cherish.

Thanks to my UPC friends –Julio Luna, Víctor Vaquero, Noè Rosanas, Anna Ayza, Felip Martí, Carlos Flores, Albert Pumarola and Gerard Canal– some of which I have known them since the master's program, I appreciate your valuable help and insights, as an inner source of advice and a reference point to share and exchange experiences.

I would like to also thank my friends, colleagues and the many students who have given their time to help me collect recordings of human shape and appearance in various activities, without these recordings this research would not have been possible.

I offer my profound thanks to professor Shin'ichi Satoh and everyone back at Satoh Lab in the National Institute of Informatics in Tokyo –Benjamin Renoust, Sang Phan, Yusuke Matsui, Yumi Awano, Chien-Quang Le, Nam Van Hoang and Ngoc T.B. Nguyen– for their kind support and the warmest welcome. It was a personal dream come true and I will always be very grateful with you all for providing me one of the best experiences I ever had.

My thanks also go to all the colleagues at the Institut d'Investigació en Intel·ligència Artificial and my hosting professors Pere García and Josep Puyol for letting me stay at such a calm, comfortable and convenient environment during the last stages of thesis writing. I want to extend my gratitude to Eloi Pérez as well, who has offered me the possibility to work alongside him and the privilege to continue my professional journey here –in a world full of "contrasts".

I want to express my thanks to Ana Sanchís who was able to decipher my horrible and poorly drawn designs and was able to transform those ideas into Figure 6.10 in Chapter 6 of this dissertation. You are truly a wizard.

My thanks must also go to all my dear friends in Barcelona and Colima who kept my spirit high at all times and were always rooting for me –namely Yara, Edgar and Cristina. Even though there might be some missing names here, I just want to express my sincerest thanks to *everyone* who shared their time and energy with me.

Being apart from my family and country was certainly hard, slowly adapting to new costumes and to a different lifestyle. That is why I am extremely grateful to my girlfriend Andrea, for her infinite moral support and understanding during my academic life. Your love kept me always cheerful.

Thanks are extended to her parents, who always lend a helping hand in times of need. There are no words to describe how much I value your empathy towards me.

I also appreciate the affections and good vibes from my family back at Mexico –my grandmothers, all my aunts, uncles, cousins and close friends– who were holding the fort on my behalf. And mainly, my father, who made sure I would not forget how proud he was of his daughter. Thanks dad, I hope I can be a good enough example of perseverance for my brother and sister, whom I want to thank warmly for tightening our bond even more over the last years.

Finally, I want to thank a very special and important person in my life. My best friend, my referent, my heroine, my source of inspiration, my main motivation. The fiercest, bravest and most dedicated woman I will ever know: my dear mother. Mom, thank you for teaching me discipline to follow my dreams, for encouraging me to fly towards them and giving me the strength to never give up on them. This is all because of you and your unconditional love. Thank you, from the bottom of my heart.

Contents

List of Figures

List of Tables

List of Abbreviations

AAM	**A**ctive **A**ppearance **M**odel
AI	**A**rtificial **I**ntelligence
AmI	**Am**bient **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
AOS	**A**dditive of **O**pperator **S**plitting
API	**A**pplication **P**rogramming **I**nterface
AR	**A**ugmented **R**eality
ASM	**A**ctive **S**hape **M**odel
CCD	**C**onditional **C**ountour **D**etector
DNN	**D**eep **N**eural **N**etwork
HCI	**H**uman **C**omputer **I**nteraction
HRI	**H**uman **R**obot **I**nteraction
HOG	**H**istogram of **O**riented **G**radients
IoT	**I**nternet **o**f **T**hings
ISO	**I**nternational **O**rganization for **S**tandardization
ISTAG	**I**nformation **S**ociety and **T**echnology **A**dvisory **G**roup
KNN	**K** - **N**earest **N**eighbor
MR	**M**ixed **R**eality
NA	**N**ot **A**pplicable
NUI	**N**atural **U**ser **I**nterface
PCA	**P**rincipal **C**omponents **A**nalysis
PSM	**P**artial **S**can **M**odel
RGB	**R**ed **G**reen **B**lue
RGB-D	**R**ed **G**reen **B**lue - **D**epth
ROI	**R**egion of **I**nterest
SDK	**S**oftware **D**evelopment **K**it
SIFT	**S**cale **I**nvariant **F**eature **T**ransfrom
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SURF	**S**peeded **U**p **R**obust **F**eatures
SVM	**S**upport **V**ector **M**achine
ToF	**T**ime **o**f **F**light
VR	**V**irtual **R**eality
XML	e**X**tensible **M**arkup **L**anguage

To my family.

Chapter 1

Introduction

In this chapter, we start by addressing our dissertation from a general perspective, explaining the basic circumstances that had inspired the work undertaken on behalf of the contributions we wanted to deliver and which will be exposed in the next chapters.

The objectives are formulated in such a manner that the reader is able to understand by what means and in what depth this doctoral thesis aims to fulfill and meet those expectations.

A short and concise description on the contents of this document and how they are subsequently organized is also provided. Finally, we have included some details about the outcomes, in the form of internships and publications, that we have produced in the course of these years of research activities.

1.1 Background and Motivation

In recent times, research in computer vision has shown great progress with the inclusion of depth sensing cameras, also known as RGB-D sensors. The rapid advancements in this technology led by increased computing power and more efficient and sophisticated sensors have given rise to more effective pattern recognition and for machine learning techniques to be applied in real time.

Smart environments are evolving accordingly. We are already envisioning a future where ambient intelligence is integrated into our daily lives, where computers and robots help to support us in everyday activities in an easy and natural way. The need thus emerge to provide personalized services in these environments, demanding some form of human identification system. Two general approaches can be considered when dealing with the identification problem: an active, voluntary, user-initiated approach, and a passive, involuntary, system-initiated approach.

The trend towards a more natural human-machine interaction prevails on the second option, where people should be identified by any smart device cognitively emulating how we, as humans, use abstract reasoning to identify other individuals. Depending on the context of their interaction, to remember the preferences and characteristics of a user

would enable these intelligent systems to perform socially useful activities. However, identifying individuals in uncontrolled settings is a challenging task.

Designing effective feature representations for humans is in itself a problematic issue. The appearance of a person can differ for a wide variety of reasons: occlusion, cluttered background, image quality or resolution, and changing viewpoints. Even something as simple as wearing an accessory that it is not uniformly coloured or that appears different when observed from another point of view can be difficult to cope with. However, the use of RGB-D sensors can assist in these challenges by making use of their depth data to provide automatic person detection and background removal via their Software Development Kits (SDKs).

And it is upon this unstoppable technological revolution that people interact with computers in diverse ways; the interface between humans and computers plays a crucial role in facilitating this interaction. RGB-D technologies supporting these interactions are continuously evolving to achieve less tangible and more natural means of communication, including voice commands, hand gestures, head and eye movements, and whole body actions.

Thus, RGB-D sensors such as Microsoft's Kinect have become an important element to consider when designing state-of-the-art Natural User Interfaces (NUIs), allowing people to engage with embodied character agents in a way that cannot be achieved with other interface paradigms. It demonstrates the growth in the human-computer interaction field has been in quality of interaction, guided by the mission to actively pursue the enhancement of the human-machine partnership.

Existing research under this framework tends to exploit any possible and competitive advantage by pooling all available resources, in order to acquire the best performance and achieve more accurate results. A major technological investment which requires both hardware and software to be working in high-end machines and top-level features –GPUs, point clouds rendering, several devices deployed in parallel, among others. Hence, we found as an interesting contrast to study the problem from a different standpoint and explore scenarios with very limited means, in pursuit of approaches which represent sufficient solutions for low-cost projects and minimalist environments, regardless of the numerical superiority other methods can provide in richer conditions.

1.2 Objectives

The objectives of this dissertation are clearly outlined, focusing on the motivation within the background previously described, and revealing an opportunity niche in which we attempt to contribute under specific circumstances and guidelines.

1.2.1 General Objective

The main objective in this research is the design and implementation of a system that recognizes and unequivocally identifies people with a RGB-D sensor, based on the retrieval of 3D information to be further analyzed as 2D information, thus lowering the complexity of data management and empowering classic or straight-forward approaches with richer knowledge of data. The primary target are human body shapes, taking into account both contour and appearance information for this identification strategy.

1.2.2 Specific Objectives

Key aspects within the overall objective comprise:

1. To test more practical approaches that follow less computationally expensive instances.
2. To reinforce the existing research conducted with human body shapes and consolidate them as a powerful target for people recognition applications.
3. To employ and provide open-source software to promote knowledge transfer in the robotics and computer vision fields of study.
    - Through cross-platform libraries or platform independent tools which are currently becoming standardized to some extent.
    - Supporting the code to be reused by anyone in a future C++/C# integration project or in any similar applications.
4. To operate with minimum resources applying novel techniques and equipment.
    - Implementing the latest RGB-D technology with state-of-the-art approaches in an open-source style using only one device.
5. To accurately identify people without regard to the time in which the subject could remain out of the sensor's field of vision.
    - Assign IDs to new subjects, remember IDs of already known subjects.
    - People recognition, re-identification and tracking tasks in real-time.
    - High execution speed to be sufficient for the system to construct an on-line database.
6. To maintain the interaction between the user and the system as natural as possible.
    - Respecting human perception.
    - Passive responses and active non-intrusive interactions.
    - Not demanding a particular pose from the user to trigger the system.

1.3 Thesis Organization

This dissertation is organized as follows.

Chapter 2 reviews the literature relevant to people recognition and person identification. It also discusses several research methodologies pursuing to address these problems in different contexts and backgrounds, which are strongly related to the objectives of this dissertation.

Chapter 3 introduces our proposal on single-camera automatic landmarking for people recognition tasks using an ensemble of regression trees. It extrapolates a face alignment approach to body shapes by adapting a machine learning algorithm within an experimental setting.

Chapter 4 contains an interesting approach on leader detection, group categorization and tracking by means of individual role assigning, after cognitively analyzing the social relations within the group that arise from motion and spatial interactions. To the best of our knowledge, it is a new problem that could open a novel perspective on social robotics and ambient intelligence.

Chapter 5 upgrades the work developed in Chapter 3 by performing 3D body contour tracking with a RGB-D sensor. A brief experiment that exposes some relevant ideas concerning human-body shape extraction under this framework and also serves to discuss the feasibility of 3D automatic landmarking.

Chapter 6 proposes a person re-identification system using 3D pose estimation. Pose estimation values are calculated from specific joint rotations which are retrieved from a RGB-D sensor and subsequently transformed into a sort of 2D viewpoint-model. This is an original approach we have named *3D Partial Scans Models* (3D PSMs) and aims to process 2D data which has been obtained from 3D information with computer vision algorithms. A significant work that unifies everything we have learned throughout the various chapters in this dissertation and integrates the concept of natural user interfaces to be explored in human-computer interaction within a smart environment setup.

Chapter 7 finally discusses the progress made in achieving the defined objectives, presents the thesis conclusions, contributions and final recommendations for future work.

For the sake of a better understanding of the synergy between the different chapters and how the elements of each chapter contribute to the general objective of the thesis, one can refer to Figure 1.1.

Starting from the left branch, Chapter 3, explores basic landmarking research on faces and elaborates towards the 2D human body modelling topic.

The center branch, Chapter 4, keeps researching on the 2D framework but focusing on motion and social strategies, since the previous 2D automatic landmarking approach could not be implemented under the harsh conditions of group occlusion and pose intravariance in a moderately crowded setup.

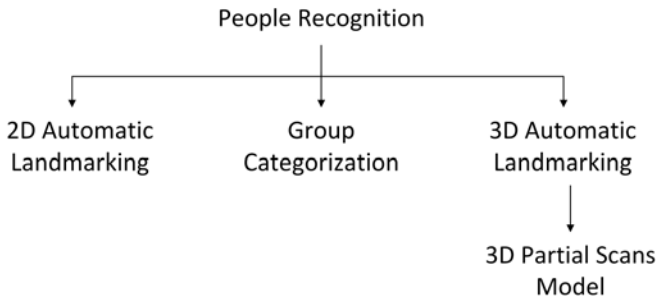


FIGURE 1.1: Alternative reading for chapters with experimental contents. First levels from left to right: Chapter 3, Chapter 4 and Chapter 5. Second level on the right: Chapter 6.

Finally, the branch on the right systematically tackles the user identification problem all the way through. In the first level, Chapter 5 performs on-line human body segmentation that will become one of the base algorithms for the 3D Partial Scans Models approach from Chapter 6. An approach which combines 2D and 3D instances, claiming the knowledge gained on positive and negative traits from preceding experimentation and comes forth with a whole new proposal.

In addition, Table 1.1 presents a brief description of the experiments carried out by chapter, their characteristics and findings. With this, the most significant information from each work is delivered in a nutshell.

1.4 Internships

In order to augment our theoretical knowledge with practical experience, and also, seizing the unique opportunity to write the thesis while working and receiving valuable guidance from experienced colleagues, the following internships were conducted:

- National Institute of Informatics (NII). Tokyo, Japan.
  Digital Content and Media Sciences Research Division.
  Hosting professor: Shin'ichi Satoh.
  Duration: 5 months (from 11th March to 11th August, 2016).

- Institut d'Investigació en Intel-ligència Artificial (IIIA-CSIC). Bellaterra, Barcelona.
  Unitat de Desenvolupament Tecnològic en Intel-ligència Artificial.
  Hosting professors: Pere García and Josep Puyol.
  Duration: 6 months (from 1st March to 31st August, 2018).

TABLE 1.1: Table of Experiments

Chapter	Samples	Techniques	Parameters	Proc. Time	Accuracy
3	46 images	Ensemble of regressors, HOG, SVM and Sliding window	C, target size, upsample, flip, epsilon, over-sampling, nu, tree-depth	126 ms	11-14% displacement error
4	Over 5400 images	HOG, SVM, Differential subtraction, Standard deviation, Optical flow and Discriminative correlation filters	hit threshold, window stride, padding, scale0, group threshold, max deviation, pyramid layers, iterations, polynomial degree expansion	270 ms	$\approx$ 70% acc. rate
5	Online streaming @ 30 fps	Coordinate mapping and Conditional Contour Detector kernel	NA	19 ms	<2mm depth acc. error
6	Online streaming $\approx$ 3084 images	Euler angles conversion, Coordinate mapping, KAZE features and k-d tree (decision trees + KNN)	uniqueness threshold, k, rotation bins, scale increments	242 ms	63% acc. rate

1.5 Publications

As a part of the outcomes of this research, the following journal and conference papers have been produced:

1.5.1 Journals

- Trejo, K., Angulo, C., Satoh, S. and Bono, M. (2018). "Towards robots reasoning about group behavior of museum visitors: Leader detection and group tracking". *Journal of Ambient Intelligence and Smart Environments*, 10(1), pp. 3–19. IOS Press [DOI 10.3233/AIS-170467]. (JCR IF 2017: 0.809, Q3)

- Trejo, K. and Angulo, C. (2016). "Single-Camera Automatic Landmarking for People Recognition with an Ensemble of Regression Trees". *Computación y Sistemas*, Thematic Issue: Topic Trends in Computing Research (Guest Editor: Claudia P. Ayala), 20(1), pp. 19-28. [DOI: 10.13053/CyS-20-1-2365].

1.5.2 Conferences

- Trejo, K., Acevedo-Valle, J.M. and Angulo, C. (2016). "2D and 3D automatic landmarking of body contours for people recognition". In *Innovation Match MX, Foro Internacional de Talento Mexicano*. "Innovation Match MX 2015-2016: 1er Foro Internacional de Talento Mexicano: Memorias de Ponencias IMMX". Guadalajara, México, pp. 1-8.

- Trejo, K., Angulo, C. and Aguado, J. C. (2015). "Body Contours AAM: Automatic Landmarking for People Recognition". In *Artificial Inteligence Research and Development: Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence*, pp. 279-282, Valencia, Spain. IOS Press. [DOI 10.3233/978-1-61499-578-4-279].
  (Best Poster Award)

1.5.3 Collaborations

- Acevedo-Valle, J.M., Trejo, K. and Angulo, C. (2017). "Multivariate Regression with Incremental Learning of Gaussian Mixture Models". In *Recent Advances in Artificial Intelligence Research and Development: Proceedings of the 20th International Conference of the Catalan Association for Artificial Intelligence*, pp. 196-205, Deltebre, Terres de l'Ebre, Spain. IOS Press. [DOI. 10.3233/978-1-61499-806-8-196].

- Acevedo-Valle, J. M., Angulo, C., Moulin-Frier, C., Trejo, K. (2016). "The Role of Somatosensory Models in Vocal Autonomous Exploration". In *Revista Internacional de Investigación e Innovación Tecnológica*, 4(23), pp. 1-11.

Chapter 2

Literature Review

This chapter explores the body of knowledge on people recognition by providing a survey of the methodologies employed in prior research within our proposed framework, thus analyzing their distinctive traits, known applications and evolution over time. The chapter concludes with a summary of the direction of the research work in this thesis.

2.1 People Recognition and Identification

Recognition is one of the core pursuits in artificial intelligence (AI) research [54, 8, 118]. This task attempts to attach semantics to visual data such as images or video. Object recognition is an important and recurring subtopic where models are built to recognize object categories or instances [51, 76, 14]. Yet another subtopic quite popular nowadays is people recognition [106]. People recognition comprises two major interests: attaching identities to pictures or video, and building descriptions from visual data of people behaviour. These descriptors lead to a variety of tasks which can be performed based on those premises, like face recognition [60, 40, 133], pose estimation [89], activity recognition [18, 94] and people tracking [5], among others.

Properly identifying humans includes several difficulties to be overtaken. Images or video of people can show a high degree of variability in shape and texture. Appearance variations are due to differences between individuals, deformations in facial expression, pose and illumination changes. It should be also taken into account visual perception parameters such as resolution, contrast, brightness, sharpness, and color balance [15, 135, 93, 16].

Therefore, person identification problems are active research topics in the computer science field with a large potential to become interesting applications, from surveillance, health care and shopping to human-robot interaction and the personalization of services in smart environments. Designing robust approaches with effective feature representations for human identification addresses one of the major challenges on this field.

Since the very beginning, the only available material to study the people recognition subject was to analyze RGB color or grayscale images obtained either from still pictures or

frame-by-frame video sequences. Ergo, 2D techniques were employed under this framework, limited by the technology available at the time but not restrained to explore a wide range of possibilities in a world with an increasing demand of artificial intelligence progress. This is the focus for the literature review in Section 2.2.

Thanks to the fast development of information and communication technologies in the last decade, computers and other electronic devices have improved in performance while decreasing in cost, which in turn has given rise to the emergence of three-dimensional technology.

Sensors can now integrate the strength of optical cameras and laser-based 3D scanners, obtaining 3D measurements of objects while retaining a compact and almost portable size. Thus, 3D computer graphics, modeling and rendering are becoming more practical and affordable, as well as meaningful for many AI applications such as object tracking, pose estimation and human-computer interaction.

Despite this, 3D information databases are scarce and they contain very little amount of data, especially for human features. Several 2D applications can potentially be improved with 3D models, but considering the processing power needed will be significantly higher. For instance, 2D face recognition continues to be highly accurate and will continue to improve as custom face recognition pipelines are built. 3D face recognition marks an important step towards further improvement in accuracy and make it nearly perfect. However, notable advances in hardware capabilities are required to make this a reality in much larger and complex targets. We will discuss more about this 3D approach in Section 2.3.

In Section 2.4 we study the people recognition problem in the framework of ambient intelligence and robotics, where the interaction between humans and machines with embedded intelligence currently makes a strong and pressing appeal on becoming more natural. To enable this interaction, a mobile service robot or intelligent system needs the ability to recognize persons in its surroundings. The recognition problem can then be decomposed into three sub-problems: detection (*how many people are there?*), localization (*where are they?*) and identification (*who are they?*).

A summary of the chapter is finally provided with an overall discussion of the works related to this dissertation and the understanding of where our research fits into and adds to the existing body of agreed knowledge in the people recognition subject.

2.2 2D People Recognition Techniques

Several 2D people recognition techniques have been proposed over the years and even some of them are still in use today, thus reflecting the great deal of robustness, precision and accuracy these methodologies had to face without sensing key information from the environment.

It is only natural that at some point, researchers' ideas complement each other and converge into hybrid solutions. Hence, to understand the general principles of the most iconic 2D people recognition techniques, these approaches have been classified into three major categories according to the nature of their base algorithm.

2.2.1 Statistical Model-Based Approaches

Model-based techniques are a promising approach where a model representing an identity of interest is matched with unknown data. This kind of techniques have shown to be able of entirely describe facial characteristics in a reduced model, extracting relevant face information without background interference [41, 39].

Models also provide the basis for a broad range of applications by "explaining" the appearance of a given image in terms of a compact set of parameters of the model. These parameters are useful for higher level interpretation of the scene. For instance, when analyzing face images they may be used to characterize the identity, pose or expression of a face.

In order to interpret new images, an efficient method for the best matching between image and model is required. Several approaches for modelling variability have been described. Nevertheless, they present some drawbacks and weaknesses, as addressed below:

Prototype variation according to some physical model. This is the most common approach, but it is computationally very expensive [9, 25].

Principal Component Analysis (PCA). It describes face images in terms of a set of basis functions, or *eigenfaces* [119]. Though valid modes of variation are learnt from a training set, and are more likely to be more appropriate than a "physical" model, the eigenface is not robust to shape changes, and does not deal well with variability in pose and expression. The main advantage is that the model can be easily matched to an image using correlation based methods.

Synthesizing new views of an object from a set of example views. They fit the model to an unseen view by using a stochastic optimization procedure [43, 57]. This is a slow procedure, but it can be robust due to the quality of the synthesized images.

3D model of the grey-level surface. It allows full synthesis of shape and appearance [28], yet there is no suggestion of a plausible search algorithm to match the model to a new image.

3D model of grey-level surface with variants. Related to the previous approach, but also combining physical and statistical modes of variation [92]. Requires a very good initialization setup.

Model shape and grey-level surface with Gabor jets. Proposed in Lades et al. [67], shape and some grey-level information is modeled using Gabor jets. However, strong shape constraints are not imposed, so the model cannot easily synthesize a new instance.

Model shape and local grey-level appearance. Using Active Shape Models (ASMs) [29] to locate flexible objects in new images, Lanitis et al [69] use this approach to interpret face images. Having found the shape using an ASM, the face is warped into a normalized frame, in which a model of the intensities of the shape-free face is used to interpret the image. There is also an extension of this work [40] to produce a combined model of shape and grey-level appearance, but again relies on the ASM to locate faces in new images.

Active Appearance Models (AAMs) [27] can be seen as a further extension of this last idea, using all the information in the combined appearance model to fit the image. This approach benefits from insights provided by two earlier works. Covell [30] demonstrated that the parameters of an eigen-feature model can be used to drive shape model points to the correct place. Black and Yacoob [17] used local, hand crafted models of image flow to track facial features, but do not attempted to model the whole face. AAMs can be thought of as a generalization of this, in which the image difference patterns corresponding to changes in each model parameter are learnt and used to modify a model estimate.

Hence, Active Appearance Models are generative models of a certain visual phenomenon. Despite being linear in both shape and appearance, overall, AAMs are nonlinear parametric models in terms of pixel intensities. Fitting an AAM to an image consists of minimizing the error between the input image and the closest model instance. Frequent applications of AAMs include medical image interpretation, face recognition and tracking. Nevertheless, a major issue lies in the construction of the 2D shape mesh as landmarks must be placed by hand on the training images, which is a very long and time consuming process to carry out.

The task of constructing AAMs without hand-marking the mesh is called *Automatic AAM* and the process to automatically localize the vertexes in that mesh is known as *automatic landmarking*. Several approaches have been performed to achieve this task on images of human faces, either static or dynamic, combining feature descriptors and predictors [10, 134, 101]. However, state-of-the-art methods employ upgraded versions of AAMs combined with decision-tree learning algorithms, leading to outstanding face alignment results [35, 44, 59], as well as 3D implementations that will be further discussed in Section 2.3.

2.2.2 Machine Learning

Machine learning is part of the artificial intelligence field of research, which creates systems that learn automatically by providing knowledge to computers through data, observations and interactions with the real world. A machine that really *learns* an algorithm, analyzes this data and it is capable to predict future behaviors. The acquired knowledge is what allows computers to correctly generalize to new settings. Also, *automatically* –in this context– implies that these systems improve in an autonomous way over time, without human intervention. Just by letting them improve with experience and, instead, allowing them to learn a few tricks on their own.

The following are the top four machine learning algorithms used in research for people recognition purposes:

Classification and regression trees. The representation of the decision tree model is a binary tree. Each node represents a single input variable and a split point on that variable, assuming this variable is numeric. The leaf nodes of the tree contain an output variable, which is used to make a prediction. Predictions are made by walking the splits of the tree until arriving at a leaf node and output the class value at that leaf node.

Trees are fast to learn and very fast for making predictions. They are also often accurate for a broad range of problems and do not require any special preparation for your data.

K-Nearest Neighbors (KNN). The KNN algorithm is very simple and very effective, since the model representation for KNN is the entire training dataset. Predictions are made for a new data point by searching through the entire training set for the K most similar instances –the neighbors– and summarizing the output variable for those K instances. For regression problems, this might be the mean output variable, for classification problems this might be the mode or most common class value.

KNN can require a lot of memory or space to store all of the data, but only performs a calculation –or learns– when a prediction is needed. The idea of distance or closeness can break down in very high dimensions which can negatively affect the performance of the algorithm on your problem. This is called the *curse of dimensionality*. It suggests to only use those input variables that are most relevant to predicting the output variable.

Support Vector Machines (SVMs). In SVMs, a hyperplane is selected to best separate the points in the input variable space by their class. The distance between the hyperplane and the closest data points is referred to as the margin. The best or optimal hyperplane that can separate the two classes is the line that has the largest margin. Only these points are relevant in defining the hyperplane and in the construction of

the classifier. These points are called the *support vectors*. They support or define the hyperplane. In practice, an optimization algorithm is used to find the values for the coefficients that maximizes the margin.

SVM might be one of the most powerful out-of-the-box classifiers and worth trying on any dataset.

Artificial Neural Networks (ANNs). ANNs are biologically inspired computer programs designed to simulate the way in which the human brain processes information. An ANN is formed from hundreds of processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organized in layers.

Each PE has weighted inputs, a transfer function and an output. The behavior of a neural network is determined by the transfer functions of its neurons, by the learning rule, and by the architecture itself. The weights are the adjustable parameters and, in that sense, a neural network is a parametrized system.

The weighted sum of the inputs constitutes the activation of the neuron, an activation signal is then passed through the transfer function to produce a single output of the neuron. Transfer functions introduce non-linearity to a network. During training, the inter-unit connections are optimized until the error in predictions is minimized and the network reaches the specified level of accuracy.

From this last approach, a new discipline called *Deep Learning* has arisen which can apply complex neural network architectures to model patterns in data more accurately than ever before. One of the most time-consuming tasks in traditional neural networks is to develop the "features" that are imported into the system. With deep learning neural networks (DNNs) it is unnecessary to define key features. Rather, the neural networks identify these features themselves and then make inferences about which ones are relevant to determine a proper output. These neural networks need samples along with ground truth data, but having the automated feature identification greatly reduces the amount of work required.

The best performing applications involve visual and speech analysis. Identification of features is often very difficult both from a subject matter perspective and the breadth of features available. Deep learning can identify all features and then make a determination as to which ones are relevant.

Still, the biggest weakness of DNNs is the high amount of data required to train them. Unlike conventional neural networks, for which feature details are provided as part of the input, DNNs need enough data to identify features on their own. As a result, they often require in excess of 10 million samples to perform reliably, which is unfeasible for our proposed target under the conditions stipulated on this dissertation.

The input data must provide greater variation in order to prevent *overfitting*, which happens when a neural network develops inferences not based on real relationships of

the data, often the result of training on too limited a set of real incidents. Output works well on the training set, but not for a real-world environment. Unless we have access to a significant amount of labeled data, it might be better off with traditional machine learning techniques.

Deep learning neural nets are also computationally intensive. Unless a project can access and pay for significant compute power, DNNs often fail to provide superior output over conventional methods.

"Interpretability" or the ability of a layperson to understand why a model used by a DNN delivered certain results, also presents challenges for DNN adoption. Many traditional machine learning models allow for interpretability, disclosing factors that resulted in a particular answer. DNNs are limited by how these tools identify features and create inferences, yielding a complex model that most likely has hundreds, or thousands, of factors. If a research project must provide a view into the nature of output, DNNs make this more problematic.

In machine learning, there is something called the *No Free Lunch* theorem. In a nutshell, it states that no one algorithm works best for every problem and any circumstance.

2.2.3 Computer Vision

Computer vision attempts to mimic the abilities of human vision by electronically perceiving and understanding an image. It is a broad field and includes several domains like gesture recognition, optical character recognition, face detection, people detection and so on.

Face recognition is definitely one of the most well-known computer vision problems. Due to its popularity it has been well studied over the last 50 years. The first intents to explore face recognition were made in the 60's. However, it was until the 90's when Turk and Pentland implemented the "eigenfaces" algorithm [119], a.k.a. PCA, that this field showed some really exciting and useful results.

From there on, the need to recognize many different targets like objects and people, led to the birth of several hardcore computer vision techniques used in detection and recognition tasks which became very well-known to the AI community, like local invariant descriptors, bag-of-visual-words (BoW) models, object detection using keypoints, deformable parts models (DPMs) and Histogram of Oriented Gradients (HOG).

Even though the HOG descriptor for object recognition is nearly a decade old, it is still heavily used today — and with fantastic results. Dalal and Triggs approach in [32] demonstrated that the Histogram of Oriented Gradients (HOG) image descriptor and a Linear Support Vector Machine (SVM) could be used to train highly accurate object classifiers — or in their particular study, human detectors. In fact, OpenCV[1] currently

[1] https://opencv.org/

has a pre-trained HOG + Linear SVM model based on this method and can be used to perform pedestrian detection in both images and video streams.

A HOG descriptor is computed by calculating the horizontal and vertical gradients that capture contour and silhouette information of grayscale and color images. This is easily achieved by filtering the image with the following kernels:

$$[-1, 0, 1] \quad \text{and} \quad [-1, 0, 1]^\intercal. \tag{2.1}$$

We can find the magnitude and direction of gradients using:

$$\begin{aligned} \|\bigtriangledown\| &= \sqrt{G_x^2 + G_y^2} \\ \Theta &= \arctan(G_y, G_x). \end{aligned} \tag{2.2}$$

Note the use of unsigned orientations such that the numerical range of the elements in $\|\bigtriangledown\| \in [0, 180]$.

The gradient image removes a lot of non-essential information and highlights outlines. At every pixel, the gradient has a magnitude and a direction. For color images, the gradients of the three channels are evaluated. The *magnitude* of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the *angle* is the angle corresponding to the maximum gradient.

In the next step, the image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells. One of the important reasons to use a feature descriptor to describe a patch of an image is that it provides a compact representation. An 8×8 image patch contains $8 \times 8 \times 3 = 192$ pixel values. The gradient of this patch contains 2 values –magnitude and direction– per pixel which adds up to $8 \times 8 \times 2 = 128$ numbers. These 128 numbers are represented using a 9-bin histogram which can be stored as an array of 9 numbers. A bin is selected based on the direction, and the vote –the value that goes into the bin– is selected based on the magnitude.

Not only is the representation more compact, calculating a histogram over a patch makes this representation more robust to noise. Individual gradients may have noise, but a histogram over 8×8 patch makes the representation much less sensitive to noise.

An 8×8 patch is a design choice informed by the scale of features we are looking for. HOG is mostly used for pedestrian detection. 8×8 cells in a scaled photo of a pedestrian are big enough to capture interesting features. The histogram is essentially a vector or an array of 9 bins corresponding to angles $0, 20, 40, 60, \ldots, 160$.

Up to this point, a histogram has been created based on the gradient of the image. Gradients of an image are sensitive to overall lighting. If you make the image darker by dividing all pixel values by 2, the gradient magnitude will change by half, and therefore the histogram values will change by half. Ideally, we want a descriptor to be independent

of lighting variations. In other words, the histograms have to be *normalized* so they are not affected by lighting variations.

Dalal and Triggs evaluated four different methods for block normalization. Let $\mathbf{v}$ be the non-normalized descriptor vector containing all histograms in a given block, $\|\mathbf{v}\|_k$ be its k-norm for $k = 1, 2$ and ϵ be a small constant. Then the normalization schemes can be one of the following:

$$\begin{aligned} L2\text{-}norm(\mathbf{v}) &= \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2 + \epsilon^2}} \\ L1\text{-}norm(\mathbf{v}) &= \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_1 + \epsilon}} \\ L1\text{-}sqrt(\mathbf{v}) &= \sqrt{\frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_1 + \epsilon}}} \end{aligned} \tag{2.3}$$

and *L2-hys* which can be computed with *L2-norm* followed by clipping (limiting the maximum values of $\mathbf{v}$ to 0.2) and re-normalizing, as in [76]. *L2-hys*, *L2-norm*, and *L1-sqrt* schemes perform equally well, while the *L1-norm* provides a slightly less reliable performance.

To calculate the final feature vector for the entire image patch, the normalized vectors are concatenated into one giant vector. So, in brief, gradient information in a HOG descriptor is pooled into a 1-D histogram of orientations, thereby transforming a 2-D image into a much smaller 1-D vector that forms the input for machine learning algorithms such as random forests, support vector machines, or logistic regression classifiers.

Objects of interest in low resolution images are often only a few pixels in size and many features typically used to identify an object may be highly blurred, leaving only object outlines as differentiable. HOG descriptors capture such outline information, and are simpler, less powerful, and faster ($\sim 20\times$) alternatives to neural networks. Furthermore, HOG features can be extracted via the CPUs of a laptop or computing cluster, and need not rely on high performance graphical processing units (GPU) that may not be available to all users. HOG feature descriptors and their extensions remain one of the few options for object detection and localization which can remotely compete with the recent successes of DNNs.

Such as HOG, a large variety of many other feature extraction methods have been proposed over the years to compute reliable descriptors. Among these descriptors, the Scale Invariant Feature Transform (SIFT) descriptor [77] utilizing local extrema in a series of difference of Gaussian (DoG) functions for extracting robust features and the Speeded-Up Robust Features (SURF) descriptor [12] partly inspired by the SIFT descriptor for computing distinctive invariant local features quickly are the most popular and widely used in several applications, including shape matching, image stitching and classification.

Both of these approaches and the many related algorithms which have followed rely on the use of the Gaussian scale space and sets of Gaussian derivatives as smoothing

kernels for scale space analysis. However, Gaussian scale space does not respect the natural boundaries of objects and smoothes to the same degree both details and noise at all scale levels. KAZE features [3] shows that by means of nonlinear diffusion filtering it is possible to obtain multiscale features that exhibit much higher repeatability and distinctiveness rates than previous algorithms based on the Gaussian scale space. At the cost of a moderate increase in computational cost compared to SURF, SIFT and CenSurE: a SURF improvement by using center-surround detectors [1].

Given an input image, KAZE features builds the nonlinear scale space up to a maximum evolution time using efficient Additive Operator Splitting (AOS) techniques and variable conductance diffusion. Alcantarilla et al. start the computation of this nonlinear scale space by taking a similar approach as done in SIFT, discretizing the scale space in logarithmic steps arranged in a series of O octaves and S sub-levels, but working with the original image resolution. The set of octaves and sub-levels are identified by a discrete octave index o and a sub-level one s. The octave and the sub-level indexes are mapped to their corresponding scale σ through the following formula:

$$\sigma_i(o,s) = \sigma_0 \cdot 2^{\{o+s\}/S},\ o \in [0,\dots,O-1], s \in [0,\dots,S-1],\ i \in [0,\dots,N], \tag{2.4}$$

where σ_0 is the base scale level and N is the total number of filtered images. Nonlinear diffusion filtering is defined in time terms, so the set of discrete scale levels in pixel units σ_i is converted into time units. In the case of the Gaussian scale space, the convolution of an image with a Gaussian of standard deviation σ (in pixels) is equivalent to filtering the image for some time $t = \sigma^2/2$. This conversion is applied in order to obtain a set of evolution times and transform the scale space $\sigma_i(o,s)$ to time units by means of the following mapping $\sigma_i \rightarrow t_i$:

$$t_i = \frac{1}{2}\sigma_i^2,\ i = \{0,\dots,N\}\,. \tag{2.5}$$

From the input image –previously convolved with a Gaussian kernel of standard deviation σ_0 to reduce noise– an image gradient histogram is computed to obtain the contrast parameter k in an automatic procedure proposed by Weickert in [123] using *Perona-Malik* diffusion equation [98]. Then, given the contrast parameter and the set of evolution times t_i, it is straightforward to build the nonlinear scale space in an iterative way using the AOS schemes (which are absolutely stable for any step size) as:

$$\mathbf{L}^{i+1} = \left(\mathbf{I} - (t_{i+1} - t_i) \cdot \sum_{l=1}^{m} \mathbf{A}_l(\mathbf{L}^i)\right)^{-1} \cdot \mathbf{L}^i, \tag{2.6}$$

where $\mathbf{L}$ is the original image and $\mathbf{A}_l$ is a matrix that encodes the image conductivities for each dimension, as discretized by Weickert et al. in [124].

Next, the response of scale-normalized determinant of the Hessian is computed at multiple scale levels to detect interest points. For multiscale feature detection, the set of differential operators needs to be normalized with respect to scale, since in general the amplitude of spatial derivatives decrease with scale:

$$\mathbf{L}_{Hessian} = \sigma^2 \left(\mathbf{L}_{xx}\mathbf{L}_{yy} - \mathbf{L}_{xy}^2 \right), \tag{2.7}$$

where $(\mathbf{L}_{xx}, \mathbf{L}_{yy})$ are the second order horizontal and vertical derivatives respectively, and $\mathbf{L}_{xy}$ is the second order cross derivative. Given the set of filtered images from the nonlinear scale space $\mathbf{L}^i$, the detector response is analyzed at different scale levels σ_i. We are looking for 2D features of interest that exhibit a maxima of the scale-normalized determinant of the Hessian response through the nonlinear scale space.

Finally, KAZE features compute the main or dominant orientation of the keypoint –similar to SURF– and obtain a scale and rotation invariant descriptor considering first order image derivatives.

HOG is widely deployed in the people detection stages of Chapters 3 and 4, due to its capability as a global descriptor to generalize a target, working quite well for shape and contour representations. However, experiments in Chapter 6 are in need of a local descriptor like KAZE to extract distinctive keypoints from people's appearance, so as to be able to recognize and identify a user among other individuals. KAZE's superiority in performance against other popular feature descriptor methods, together with its open-source nature –unlike SIFT and SURF– are the main reasons why we are implementing KAZE features in Chapter 6.

2.3 3D People Recognition Techniques

The reason behind the low propaganda of 3D people recognition techniques *expressly* acknowledged as such is because they actually fuse 2D implementations with 3D information, rather than going for a full three-dimensional approach. Either by using multiple sensors to retrieve 3D data or full scanning of the targets, is still expected to improve modeling efficiency, accuracy and convenience, with a 2D image processing method.

2.3.1 3D Active Appearance Models

As we saw earlier in this chapter, AAMs is a statistical approach that models the shape and texture of a target object. Unfortunately, the traditional AAMs framework could fail when the subject pose changes as only 2D information is used to model a 3D object. To overcome this limitation, a 3D AAMs framework is proposed in [24], in which a 3D shape model and an appearance model are used to model human faces.

The traditional 2D AAMs has a great success as long as only frontal images are matched. Due to head pose variations in the 3D world, 2D AAMs may fail to converge as already mentioned on previous paragraphs. To deal with this issue with the 2D AAMs framework, one way is to collect data in every possible pose which is unfeasible in practice.

In [128], a 2D+3D AAMs approach exploits the 2D shape and 3D shape models simultaneously. The shape instance generated by 2D AAMs is varied to be consistent with a possible 3D shape. This constraint is formulated as a part of the cost function. To combine this constraint into the original cost function, a balancing weighting is added.

The value of this weighting constant is empirically determined. Based on the existing 2D AAMs framework [81], it is proposed in [24] to directly use a 3D shape model with the appearance model. In this 3D AAMs framework, only frontal face images are needed for learning. While the aligning is carried out, the 3D shape model generates faces in different poses by varying rotation and translation of a frontal face instance. Their main contribution is to extend the search space of 2D AAMs to the 3D world and no balancing weighting between the 2D shape and 3D shape models is needed.

An updated version of this approach has been presented by Dopfer et al. [38] based on their previous work in [122], where they compared it against 3D Morphable Models by Blanz et al. [19] and with 3D AAM using 2D images, claiming comparable success rate but higher performance in angle estimations. Thus, the authors unify the methods of both papers and make the theoretical framework clearer while adding significant progress in the experimental evaluation. The experimental results demonstrate that the proposed algorithm using intensity and range data has a 86% success rate for face alignment while the 3D AAMs using intensity images only has a 68% success rate.

However, the treatment of the training data for 3D AAMs construction is a complex task which could be spread out over time for other targets that are not faces.

A shape $\mathbf{S}$ consists of a fixed number of points which describe the shape of the target object. The classical AAM used 2D points to describe the shape—here the shape is extended to 3D. The shape $\mathbf{S}$ can be written as:

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{pmatrix}. \tag{2.8}$$

A model is then constructed using a data set of m exemplar faces, each represented by its shape-vector $\mathbf{S}_i$. Since it is assumed all faces are in full correspondence, new shapes $\mathbf{S}_{model}$ can be expressed in barycentric coordinates:

$$\mathbf{S_{mod}} = \sum_{i=1}^{m} a_i \mathbf{S}_i, \tag{2.9}$$

where the set of face-shapes are parametrized by the coefficients $\vec{a} = \{a_i\}_{i=1}^{m}$.

The 3D shape of an object and its possible shape variations are learned from training data. Scale, rotation and translation variations are first removed from this training data. Next, orthogonal Procrustes analysis [111] should be performed to align the shapes. From the aligned shapes, the mean shape $\mathbf{S}_0$ is computed and subtracted to all the data in order to obtain the shape variations respect to the mean. Principal component analysis (PCA) [56] is then applied to the aligned shapes to find the shape variation bases:

$$\mathbf{S_{model}} = \mathbf{S}_0 + \sum_{i=1}^{m-1} \alpha_i \mathbf{s}_i, \tag{2.10}$$

where $\mathbf{s}_i$ are the eigenvectors of the covariance matrix $\mathbf{C}_S$ computed over the shape difference $\Delta\mathbf{S}_i = \mathbf{S}_i - \mathbf{S}_0$. The probability for coefficients $\vec{\alpha}$ is given by:

$$p(\vec{\alpha}) \sim \exp\left[-\frac{1}{2}\sum_{i=1}^{m-1}\left(\alpha_i/\sigma_i\right)^2\right], \tag{2.11}$$

with ${\sigma_i}^2$ being the eigenvalues of the shape covariance matrix $\mathbf{C}_S$.

As for the appearance model, *texture* or appearance is the color or intensity of a target object and can be represented by an appearance-vector $\mathbf{T}$ as:

$$\mathbf{T} = \begin{pmatrix} R_1 & R_2 & \dots & R_n \\ G_1 & G_2 & \dots & G_n \\ B_1 & B_2 & \dots & B_n \end{pmatrix}, \tag{2.12}$$

that contains the R, G, B color values of the n corresponding vertices, since it is assumed that the number of valid texture values in the texture map is equal to the number of vertices. A texture model is also constructed using the same conditions as in equation 2.9, expressing new textures $\mathbf{T}_{model}$ in barycentric coordinates as a linear combination of the shapes and textures of the m exemplar faces:

$$\mathbf{T_{mod}} = \sum_{i=1}^{m} b_i \mathbf{T}_i, \tag{2.13}$$

where the set of face-textures are parametrized by the coefficients $\vec{b} = \{b_i\}_{i=1}^{m}$.

To define an appearance space, we have a triangular mesh where its vertices correspond with the shapes landmarks: the mean shape (a.k.a. the reference shape). All texture from training images is mapped onto the mean shape by warping. In addition, the texture may change due to the lighting or camera settings. Therefore, normalization of the texture is necessary. From all mapped and normalized training images the mean appearance $\mathbf{T}_0$ is computed, then subtracted to all data and finally, PCA is performed to

compute the appearance variation bases:

$$\mathbf{T_{model}} = \mathbf{T}_0 + \sum_{i=1}^{m-1} \beta_i \mathbf{t}_i, \tag{2.14}$$

where $\mathbf{t}_i$ are the eigenvectors of the covariance matrix $\mathbf{C}_T$ computed over the texture difference $\Delta\mathbf{T}_i = \mathbf{T}_i - \mathbf{T}_0$. The probability $p(\vec{\beta})$ is computed similarly to $p(\vec{\alpha})$ (Eq. 2.11).

To apply the procedures described above, assumptions have to be met: correspondences between landmarks from all the training data is imperative, which is only achieved with the so-called *anatomical landmarks*. Meaning that every i-th landmark has to have the same anatomical significance to every and each one of our data vectors.

Building from scratch an anatomically landmarked database under these specific characteristics for other targets that are not faces is a very time consuming task –specially for larger targets like the human body–, as well as too computationally expensive if point clouds were employed.

2.3.2 RGB-D Sensor Information

A RGB-D sensor delivers a combination of three color channels (one for each color: red, green and blue) known as *RGB* image and its corresponding *depth* data in real time. A depth image is an image channel in which each pixel relates to a distance between the image plane and the corresponding object in the RGB image. These sensors have been exploited in computer vision for several years, but the high price and the poor quality of such devices have limited their applicability. However, with the development of the low-cost consumer RGB-D products –like Microsoft Kinect and Asus Xtion– high-resolution depth and visual sensing has become available for widespread use as an off-the-shelf technology.

Microsoft's original Kinect for Xbox 360 (Kinect for Windows v1) has been responsible for the uprising of low-budget 3D Scanning and became one of the most common RGB-D devices on the market. Kinect v1 is now outdated since the quality of data is not good enough for today's standards. Nevertheless, the release of Kinect for Xbox One (Kinect for Windows v2) –a new version with significantly expanded hardware capabilities– brought a promising device within reach. Table 2.1 summarizes a v1 versus v2 side-by-side comparison, where it is very noticeable how Kinect v2 outperforms Kinect v1.

The increase in resolution has been impressive with the v2 reaching full-HD and a 60% wider field of vision, which lessens the likelihood of losing track of the user. Kinect v2 sensor uses Time-of-Flight (ToF) technology, meaning that it computes the depth of objects it has in front of it by throwing infrared light rays and estimating how much time these rays need to bounce on surfaces and come back. Unlike v1 that calculates depth

using an infrared light pattern projection (Structured Light), which requires a production-time calibration step and a careful preparation of the environment. Brightness of the ambient light, dust, transparencies and reflective surfaces would give problems to Kinect v1 sensor, since it is difficult to distinguish the pattern projection under these conditions.

Kinect v2 tracks more people, with more joints, faster and with greater precision. Microsoft Kinect still continues to be the most known sensor and well the most powerful one on terms of intrinsic features.

Such premises make RGB-D cameras like Kinect sensor to have a huge impact on research, not only in the computer vision community, but also at related fields like robotics, human-machine interaction and image processing. This considerably pushed forward several areas such as: 3D reconstruction, camera localization and mapping (SLAM), gesture, object and activity recognition, people tracking, bilateral filtering, and many more.

The practical use of depth information is recognized as a key trait for many 3D multimedia applications. Over the years, researchers have attempted to develop technologies that generate a high-quality three-dimensional view. By using depth information, high-quality three-dimensional images can be generated in the form of a stereoscopic image, which provides the necessary sense of reality. Accordingly, extensive multimedia research based on depth information has been conducted, such as depth image-based rendering (DIBR), augmented reality (AR), virtual reality (VR) and mixed reality (MR).

And this is how the arrival of RGB-D technology in people recognition tasks has widened the target scope in recent years. Now it is possible to detect and track the human body shape, mapping its appearance with amazingly high precision –as will be seen in Chapter 5– and going as far as rendering body joints to construct a basic skeleton with ease. However, despite these new perceptions and their undeniable potential, human body shapes, their pose and the information they contain as a whole have rarely been targeted in research, or at least not as much as facial features.

For a long time, researchers have been challenged by many problems such as detecting and identifying humans in real-world situations. Traditional object segmentation and tracking algorithms based on RGB images are not always reliable when the environment is cluttered or the illumination conditions suddenly change, both of which occur frequently in a real-world setting. Nonetheless, effectively combining depth and RGB data can provide new solutions to these problems, where segmentation based on depth information has lighting invariance at medium-low ambient light levels and also surface texture invariance. Meanwhile, the accuracy of human tracking and identification has been improved by considering the depth, motion and appearance information of a subject, allowing us to take people recognition out of the lab and into real environments (e.g. homes, schools or hospitals).

There are great expectations that systems endowing RGB-D technology will lead to a boost of new 3D perception-based applications in the fields of ambient intelligence,

TABLE 2.1: Kinect v1 versus Kinect v2 technical specifications

	Kinect for Windows v1	**Kinect for Windows v2**
Color	640×480 @ $30fps$	1930×1080 @ $30fps$
Depth	320×240 @ $30fps$	512×424 @ $30fps$
Sensor	Structured Light (PrimeSense Light Coding)	Time of Flight (Tof)
Range	$0.8 \sim 4.0m$	$0.5 \sim 4.5m$
Angle of View (Horizontal/Vertical)	57/43 degree	70/60 degree
Microphone Array	◯	◯
Body Index	6 people	6 people
Body	2 people	6 people
Joint	20 joint/people	25 joint/people
Hand State	Open/Closed	Open/Closed/Lasso
Gesture	×	◯
Face	◯	⊙
Speech/Beamforming	◯	◯

robotics and visual augmented reality. It is already attracting a growing interest in natural user interfaces (NUIs), which, if combined with general human skills, allow the intuitive development of human-computer partnerships. A key premise for the human-computer interaction field and one of the main cornerstones of this dissertation. Experimentation with RGB-D sensors technology under a NUI framework will be explored in Chapter 6, working as a prelude towards a more natural interaction in a robotic or smart environment implementation.

2.4 People Recognition in Ambient Intelligence and Robotics

The Information Society and Technology Advisory Group (ISTAG) in the European Commission proposed the *Ambient Intelligence* (AmI) concept in the late 90's: Environments integrated with sensors and intelligent systems. The environments having the following properties:

- Awareness of the presence of individuals
- Recognition of the individual's identities
- Awareness of the contexts
- Recognition of activities
- Adaptation to changing needs of individuals

In consequence, AmI is capable to deliver personalized services automatically in anticipation of the needs of inhabitants and visitors. None of these AmI specifications were tightly defined by ISTAG, as it aptly recognized the complexity and rapid evolution of the technologies and markets involved. Instead, it took a holistic approach and identified which areas were more susceptible for the realization of AmI in aspects like technology, society and business.

The *ambient* side of a system includes sensors, processors, communications and adaptive software. They have all made huge progress in the past few years. Smartphones and many other wearable devices on Internet of Things (IoT) markets will continue to drive the miniaturization and cost reduction. In a not too distant future, the technologies will practically be built into everything around us.

The *intelligence* side requires an AI system that can automatically and accurately track people and their interactions with the environment. Intelligent agents or robots can automatically perform tasks within the environments to serve the needs of the people.

ISTAG proposed a user-centered participation view in approaching AmI development [108]: "AmI needs to be driven by humanistic concerns, not technologically determined once" and should be "controllable by ordinary people". The interaction within ambient intelligence should be unobtrusive and not involving a steep learning curve. You should not need a manual to participate in AmI environments. It should just work as you walk into an AmI environment, enhancing social interactions of all the participants within the environment.

However, AmI needs to work reliably within the constraints of the state-of-the-art technologies. Factors like accuracy, capacity and fail-safe measures for all the hardware and software components need to be taken into consideration. Recently, Microsoft's *Tay bot* [55] is a clear real-life example on how artificial intelligence can unexpectedly go rogue. There is still a lot of work to be done in order to achieve fully understanding on how complex AI systems behave.

There has been also much interest in the so-called *service robots* which operate in populated environments [7, 62], and the topic of human-robot interaction has attracted even more attention recently.

According to ISO 8373 robots require "a degree of autonomy", which is the "ability to perform intended tasks based on current state and sensing, without human intervention". For service robots this ranges from partial autonomy –including human-robot interaction– to full autonomy –without active human-robot intervention. The statistics of the International Federation of Robotics (IFR) for service robots therefore include systems based on some degree of human-robot interaction or even full teleoperation as well as fully autonomous systems.

Service robots are categorized according to personal or professional use. They have many forms and structures as well as application areas, particularly in *smart environments*,

where ambient intelligence technologies are heavily implemented. Potential smart environments can be found in homes, hospitals, transportation, roads, offices, classrooms and museums.

Most research on classical robotic platforms has focused on perceiving its environment and storing data onto its database for later use. In this approach, robots are then equipped with many expensive sensors and work based on complex algorithms and a huge database for decision-making. Despite this heavy configuration, the performance of the robotic system is not very satisfactory.

These systems are not immune to occasional failures, and the manner in which they fail can seriously effect users' perception of those systems and the services they provide. Studies indicate that failure by a robotic service lowers users' trust and can make them reluctant to use the service again. Other potential negative consequences of failures include negative feelings toward the service, inability to recover the task that was being performed and feelings of helplessness or uncertainty. Unexpected behaviors resulting from failing autonomous robots can also generate negative side effects. Aside from the possibility of causing additional problems or even being potentially dangerous, unexpected behaviors can have far reaching consequences such as instigating social conflicts between people.

As fully autonomous robotic services are integrated into society, they will need to be capable of interacting with people in a variety of relationships with the robot and not just the operator. These people will require different kinds of information and expect different kinds of recovery behaviors based on their relationship with the robot.

On the other hand, a light-weight robot in a smart environment can perform complex tasks and provide more reliable services in which the environment is full of sensors, actuators and service robots interwoven through wireless communications networks. Smart environments can achieve a symbiosis between the robot and the environment, which in return gives a positive impression to the people interacting with the system, since humans participate in a more natural –explicit and implicit– communication process.

2.4.1 Natural Interaction

NUIs are interfaces which are designed to feel as natural as possible to the user by reusing general human skills, in order to achieve appropriate interaction with specific contents related to a machine. NUIs are here to replace and break away from the traditional keyboard and mouse, birthing a new concept for modern interaction: *natural interaction*.

In real life situations, people communicates naturally through gestures, expressions and movements. Research work in natural interaction means to create and develop systems that understand these actions and engage people to interact with a system or sensor-intelligent environment, in a more *natural* and effective way.

Echoing the same viewpoint as Valli's in [120], "people do not need to wear any device or learn any instruction", interaction should be intuitive. A person experiencing a system based on natural interaction is not necessarily an active or willing user –as we shall see in Chapter 6– this person can be simply passing by and enjoying passively the encounter.

A natural interface activates the cognitive and cybernetic dynamics that people commonly experience in real life, thus persuading them they are not dealing with abstract digital media, but with physical and real objects. This results in a reduction of the cognitive load, hence increasing the amount of attention on a content.

From this perspective, the success of a natural interaction system depends on how it influences people experiencing it, which is a synthesis of a broad number of aspects, like non-obtrusive sensing, visualization, cognitive load and response times.

2.4.2 Group Tracking with Social Cues

Visual analysis of human motion attempts to detect, track and recognize people, and more generally, the understanding of human behaviors from image sequences involving humans [85, 97, 42]. This strong interest is driven by a wide spectrum of promising applications in many areas such as content-based image storage and retrieval, video conferencing, smart surveillance [50], perceptual interfaces, virtual reality and robotics [6], just to name a few.

Tracking groups of people is an important skill for surveillance systems and robots which operate in populated environments. Research has found that up to 70% of pedestrians walk in groups [90]. People are social beings and as such they form groups, interact with each other, merge to larger groups or separate from them. Group dynamics have been widely discussed on earlier works [82, 31, 114]. The harvest of powerful knowledge about groups, their position, size, motion state and social behavior, can enable systems to gain a deeper understanding of human environments and provide better services to users.

What makes an assemblage of people a *group* is an extremely complex question in general which involves difficult-to-measure social relations among subjects. A concept related to this topic is the proxemics theory introduced by Hall in [52]. It was found from a series of psychological experiments that social relations among people are reliably correlated with physical distance during interaction. Correlation allows to infer group affiliations by means of available body spacing information, an approach further explored in Chapter 4.

The ability of robots endowing cameras to keep track of people in their surroundings is a major issue. While tracking individual people is a highly studied problem in target tracking, computer vision and robotics, the problem of tracking groups of people had

been barely explored [48]. However, the number of related works has been recently increasing due to activities in the visual surveillance and social computing communities [36, 95].

Arras et al. [71, 70] have been constantly working on their recursive multi-model hypothesis theory for tracking groups of people. This approach seeks to reflect group formation process and gain efficiency in situations where maintaining the state of individual people would be intractable. The theory relies heavily on learning group models and their cohesion probabilities.

State-of-the-art in the group tracking area undoubtedly includes these works, where even an outstanding tracking performance is achieved in real-time using RGB-D data [73]. Nonetheless, research in which a group of observed people is not only detected as a whole, but accurately categorized by means of a cognitive role assigning is non-existent to our knowledge.

Recognition of human activities under logic-based scenarios has been adopted by various event detection algorithms including hierarchical constraint satisfaction [112], multithread parsing [132], randomized stochastic local search [21], probabilistic inference on constraint flow [66] and Markov logic networks [87]. However, most of these approaches focus on events with a single agent only [72, 84], or events that can be detected without role identification [21, 87]. Otherwise, roles are initialized by prior information or human intervention [66, 112, 132].

Our study in Chapter 4 revolves around an approach in which, rather than detecting events, it detects roles attached to a motion trigger that analyzes the cohesion of groups as well as the membership or non-membership of the people involved. It performs dynamic group tracking by matching spatio-temporal occurrences through proxemics, and role-logic assignments based on motion directives, without any manual annotations [65] or trained models [68]. As such, it opens up new avenues for research featuring valuable information about the social relationships and interactions within the tracked group, which are highly pursued objectives in the robotics field.

2.5 Summary

We have explored the literature and state-of-the-art in the interdisciplinary fields relevant to the research conducted in this dissertation and provided insights into the trends of people recognition and identification which motivate this research.

Consequently, a critical evaluation of the works related to our research topic has been sustained, giving rise to the identification of gaps in current knowledge. From 2D and 3D techniques, through hybrid approaches and right up to natural interaction in intelligent environments, several unpractical approaches and computationally cost-effective procedures have been detected.

Quite a variety of highly reputed techniques and methodologies were thoroughly analyzed, taking into consideration both the advantages and the disadvantages linked to the thesis statement and our particular target. This has resulted not only in acknowledging or dismissing them, but in discussing the reasons behind their likely support or opposition, pointing to where we are and where we are going, including the challenges we will face getting there.

Hence, the existing limitations -whether of hardware or software- highlighted in some theories, set the starting point for the ideas that are going to be presented in the following chapters of this dissertation. Ideas with a straightforward train of thought: harnessing 3D resources and exploiting machine learning and computer vision algorithms to obtain practical solutions that boost the overall performance of people recognition tasks by targeting human body shapes, so we can optimize the naturalness of human-machine interactions.

Chapter 3

Single-Camera Automatic Landmarking with an Ensemble of Regression Trees

Landmarks are key features which identify an object. They are spatial items such as the tip of the nose, the corners of the eyes and the edges of the mouth in a face. The localization of landmarks provide reference points from which further correspondences can be interpolated or inferred and thereby enable a reliable matching.

When performing the shape learning problem, *automatic landmarking* refers to the automated detection and location of landmarks. This is a difficult task since landmarks appearance can have a large degree of variability. This variability can be ascribed to intrinsic differences between individuals and to extrinsic factors due to illumination conditions when an image is captured. The problem appears even more challenging when the target is not a rigid object and large changes in pose are very likely to occur, as in the case of human body shapes.

A vast body of work exists on the estimation of articulated human pose from images and video using automatically generated landmarks. Recent studies on pose estimation from still images reveal prominent results using regressors [34, 105, 79]. According to this knowledge, we ask ourselves if a successful approach for face alignment [59] could be extrapolated to the automatic landmarking of human contours on still images. Hence, in this chapter we propose to extend Kazemi and Sullivan's algorithm from face alignment to body shapes. The algorithm employs a cascade of regressors combined with decision-tree learning, addressing the latest techniques on model-based approaches and pose estimation.

3.1 Introduction

We will check our hypothesis about the feasibility to extrapolate the Kazemi and Sullivan's algorithm from face alignment to body shapes into an experimental framework. Experimentation will follow this flow: From a collection of 46 images with 4 different subjects, a fraction is used for training the first stage, the *people detector*; next, another fraction is used for training and testing in the *body shape predictor* stage; finally, these two

stages converge in the last stage: *automatic landmarking*, where the remaining images are used for testing.

One of the subjects that appears in the training set for the people detector stage is not considered in the shape predictor training. We have done this as a way to validate our algorithm. The output of this trainer produces a general shape model for human bodies which, combined with a people detector algorithm, will be used to fit in and automatically landmark the body contours of new images, either from the 3 trained subjects or the unseen fourth subject.

Results will confirm that, with proper adjustments, the considered face alignment algorithm can be exported to human bodies with a very low computational time for testing, which fosters real-time implementation for robots equipped with a single camera.

The landmarking task is carried out only annotating a significantly small dataset, without any motion capture data [46] or multiple camera views [37]. System execution is validated in real-time by means of a speed test, which provides more solid evidence to our purposes.

RGB-D sensors technology could also contribute to widen the method's scope [38], but the aim in this chapter is to start with the 2D data scenario and analyze the accuracy for less computationally expensive instances. The applicability of active appearance models (AAM) and other model-based techniques could be enlarged under this framework in the people recognition and tracking domain.

3.2 People Detector

The first phase of the overall proposed system is a people detector training stage. It will be implemented using the `imglab` tool from the *dlib C++ Library*[1]. From a set of images of different subjects obtained by webcam streaming, a subset is used to train the people detector, and they will not be used in further stages of the system.

Hence, from a set of 46 images provided by 4 different subjects from a webcam streaming with a 640×480 standard size, 26 of them were taken only to train the people detector algorithm. As a matter of fact, none of these 46 images duplicate in other subsets. Each dataset has their own share so as to demonstrate the algorithm is not influenced by preferences for a particular subject or shape instance.

Using these 26 images -from Subject 2, Subject 3 and Subject 4- as an input in the `imglab` tool we were able to annotate our dataset with red bounding boxes over the whole body of each subject and label them as *body_detected*, as depicted in Figure 3.1.

Apart from the subject, there were no other people on the scene nor the presence of any object with anthropomorphic characteristics within the images. Therefore, crossed

[1] http://dlib.net/

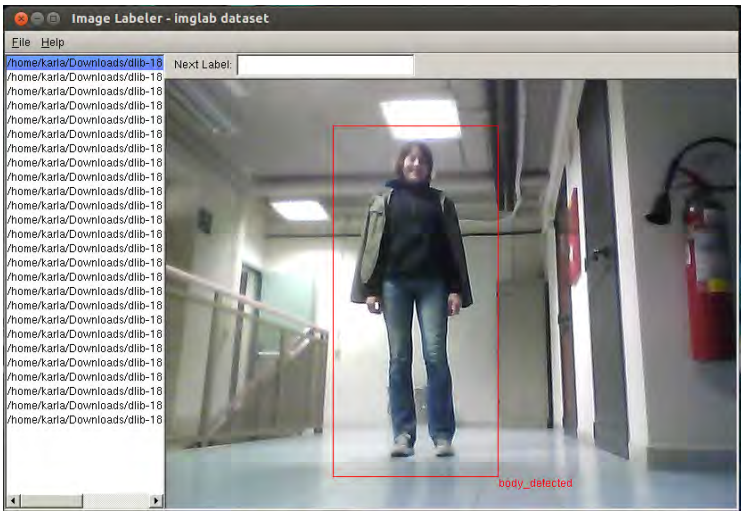


FIGURE 3.1: Annotating images with `imglab` to train the people detector stage.

bounding boxes were not necessary to indicate false positives which *dlib C++* code should ignore when performing the training detections.

3.2.1 Training

Several tests were conducted to obtain suitable training parameters for the best possible detection results. Moreover, variables like number of training images and tightness of the bounding box were taken into account.

The most significant training parameters for the people detector system are shown in Table 3.1 with their selected values. The *dlib C++ Library* goes through the steps to train a kind of sliding window object detector as the one published in [32] and summarized in Figure 3.2, applying HOG as feature descriptor and linear SVM as baseline classifier. Consequently, the trainer endows the usual SVM's *C* hyper-parameter (see Table 3.1). The most favourable value for *C* was empirically found by analyzing the performance of the trained detector on a test set of new images, all of them images not used on the training stage. In general, higher values for *C* encourages it to fit the training data better but might lead to overfitting.

The trainer keeps running until the "risk gap" is small enough (smaller than the *epsilon* value in Table 3.1). For most problems a value in the range from 0.1 to 0.01 is plenty accurate. Smaller values prompt the trainer to solve the SVM optimization with a higher accuracy, although it will take longer to train. The term *target-size* in Table 3.1 refers to the size in pixels of the sliding window detector.

For this training data in particular, the *upsample* function does not improve the results. Acting as a pre-processing step, *upsample* increases the size of the images by a factor of 2 allowing us to detect smaller features on the target object. This is not our case since

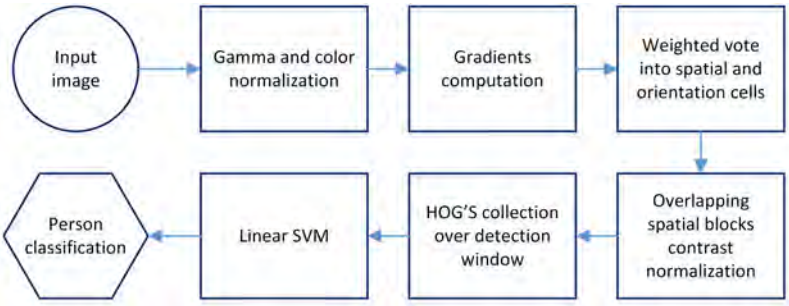


FIGURE 3.2: An overview on the feature extraction and object detection algorithm by Dalal and Triggs. First, HOG feature vectors are extracted from the detector window tiled with a grid of overlapping blocks. Next, the detection window is scanned across the input image at all positions and scales. Finally, the combined vectors are fed to a linear SVM for object or non-object classification.

we are focusing just in human silhouettes, which already are quite large to be detected in any circumstance.

Since human bodies are usually left-right symmetric we can increase our training dataset by adding mirrored versions of each image with the *flip* option. This step is really useful as it doubles the size of our training dataset from 26 to 52 images, improving significantly the results. Another remarkable detail is about training the bounding boxes a bit loose from the bodies. If trained too tight, detections tend to be partial. Under final corrections, our people detector started with detections like the one shown in Figure 3.3a, but with proper adjustments achieved results as in Figure 3.3b. Bodies with complicated poses are difficult to entirely contain them in the detection box (Figure 3.3c). Nevertheless, setting the parameters to their final values on Table 3.1 led to obtain better detections of this particular instance (Figure 3.3d).

TABLE 3.1: Default parameter values set by the example program versus selected values after running some tests.

Parameter	Default Value	Selected Value
C	1	0.65
epsilon	0.01	0.01
target-size	80×80	100×100
upsample	no	no
flip	no	yes

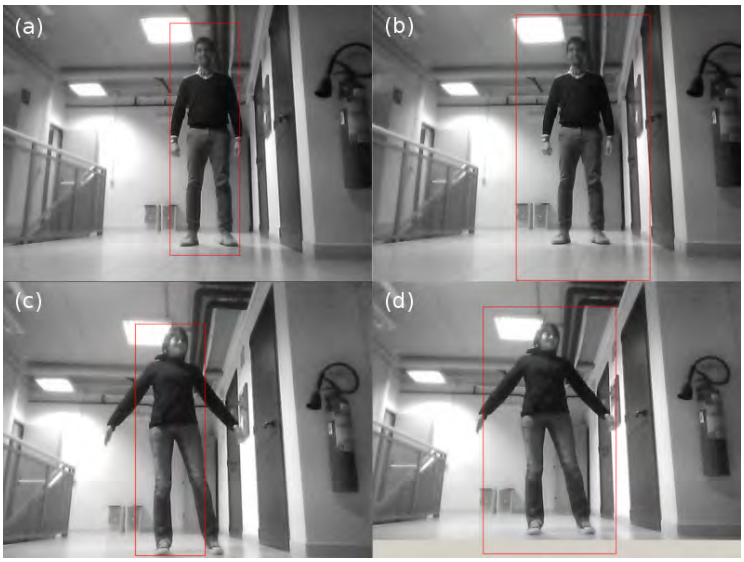


FIGURE 3.3: People Detector results. (a) Natural standing pose, detection box is tight and subject's left hand is partially missed. (b) Same image as (a) with final selection of parameter values, albeit the detection box is now too loose for the target, detection is complete. (c) Subject spreading her limbs, detection box not able to contain both arms nor her left foot. (d) Same image as (c) with final selection of parameter values, detection box only misses the tip of the right hand.

3.3 Body Shape Predictor

Functions in the *dlib C++ Library* implement methods described in [59], where they show how an ensemble of regression trees can be used to estimate face landmark positions directly from a sparse subset of pixel intensities, achieving real-time performance with high quality predictions. Similarly, we have trained a shape model by estimating body landmark locations to evaluate and demonstrate whether the algorithm can be extrapolated to body contours as well.

The directory for the body shape predictor contains a training dataset and a separate testing dataset. The training set consists in 4 images from Subject 1, Subject 2 and Subject 3; while the testing set comprises mirrored versions of the training set images. Annotated images contain a bounding box with 180 landmarks on each body shape. The objective is to use this training data for learning to identify the position of landmarks on human bodies in new images. Once the shape predictor has been trained, it is tested on previously unseen data, the testing dataset.

It is important to emphasize the flexibility in the number of proposed landmarks. Depending on the application, one could envision to train a new model with more or less landmarks, which would directly affect the smoothness of the shape contour and the accuracy of the fit, becoming a trade-off between faster/slower training and a rough/clean shape.

The baseline method from Kazemi and Sullivan uses 68 landmarks and it is applied on *faces*. Face shapes are rather small compared to the entire human body. It is only natural that while studying this reference, we try to act proportionally when deciding on the number of landmarks to be trained for the body shape predictor, as well as their allocation criteria.

Thus, after analyzing the average closeness of the landmarks and the output we would prefer to obtain –the general expectation in research is a smooth humanoid shape– it prompted us to aim for a generalist proposal by defining 180 landmarks.

3.3.1 Training

Along this study, we are assuming to work on a very small training dataset since: (i) there are no available landmarked datasets for 2D human body contours, so we built the training dataset ourselves by hand, and (ii) it is unrealistic to assume large datasets of hand-made landmarked people. Training data is loaded from a XML file having a list of the images in each dataset and also containing the positions of the body detection boxes and their landmarks. A total of 180 landmarks were distributed by hand as shown in Figure 3.4, quite close to each other so that the body shape predictor trainer could be more efficient in generating a smooth shape model.

The body shape predictor trainer has several parameters to be tuned. A general overview of this training is depicted in Figure 3.5, for a more detailed explanation we refer to [59]. Default values from the *dlib C++ Library*'s function are used in our experimentation, except for: higher *oversampling* value to effectively boost the training set size, which is really small in our experiments; smaller *nu* value to reduce the capacity of the

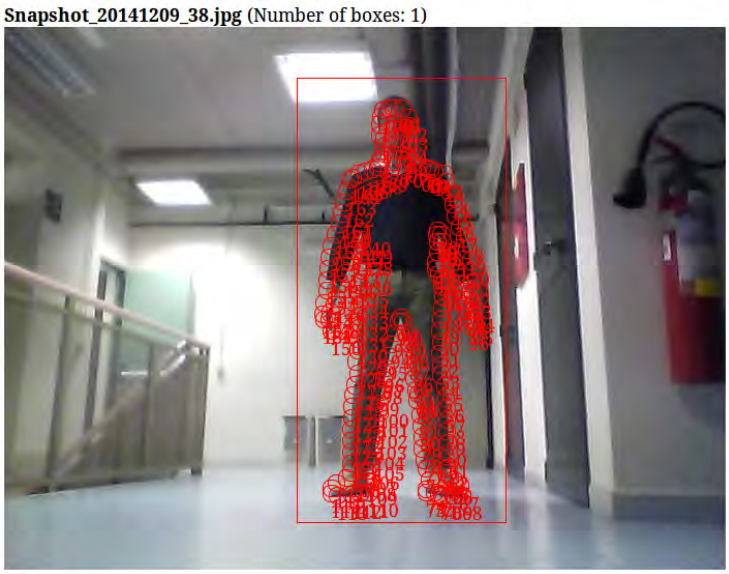


FIGURE 3.4: Target landmarked by 180 enumerated points.

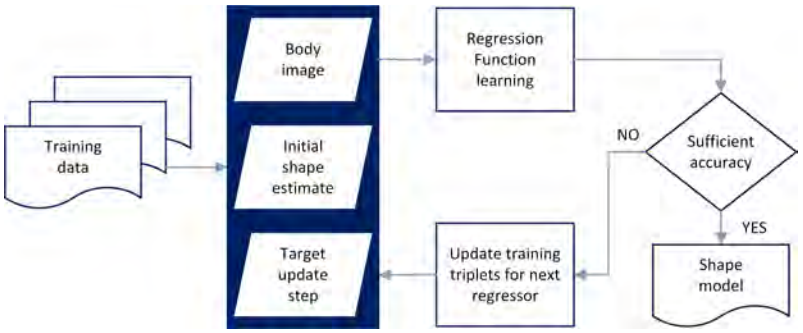


FIGURE 3.5: Cascade of regressors training from the algorithm by Kazemi and Sullivan to estimate position of facial landmarks. Regression functions are learned using gradient tree boosting with a sum of square error loss. The core of each regression function is the tree-based regressors fit to the residual targets during the gradient boosting algorithm. Decision at each split node in the regression tree is based on thresholding the difference between two pixels intensities.

model by explicitly increasing the regularization; and smaller value for the *tree-depth*, to reduce the model as well.

With this information, the algorithm generates the body shape model. The generated model is validated using a measure of the average distance (in pixels) between body landmarks obtained from the shape predictor and where they should be according to the training set, which is called *mean training error*. Yet, the real test lies in how well the predictor performs on unseen data. We test the recently created shape model on a testing set of landmarked mirrored images to avoid hand-marking all over again, obtaining a *mean testing error* which validates in the exact same way as its "training error" counterpart.

Only one parametrized testing for the body shape predictor has been conducted so far. Setting values to 500 for *oversampling*, *nu* equal to 0.05 and a *tree-depth* value of 2, the shape predictor takes around 3 minutes to train, obtaining a *mean training error* of 9.03275 pixels and a *mean testing error* of 68.8172 pixels. The overall accuracy of the method lies in this *mean testing error* value.

Taking into account the standard resolution of all the images submitted for experimentation –640 $\times$ 480 pixels– the mean error of 68.8172 pixels represents approximately a 11% displacement error in width and a 14% displacement error in height, with respect to the input images size. As we proceed onto the next stage of the system with the generated shape model and its characteristics, the displacement error will be reflected in the results from Section 3.4, noticing the shifting between the predicted position of the shape and where it actually should be.

3.4 Automatic Landmark Detection

Once the people detector and the body shape model have been trained, it is time to combine them to find front-position human bodies in an image and estimate their pose. The pose takes the form of 180 connected landmarks describing the shape or silhouette of the subject on a given image. Again, this pose estimator was created using *dlib C++ Library*'s implementation of [59]. A new testing dataset was built with the 12 remaining images from the initial 46 that were provided for the experimentation. This time with images from all 4 subjects. Subject 4 appears in the training stage of the people detector, however, it was not included in the training of the shape model. Hence, Subject 4 is a *completely unseen* target for our automatic landmarking process, as the people detector stage just helps the overall system to narrow the body search to one area of the image but does not influence the shape fitting procedure.

We will now focus our attention on the automatic landmarking results shown in Figure 3.6. Throughout the first three frames, the subject maintains a natural standing pose with minor variations. It is noticeable from the first (Figure 3.6a) to the second frame (Figure 3.6b) how the algorithm is able to refine the detection and make a better fit of the actual position of the head and feet. It keeps improving until the third frame (Figure 3.6c), where the algorithm actually manages to obtain a good estimation of the real location of the arms and hands of the subject as well. In Figure 3.6d the algorithm still tries to catch a pose although the subject is partially sided while stepping out from the visible range of the webcam, which is quite remarkable.

For the experiments introduced in this chapter, it is important to stress that natural standing poses were a requirement due to general objectives of the project. Which means

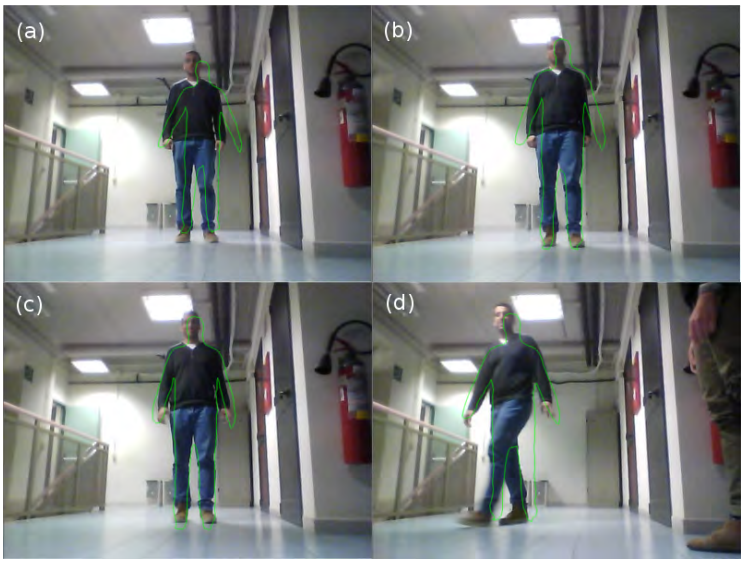


FIGURE 3.6: Automatic Landmarking. *Natural standing* case with a subject at a regular distance from the webcam.

we do not want to depend on a particular flashy pose that triggers a response from the robot, but rather make the robot or intelligent agent to notice and identify you by just normally passing by, as in any human notion of another human presence.

However, it was important for us to have some images in the dataset with people also displaying slightly out of the ordinary poses (see Figure 3.7b), so we could reveal the scope and limitations of the algorithm facing this type of cases as well. It is evident we have a re-scaling distance issue as seen in Figure 3.7, but at least the algorithm makes an effort to fit into the spread arms of our subject in Figure 3.7b. Probably, with more similar frames to this one, the algorithm would end refining that pose estimation.

All these automatic landmarking experiments were performed in about 126 milliseconds in a laptop with Intel Core i5-480M and 4GB RAM memory. Depending on the image it could take more or less than this mean test time. Yet, the procedure in general does not overpass 130 milliseconds, where 80 to 90 milliseconds correspond merely to the people detector stage.

Finally, we present a set of images from the same subject in Figure 3.8, the unseen Subject 4. The images leaded to problems in shape prediction due to the training stage of the people detector (Figure 3.8a). With the final tuning of the parameters we achieved a better detection. Even though the box does not reach the top of the head (Figure 3.8b) it is good enough for the next stage, as the shape predictor starts its landmarking at the top of the forehead instead. While in the process of strategically placing the 180 landmarks on the training data for the shape predictor, we considered *hair* as a very unstable feature that can be dismissed without major consequences. Thus, landmarking starts on the forehead to encircle just the face of the person, becoming a constant human body feature

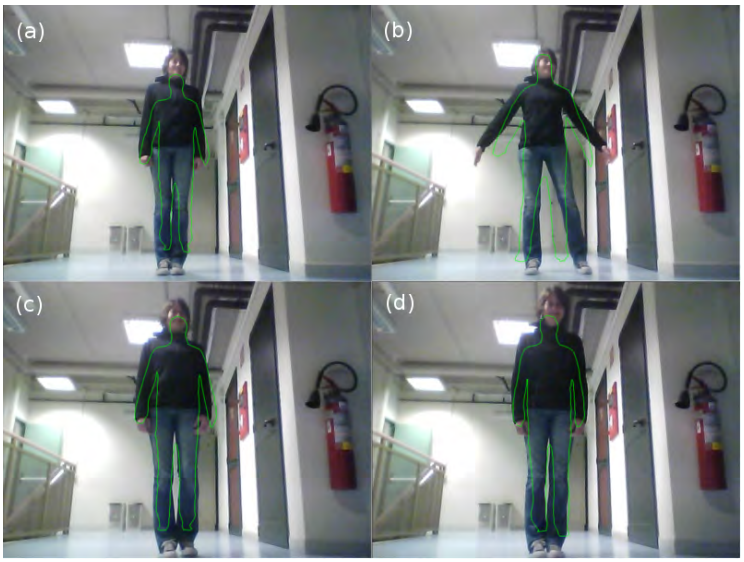


FIGURE 3.7: Automatic Landmarking. *Special pose* case with a subject approximating to the webcam.

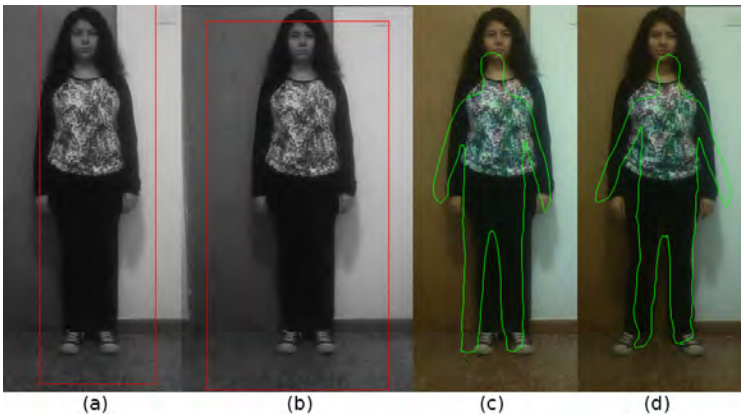


FIGURE 3.8: A difficult case. (a) First test output from people detector. (b) Output with selected values for people detector in its last testing session. (c) Worst automatic landmark detection of the dataset. (d) The automatic landmark detection does not get any better after a second frame of the subject as seen in other cases.

implicitly.

Figure 3.8c and Figure 3.8d present the most disappointing outcomes for automatic landmarking. There is no real improvement respect to the first frame but rather grows worse estimating spread arms instead of spread legs and shortening them; however, it achieves to fit well the left leg. This last fact can be regarded to a significantly slower learning of the unseen body shape. The subject maintains almost the same pose throughout the frames, but apparently this does not help the algorithm to work easier on the image. The problem seems to lie in the subject's clothing rather than in the *unseen target* fact. The clothes worn by Subject 4 are almost completely black and there are no considerable disparities in color between body parts, which has allegedly become more difficult to overcome for the algorithm.

3.5 Conclusions

In this chapter we have presented an automatic landmarking approach for human body shapes to be carried out on still images from a single camera by training a dataset of only 4 manually-annotated images. It demonstrated an acceptable performance in terms of accuracy and computational time, proving –to a certain extent– that facial automatic landmarking algorithms from the current literature can be adequate for targeting body contours.

The approach could be really practical and suitable for domestic service robots equipped with basic technological resources due to economic reasons or efficiency purposes, which becomes a valuable contribution to the cause.

Despite leaving aside some areas requiring attention for improvement, the results obtained are quite promising and fit in the established overall objectives of this dissertation. With adjustments, our proposal could eventually accomplish to be a fast and easy implementation on the primary stages of 2D body-shape models construction, and subsequently, on people recognition activities.

Accuracy could be refined with further experimentation and tuning of the shape predictor parameters, besides feeding more landmarked images to the training dataset. A face detector stage could be also useful to reduce the fitting error. The system shows potential to perform in real-time with live video streaming from a webcam, thus enabling the system for people tracking tasks.

In order to set the tone for future work, it would be interesting to see an attempt to implement *inter-shoulder distance* on the approach proposed in this chapter. Analogue to what inter-ocular distance achieves on face alignment problems, inter-shoulder distance could become the appropriate solution to re-scaling distances. *Inter-hip* distance may as well reinforce the body landmarking stage and exceed upcoming results.

Chapter 4

Towards robots reasoning about group behavior of museum visitors: Leader detection and group tracking

In the previous chapter, a purely technical approach have been devised to tackle just one of many facets in the people recognition problem. However, major challenges concerning the implementation of real-world applications remain. In order to ensure full immersion of the systems which will bring us closer to a more natural interaction, we should look for solutions embedding logical and cognitive reasoning into a machine.

The field of human-robot interaction (HRI) is a broad community encompassing robotics, artificial intelligence (AI), human-computer interaction (HCI), psychology and social science. HRI in social robotics explores important issues in designing a robot system that works with people in daily life environments, capable of interacting with, modeling, and learning from humans. Robotic systems should improve their capabilities to not only understand humans but also to convey their intention within their actions.

In this chapter we show that this kind of behaviour is achievable through a field study conducted at a science museum. We introduce a computer vision algorithm which is able to detect and track a leader within a group of people –the science communicator, for this particular case– and distinguish between group members and non-group members as well, all by means of a cognitive and logical behaviour analysis of their interactions on scene. The leader's direction is also computed as an attention reference for this approach.

The computer vision system is supervising people within a group following a guide to prevent accidents and missing people. This proposal represents one of a wide range of possible applications and future scenarios where group interactions are a key aspect for robots to understand and effectively participate in social environments.

4.1 Introduction

A museum is one of many HRI friendly environments where strong *leader-group* roles are relevant and quite valuable to maintain people organized and safe, such as schoolchildren walking, therapy groups and guided tours on industrial or commercial facilities.

Large groups of people require heightened attention on the part of the leader guiding them. A robotic companion, acting as an autonomous and independent being that could interact with the crowd and assist the group leader by processing their cohesion information, can alleviate the workload. Rather than relying on a smartphone or tablet platform to provide this data, an activity that could easily distract the leader and break their concentration in many possible ways.

Since the 1990's research has explored possible implementations of companion robots as robot museum guides [23, 22]. Some studies have been focusing on the human-robot interaction addressing the robot's ability to create a short-term interaction with visitors at the museum, as well as the robot's ability to navigate in populated, dynamic and unpredictable places such as the museum environment [117]. Other studies have been looking into creating believable social robots, exploring the robots abilities to create emotional contact with museum visitors through eye contact or engaging dialogues with the robots audiences on the museum tour [49, 64, 63]. These studies have mainly been conducted with a focus on the technological abilities of the robot in a quest to optimize the robot's functionalities to become the guide. However not many have questioned to pose the robot as an assistant to a human museum guide. Thus providing support and empowering the human experts in their tasks rather than acting as a replacement [80].

One of these tasks would be to prevent visitors from hiving off the group and inevitably get lost. A recurrent incident, specially for children, which is the reason why missing persons is a major concern in crowded environments such as a museum. Moreover, a robotic assistant would keep visitors safe and aware from restricted areas in the museum that could be a source of accidents as well. It is important to have support on this matter, especially when a science communicator deals with large groups of visitors.

The aim of this application is to track and reason about social grouping with highly defined roles using a simplistic cognitive process, rather than learning a priori social relations through labeled training and models [68], reference objects [65] or online-human expert encoding [113]. This cognitive development detects the subject holding a leader role within the group and, subsequently, assigns the rest of corresponding roles as either, group members or non-group members. The assignment is based on a behavioral analysis of the group in reference to their selected leader, motion and spacing interactions.

Experiments in this work are a first attempt to prove the feasibility of the proposed approach. The intention is to later demonstrate that a robotic system endowed with computer vision capabilities can track a group of museum visitors following the lead of a science communicator. Hence, further implementation on robots of other HRI areas with

a similar behavioural setup would help in supervision tasks by adding a new technological support.

4.2 Methodological Approach

To detect a person holding the leadership position within a group of people is an easy task for a human being. For us, it just takes a few seconds after witnessing the interactions between a group, to identify the members and recognize the leader among them. Thus, we should formulate how to approach the problem to convey this natural and cognitive capacity to discern into a machine.

However, before entering into specifics, it seems necessary to clarify why we are not going to use the automatic landmarking approach from Chapter 3 to handle people detection tasks in the experimentation that will be depicted on this chapter.

Automatic landmarking in Chapter 3 was trained only on *frontal* poses. People detection in a moderately crowded environment like a museum, presents a higher complexity and variability of human poses. Re-utilizing this method to tackle people detection tasks in the current work would have meant to re-train the algorithm with several different poses by hand-marking each input image with 180 landmarks.

Despite the method's advantage of requiring small datasets to deliver sufficient accuracy, it represents a workload we cannot not afford at the moment, since there are even more important objectives to pursue: group categorization and leader's direction estimation. Both problems are new and unexplored yet to the scientific community, and so they demand more time investment in their research.

4.2.1 Analyzing behavior to select and track a leader

Beyond any distinctive piece of cloth, color or equipment, a leader is all about attitude. There are distinctive traits or inferences that could rapidly spot the person in charge. One of them is body language: leaders tend to gesticulate the most within a group, rather because they are expressing an idea or giving directions to other members. Another indicator, surely the most obvious one, is leading translations: is this person always ahead of the group? is she/he just being followed by most of the members, guiding their way?

These features can be reflected in computer vision algorithms by means of a high variation in position of the bounding boxes obtained with a people detector. Major gesticulation leads to variations in size (width and height) of the bounding box, whereas quick displacements become important translations in Cartesian directions. Both assertions represent significant changes in the bounding box position that other members of the group certainly would not display throughout the frames of a given scene.

Nevertheless, the 'bounding box' approach alone is scarcely effective. People detector algorithms lose stability when it comes to precision of position recall. Even if a person

remains in the same place with the same pose from one frame to another, the people detector would likely retrieve a bounding box with a different position and size each time, despite being a true positive. In Section 4.3.1 it is described how this premise is reinforced with a motion detection strategy which results in a robust group categorization algorithm.

According to the features selected, which strongly relies in building an artificial cognition system based on human motion behavioral analysis, the general methodology for the group-leader tracking is now exposed. Figure 4.1 shows the structure behind the *Group-Leader Tracker* algorithm, which detects the leader within a group of observed people and, consequently, classifies subjects on the scene as either, part of the group or not.

4.2.2 Detections correspondence and filtering

The initial step is detecting all the individuals on the image scene using OpenCV's *People Detector* algorithm[1] [32] with a fine tuning of its parameters. In the interest of achieving the best possible performance for the detector, more than 30 training tests were carried out with several different parameter settings in the experimentation phase.

Unfortunately, people detection does not preserve the same appearance order throughout all the analyzed frames. *Match Detection* makes sure to relate a human detection in the current frame with its corresponding detection on a previous frame. In this stage, the coordinates of each detection box in the current frame are compared with the coordinates of all the detections on the previous frame. The minimum distance between coordinates is assumed to be the same detection in both frames.

FIGURE 4.1: General scheme for the Group-Leader Tracker algorithm.

[1] https://opencv.org/

Once detections are matched with people in the scene, the areas' difference between bounding boxes $\Delta x, \Delta y$ is quantified and stored as a cumulative sum. The *Accumulate Differences* stage applies a threshold filter which prevents outrageous values to sum-up, originating from possible grouping errors or false-positive detections from the people detector, as generally these values may disturb the algorithm and affect the performance of upcoming stages.

4.3 Motion Detection and Group Categorization

In order to categorize people in the video stream, both motion and gesture detection algorithms will be employed in a fast and accurate form. From the detected motion, tags will be assigned to the people present in the scene according to the their role in the group.

4.3.1 Motion detection

The major improvement in the bounding boxes basic treatment for motion detection that was mentioned in Section 4.2.1 has been achieved implementing a Differential Subtracting algorithm from Collins et al. [26]. The technique allows to erase the *ghosting phenomenon* which generates bounding boxes around elements in the image that are not humans. Moreover, combining this method with the optimization from Cèdric Verstraeten [121] using standard deviation offer the prospect of better results.

Robust motion algorithms go as far as tackling outdoors environment problems with ease. This results from the fact that motion detection for outdoor scenarios have to deal with constant light changing and excessive wind, which generates a "weather noise" problem that is part of the environment's layout and has to be discriminated. As such, motion detection under these circumstances must employ sufficient computing power –sometimes even GPU accelerated computation– for a robust motion tracking implementation. The presented data was collected indoors. Indoor scenarios are not prone to these kind of problems, hence it is not necessary to compromise CPU power on this task. A simple trick to erase false positives is assuming motion only occurs in a sequence of images larger than one.

Another interesting parameter which can be used to neglect false positives is the standard deviation. Standard deviation describes the distribution of the motion. When motion is specific at a single point, like when a human is moving far away from the camera, then motion will be mainly concentrated around a single point or pixel, hence standard deviation will be near to zero. On the other hand, when a lot of motion is detected and is distributed over the entire image, then standard deviation will be very high. A huge distribution mostly indicates no real motion, e.g. indicate aggressive wind or other abrupt changes. Notice that in some scenarios, such as public places, high distributions are very usual and this assumption fails. Instead of working with a rectangle shape as bounding

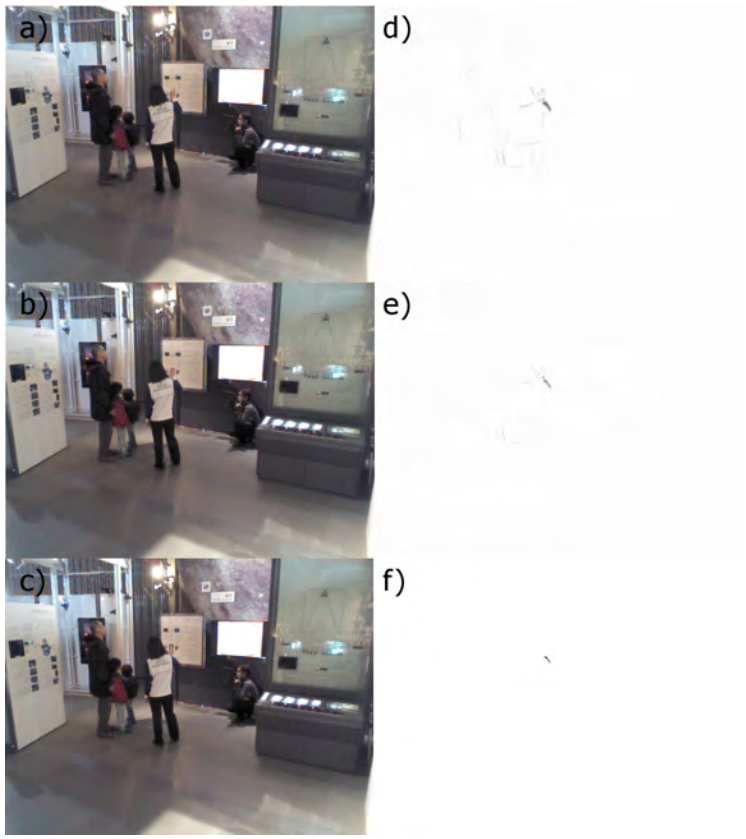


FIGURE 4.2: Motion detection example: a) previous, b) current and c) next image frames. The absolute difference between a) and c) is shown in d), whereas the absolute difference between b) and c) is depicted in e). The final result f) is the threshold obtained from an AND operation on d) and e).

box, a concave hull can be defined. Again, as it is not the case for this problem, keeping things simple is preferable.

Accordingly, processing for motion detection is based on three images from the video stream, which are called *previous*, *current* and *next* (examples in Figure 4.2a–c). The first step is performed by subtracting the *previous* and *next* images (see example in Figure 4.2d) and then the images *current* and *next* (Figure 4.2e). A bitwise AND operation is performed between both results and the final outcome is thresholded to make it accurate for larger changes only (Figure 4.2f).

The thresholded result is placed as a window on the current image looking for motion changes, that is for pixels with values equal to 255 (colors inverted in Figure 4.2 to help visualization) which will indicate motion. When motion is detected *min* and *max* values are evaluated, which are used to compute a bounding rectangle containing all changed pixels on the scene. If motion is detected, a yellow rectangle is drawn on the resulting image (Figure 4.3).

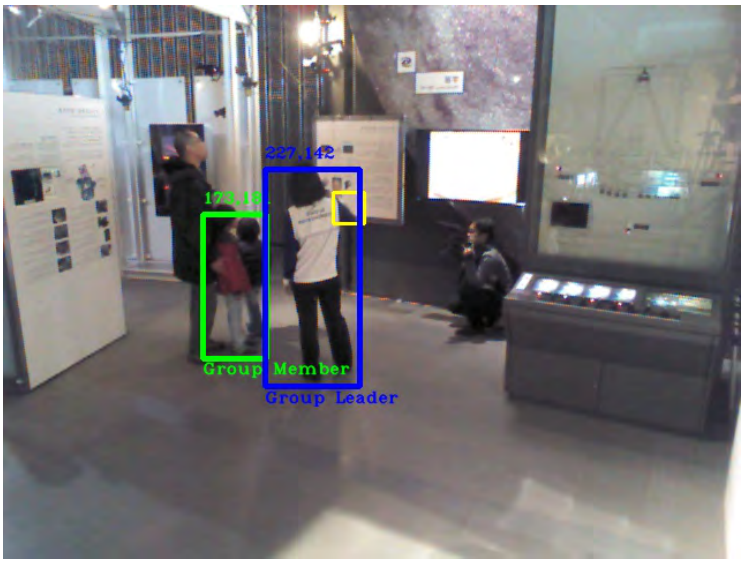


FIGURE 4.3: A yellow bounding box is computed containing all pixels with values equal to 255 in Figure 4.2f, which represent the area with the most significant motion in the given scene: the science communicator's gesturing hand.

Besides the method is hold very simple and fast, it obtains a high performance. Selection of the threshold value is a key point for accurate motion detection. Some algorithms even suggest to pick a dynamic or adaptive threshold. Nevertheless, this feature would delay the overall procedure and it does not completely avoid false positives, hence it is not considered.

4.3.2 Group categorization

Pursuing to categorize people detected on the scene as part of a group or not, a definition of 'group' must be established. *Group* stands for a set of people having one leader and his/her current followers. It will be assumed that a leader will act as such throughout the entire video sequence while followers conforming the group remain the same in number from start to finish. In other words, no new leadership will arise among the group and no new members will be allowed to join. Despite the hard constraints in this definition, they typically apply to guided tours in museums. Hence, the required focus to monitor and track a group of museum visitors has been established with such restrictions, as they highly relate to the actual environment, the specific circumstances and the case to be studied in this chapter.

Depending on the conditions they meet while in a Transitory or a Stationary state, people detected on the scene earn a permanent or a transient –modifiable, overwritable– tag which identifies and assigns the corresponding roles to each individual (see Table 4.1).

A *Transitory state* is considered when the video stream starts depicting a leader moving with the followers from one place to another (Figure 4.4), whereas a *Stationary state*

TABLE 4.1: Logic behind people detection categorization

Tag	Type	Role	Transitory State	Stationary State
1	Permanent	Group Member	Detection box overlapping with motion box or with group members boxes	Potential group member not selected as the leader
2	Permanent	Non-Group Member	Detection box not overlapping with motion box nor group members boxes	Detection boxes not tagged as potential group members
3	Permanent	Group Leader	First detection box to overlap with motion box	Potential group member which first overlaps with motion box
4	Transient	Potential Group Member	NA	Detection boxes significantly close to each other

is encountered when a video stream starts with the leader already settled on the scene surrounded by the followers (Figure 4.5).

In this sense, a leader in a Transitory state is prone to *initiate* the movement on screen, as is the one guiding the group. Whereas in a Stationary state a leader tends to gesticulate more than anyone else, sometimes excessively, which is clearly justified for science communicators in a museum. These gesturing actions also *start* displaying a notion of movement over the scene. The logic for both assumptions and their subsequent role assignment effects are reflected in Table 4.1, evaluating in detail the main interactions between the motion bounding rectangle and the people detection boxes on scene.

The leader is susceptible to be incorrectly detected in the Stationary state, since detecting motion from gesturing is more discrete, it takes a longer period of time to be categorized than in its Transitory counterpart. Hence, when no relevant movement is taking place on the first frames of the image sequence while quite a number of detections are occurring, images are experiencing a *Soft case*. In the case of being able to detect only one person on several frames of the scene, then we are going through a *Hard case*.

A Soft case is easier to overcome than a Hard one, since it is just a matter of time for the leader to make a significant gesture and activate the motion detection algorithm (Figure 4.6). A Hard case, in contrast, deals with the difficulty of working through the people detector's failure due to complex view angles or occlusions with other individuals (Figure 4.7).

Group members have a tendency to get close together and maintain a certain distance from the leader figure. In consequence, there exists a high probability that the leader is the only person not occluded and the one successfully identified by the people detector

FIGURE 4.4: Example of a video sequence undergoing a Transitory State.
Top: Frame 1 of *Training2* video. Bottom: Frame 100 of *Training2* video.

in a Hard Case. However, it is not possible to rely entirely on the previous assumption and assign roles based exclusively on this.

Hence, to properly handle any ambiguous role situation, in conjunction with the Permanent-type tags, a special Transient-type tag associated to the "Potential Group Member" role has been created. When the video stream starts and no motion is detected yet, the algorithm searches for the people detection boxes which are closer to each other and designate those detections as members of a potential group using the transient tag. Once the leader is spotted, tags become permanent and roles are automatically assigned. This cognitive development is triggered by the process of perceiving which individuals are grouping from the very beginning, monitoring their behaviour and analyzing the scene. In this form, a general solution for categorization in the Stationary state is achieved without isolating Hard and Soft cases.

Certainly, the transient tag is not affecting the Transitory state where motion detection shows up rather fast. Nevertheless, a *True Member* algorithm is internally serving as a last filter before assigning tags for the Transitory State. Descriptions in Table 4.1 refer to a set of actions which trigger role selection sequences that finally transform into role assignments. The True Member algorithm is responsible for a second level analysis on this behavioral actions to ensure – as much as possible – a definitive role assignment. True Member evaluates box overlapping with other role-assigned boxes or, failing that,

FIGURE 4.5: Example of a video sequence undergoing a Stationary State. Top: Frame 1 of *Training4* video. Bottom: Frame 100 of *Training4* video.

how close they are to these boxes in terms of pixels distance.

Up to this point, there is a clear distinction between who is a group leader and who is a group member. What about the cameraman? Is the cameraman considered a group member or not? Technically, the cameraman can be treated as a passive group member after he follows the group and keeps track of them from start to finish and at a – sometimes – quite short distance. Nevertheless, given the nature of this work, the choice has been made through the line of reasoning in which the cameraman is not considered a group member since he is not a museum visitor, strictly speaking. The cameraman and any false-positives are categorized as non-members, together with detections of people wandering on their own in the scene.

A 25 frame rate sampling is applied for detection and elimination of possible false positives, thus minimizing errors for upcoming frames. False positives have their origin in the *People Detector*'s phase, the first stage in our overall procedure, where algorithms were maintained fast, but maybe inaccurate, for real-time processing purposes. When people's bounding boxes and their category labels have been assigned, computing time is available to evaluate if all registered detections have been active or not. An *activeness detection* procedure is applied based on a set of features. These attributes are: the coordinates of the bounding boxes (x, y) the areas' difference between bounding boxes

FIGURE 4.6: *Training3* video exemplifying a Soft case. Top: Frame 1, people detector outputs three humans on scene. Bottom: Frame 35, the algorithm is able to rapidly categorize all detections after the group leader has been identified.

$(\Delta x, \Delta y)$ and the assigned categorization tag. The (x, y) coordinates provide the location of the upper-left corner of a detection bounding box. If any of these features have changed within 25 frames then the detection is considered as a true and active one; else, in the presence of a detection with no attribute changes, it is considered as a false positive since the probability that this detection has been assigned to a still object rather than a human being is extremely high.

It should be emphasized that this filtering is applied to all the detections obtained so far at the time of evaluation, free from any role assignment consideration.

4.4 Leader Tracker and Direction Estimator

When the categorization phase has been completed after a few frames in the video stream and roles have already been assigned, now is time for the algorithm to continuously track the selected group leader and occasionally compute his/her direction.

FIGURE 4.7: *Training1* video exemplifying a Hard case. Top: Frame 1, people detector outputs only one human. Middle: Frame 6, leader is categorized with no group members around despite the existence of more humans on the scene. Bottom: Frame 97, one of the group members is finally detected and it is automatically categorized by the algorithm.

4.4.1 Leader tracker

Dlib's[2] implementation of the winning algorithm from 2014's Visual Object Tracking Challenge has been selected to handle this stage. Robust scale estimation is a challenging problem in visual object tracking. Most existing methods fail to handle large scale variations in complex image sequences. The proposed approach by Danelljan et al. [33]

[2]http://dlib.net/

works by learning discriminative correlation filters based on a scale pyramid representation. They learn separate filters for translation and scale estimation, and demonstrate that this improves the performance compared to an exhaustive scale search. Scale estimation approach is generic as it can be incorporated into any tracking method with no inherent scale estimation. The method is shown to outperform the best existing tracker by 16.6% in median distance precision, while operating at real-time.

The centre of coordinates from the leader's bounding box, obtained from the first frame where the leader has been detected, is the only information fed to the algorithm, which is capable to predict the position of this bounding box throughout all the upcoming frames with high accuracy. The 25 frame rate sampling in Section 4.3.2 is also employed in the computation of an averaged optical flow to determine the group leader's direction. A matter thoroughly discussed in the next subsection, which is also a relevant part from the last stage of the algorithm and brings conclusion to the overall methodology of the project.

4.4.2 Leader's direction by constrained optical flow

Two simple optical flow algorithms are currently available in OpenCV, Tao's [116] and Lucas-Kanade methods [78] which can be combined with Shi-Tomasi [115] algorithm to select interest points, as well as two dense flow implementations from Weinzaepfel [125] and Farnebäck [45]. Even though Tao's and Weinzaepfel's approaches are state-of-the-art procedures, Farnebäck's is a long-established algorithm that should be tried out first. Tao's algorithm works well with high-resolution videos – which is not the case – leaving Weinzaepfel's as an unquestionable candidate for future upgrades.

Therefore, in order to determine and indicate in which direction the leader is facing on a given scene, OpenCV's implementation of Farnebäck's optical flow is chosen. The main idea is to constrain the optical flow to be calculated only in the bounding box area of the group leader by defining a ROI (Region Of Interest), rather than running the algorithm over the whole scene. Three options are proposed: (a) to compute the leader's direction by an average of all the directions the body is manifesting; (b) to calculate the average direction of all the pixels composing the central axis of the bounding box; (c) to obtain the average direction of the body's centroid pixel. Only averaged approaches are considered in the interest of stabilizing the flow's behavior and the way to display it, exposing a relevant improvement not only visually but in terms of performance as well.

For the time being, option (c) has been selected, as it is a simple and straightforward strategy. It makes sense to retrieve the centroid pixel direction computed by optical flow: a person's torso is a stable body part that inevitably describes the motion reality of the individual. In other words, a person turning the head into a certain direction does not necessarily imply she/he is planning to move in this direction, they could be explaining and looking at something next to them or someone could be momentarily catching the

person's attention from that location. While the action of moving the torso *does* determine the person's direction intention, as you cannot move around your torso without conducting your whole self, and states genuine focus, since your torso is always standing in front of your primary attention. Therefore, the torso is a more reliable body feature to resolve which direction a human is facing and also, a less complicated position to retrieve when confronting the challenges of inter-individual variance.

4.5 Case Study and Experimentation

The National Institute of Informatics (NII) in Tokyo, Japan, performed an experiment at The National Museum of Emerging Science and Innovation (Miraikan), placing four Kinect v1 sensors on different spots across a few rooms. This experiment consisted in gathering color, depth and skeleton information provided by the RGB-D sensors stream from February to March 2014 within several sessions. The collected scenes mostly contained visitors interacting with science communicators from the museum.

Retrieved data was meant to be replicated offline for further analysis in 3D research lines. Unfortunately, posterior data processing was taking more time than expected as essential information was missing. Image frames from the same scene have different angles and, hence, different camera pose estimations for each case scenario. Without any calibration parameters available, camera re-sectioning and 3D-2D mapping for Color-Skeleton streams is futile. A Depth-Color mapping approach was considered, yet, the collected depth frames have two major drawbacks: there is no background subtraction and no timestamp per frame. Only hand-made workarounds could be actually performed, which is a huge amount of work for results with a virtually low precision rate.

At present, computing a 3D-2D mapping to optimize results is almost unreasonable under these conditions. However, it is genuinely interesting to exploit all the 2D possibilities this valuable information is willing to offer, as many robotic platforms and systems run with basic hardware resources due to short budgets or efficiency purposes. Which, in any case, makes high-end, fast and easy-to-use technology affordable for everyone who needs it. Consequently, it has been decided that only color data in the form of pictures and videos is currently being used. Leaving aside, for the moment, the rest of data for a 3D upgrade in the near future.

4.5.1 Goal of the experiment

The first objective is to use this information for leader detection: to detect and track the leader within a group of people – for this particular case, the science communicator –and determine which direction the leader is facing.

The second objective is group tracking: to identify the leader's followers on the scene, i.e, the members composing the group guided by the science communicator. In addition

to acknowledging other people are not part of the group, classifying their roles by reasoning over their group behavior and interactions.

This understanding of the scene and the individuals comprising it are key elements for an eventual development of a robot to socially interact with the environment and actively participate on it. Role interpretation must be remarkably accurate, since the robotic system has to recognize the entire group and keep track of them to fully supervise the museum visitors, working together on this task with the science communicator.

4.5.2 Testbed

Five videos comprising more than 3000 frames in total were selected to train, in a supervised fashion, the proposed Group-Leader Tracker algorithm. Four of these videos share the same background as they were obtained from the same Kinect device, although one of them was recorded prior to fixing the device into a tripod base to gain stability. Hence, color data in this video reflects some angle differences and related illumination changes.

Six videos, different from the five used for training, containing a total of 2405 frames were submitted for testing with respect to the ground truth. Video 1 and Video 6 share the same background, yet they have slightly different view angles, as the device apparently slid a bit between sessions. The four remaining videos share a similar fate since all were recorded on the same room, however, Video 2 and Video 3 have the same angle of view, which differs from the view angle found in Video 4 and Video 5. Contrasting these background conditions with the results on Table 4.2 it is safe to say there is no relation between good outcomes and a certain background, nor the other way around.

4.5.3 Results

Ground truth and accuracy formulations

A quantitative evaluation of the proposed methodology is imperative so as to measure its actual performance. Ground truth is an objective way to obtain the real accuracy of the algorithm. To create and annotate ground truth data becomes a time consuming process in the absence of available datasets to employ. It is a rather new problem, which implies some manual labelling and scoring when it comes to this matter.

The percentage of accuracy for each test video is depicted on Table 4.2. Accuracy measurement is based on the number of correct categorizations from the total number of detection boxes, this applies to all the sample frames of every video. Moreover, we found the average processing time to be around 270 miliseconds per frame.

The first column of Table 4.2 displays the name of the test video, followed by the total number of frames composing that video and the sampling rate applied. To test a video, a sample is taken every x frames, calling this value *Sampling Frame Rate*. For this reason,

TABLE 4.2: Categorization - Accuracy test

Test Name	Total Frames	Sampling Frame Rate	Sample Frames	Detection Boxes	Leader Correct	Leader Incorrect	Member Correct	Member Incorrect	Acc. (%)
Video1	910	10	91	139	3	33	47	56	36
Video2	450	10	45	100	41	0	57	2	98
Video3	200	10	20	35	0	13	17	5	46
Video4	500	10	50	65	50	0	15	0	100
Video5	230	10	23	48	20	0	27	1	98
Video6	115	5	23	20	3	6	11	0	70

the number of *Sample Frames* is obtained as a result of dividing the number of *Total Frames* by the *Sampling Frame Rate*.

The *Detection Boxes* column states the total number of detection boxes encountered on the sample frames, whereas the next four columns determine which of those detections were correctly or incorrectly categorized by our algorithm, classifying them as leader or members detections. Accuracy is then computed with the sum of correct categorizations –*Leader Correct* and *Member Correct*– over the *Detection Boxes* value as the 100% goal.

To better understand these formulations take, for instance, the next example. Figure 4.8 presents the annotated ground truth versus the testing result of sample frame number 38 in Video 2, where four out of five detections are categorized correctly: the group leader, two group members and the cameraman classified as non-member. If any false-positives are encountered, the algorithm should categorize them as non-members until the filter is able to eliminate them. Which means the false positive in this case has been incorrectly categorized as a group member.

Hence, the accuracy test results on sample frame number 38 of Video 2 are allocated in Table 4.2 as: one correct categorization for the leader (*Leader Correct* +1), three correct categorizations for two of the group members and the cameraman (*Member Correct* + 3) and one incorrect categorization for the false positive (*Member Incorrect* + 1).

Accuracy of the categorization

Very promising numbers for the algorithm introduced in this chapter can be obtained from Table 4.2. The averaged accuracies from all the videos lead to a general average accuracy of 75%, while the actual average accuracy of the algorithm is about 71% when computed on the total of detection boxes rather than giving the same importance to videos of different length.

Videos 2, 4 and 5 reveal astonishing results as the one presented in Figure 4.9, whereas

FIGURE 4.8: Sample frame 38 of Video 2. Top: Ground truth annotation. Bottom: Testing result. A blue box represents the Group Leader, green boxes are for Group Members and red boxes suggest Non-Group Members. A yellow box describes the motion detection area, which is rather large on this scene undergoing a Transitory state.

Video 6 does not lag behind with a 70% accuracy rate. These results merit further in-depth analysis just as much as Video 1 and Video 3 demand with a rather poor performance, actively seeking feedback in order to improve. All tests intend to provide relevant information on any found issue, which is thoroughly examined in Section 4.5.4.

Leader's direction performance

The constrained optical flow approach has manifested exceptional results. For example, Figure 4.10 depicts a Transitory state from Video 5 where the science communicator is guiding two museum visitors through another exhibition in the room. Notice that the direction arrow in sample frame 5 (Top) is rather small compared to the arrow shown in sample frame 15 (Bottom).

These frames were taken from the same video sequence, both portraying the displacement of the leader and her group throughout the room, and from Figure 4.10 we can observe that the magnitude of the motion has been calculated. The length of the arrow reflects the magnitude of the averaged optical flow from that point in space, which is proportional to the magnitude of the motion on scene.

FIGURE 4.9: Video 2 with 98% of accuracy. Science communicator is displayed as the group leader in a blue bounding box, while museum visitors have been recognized in green-colored boxes as members of this group. Pixel coordinates on the upper-left corner of each box.

Science communicator in sample frame 5 has recently started moving towards the appointed direction, which is why the arrow's magnitude is smaller than the one seen in sample frame 15, where the science communicator has reached her final destination. Magnitude also reveals the speed of displacement. Hence, large direction arrows are easily encountered in Transitory State videos, whereas short arrows are common in Stationary State since science communicators do not move from their position while explaining an exhibit (see Figure 4.11). Leader's direction information can alert a robotic assistant of large or quick displacements from the science communicator. After all, museum visitors under these circumstances are more likely to split from the group accidentally, if they get distracted. A missing persons situation could be prevented monitoring this particular attention reference.

4.5.4 Found issues

This is a list of issues that can be reported from experimentation.

Distance from the camera

A certain distance from the camera is necessary for the people detector to work properly. Occlusion between individuals and their closeness to the camera view makes it difficult for the algorithm to detect all the group members on the scene, which is a recurrent issue (see Figure 4.11). In fact, it can quickly escalate into a major one if the people detector

FIGURE 4.10: Video 5 with 98% of accuracy. Top: Sample frame 5. A blue arrow with the leader's torso as origin estimates her direction. Bottom: Sample frame 15. Leader's direction arrow indicates the direction of movement and also displays its proportional magnitude.

is partially detecting members of the group despite these circumstances, just as it did on Video 1.

Failures related to the people detector

Video 1 holds the lowest performance from the tests, owning a 36% of accuracy. The cause behind it is that the first frame where the leader bounding box would collide with the motion box, is the exact same moment where the people detector does not detect the leader.

When the motion box finally makes an appearance as a result from the science communicator's gesturing, the people detector has made only one detection on scene. Unfortunately, this detection does not correspond to the science communicator but to a museum visitor instead.

The algorithm immediately performs the role assignment since the visitor's oversized detection box overlaps with the motion rectangle (see Figure 4.12). The outcome of this Hard case scenario is a wrong leader selection, an error which propagates through the entire video sequence.

FIGURE 4.11: Video 4 with 100% of accuracy. A Stationary state scenario where it is hard to detect the museum visitors being assisted by the science communicator. And yet, the cameraman is successfully classified as a non-group member in a red color box.

The effect of false-positives

False-positives are considered as non-group members within the algorithm's logic and a periodic filtering function has been set to eliminate them. However, there is still a considerable amount of object-triggered false-positives drawn by the people detector.

Losing track

An unusual problem was encountered in one of the testing videos: the leader tracker got lost. The tracker algorithm is extremely robust and normally the manifested errors come from people detector's weaknesses. Hence, we are not facing this issue because it did not occurred in any training sequence before.

Figure 4.13 shows how the algorithm was performing sufficiently well in Video 6 until the tracker looses the science communicator's location. Despite some missing tracking information, the algorithm's categorization task remains in force.

The cameraman controversy

The *cameraman case* has been a difficult one to overcome. Opposing opinions exist on the question as to whether a cameraman should be considered as a group member or not, even among human beings. So, what can it be expected from the reasoning of a machine?

If the cameraman is close enough to the group or moves fast to catch up with them, it is very likely for the cameraman to be confused with a group member. A situation also exists where the cameraman could be addressed as the group leader if he starts moving

FIGURE 4.12: Video 1 with 36% of accuracy. Top: Frame 1, science communicator has been detected, no role assigned yet. Middle: Frame 4, science communicator gestures and motion box overlaps with the only detection obtained, selecting a museum visitor as the group leader. Bottom: Frame 6, science communicator is detected after the leader role has been given, appointing him as a group member.

earlier than the leader and even faster, as he is trying to avoid a possible collision with the group.

Keeping pace with the leader

Yet, false categorizations for the leader role can occur under a certain course of events. Video 3 portrays an image sequence dealing with a Transitory State where the video starts depicting a science communicator guiding the museum visitors to the next exhibition.

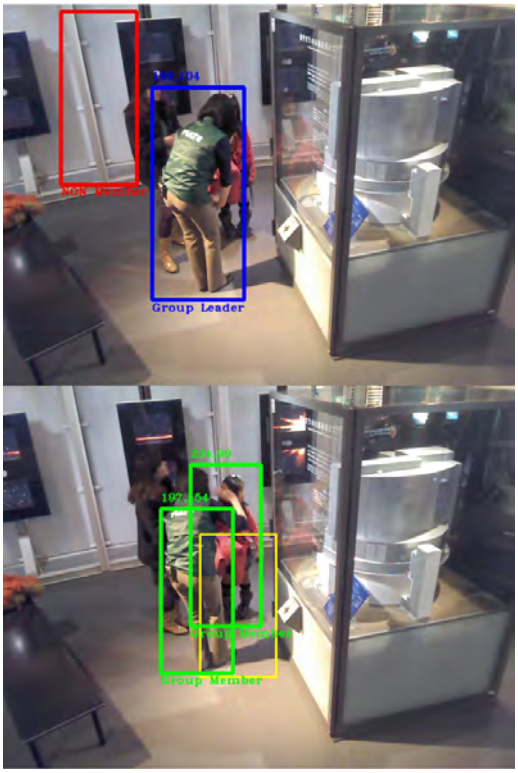


FIGURE 4.13: Video 6 with 70% of accuracy. Top: Science communicator is being tracked as the leader. False positive object detection is classified as non-group member until the algorithm is able to eliminate it. Bottom: Leader tracking is lost in some point around frame 35. Yet, the algorithm still considers the leader as part of the group.

Everyone on the scene moves at the same time and as fast as the science communicator leading their way, thus creating a large motion rectangle which overlaps with all the detection boxes (see Figure 4.14).

This behaviour unveils a critical predicament for the algorithm's logic so far. As all the detection boxes are the "first" ones to overlap with the motion rectangle, the system will choose as group leader the first box that was detected and analyzed by the people detector, which is random.

4.5.5 Discussion

As mentioned in the previous subsection, most of these errors occur and come from the people detector's performance. While it is true that some cases have a certain degree of complexity, a better people detector should be implemented to mitigate this issue as much as possible.

Errors arising from self occlusions between individuals and difficult view angles are very likely to dissipate once the system is installed on a robotic assistant. The experiments presented in this work contain image sequences from fixed cameras of the Kinect devices

FIGURE 4.14: Video 3 with 46% of accuracy. Top: All individuals are detected, no roles assigned yet. Bottom: Cameraman has been addressed as the group leader, whereas the science communicator is not even detected on this frame. A large motion rectangle is displayed containing all the detections on scene.

recording on the upper part of different rooms. In contrast, a robot would have a better point of view of the scene in general and even the flexibility to move around to get one.

With the purpose of extending the average accuracy of the system and boost the algorithm's performance to another level, an *exponential motion* algorithm should be carefully designed for further implementation.

A human cognitive knowledge about roles within an observed group of people is based on their exponential interactions and behavioral attitudes over time. Meaning that, it is sometimes difficult for human beings to immediately categorize a group and classify their roles when the group is extremely homogeneous a priori.

All members behave somehow equally and adopt the same attitude towards each other, regardless of whether there is a visually distinctive subject among them or not. As a consequence of this complexity, we need to observe their interactions for a while longer until these members seem to develop self-identities, defining their own group roles (consciously or not).

Once we are satisfied with this development then it looks only natural to deliberately

determine who is a leader and who are the followers. Arriving to that conclusion implies we had observed a series of interactions evolving, where the leader's attitude grew exponentially within a certain time period. To convey this human-like way of reasoning into a machine is the missing piece of the puzzle. Quantifying the motion interactions over time of all the detections, in an exponential fashion, will reinforce the algorithm and improve significantly its overall structure.

4.5.6 Exponential Motion Algorithm Implementation

A first version of the *exponential motion* approach has been developed and subsequently tested on all the videos. The exponential analysis is made every 50 frames, where the most targeted role for each detection becomes its definite categorization. At least until the next 50 frames, since the role computations reset to zero and start all over again, eliminating the initial hard constraints and delivering a relatively *dynamic* role attribution.

Table 4.3 refers to the accuracy test results obtained after implementing the exponential motion algorithm. In contrast with Table 4.2, the general averaged accuracy and the actual average accuracy both have dropped by 2%, from 75% to 73% and from 71% to 69%, respectively.

Following this comparison but being more specific, Video 1 and Video 6 have shown fair results as some errors have effectively diminished (see Figures 4.15 and 4.16). However, accuracy in Video 1 increased by 12% with the exponential motion implementation, while accuracy on Video 6 decreased 7%.

On the other hand, Video 3 increased its accuracy by 12% as well, due to an improvement in correct leader detections (see Table 4.3). However, results for members categorizations on Video 3 are poor (Figure 4.17). This problem also affects the rest of videos –the ones depicting the best accuracies on the previous set-up– just as much.

Notice that the accuracy on Video 2, Video 4 and Video 5 fell by 9% with respect to the original algorithm. The cause behind these fluctuations can be observed in Table 4.3,

TABLE 4.3: Categorization with exponential motion - Accuracy test

Test Name	Total Frames	Sampling Frame Rate	Sample Frames	Detection Boxes	Leader Correct	Leader Incorrect	Member Correct	Member Incorrect	Acc. (%)
Video1	910	10	91	178	36	46	51	45	48
Video2	450	10	45	100	39	0	50	11	89
Video3	200	10	20	36	8	3	13	12	58
Video4	500	10	50	55	45	0	5	5	91
Video5	230	10	23	57	16	1	35	5	89
Video6	115	5	23	24	9	0	6	9	63

FIGURE 4.15: Video 1 results under exponential motion implementation. Top: Frame 51, one group member is correctly categorized and the other is wrongly addressed as the group leader, whereas the true leader is not even detected. Bottom: Frame 419, after several frames of analysis all the detections are correctly categorized on scene.

a quite visible and general trade-off between leader and members categorization. Improvement in the correct identification of the leader has affected the membership in a negative way.

It seems that the dynamic role assignment interferes with the proxemics of the original algorithm, an issue to be considered on a second version of the *exponential motion* implementation.

On a side note, people detection issues are still backing down the algorithm's performance, despite the efforts of creating an array with a historical record of detections as an attempt to maintain track and control of all the detections boxes and avoid further duplication or sub-detection (identifying the same detection in a smaller bounding box).

4.6 Conclusions

In this chapter, a new problem has been addressed within the group tracking research area. Detecting the leader of a group and categorize its members attracts a great deal of interest in the study of group interaction and social environments. Role assignment

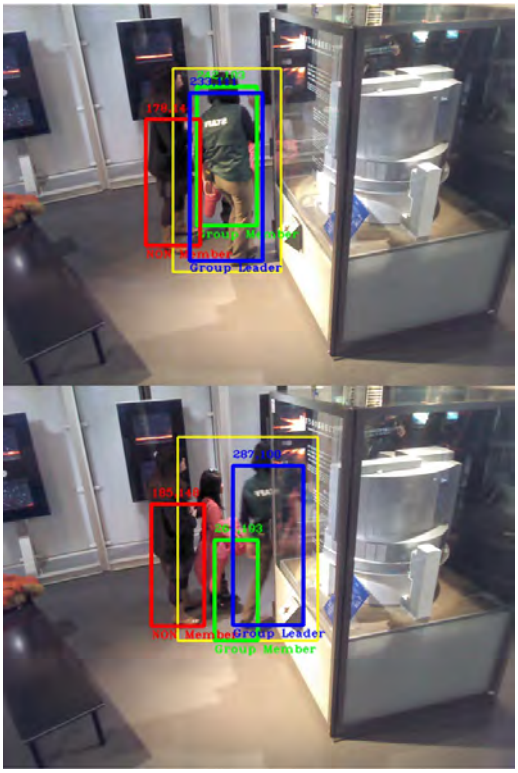


FIGURE 4.16: Video 6 results under exponential motion implementation. Top: Frame 51, only one false categorization made on a group member and the leader tracking is working properly. Bottom: Frame 70, incorrect categorization on the same group member remains, yet the tracker keeps following the leader almost exiting the scene.

and behavioural analysis by means of a cognitive approach based on motion logic and proxemics theory is a novel methodology which is apparently naive but its simplicity has proven to be quite successful.

The prominent results of this contribution are considered to be relevant for a wide scope of possible applications, specially for robotic assistants in similar environments. In order to achieve the sufficient accuracy for the algorithm and later impact on more complex environments, a number of improvements are required, besides the ones mentioned in Section 4.5.5 and Section 4.5.6, after exhaustive analysis of experimental results.

Hence, depth data related to the color images obtained from Kinect's RGB-D sensor should be used to create a more refined detection of the subjects, despite the challenge of restoring faulty information. Another improvement refers to the location of people's bounding boxes, which could be used as a region of interest over the corresponding depth arrays, which combined with thresholded distances, background subtraction algorithm or a deep learning segmentation method, could provide better results.

It is important to fully exploit the collected 3D data on the field for the sake of future experiments in this research line. However, it is also natural to explore immediate

FIGURE 4.17: Video 3 results from exponential motion implementation. Top: Frame 51, even though the four people on the scene are moving as much as the group leader does, the leader is correctly categorized along with the cameraman and one of the two group members. Bottom: Frame 102, two out of three detections are wrongly categorized. Purple boxes reference to previous detections.

upgrades.

Kinect v2 uses time-of flight by which the sensor can see just as well in a completely dark room as in a well lit room. The first Kinect also accomplishes this feature using structured light to reconstruct the depth data with approximations for pixels between the projected points. Still, Kinect v2 has far superior performance since each pixel now has a unique depth value. This method is more stable, precise and less prone to interferences.

This RGB-D state-of-the art technology is certainly more powerful and complex than the one embedded in the first generation of Kinect. A significant improvement that comes along in real-time, processing 2 gigabytes of data per second with a faster broadband for data transfer. Therewith, further on-line implementation of background subtraction and coordinate mapping would result in a people detector with higher accuracy and virtually zero false-positives.

Chapter 5

3D Automatic Landmarking with a Conditional Contour Detector Kernel

Contours can be explained simply as a curve joining all the continuous points along the boundary of an object. Unlike edges, which refer to any abrupt change in local image features, such as luminance and color, contour represents changes from object to background or one surface to another, which need deeper levels of image information to detect.

Conversely, segmentation is the process of clustering image elements that *belong together*. It is achieved by partitioning or dividing the image into regions with coherent internal properties and grouping them to identify sets of coherent tokens in that image.

When the target object in an image is a human being and we are only interested in segmenting a person from the background to find the contours of their body, we reduce the problem of image segmentation to that of contour detection.

Contour detection is a crucial task in computer vision, not only because of the obvious aspect of detecting contours of subjects contained in an image or video frame, but because of the derivative operations connected with identifying contours.

These operations are –namely– computing bounding polygons, approximating shapes, and generally calculating regions of interest, which considerably simplify interaction with image data and turns to be an effective tool for shape analysis, object detection and recognition.

We consider that human body segmentation and contouring on RGB-D images could be seen as a 3D automatic landmarking process were, instead of learning a shape model by manually hand-marking 2D training images which are fed to a predictor –as seen on Chapter 3–, the shape is directly obtained from the mapping of depth and RGB values of a person detected on the scene by a sensor.

The introduction of RGB-D information in applications where the accuracy in people detection tasks is critical, becomes a suitable solution to suppress false-positives and diminish the effects of occlusions and complex view-angles, an issue thoroughly discussed in the previous chapter.

Hence, in this chapter, a novel methodology for body contour tracking is proposed with the design and implementation of a *Conditional Contour Detector Kernel*. A cross-type

kernel that scans every output image frame provided by a background subtraction algorithm using a RGB-D sensor in real-time. This type of on-line 3D body contour tracking could be of great importance in several computer vision applications coping with limited processing resources, which still need to obtain human inquiries without sacrificing accuracy.

5.1 Introduction

Early works in human body segmentation were based on implicit surface models of the body in order to predict its location throughout a given video sequence [99] or by learning a set of mean posture clusters and their local shape distributions for humans in various postures [109]. The application of boundary edge selection methods [110] and active contour tracking [4] achieved more refined results, modelling shape in terms of the contour points. Freifeld et al. [47] learned a Contour Person Model which instead models deformations of 2D contours to explicitly represent articulations, factoring different types of poses.

Latest proposals include body tracking based on region intensities and the motion vector of its interest points [2], as well as iterative human contour segmentation using oriented graphs [83]. However, the state-of-the-art approach takes it to the next level by adaptively combining color and depth cues of RGB-D images to set the evolution of the active contour model of a human body [129].

In image processing and computer vision, a kernel is a small matrix used to apply effects such as blurring, sharpening, outlining, embossing and more. They are also used in machine learning for feature extraction, a technique for determining the most important areas of an image. In this context, the process to accomplish those results is generally referred as *convolution*.

For the approach presented here, we have designed a kernel that does not fall within the traditional procedure of convolution, since it properly adapts to the data delivered by state-of-the-art RGB-D technology. Rather than adding each element of the image to its local neighbors, weighted by the kernel in a mathematical fashion, our kernel instead asks if the pixel and its neighbors meet a certain condition, which is directly related to the nature of the RGB-D image information provided by the sensor: *Does this pixel belong to a user or to the background?*

5.2 3D Human Body Contour Tracking with a RGB-D sensor

In order to improve a possible description on human shape, we need to retrieve more precise and accurate information from the real world. Hence, our method incorporates

the latest RGB-D technology, Kinect for Windows v2 sensor (a.k.a. Kinect v2) to fulfill this task in real-time.

We firstly introduce an on-line background subtraction algorithm performed with Kinect and coded in C# [102]. This algorithm acts as the baseline to obtain the image output which is going to be analyzed by the Conditional Contour Detector Kernel –our contribution in this work– and finally transformed into 3D body contour tracking.

5.2.1 Background Subtraction

Kinect v2 effectively combines depth and RGB data. Segmentation based on depth information from Kinect v2 has lighting invariance at medium-low ambient light levels, as well as surface texture invariance. Hence, Kinect's depth frames are essential to perform background subtraction tasks in our experiments, since depth points are mapped to color points which are later selected through their *human body* membership.

Based on a very exhaustive evaluation from Yang et al. [130] and Woolford [126] where the high accuracy of the depth frame data is acknowledged via extensive quantitative analyses (accuracy distribution, depth resolution, depth entropy, among other measures), we have considered that qualitative results supporting those studies should be delivered through our work in this chapter.

Moreover, experimentation in [130] encountered an average depth accuracy error of <2mm within the distance range we are acting in our own experiments (0.5 to 3m), which is considered negligible. RGB-D technology does not manifest any significant displacement errors to be reported like in Chapter 3. Therefore, performing a Jaccard index or Dice coefficient evaluation on this particular task is not imperative, since work from Moreira et al. [88] using a Kinect v1 sensor already proves both coefficients are close to 1, while it is well-known Kinect's v2 performance is greater than its predecessor.

As previously mentioned, background must be removed in order to isolate human-user pixels. Background subtraction will be implemented using the `BackgroundRemovalTool` from *Vitruvius*[1]. The key is to map the RGB color values which correspond to the person's depth distances and viceversa. This procedure is called *Coordinate Mapping* [103]. `CoordinateMapper` is a useful property of the `KinectSensor` class, so it is tighted to each Kinect sensor instance.

RGB frames (1920×1080) are wider than the depth frames (512×424). As a result, not every color pixel has a corresponding depth mapping. However, body tracking is performed primarily using the depth sensor of Kinect, so there is no need to worry about missing values since the body index frames have a 512×424 size. Body index frames contain information on the pixels belonging to a human-user in the depth space (see Listing 5.1).

```
1 // Color frame (1920x1080)
```

[1] https://vitruviuskinect.com/

```
int colorWidth = colorFrame.FrameDescription.Width;
int colorHeight = colorFrame.FrameDescription.Height;

// Depth frame (512x424)
int depthWidth = depthFrame.FrameDescription.Width;
int depthHeight = depthFrame.FrameDescription.Height;

// Body index frame (512x424)
int bodyIndexWidth = bodyIndexFrame.FrameDescription.Width;
int bodyIndexHeight = bodyIndexFrame.FrameDescription.Height;
```

LISTING 5.1: Obtaining dimensions from all data streams.

First, we need to initialize the arrays (Listing 5.2). Initialization happens only once, so to avoid allocating memory every time we have a new frame.

```
1    _depthData = new ushort[depthWidth * depthHeight];
2    _bodyData = new byte[depthWidth * depthHeight];
3    _colorData = new byte[colorWidth * colorHeight * BYTES_PER_PIXEL];
4    _displayPixels = new byte[depthWidth * depthHeight * BYTES_PER_PIXEL];
5    _colorPoints = new ColorSpacePoint[depthWidth * depthHeight];
6    _bitmap = new WriteableBitmap(depthWidth, depthHeight, DPI, DPI, FORMAT,
     null);
```

LISTING 5.2: Initializing arrays of data.

From lines 1 to 6, the contents of these arrays are:

`ushort[] _depthData`: The depth values of a depth frame

`byte[] _bodyData`: The information about the bodies standing in front of the sensor

`byte[] _colorData`: The RGB values of a color frame

`byte[] _displayPixels`: The RGB values of the mapped frame

`ColorSpacePoint[] _colorPoints`: The color points we need to map

And `_bitmap` as source for a `WriteableBitmap` class in order to be a perpetually updated image that will display in real-time the RGB values obtained from the coordinate mapping. Then, the output is going to be a continuous image stream of a person cropped from the background. Once the arrays are initialized, we populate them with new frame data (see Listing 5.3).

```
1  void Reader_MultiSourceFrameArrived(object sender,
       MultiSourceFrameArrivedEventArgs e)
2  {
3      var reference = e.FrameReference.AcquireFrame();
4
5      using (var colorFrame = reference.ColorFrameReference.AcquireFrame())
6      using (var depthFrame = reference.DepthFrameReference.AcquireFrame())
```

```
7       using (var bodyIndexFrame = reference.BodyIndexFrameReference.AcquireFrame
        ())
8       {
9           if (colorFrame != null && depthFrame != null && bodyIndexFrame != null)
10          {
11              camera.Source = _backgroundRemovalTool.GreenScreen(colorFrame,
        depthFrame, bodyIndexFrame);
12          }
13      }
14  }
```

LISTING 5.3: Reading data streams.

The code for the function described in Listing 5.3, first sets the current frame as the reference (line 3); next, it reads the color, depth and body index streams from the reference (lines 5, 6 and 7); and finally, the program calls the `_backgroundRemovalTool` class to feed the image source to be displayed with our resulting image by providing the data of these streams (line 11).

Notice we opted to create a `MultiSourceFrameReader` since this allow us to read frames from multiple data sources relatively synchronously without having to open multiple readers. Keep in mind that there is no Data Source called `MultiFrameSource`. It is mostly an abstraction of convenience, and we could have called the readers for each data source individually instead.

Now, within the `_backgroundRemovalTool` class we call the coordinate mapper, which associates depth values to their corresponding color points. `CoordinateMapper` is passed as a parameter from the connected Kinect sensor.

```
1       _coordinateMapper.MapDepthFrameToColorSpace(_depthData, _colorPoints);
```

After the mapping has been done it must be specified which pixels belong to human bodies and add them to the `_displayPixels` array. Hence, we loop through the depth values and update this array accordingly.

```
1   for (int y = 0; y < depthHeight; ++y)
2   {
3       for (int x = 0; x < depthWidth; ++x)
4       {
5           int depthIndex = (y * depthWidth) + x;
6           byte player = _bodyData[depthIndex];
7
8           if (player != 0xff) // Check whether this pixel belongs to a human
9           {
10              // Refer to cartesian coordinates on the color camera image
11              ColorSpacePoint colorPoint = _colorPoints[depthIndex];
12              int colorX = (int)Math.Floor(colorPoint.X + 0.5);
13              int colorY = (int)Math.Floor(colorPoint.Y + 0.5);
14
```

```
15              if ((colorX >= 0) && (colorX < colorWidth) && (colorY >= 0) && (
    colorY < colorHeight))
16              {
17                  int colorIndex = ((colorY * colorWidth) + colorX) *
    BYTES_PER_PIXEL;
18                  int displayIndex = depthIndex * BYTES_PER_PIXEL;
19
20                  _displayPixels[displayIndex + 0] = _colorData[colorIndex];
21                  _displayPixels[displayIndex + 1] = _colorData[colorIndex + 1];
22                  _displayPixels[displayIndex + 2] = _colorData[colorIndex + 2];
23                  _displayPixels[displayIndex + 3] = 0xff;
24              }
25          }
26      }
27  }
```

LISTING 5.4: Identifying human body pixels.

The result is a bitmap with transparent pixels for a background and colored pixels for the human bodies (Figure 5.1). Kinect sensor retrieves all the appearance information available for the detected human body. No matter how complex the pose (Figure 5.1c), the subject is instantly and continuously tracked, even when partially sided (Figure 5.1d).

As long as the Kinect sensor is live and running, the `WriteableBitmap` array keeps updated frame after frame (Listing 5.5), constantly fed by the `_displayPixels` array.

```
1 _bitmap.Lock();
2 Marshal.Copy(_displayPixels, 0, _bitmap.BackBuffer, _displayPixels.Length);
3 _bitmap.AddDirtyRect(new Int32Rect(0, 0, depthWidth, depthHeight));
4 _bitmap.Unlock();
```

LISTING 5.5: Bitmap handling.

For greater control over updates, and for multi-threaded access to the back buffer of the `WriteableBitmap`, the following workflow is used in the piece of code of Listing 5.5:

1. Call the `Lock` method to reserve the back buffer for updates (line 1).
2. Obtain a pointer to the back buffer by accessing the `BackBuffer` property (line 2).
3. Write changes to the back buffer. Other threads may write changes to the back buffer when the `WriteableBitmap` is locked (line 2).
4. Call the `AddDirtyRect` method to indicate areas that have changed (line 3).
5. Call the `Unlock` method to release the back buffer and allow presentation to the screen (line 4).

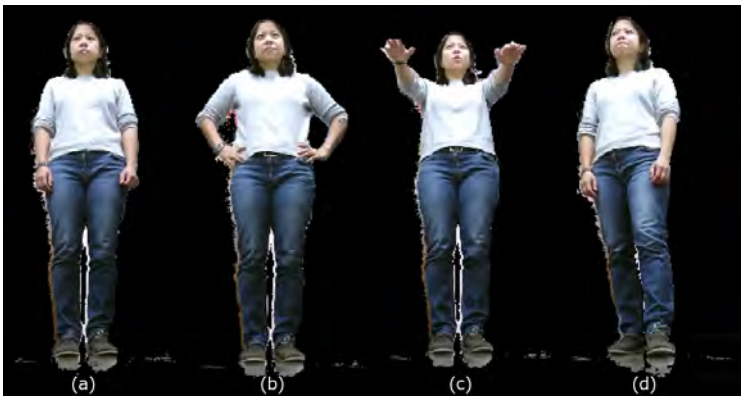


FIGURE 5.1: Background removal.

5.2.2 Conditional Contour Detector Kernel

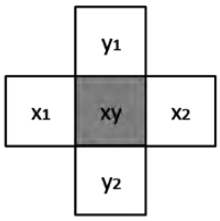
A shape contour extraction should be performed on the displayed color pixels of the human body subtracted from the background and currently tracked by Kinect sensor in order to obtain a 3D silhouette.

A special kernel has been designed and created to scan the output image from the background subtraction algorithm (Figure 5.2). We chose a cross-type kernel for efficiency. A cross pattern sweeps the frames faster than a square pattern would do, as more evaluations are executed. Certainly, more refined shapes would be acquired with a square design, but the trade-off accomplished with the *cross* is sufficient and very practical. When certain conditions are met, this kernel can determine whether the current pixel belongs to the subject's shape contour or not.

Certainly, at first sight, the morphological operation called *remove* inside `bwmorph` function from MATLAB's Image Processing Toolbox[2], seems to perform the same action and one might think it will provide the same result than our Conditional Contour Detector (CCD) kernel. However, `bwmorph` is actually the only function in the Toolbox that does not support 3D images, hence, we cannot expect this function to deliver any sense of depth like our CCD will prove to show in the results of Section 5.2.3.

It became apparent from the code in Listing 5.4 that values retrieved in the body index frames are binary, where $0 \times FF$ stands for *no player* and 0×00 for *player exists* (see Line 10). In fact, each pixel from the `BodyIndexFrameSource` is represented by an 8-bit unsigned integer. Each pixel within the "body area" of the array will have a value from 0 to 5 –since Kinect v2 can track up to 6 bodies simultaneously– that corresponds to the index of the `_bodyData` array provided by `bodyIndexFrame`. The rest of pixels –not part of the body area– will have a $0 \times FF$ value. A pixel with the `depthIndex` value of 0×02 represents a depth pixel occupied by the player found in `_bodyData[2]`. Hence, since we are performing the experiments on one person at a time, for our particular case the body

[2]https://www.mathworks.com/help/images/ref/bwmorph.html



if $[xy] != 0xFF$ ***and*** $[x1|y1|x2|y2] == 0xFF$
then $[xy]$ *is a* ***BODY CONTOUR PIXEL***

FIGURE 5.2: Conditional Contour Detector Kernel.

index value is always 0×00 for the user, whereas the background pixels are taken as $0 \times FF$ values.

We refer to the *human user* as a "player" since Kinect software acknowledges this technical term. Ergo, background values are 1's and people values are 0's in every processed frame. In this context, the CCD kernel evaluates each pixel and its neighbours like depicted in Figure 5.2, where the center of the kernel *xy* is the currently evaluated pixel which belongs to a `depthIndex` value in the currently analyzed frame.

In order to implement our CCD, the piece of code from Listing 5.4 changes accordingly. We extract only the pixels related to the perimeter from the body currently detected by Kinect sensor (see Listing 5.6).

```
1  for (int y = 1; y < depthHeight-1; ++y)
2  {
3      for (int x = 1; x < depthWidth-1; ++x)
4      {
5          int depthIndex = (y * depthWidth) + x;
6          int depthX1 = (y * depthWidth) + (x - 1);
7          int depthX2 = (y * depthWidth) + (x + 1);
8          int depthY1 = ((y - 1) * depthWidth) + x;
9          int depthY2 = ((y + 1) * depthWidth) + x;
10
11         byte player = _bodyData[depthIndex];
12         byte neighbourX1 = _bodyData[depthX1];
13         byte neighbourX2 = _bodyData[depthX2];
14         byte neighbourY1 = _bodyData[depthY1];
15         byte neighbourY2 = _bodyData[depthY2];
16
17         if ((player != 0xff) & (neighbourX1 == 0xff | neighbourX2 == 0xff |
       neighbourY1 == 0xff | neighbourY2 == 0xff))
18         {
19             // Lines 11 to 23 of the original code remain the same
20         }
21     }
22 }
```

LISTING 5.6: Code for the CCD kernel.

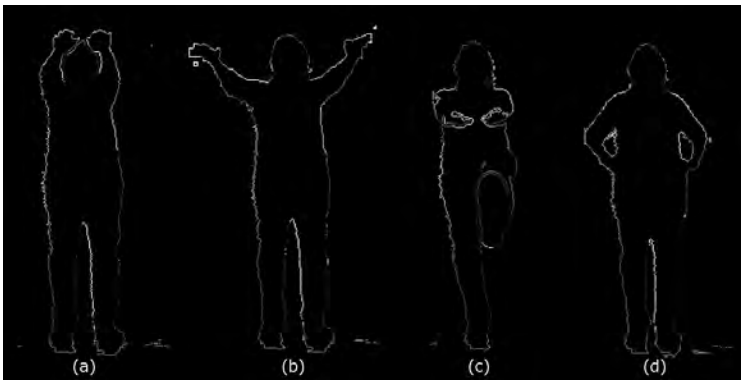


FIGURE 5.3: Real-time 3D body contour tracking.

The *for* loops in lines 1 and 3 are modified to fit the cross pattern of the kernel and reduce the area of search. Line 5 represents the depth index, whereas line 11 contains the depth value from the analyzed xy pixel. Below them are the same instances for the neighbouring pixels –respectively– matching with the variables depicted in Figure 5.2.

On the other hand, an AND followed by four OR operators are added to the *if* statement in line 17. They build a condition in which: if the currently analyzed xy pixel belongs to the body of a player or human user, and if any of the neighbouring pixels is a background pixel, then we consider the xy pixel as a *body shape pixel*.

The code from lines 11 through 23 in the original code for background subtraction in Listing 5.4 remains unchanged, since this is the part responsible for constructing the array of pixels we want to be displayed in real-time.

5.2.3 Results

Outstanding and highly accurate results have been obtained so far and are displayed in Figure 5.3. Average processing time is of about 19 miliseconds per frame. Depth differences between limbs are not an issue at all (Figures 5.3a and 5.3b). Notice that even with occlusions, the continuity of the shape contour is not compromised (Figure 5.3c). It only shows less detail in small contouring areas, like hair or fingers, depending on the closeness of the human-user to the sensor device. However, this phenomenon appears to be more pronounced in fingers contouring.

CCD kernel is able to convey a notion of depth by achieving a visual display of it, as seen in Figure 5.3c, where the user has the tip of the foot and both hands stretched towards the z direction, so he/she can be properly acknowledged in the resulting image. If we were processing the same case with MATLAB's `bwmorph` those features would disappear, omitting visually valuable pose information from the user.

Approximated details about a person's hair type and complexion can be deduced with these simple contour images, features which could be sufficient for computer vision

tasks in e-fashion and health. The accuracy achieved with full body contours it is also very acceptable for robotic systems and applications with people recognition properties.

5.3 Conclusions

A practical on-line approach for 3D contour tracking of human bodies was introduced in this chapter. By developing a kernel to leverage the features of the Kinect sensor, our proposal shows a good performance and encouraging results.

RGB-D state-of-the-art technology has plenty of advantages to be exploited in real-time applications with great precision. The structure of the output parameters from a RGB-D sensor and the complete information they retrieve enables experimentation for more specific applications like the one portrayed here.

It might seem that model-based approaches for people recognition where body texture and human shape must be processed separately, e.g. 3D AAMs construction, would make an interesting target for this work to be applied, since shape and texture are often modelled and represented as a mean, requiring a lot of training examples. Unfortunately, we did not succeed in this mission.

All the 3D AAMs related literature [24, 58, 122, 38] have a face target and a set of landmarks with slight variations from one work to another in order to obtain the mean shape model. Nevertheless, these conventional approaches could be difficult or even unfeasible to apply on more complex and larger targets –like human bodies– since the landmarks should be *anatomical*, and thus, have stable locations and characteristics that do not vary significantly in the presence of abnormalities.

The method exposed here does not have anatomical landmarks, as we directly segment and provide the body shape without using any of those keypoints. In fact, we have regrettably come to the conclusion that contour landmarks are not suitable for the study of human bodies under model-based approaches. A face –which is a very popular target for AAMs and others– contains features with fixed positions (eyes and nose) or a low degree of deformations (mouth and eyebrows), in contrast, a body contains limbs with high variability in position and deformation, namely the arms and legs of a person. Correspondence issues are unavoidable in the absence of a constant alignment reference.

In spite of this, our 3D body contour tracking approach could be as useful and –in some cases– even more practical than employing full RGB-D data to work with human appearance and skeletal joints, since it delivers more defined spatial information than skeletal tracking [131] and, at the same time, is less computationally expensive than processing skin meshes [75, 53]. Surveillance systems would give more value to the optimum use of the available resources by providing less data-processing, high accuracy and real-time performance to fulfill their tasks, just enough to meet their standard requirements and to serve their purposes effectively.

Chapter 6

3D Partial Scans Models for People Recognition with a RGB-D sensor

Camera network-based people tracking systems have recently attracted much attention in the computer vision community. Application areas range from surveillance up to ambient intelligence (AmI) and human-robot interaction (HRI) tasks. And yet, one of the essential problems for people tracking is person re-identification.

In this context, person re-identification consists in recognizing an individual across different –possibly– non-overlapping views of a camera network [100]. However, in this dissertation we focus on a concept of re-identification so as to recognize and keep track of people who left the camera view *regardless* of the number of cameras. That is, without having to rely on the interrelation between the different camera views to obtain user information.

According to an exhaustive survey of approaches and trends in person re-identification [13], those approaches can be broadly classified into Contextual and Non-contextual methods. Contextual methods use external information such as cameras calibration or geometry to aid the re-identification task. Non-contextual methods can be sub-divided into active and passive. Passive methods do not use any machine learning for descriptor extraction and matching whereas active methods do. Active methods can comprise color calibration between a camera pair, descriptors learning and metrics learning.

However, with the introduction of RGB-D technology those methods have been upgraded or studied from newer and different perspectives such as deep learning [107], on-line metric model update [74], thermal features [86] and depth + skeletal data [20], obtaining very promising results.

With the aid of the background subtraction API introduced in the previous chapter, our intention is to perform people recognition and re-identification not based on a camera network, instead we seek to employ only *one* operative RGB-D sensor as a service or an embedded technology within the environment of the human users.

As such, the system we have introduced in Chapter 4 could be implemented on a robotic platform serving as a social assistant with only one device mounted on its structure, but it also has the potential to become part of an ambient intelligence framework;

whilst the approach portrayed here is closer to a smart environment setup, enhancing human-robot interaction by exploiting the capabilities of natural user interfaces (NUIs).

People detection and tracking is a necessary skill that a robot possesses to achieve a natural and intuitive human–robot interaction. Understanding how people move through the scene and recognizing them is a key issue for decision-making in a social robot. Therefore, accurately perform these tasks would help improve interaction effectively and efficiently. Our contribution is the design and implementation of a human re-identification system for small social environments where the human-robot interactions belong to a closed group of individuals –family, work-group, classroom– that can be adaptable to similar HRI applications by using a unique integration of new ideas and established techniques.

The capability of identifying people over days, known as "long-term re-identification", is required in long-term service scenarios. In this chapter we are not taking the *long-term* factor into account, since there are several works already in existence that handle the problem rather well with approaches such as biometrical or anthropometric measures [96, 11], face recognition [61] and robust depth-base recognition [127] with RGB-D sensors, namely, Kinects.

Instead, we propose a new approach –using a Kinect v2 RGB-D sensor– that we have called *3D Partial Scans Models* (3D PSMs), an approach completely different from the previous ones in terms of reduced hardware investment (no RGB-D camera networks) and less computationally expensive methods (no point clouds nor skin meshes), in order to streamline the re-identification problem as much as possible with simpler procedures and minimum resources.

6.1 Introduction

Kinect device gives eyes, ears and brain to a computer by simple gesturing and speaking and, with that, it has brought a new era of NUIs. Kinect collects data from its various sensors and sends it to either a local or a remote computer, serving as an input device for a digital reality. A RGB-D sensor that can be adapted to many different scenarios and can be prototyped against before a more integrated and custom solution is devised. Kinect was initially designed for home entertainment and gaming; however, being light-weight, reliable and processing a high speed of measurement, Kinect has found use in indoor robotics, 3D scene reconstruction and object detection.

In this work, we employ *one* Kinect for Windows v2 and a laptop with Intel Core i5-4210M and 8GB RAM memory as our hardware setup. The reader will notice that we are further examining some technical aspects of Kinect that we did not revised before. Complete knowledge on the operational structure, the mathematical references, the nature of

the data inputs and their workflow are a vital requirement in this chapter to fully understand the solutions proposed under those specifications, but also to address more easily the challenges we have encountered along this journey. Since this Kinect is actually the same RGB-D sensor we used for experiments in Chapter 5, one can refer to subsections 6.2.1 and 6.2.2 from the current chapter in order to gain a deeper knowledge on certain functions behind that project, albeit not mandatory.

Our novel proposal consists in a partial scanning strategy of 8-views triggered by the following poses: Front, Front-Side Right, Front-Side Left, Side Right, Side Left, Back, Back-Side Right and Back-Side Left. Hence, with the captured frames from a single Kinect v2 sensor we virtually construct the body of a human user with only 8 images, reducing the workload of whole-body 3D scanning dramatically. We also take advantage of knowing at all times the position and orientation of the body joints collected from the current individual in the scene, by accessing the information related *only* to the corresponding pose. A method that provides a faster user recognition on the database. To recognize a person, we compare the appearance similarity from the current user –using a score computed with KAZE features and K-Nearest Neighbors– against the existent users in our database. If this score hits and surpasses a certain threshold, then the individual is re-identified; otherwise, the person is registered as a new user.

The system works in real-time, automatically constructing the users database on-line as time passes and people come across the field of view of the RGB-D sensor: appearing, disappearing and re-appearing in the scene, since this behaviour simulates a *natural interaction* supported by both the users and the system itself. From the users –on the one hand– when they encounter a NUI which does not interfere with the naturalness of their actions, i.e. casually introducing themselves or chatting with another individual, or just passing by, as this is how would a person normally behave around others within a small crowd. The system –on the other– has artificially embedded the cognitive and intuitive way in which we as human beings recognize and re-identify another person, a reasoning that should be conveyed into any NUI to establish a better "human-machine" interaction.

6.2 Understanding Kinect's Software Interface

The Kinect for Windows v2 SDK has a robust API that makes extracting any data of interest a cinch. All Kinect sensors have similar design patterns and developer experiences. A pattern that generally involves starting the Kinect, delineating which data types we want, and then performing our data manipulation and analysis on frames from a stream of a specific data type. At the forefront of all of this are Data Sources, an API construct that allows us to access data of a specific type.

6.2.1 Basic Flow of Programming

Kinect's data can be conveniently accessed through *Data Sources* defined by the API. Data Sources are actually similar to what in Kinect for Windows v1 were known as *Streams*. The name's change was the result of a stronger adherence to the Windows Design Guidelines at the time of the API's development, which required a different definition for the concept of "stream".

What is more, the properties of the Stream and Data Source modules are different, altering the way information is retrieved for further processing (see Figure 6.1).

Each Data has an independent Source in Kinect v2 (ColorSource, DepthSource, InfraredSource, BodyIndexSource, BodySource, etc.) and they do not depend on any other Source but their own, which supports simultaneous access to information. This accessibility feature was not possible with Kinect v1 and, as a result, led to severe problems in data synchronization. We have already faced this problem in Chapter 4, where the data for experimentation was not only poorly collected but also presented with timestamp discrepancies within Color, Depth and Skeleton frames. Each Data Source now focuses on one type of data provided by Kinect v2 and provides metadata about itself (e.g., whether this Data Source is being actively used by the Kinect sensor) and enables us to access its data through *readers*. Readers are devoted to read from the Data Sources and trigger an event each time there is a data frame available for use.

A frame contains all our valuable data along with the corresponding metadata such as dimensions or formatting. Frames are obtained from `FrameReferences`, which are in turn obtained from readers. `FrameReferences` exist mainly to help us keep our frames in sync.

6.2.2 Coordinate Systems

One might be wondering about how the depth camera and color camera have different images that by default do not align. Quite simply: if you look at the front of a Kinect, the cameras are not in the same spot (see Figure 6.2), not to mention have different resolutions.

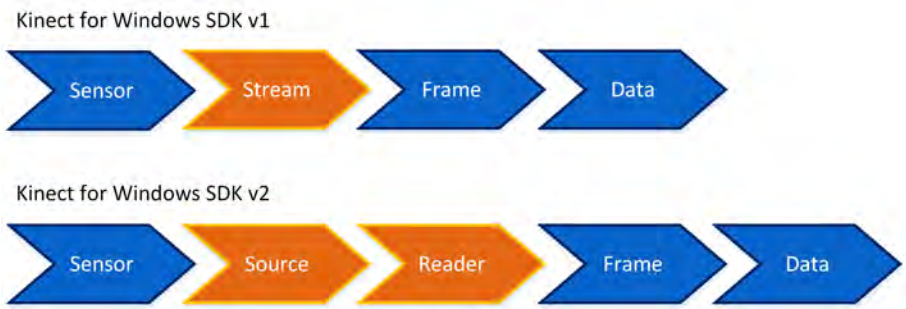


FIGURE 6.1: Kinect v1 versus Kinect v2 flow of programming.

FIGURE 6.2: Kinect v2 sensor components.

Kinect v2 can acquire RGB camera images with a 1920 × 1080 resolution and running at a rate of 30 fps or 15 fps depending on the lightning conditions of the room. Depth images can be acquired with a 512 × 424 resolution. The measurable distance range is from 500 mm to 8000 mm, but the range to recognize human beings is from 500 mm to 4500 mm.

Since the position and resolution of each sensor is different, the data is obtained as a value expressed in the coordinate system of each sensor. When using data obtained from different sensors at the same time, it is necessary to convert the coordinates to match. Thus, Kinect v2 has 3 coordinate systems: *ColorSpace*, *DepthSpace* and *CameraSpace*, and as a consequence, there are 3 data types in the SDK: `ColorSpacePoint`, `DepthSpacePoint` and `CameraSpacePoint` representing coordinates in each coordinate system.

For the RGB image, depth image, and skeleton information, the coordinate system is different. The coordinate system of the RGB image is ColorSpace, that of the depth image is DepthSpace, and that of the skeleton or body joint information is CameraSpace (see Table 6.1).

`CameraSpacePoint` refers to the 3D coordinate system used by Kinect. Its origin is located at the center of the depth camera and each unit of measure is equivalent to one meter. `ColorSpacePoint` marks the location of coordinates on a 2D color image (as garnered from a ColorFrame). Likewise, `DepthSpacePoint` marks the coordinates on a 2D depth image.

CameraSpace is a 3D coordinate system (Figure 6.3) with the following features:

- Kinect v2 is located at the origin of the coordinate system.
- The direction of the camera lense is the positive direction of the z-axis.
- Vertical upward direction is the positive direction of the y-axis.
- Right-handed.

FIGURE 6.3: CameraSpace coordinate system used by Kinect v2.

In all 3 types of coordinate systems –CameraSpace, ColorSpace, and DepthSpace– "the horizontal direction from left to right seen from the user facing Kinect v2" is the positive direction of the x-axis.

Hence, we should align the images obtained from the ColorSpace and the DepthSpace coordinate systems to be able to work with the information captured from the CameraSpace coordinates. We can perform this alignment mathematically, but fortunately Kinect v2 already solves this critical problem. As seen in Chapter 5, we declare an instance of the `CoordinateMapper` utility class to let us align points in both images along with the positions of skeletal joints. It is not a static class and is a property of each individual `KinectSensor` object. `CoordinateMapper` is able to convert coordinates from: CameraSpace to ColorSpace, CameraSpace to DepthSpace, DepthSpace to ColorSpace and DepthSpace to CameraSpace.

6.2.3 Joint Orientations

Joint orientation can be one of the most difficult concepts to grasp in the Kinect SDK. However, it is especially useful in applications where absolute metric coordinates are not as helpful. Such is the case of avateering, for example, where the proportions of an animated character might not correspond to a user's actual body.

Joint orientations sidestep this issue by ignoring bone length and instead permitting us to rebuild the skeleton with knowledge of the joints' local quaternion orientation values. These quaternion values indicate the rotation of a joint about the bone originating from its parent joint. Each joint has its place in a hierarchy, as shown in Figure 6.4, which can be traversed to reconstruct a skeleton (see Figure 6.5).

Almost all of the Kinect's raw skeletal and body data is embodied in `BodyFrameSource`. Similarly to `DepthFrameSource`, `InfraredFrameSource` and `ColorFrameSource`, the object

TABLE 6.1: The three types of coordinate systems in Kinect v2

Coordinate System	Type of Coordinates	Captured Data
ColorSpace	ColorSpacePoint	RGB image
DepthSpace	DepthSpacePoint	Depth, BodyIndex, Infrared image
CameraSpace	CameraSpacePoint	Skeleton information

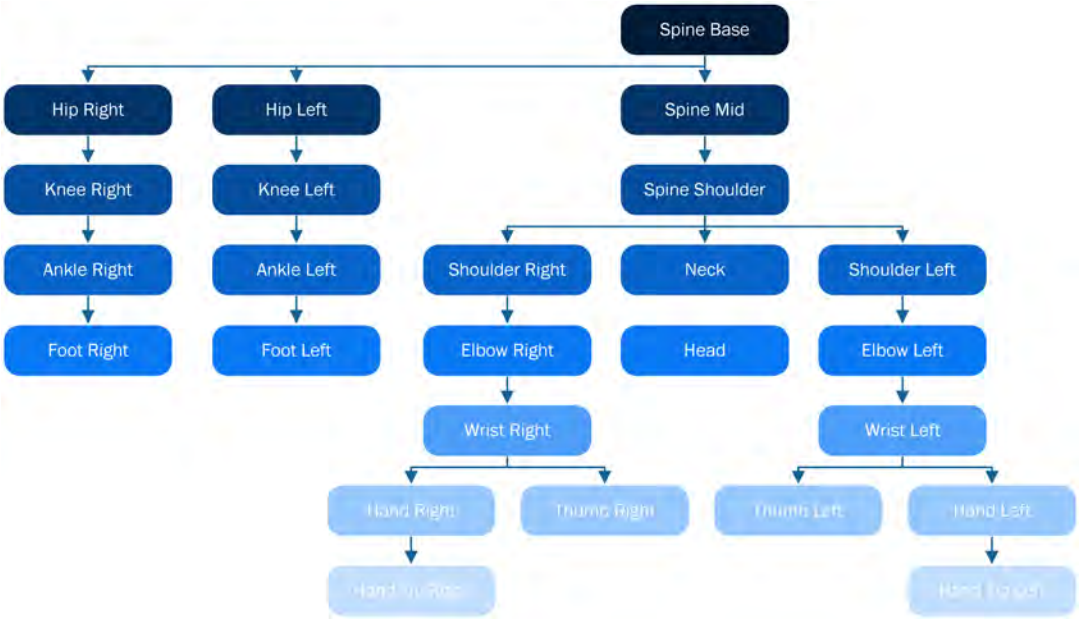


FIGURE 6.4: Joint hierarchy in the Kinect v2 SDK.

`BodyFrameSource` gives data frames roughly in sync with the other data sources, each one detailing positional and directional data about the body joints observed by Kinect.

In Chapter 2, Table 2.1 mentioned that the Kinect for Windows v2 tracks 25 joints, 5 more than the Kinect for Windows v1 did. These 25 joints are depicted in Figure 6.5 and include the new Neck, Left Hand Tip, Right Hand Tip, Left Thumb and Right Thumb joints. Overall, the skeletal-tracking accuracy has been drastically improved since the previous generation. This improvement enables a much more fluid experience for a user employing gesture-based applications.

Using the LINQ namespace, we can loop through each *body* in our *bodies* array that is being tracked to apply any code requiring joint data. The `Joints` property of the `Body` structure contains a dictionary of key-value pairs for each joint in the tracked body. Hence, if we would want to access a specific joint directly, we could query the `JointType` in the dictionary as such:

```
1 Joint j = body.Joints[JointType.ShoulderLeft];
```

Note that the identifiers for the `JointTypes` are all written as camel-cased renditions of those listed in Figure 6.5 (i.e., `Shoulder_Left` is written as `ShoulderLeft` in this context). With this line of code, however, we are only able to access the *position* and *tracking state* information of the `ShoulderLeft` joint. To obtain the *orientation* of that joint we should invoke:

```
1 Vector4 j = body.JointOrientations[JointType.ShoulderLeft].Orientation;
```

where the `Vector4` structure properly stores the joints' local quaternion orientation values.

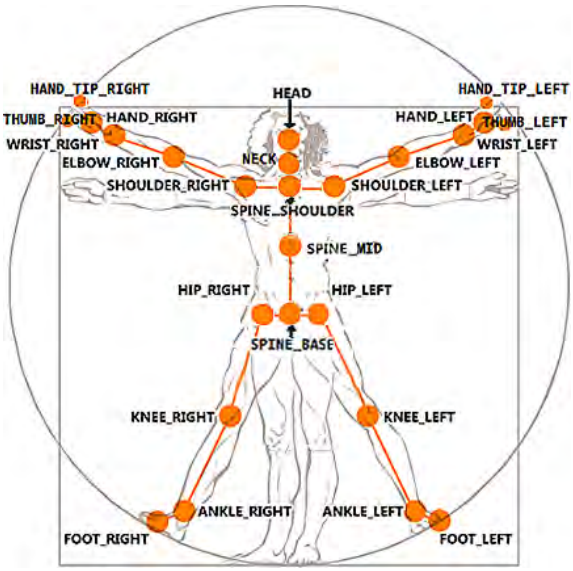


FIGURE 6.5: Kinect v2 joint types with positions relative to the human body.

6.3 Pose Estimation

From computer graphics, the application domain of quaternions soon expanded into other fields such as robotics and virtual reality, since researchers and game programmers have discovered the true potential of quaternions and started using them as a powerful tool for describing rotations about an arbitrary axis.

In this work, we are giving particular emphasis to computer vision algorithms for 3D pose estimation from joint orientations. Quaternions are a compact representation provided by the RGB-D sensor which allow us to derive closed form solutions to finite rotations in space.

6.3.1 Quaternions

As mentioned earlier in Subsection 6.2.3, Kinect v2 provides joint orientation information in the form of quaternions. When composing several rotations on a computer, rounding errors necessarily accumulate. A quaternion that is slightly off still represents a rotation after being normalized, whereas a rotation matrix under the same circumstances may become non-orthogonal and computationally more expensive to convert back to a proper orthogonal matrix. Hence, it is only logical that Kinect's internal operations regarding joint rotations are performed and delivered in the quaternion domain.

Strictly speaking, a *quaternion* is represented by four elements:

$$\mathbf{q} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3, \tag{6.1}$$

where q_0, q_1, q_2 and q_3 are real numbers, and **i**, **j** and **k** are mutually orthogonal imaginary unit vectors. The q_0 term is referred to as the "real" component, and the remaining three terms are the "imaginary" components. In practice, the imaginary notation is implicite, and only the four coefficients are used to specify a quaternion, as in:

$$\mathbf{q} = (q_0, q_1, q_2, q_3). \tag{6.2}$$

Given the rotation quaternion $\mathbf{q}$, the corresponding rotation matrix is:

$$\mathbf{R}(\mathbf{q}) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}. \tag{6.3}$$

Rotation quaternions are a mechanism for representing rotations in three dimensions, and they can be used as an alternative to rotation matrices in 3D graphics and other applications. A quaternion in Kinect is a set of four values: w, x, y and z; which are the conceptual equivalents of the four coefficients q_0, q_1, q_2 and q_3 in Equation 6.2,

$$\mathbf{q} = (w, x, y, z). \tag{6.4}$$

One of the main benefits of using quaternions over other representations such as Euler angles is the singularity free kinematics relation (no gimbal lock). However, conversion to Euler angles results very convenient when you want to use each axis for different operations or you only care about a certain axis of movement. The latter is our case, since we are interested in one particular axis of rotation from certain selected joints. A more detailed explanation will be provided in Subsection 6.3.3. In the meantime, we know that the easiest way to achieve that is to convert the quaternion to *Euler angles*.

Euler angles use the composition of 3 successive –simple– rotations about different axis to generate any possible rotation. They are a complicated subject, primarily because there are dozens of mutually exclusive ways to define them. Different authors are likely to use different conventions, often without clearly stating the underlying assumptions, which makes it difficult to combine equations and code from more than one source.

Kinect's convention on Euler angles uses a right-handed coordinate system following a pitch-yaw-roll rotation order, rotating around the x, y and z axes respectively (see Figure 6.6). *pitch* describes rotation around the x-axis, such as when a person is nodding in agreement; *yaw* describes rotation around the y-axis, such as when a person is shaking their head from side to side in disagreement; and *roll* describes rotation around the z-axis, such as when a person is tilting their head. When the pitch, yaw and roll values are all 0 degrees, the user is facing the Kinect camera directly.

Before performing any conversion or operation with quaternions –even to compute the rotation matrix from Equation 6.3– they must be normalized to obtain *unit quaternions*:

quaternions with norm 1. An orientation is represented by a unit quaternion. If the quaternion is not a unit quaternion, then it is not valid in this context.

To normalize a quaternion, each component is divided by the norm. According to the Kinect's quaternion representation in Equation 6.4, the norm of a quaternion is:

$$N(\mathbf{q}) = \sqrt{w^2 + x^2 + y^2 + z^2}. \tag{6.5}$$

Once the quaternion has been normalized, we can proceed to convert it to Euler angles. We are interested in extracting the Euler angles by re-formulating the rotation matrix $\mathbf{R} = \mathbf{R}(\psi, \theta, \phi)$:

$$\mathbf{R} = \begin{pmatrix} \cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\psi\cos\phi\sin\theta + \sin\psi\sin\phi \\ \cos\theta\sin\psi & \sin\psi\sin\theta\sin\phi + \cos\phi\cos\psi & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}. \tag{6.6}$$

This matrix can be interpreted as a consecutive composition of the next simpler rotations:

- A rotation of ϕ about the x axis,
- A rotation of θ about the y axis, and
- A rotation of ψ about the z axis.

Hence, from a given matrix $\mathbf{R}$ and in reference to Equation 6.6, we could expect to extract the Euler angles by:

$$\begin{aligned} \theta &= -\arcsin(R_{31}) \\ \phi &= \arctan(R_{32}/R_{33}) \\ \psi &= \arctan(R_{21}/R_{11}). \end{aligned} \tag{6.7}$$

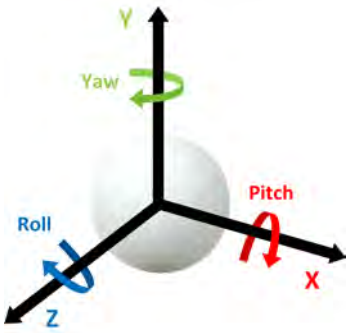


FIGURE 6.6: Euler angles rotation for joint orientations with Kinect v2.

The Kinect SDK is encapsulating the quaternion into a structure called `Vector4`. We need to transform this orientation quaternion into a set of three numeric values –the Euler angles– to calculate the rotation of the joint around the x, y, and z axis.

Thus, based on Equation 6.7 and taking the values from the rotation matrix in Equation 6.3 by the inhomogeneous expression, Euler angles can be obtained from a quaternion $\mathbf{q}$:

$$\begin{aligned} \theta &= -\arcsin(2q_1q_3 - 2q_0q_2) \\ \phi &= \arctan\frac{2q_2q_3 + 2q_0q_1}{1 - 2(q_1^2q_2^2)} \\ \psi &= \arctan\frac{2q_1q_2 + 2q_0q_3}{1 - 2(q_2^2q_3^2)}, \end{aligned} \tag{6.8}$$

changing accordingly to the quaternion coefficients in Kinect (Equation 6.4), then:

$$\begin{aligned} \theta &= -\arcsin(2xz - 2wy) \\ \phi &= \arctan(2yz + 2wx/1 - 2(x^2y^2)) \\ \psi &= \arctan(2xy + 2wz/1 - 2(y^2z^2)). \end{aligned} \tag{6.9}$$

Note, however, that the arctan and arcsin functions implemented in computer languages only produce results between -$\pi/2$ and $\pi/2$, and for three rotations between -$\pi/2$ and $\pi/2$, one does not obtain all possible orientations. To generate all the orientations just replace the arctan functions in computer code by *atan2*.

6.3.2 Face detection

Kinect's face-tracking APIs are in a different namespace called `Microsoft.Kinect.Face` and they are divided into two subsets. One is *Face*, which provides a general 2D overview of the face's features, while the other is *HD Face*, which offers a more comprehensive 3D model of the face's structure. An application that merely needs limited or brief tracking of facial features, such as detecting whether a user is looking at the screen, should rely on the Face API. Whereas an application that wants to recreate the user's face on an avatar or track minute differences on the face should make use of HD Face.

As previously stated, when the pitch, yaw and roll of a joint are all 0 degrees, means that the user is facing the Kinect camera directly. Yet, it is also the case when the user is on his/her back, facing away from the Kinect. Hence, we employ a face detection stage with the Face API to overcome this issue, allowing the system to discern between "Front" and "Back" views from the user in an active or passive interaction.

Data from the Face API is obtained as other Data Sources (see Subsection 6.2.1). The main differences are two: we assign a separate `FaceFrameSource` and `FaceFrameReader`

for each face and we must extract a `FaceFrameResult`, which is what contains the recognized facial features and data, from each `FaceFrame`.

Thus, once we declare arrays in which to contain the frame sources, readers and results for all the faces that can be simultaneously tracked at an instant, there is also the need to declare which features we plan to extract from the `FaceFrame` (see Listing 6.1).

```
1  _sensor = KinectSensor.GetDefault();
2
3  if (_sensor != null)
4  {
5      _sensor.Open();
6
7      // Initialize the background removal tool.
8      _backgroundRemovalTool = new BackgroundRemovalTool(_sensor.CoordinateMapper
       );
9
10     _reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
       FrameSourceTypes.Depth | FrameSourceTypes.BodyIndex | FrameSourceTypes.Body
        | FrameSourceTypes.Infrared);
11     _reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;
12
13     bodies = new Body[_sensor.BodyFrameSource.BodyCount];
14
15     FaceFrameFeatures faceFrameFeatures =
16         FaceFrameFeatures.BoundingBoxInInfraredSpace
17         | FaceFrameFeatures.PointsInInfraredSpace;
18
19     faceFrameSources = new FaceFrameSource[6];
20     faceFrameReaders = new FaceFrameReader[6];
21     faceFrameResults = new FaceFrameResult[6];
22
23     for (int i = 0; i < 6; i++)
24     {
25         faceFrameSources[i] = new FaceFrameSource(_sensor, 0, faceFrameFeatures
       );
26         faceFrameReaders[i] = faceFrameSources[i].OpenReader();
27         faceFrameReaders[i].FrameArrived += FaceReader_FrameArrived;
28     }
29 }
```

LISTING 6.1: Constructor for face detection using the Face API.

It can be highlighted from line 8 in Listing 6.1 that we are merging this code with the background removal algorithm of Chapter 5 and its `_backgroundRemovalTool` class, maintaining all the variables, methods and constructors names, in order to encourage consistency throughout the code shared on this dissertation.

In addition to the features listed in lines 16 and 17, other ones are also:

- FaceFrameFeatures.BoundingBoxInColorSpace

- FaceFrameFeatures.PointsInColorSpace
- FaceFrameFeatures.RotationOrientation
- FaceFrameFeatures.FaceEngagement
- FaceFrameFeatures.Glasses
- FaceFrameFeatures.Happy
- FaceFrameFeatures.LeftEyeClosed
- FaceFrameFeatures.RightEyeClosed
- FaceFrameFeatures.LookingAway
- FaceFrameFeatures.MouthMoved
- FaceFrameFeatures.MouthOpen

The infrared space equivalent to `BoundingBoxInColorSpace` and `PointsInColorSpace` is employed since Kinect's face-detection algorithms are able to recognize faces using the infrared camera. They use infrared because it is lighting independent, and thus faces are bound to be accurately recognized in any lighting condition. Feature coordinates are internally translated by the API to color space using the `CoordinateMapper` class.

Arrays are created (lines 19-21) for the face frame sources, readers and results for the number of faces that we are interested in tracking. In our case, the experiments of this chapter are a first attempt to prove the feasibility of the system, not its full performance. Ergo, we have started with only one user and one face at a time; however, a detailed explanation is provided about how to address all six of them for the sake of future implementations.

Lines 23 to 28 loop through the `FaceFrameSource` and `FaceFrameReader` arrays and generate the `FaceFrameSources` to set up their readers. The constructor method in line 25, `new FaceFrameSource(_sensor, 0, faceFrameFeatures)`, takes the Kinect sensor, initial tracking ID, and desired face frame features as inputs. The tracking ID is a property of the `Body` class, and we can use it to set the `FaceFrameSource` to track the face of a specific body and to ensure that the system is tracking faces from bodies visible in the scene.

In Listing 6.2, `FaceFrameResult` is saved –which is a property of `FaceFrame`– into our array of `faceFrameResults`. Keeping track of its index is possible to work with its respective frame source, frame reader and body. Since a reference to its `FaceFrameSource` is available in the `FaceFrame`, the initiative of finding its index using a `GetFaceSourceIndex (FaceFrameSource faceFrameSource)` method is taken, as shown in Listing 6.3.

```
1 private void FaceReader_FrameArrived(object sender,FaceFrameArrivedEventArgs e)
2 {
3     using (FaceFrame faceFrame = e.FrameReference.AcquireFrame())
```

```
4       {
5           if (faceFrame != null)
6           {
7               int index = GetFaceSourceIndex(faceFrame.FaceFrameSource);
8
9               if (ValidateFaceBoundingBox(faceFrame.FaceFrameResult)){
10                  faceFrameResults[index] = faceFrame.FaceFrameResult;
11              }
12              else{
13                  faceFrameResults[index] = null;
14              }
15          }
16      }
17  }
```

LISTING 6.2: FaceFrameReader Event Handler.

Finding the index is a straightforward matter: loop through the `faceFrameSources` array and check whether any of them match the `FaceFrameSource` from `FaceFrame`.

```
1  private int GetFaceSourceIndex(FaceFrameSource faceFrameSource)
2  {
3      int index = -1;
4      for (int i = 0; i < 6; i++)
5      {
6          if (faceFrameSources[i] == faceFrameSource)
7          {
8              index = i;
9              break;
10         }
11     }
12     return index;
13 }
```

LISTING 6.3: Obtaining the index of a FaceFrameSource.

We can also check if the bounding box is within the confines of the image with a custom `ValidateFaceBoundingBox (FaceFrameResult faceFrameResult)` method and reject the `FaceFrameResult` if it fails to do so, as a prudent measure to ensure that the box actually fits inside the image and it is not a false positive (see Listing 6.4).

```
1  private bool ValidateFaceBoundingBox(FaceFrameResult faceFrameResult)
2          {
3              bool isFaceValid = faceFrameResult != null;
4              if (isFaceValid)
5              {
6                  RectI boundingBox = faceFrameResult.
       FaceBoundingBoxInInfraredSpace;
7                  if (boundingBox != null)
8                  {
9                      isFaceValid = (boundingBox.Right - boundingBox.Left) > 0 &&
```

```
10                      (boundingBox.Bottom - boundingBox.Top) > 0 &&
11                  boundingBox.Right <= frameDescription.Width &&
12                  boundingBox.Bottom <= frameDescription.Height;
13              }
14          }
15          return isFaceValid;
16      }
```

LISTING 6.4: Face bounding box validation.

Finally, notice that the `MultiSourceFrameReader` in Listing 6.5 shares basically the same structure with the one portrayed in Chapter 5 (Listing 5.3). Difference is that the event handler for the `MultiSourceFrameReader` in this chapter also collects body and infrared data (lines 5 and 8, respectively). Moreover, added lines 14 to 39 inside the block `using (var bodyIndexFrame = reference.BodyIndexFrameReference.AcquireFrame()) {[...]}` will access facial and body information and determine whether a face has been detected or not.

```
1  void Reader_MultiSourceFrameArrived(object sender,
       MultiSourceFrameArrivedEventArgs e)
2  {
3      var reference = e.FrameReference.AcquireFrame();
4
5      using (var bodyFrame = reference.BodyFrameReference.AcquireFrame())
6      using (var colorFrame = reference.ColorFrameReference.AcquireFrame())
7      using (var depthFrame = reference.DepthFrameReference.AcquireFrame())
8      using (var infraredFrame = reference.InfraredFrameReference.AcquireFrame())
9
10     using (var bodyIndexFrame=reference.BodyIndexFrameReference.AcquireFrame())
11     {
12         if (colorFrame != null && depthFrame != null && bodyIndexFrame != null
       && bodyFrame != null && infraredFrame != null)
13         {
14             //  Update the array of bodies
15             bodyFrame.GetAndRefreshBodyData(bodies);
16
17             for (int i = 0; i < 6; i++)
18             {
19                 if (faceFrameSources[i].IsTrackingIdValid)
20                 {
21                     if (faceFrameResults[i] != null){
22                         //Face detected
23                         face_flag = 1;
24                     }
25                     else{
26                         //No face detected
27                         face_flag = 0;
28                     }
29                 }
```

```
30                  else
31                  {
32                      if (bodies[i].IsTracked){
33                          //Body detected but no face
34                          faceFrameSources[i].TrackingId = bodies[i].TrackingId;
35                      }
36                  }
37              }
38              //Update the image source
39              camera.Source = _backgroundRemovalTool.GreenScreen(colorFrame,
    depthFrame, bodyIndexFrame, bodyFrame, face_flag);
40          }
41      }
42  }
```

LISTING 6.5: MultiSourceFrameReader event handler for face detection.

Face detection is performed only if the `faceFrameSources[i].IsTrackingIdValid` property returns *True*. Otherwise, it is considered whether a body is being checked for index *i* and, if so, assign its `TrackingId` to its respective face (line 34).

A special `BodyFrame` method called `GetAndRefreshBodyData(IList<Body> bodies)` is invoked in line 15 to get our body data. The SDK directly updates the necessary parts of the `Body` array that are inputted into this method to reduce memory consumption, and it should be maintained throughout the lifetime of the `BodyFrameReader`'s operation.

Minor changes with respect to the same call of the `_backgroundRemovalTool` class in Listing 5.3 are present in line 39. Now, it also feeds the class with body data necessary to read and manipulate joints information, and a face flag containing the face detection result.

Up to this point, it is possible to inform the system if the user's face has been found by Kinect sensor, and support pose estimation for "Front" and "Back" views computed from the joints orientation information we will obtain in the following sections.

6.3.3 3D Partial Scans orientations

Hierarchical rotation in Kinect v2 provides the amount of rotation in 3D space from the parent bone to the child. This information tells us how much we need to rotate in 3D space the direction of the bone relative to the parent. It is the equivalent to consider the rotation of the reference Cartesian axis in the parent-bone object space to the child-bone object space, considering that *the bone lies on the y-axis* of its object space (see Figure 6.7).

In the hierarchical definition, the rotation of the Spine Base joint yields the absolute orientation of the player in CameraSpace coordinates. It assumes the player's object space has its origin at the Spine Base joint: the y-axis is upright, the x-axis is to the left, and the z-axis faces the camera (Figure 6.8).

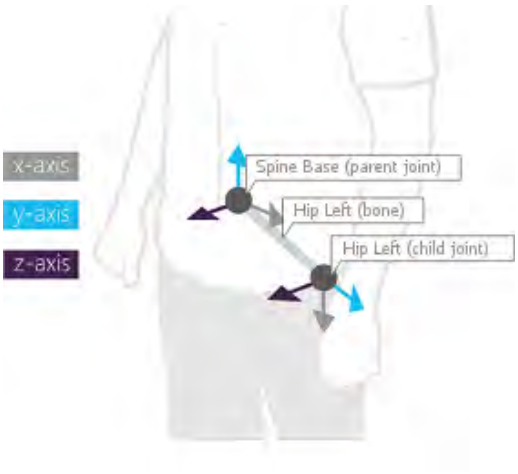


FIGURE 6.7: Bone rotation is stored in a bone's child joint. For instance, the rotation of the left hip bone is stored in the Hip Left joint.

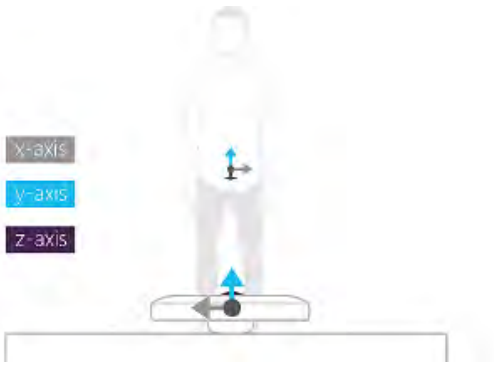


FIGURE 6.8: Absolute player orientation is rooted at the Spine Base joint.

Despite this fact, we should not entirely trust on the orientation information provided by the Spine Base joint alone. The user's own body can cause occlusions between joints by not facing the Kinect sensor directly or due to the position of their extremities. Therefore, we ought to have a couple of support joints to enrich the accuracy of our pose estimation.

It is only natural to assume that if a user walks into the scene and turns around in any direction, we should focus on that *one* axis of movement: the vertical. Transforming Kinect's quaternions into Euler angles help us to understand the user's orientation by computing a particular rotation at the selected joints.

From that orientation information, we intend to estimate the following poses from the user: Front, Back, Side Right, Side Left, Front Side Right, Front Side Left, Back Side Right and Back Side Left. These eight views or poses are what we call the *3D Partial Scans*.

Thus, at first, we were particularly interested in three joints: *Spine Base*, *Hip Right* and *Hip Left*; since their orientations can provide the most stable, truthful and useful information about the absolute orientation of the user in front of the RGB-D sensor. Relying on pose estimation over those specific joints is due to their own nature: they compose an area

of the human body that genuinely indicates the direction of attention of an individual. A topic already discussed in Chapter 4 and justified in Section 4.4.2.

Taking the absolute orientation of the user as a reference point, and that rotation around the y-axis of the Spine Base joint means the user is rotating around its own axis –which implies the user's pose is changing– so we compute the *yaw* Euler angle for this joint (Listing 6.6), based on the formulas of Equation 6.9.

```
1  public static double Yaw(this Vector4 quaternion)
2  {
3      double value = 2.0*(quaternion.W * quaternion.Y - quaternion.Z * quaternion
       .X);
4
5      value = value > 1.0 ? 1.0 : value;
6      value = value < -1.0 ? -1.0 : value;
7
8      double result_yaw = Math.Asin(value);
9      return result_yaw * (180.0 / Math.PI);
10 }
```

LISTING 6.6: Rotates the specified quaternion around the y-axis.

As for the Hip Right and Hip Left joints, their *pitch* Euler angles are computed accordingly (Listing 6.7). It is a rotation around the x-axis of movement of both joints, following the motion on the vertical axis of the Spine Base joint.

```
1  public static double Pitch(this Vector4 quaternion)
2  {
3      double value1 = 2.0*(quaternion.W * quaternion.X + quaternion.Y *
       quaternion.Z);
4      double value2 = 1.0 - 2.0 * (quaternion.X * quaternion.X + quaternion.Y *
       quaternion.Y);
5
6      double result_pitch = Math.Atan2(value1, value2);
7
8      return result_pitch * (180.0 / Math.PI);
9  }
```

LISTING 6.7: Rotates the specified quaternion around the x-axis.

Notice that lines 5 and 6 from Listings 6.6 attempt to cover the gimbal lock singularity when the yaw approaches $\pm 90°$, although it is very unlikely to occur in the rotation of human body joints, some preventive measures should be taken.

Nevertheless, while performing the first experiments, we found that the Hip joints tend to be occluded when the user is standing on a relaxed position in a Side Right or Side Left view. The user's arms happen to be relaxing on the sides of his/her body, naturally placing the hands in an overlapping position over the Hip joints. To overcome this issue, we exchanged our Hip Right and Hip Left support joints for the *Shoulder Right*

and *Shoulder Left* joints, to assist the Spine Base joint when retrieving information from poses with a high tendency towards occlusion.

6.4 User Identification and Re-Identification

The process of identifying and re-identifying a user actively or passively interacting with the system presented here starts by detecting a human user in front of the Kinect sensor, read the user's pose through joint orientation information and, find a correspondence to any user already seen before by comparing the appearance retrieved from the current pose against a database of user's images.

The user on the scene is not obliged to adopt any particular or flashy pose in order to interact with the system. The RGB-D sensor is aware of what is going on in its surroundings and responds accordingly by capturing information of any human naturally engaging with our proposed system.

6.4.1 Methodological Approach

The main idea is to estimate the pose of a user with 3D data (joint orientations), save it as 2D information (images) and virtually construct a model of that individual, which is made of partial views or *scans*, birthing the name given to our original approach: 3D Partial Scans Models (3D PSMs).

In this sense, it is a way of "scanning" a human without capturing the full body of a person in three dimensions, which enables us to work with data of lower complexity that is not computationally intensive. We are only benefiting from the accuracy of 3D information deployed by the RGB-D sensor, in order to be exploited on two-dimensional instances.

The approach does not interfere with natural interaction between the user and the sensor, since a person does not have to enter any special booth or cabin to capture the shape and appearance of their body with great accuracy. However, we understand it is not fairly comparable to proper 3D body scanners meant for robust applications that demand superior levels of accuracy to generate highly detailed 3D models. Since it is imperative to obtain exact measurements for their implementation in fitness, personal health and medical tasks.

Figure 6.9 depicts the overall structure of the proposed system: from the 3D PSMs formulation, through the ID assignation process, and on to the re-identification phases as well. The specific methodologies that have been adopted in the development of such an approach are detailed in the following subsections.

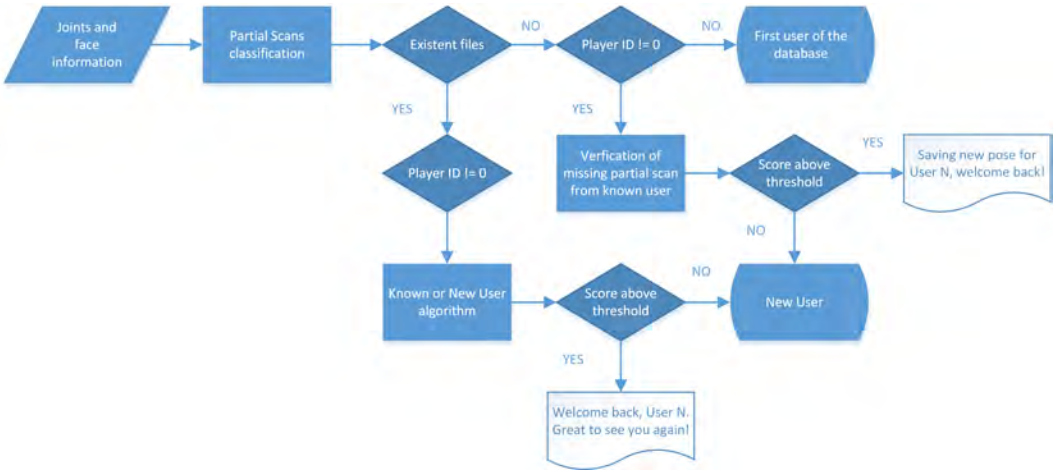


FIGURE 6.9: General scheme for the 3D PSMs user re-identification system with a RGB-D sensor.

6.4.2 3D Partial Scans Model

As previously addressed in Subsection 6.3.3, 3D partial scans are the eight image views obtained from pose estimation performed with RGB-D information. Together, these images construct the model of a user, called a *3D Partial Scans Model*. Figure 6.10 embodies how we conceive the 3D PSM concept. At the best of our knowledge, 3D PSMs represent a new 2D modelling approach with an integrated understanding of 3D data.

Rather than necessary, the eight-view proposal is more of a *compromise* of complexity. Less views would mean less user information to compare with, leaving our NUI system "visually impaired" and raising the probabilities of identifying a known user on a different (missing) pose as a new user. On the other hand, more views would extract more information from the user, at a cost of higher processing to cope with that amount of resources in the system. By selecting the eight most common and standard points of view, we try to settle in the middle ground of this trade-off, applying sensitive ranges for each view as well. Nevertheless, it would be interesting to explore those scenarios and their possible outcomes in the future.

Partial scans are defined by different intervals on the rotation values around certain axes of movement in the Spine Base, Shoulder Right and Shoulder Left joints. The rotation values are computed and retrieved from the Yaw and Pitch functions in Listings 6.6 and 6.7, accordingly.

Experimental readings were conducted on the absolute orientation information retrieved from the Spine Base joint, in order to establish range intervals to which every partial scan would belong to. Figure 6.11 displays the average value read in the output thread for each partial scan. These values served as a baseline to calculate all the intervals for: Front (F), Back (B), Side Right (SR), Side Left (SL), Front Side Right (FR), Front Side Left (FL), Back Side Right (BR) and Back Side Left (BL) partial scans.

FIGURE 6.10: 3D Partial Scans Model.

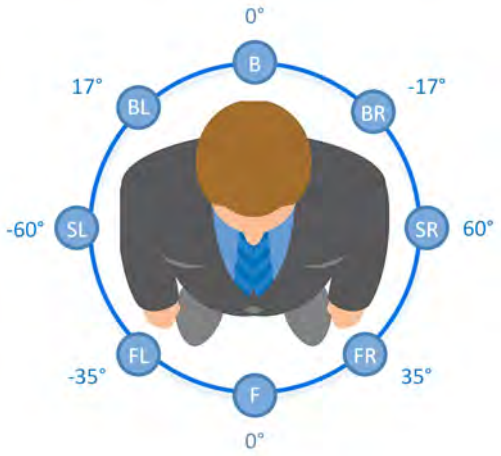


FIGURE 6.11: Yaw rotation values for the Spine Base joint.

Since the orientation at the Spine Base joint results in 0 degrees for the Front and Back partial scans, as Kinect developers did not train body tracking for people not facing the sensor, we set an interval between $-11°$ and $11°$ and classify the view based on the presence or absence of a face (Listing 6.8). The sensor readings are very sensitive, hence, the security range of almost every base value seen on Figure 6.11 is ± 10 degrees, approximately. Face detection information relies on the `face_flag` outcome from the code listings in subsection 6.3.2.

It is important to notice in lines 4 and 7 from Listing 6.8, that we are raising a flag in the event of detecting *one* image which corresponds to a partial scan pose. We propose only one image per partial scan to construct the user's model.

```
1 if (rotationYSB >= -11 && rotationYSB <= 11)
```

```
{
    if (face_flag == 1){
        front = 1;
    }
    else{
        back = 1;
    }
}
```

LISTING 6.8: Estimating Front and Back partial scans.

Front, Front Side Right and Front Side Left partial scans, use positive face detection to be acknowledged (`face_flag == 1`), whereas Back, Back Side Right and Back Side Left require the opposite trait (`face_flag == 0`), as shown in Listing 6.9.

```
if (rotationYSB >= 25 && rotationYSB <= 45 && face_flag == 1){
    fsr = 1;
}
if (rotationYSB >= -45 && rotationYSB <= -25 && face_flag == 1){
    fsl = 1;
}
if (rotationYSB >= -23 && rotationYSB <= -12 && face_flag == 0){
    bsr = 1;
}
if (rotationYSB >= 12 && rotationYSB <= 23 && face_flag == 0){
    bsl = 1;
}
```

LISTING 6.9: Estimating intermediate poses between Front and Back.

On the other hand, Side Right and Side Left partial scans need the assistance of Shoulder Right and Shoulder Left joints information, which is reflected in Listing 6.10.

```
1 if ((rotationYSB >= -70 && rotationYSB <= -50) || (rotationXSR >= -125 &&
    rotationXSR <= -105) || (rotationXSL >= 130 && rotationXSL <= 150))
2 {
3     side_left = 1;
4 }
5
6 if ((rotationYSB >= 50 && rotationYSB <= 70) || (rotationXSR >= 130 &&
    rotationXSR <= 150) || (rotationXSL >= -130 && rotationXSL <= -110))
7 {
8     side_right = 1;
9 }
```

LISTING 6.10: Estimation of Side Right and Side Left partial scans.

It is fairly curious how shoulder joints rotation complement each other perfectly. The intervals are virtually the same numerical ranges but inverted on the side they are acting on (see lines 1 and 6). That did not happen with hip joints, readings were quite similar to each other rather than complementary. On other occasions, when we expected these

joints to behave disruptively, the readings would remain unaltered. Surprisingly, further experimentation found that situation of instability was caused by occlusions from the hips with the upper limbs, and called for a change of strategy: to read shoulder joints instead.

6.4.3 Narrowing the Search

One of the advantages offered by 3D PSMs is the speed and quality of the "partial scan match" search process. It all comes down to a classic image search. No need to use computationally expensive procedures, like point cloud matching or 3D warping.

When the system interacts with an unknown user, it tries to capture as many partial scans as possible, assigning a user ID and storing these images, which are part of the user's partial scan model. However, when the system attempts to verify if the user already exists on its database, it uses the pose currently detected to compare it with the already saved partial scans of the same pose. An action that narrows the matching search by a 1/8 proportion.

Since capturing all the partial scans from a given user largely depends on the length and the type of interaction (active or passive), there is the ever present possibility of missing poses from a known user. For the sake of experimentation, we asked our volunteers to turn through 360 degrees to be able to capture most of them. And yet, this does not ensure that we will retrieve all eight partial scans.

At first, the idea was to complete the 3D PSM by "crafting" the missing views using mirrored images from the complementary poses that were actually captured in real-time. It was a fair solution to substitute real lectures with estimated lectures, although efforts should be made to retrieve the real data.

Hence, the system always verifies if the current partial scan detection relates to a missing partial scan from a known user, saving the new pose if that is the case. The "Verification of missing partial scan" stage depicted in Figure 6.9, looks for a match in the complementary view of the pose detected from any player in front of the Kinect sensor.

The complementary partial scan pairs are: Front – Front Side Right, Back – Back Side Right, Front Side Right – Front Side Left, Side Right – Side Left, Back Side Right – Back Side Left. As human bodies are usually left-right symmetric, it means that these pairs contain a high probability of being similar between them, and consequently, of obtaining a known user match if only one of them is missing.

6.4.4 Matching by Similarity Score

First, we start by setting the current partial scan detection as the *reference* image. This image must be compared with the rest of existing images from that particular partial scan database. However, in order to analyze an arbitrary number of images and detect

the one which contains our reference image, it is necessary a score of similarities for each analyzed image.

The idea is simple: given a certain folder(s) containing images, and a reference image to be localized, the system must process each one of them, determining a value which represents how many key points from the reference image have been spotted into the cycled image. At the end of the loop, the image(s) returning higher values are more likely to be the results we expect. In other words, higher the score, higher the chances of our reference to be contained into those images. We have made a customized implementation of the `FindMatch()` method from EmguCV[1] to expose the count of matches found between images and determine a similarity score (Listing 6.11). Emgu CV is a cross platform .NET wrapper to the OpenCV image processing library, which allows OpenCV functions to be called from .NET compatible languages such as C#.

```
1  public static void FindMatch(Mat modelImage, Mat observedImage, out long
       matchTime, out VectorOfKeyPoint modelKeyPoints, out VectorOfKeyPoint
       observedKeyPoints, VectorOfVectorOfDMatch matches, out Mat mask, out long
       score)
2  {
3      int k = 2;
4      double uniquenessThreshold = 0.80;
5
6      Stopwatch watch;
7
8      modelKeyPoints = new VectorOfKeyPoint();
9      observedKeyPoints = new VectorOfKeyPoint();
10
11     KAZE featureDetector = new KAZE();
12     Mat modelDescriptors = new Mat();
13
14      featureDetector.DetectAndCompute(modelImage, null, modelKeyPoints,
       modelDescriptors, false);
15
16     watch = Stopwatch.StartNew();
17
18     if (modelDescriptors.IsEmpty == false)
19     {
20         Mat observedDescriptors = new Mat();
21
22         featureDetector.DetectAndCompute(observedImage, null, observedKeyPoints
       , observedDescriptors, false);
23
24         // KdTree for faster results / less accuracy
25         using (var ip = new Emgu.CV.Flann.KdTreeIndexParams())
26         using (var sp = new SearchParams())
27         using (DescriptorMatcher matcher = new FlannBasedMatcher(ip, sp))
```

[1] http://www.emgu.com/

```
28          {
29              matcher.Add(modelDescriptors);
30
31              matcher.KnnMatch(observedDescriptors, matches, k, null);
32              mask = new Mat(matches.Size, 1, DepthType.Cv8U, 1);
33              mask.SetTo(new MCvScalar(255));
34              Features2DToolbox.VoteForUniqueness(matches, uniquenessThreshold,
    mask);
35
36              // Calculate score based on matches size
37              // ---------------------------------------------------
38                  score = 0;
39                  for (int i = 0; i < matches.Size; i++)
40                  {
41                      if (mask.GetData(i)[0] == 0) continue;
42                      foreach (var e in matches[i].ToArray())
43                          ++score;
44                  }
45              // ---------------------------------------------------
46
47              int nonZeroCount = Emgu.CV.CvInvoke.CountNonZero(mask);
48              if (nonZeroCount >= 4)
49              {
50                  nonZeroCount = Features2DToolbox.VoteForSizeAndOrientation(
    modelKeyPoints, observedKeyPoints, matches, mask, 1.5, 20);
51              }
52          }
53      }
54      else
55      {
56          mask = null;
57          score = 0;
58      }
59      watch.Stop();
60
61      matchTime = watch.ElapsedMilliseconds;
62  }
```

LISTING 6.11: Determining a similarity score with EmguCV.

Listing 6.11 depicts how KAZE features detects keypoints in both the reference and the target images (lines 14 and 22, respectively) and computes the descriptors from those keypoint locations. Next, the *k*-nearest match is found with a nearest neighbour search, using the *k*-d tree approach (line 25). *k*-d tree is an algorithm that uses a mixture of decision trees and K-Nearest Neighbours (KNN) to organize points within a multidimensional search. In EmguCV library, *k*-d tree is contained in a class from the Fast Library for Approximate Nearest Neighbors (FLANN) [91] and, as such, it will find the nearest neighbor in a majority of cases, but this strongly depends on the dataset being queried.

It becomes a trade-off between improving speed and dropping accuracy. Considering that our system is working on real-time, *k*-d trees are preferred over linear brute-force methods which might bring more accurate results, but the search is performed at a lower speed. KNN matcher in line 31 then finds, for each keypoint that now has a descriptor associated to it, the first *k*-neighbors.

The `VoteForUniqueness()` call filters the matched features, such that if a match is not unique, it is rejected (line 34). The *score* can be calculated after it (lines 38 through 44) looping between the vector of matches to increase score value every time a match is encountered.

At this point, calculating the score for a list of images becomes trivial. From a given image directory, namely our database of 3D PSMs, one needs to traverse the entire directory tree and execute a comparison for every file found. The calculated score can be saved in an apt memory structure, to be listed at the end of the process.

A `WeightedImages` structure/class is defined by:

```
1 class WeightedImages
2 {
3  public string ImagePath { get; set; } = "";
4  public long  Score { get; set; } = 0;
5 }
```

and it memorizes the path of a complete image with its calculated score. This class is used in conjunction with a `List<>`:

```
1 List<WeightedImages> imgList = new List<WeightedImages>();
```

Next, all it takes is for a simple `ProcessFolder()` method to parse every single file of each directory in the structure, recursively. For each of them, it will call a `ProcessImage()` method, which is the heart of the process and where the `FindMatch()` method is invoked. After that call, the `imgList` should be fed with the following line:

```
1 imgList.Add(new WeightedImages() { ImagePath = completeImage, Score = score });
```

The `imgList` is ordered by descendant score value and the first item is selected from this list, as it holds information related to the image with the highest similarity score. Finally, if this similarity score is greater than a fixed threshold value of 10 –established by experimentation– then, the system acknowledges the user currently interacting with Kinect as a *known user*, otherwise the system identifies a *new user*.

6.5 Results

Outstanding accuracy in pose estimation has been obtained for partial scans. One of our best results is shown in Figure 6.12, where all eight partial scans were captured in a single run –when the user is identified for the first time– achieving the construction of a complete 3D Partial Scans Model for the user. Back views are often confused with front

views, and they have become a recurrent problem in our experiments (see *back_0* in the upper-left corner).

However, we do not consider this as an alarming issue, since there are plenty of face detection algorithms for images and video outside from Kinect's SDK with an excellent reputation for their high accuracy which could be used as a way forward strategy.

Kinect can focus on the face of a user, conduct tracking and assay its structure and anatomy. It can do this for six people simultaneously, though certain facial-analysis features are computationally intensive and should be applied to one person at a time. In such cases, HD Face API comes in handy. The HD Face API is one the most advanced face tracking libraries out there. Not only does it detect the human face, but it also allows real-time access to over 1000 facial points in the 3D space.

Certainly, when we performed some experiments with HD Face instead of the regular Face API, face detection was far more accurate. Nevertheless, it can still have occasional failures due to non-ideal positioning of the Kinect sensor.

To achieve optimum performance, the Kinect sensor must be positioned in a location where it can see your entire body. According to Kinect recommendations (NUI Kinect for Windows - Human Interface Guidelines v2.0) the accuracy of the sensor decreases when these ideal conditions are not satisfied.

For those *sensor positioning* expectations to be met, i.e. $\approx$ 2m away from the subject at a convenient height, it can be challenging under our experimental framework. We allowed the users to passively interact with the system as well, roaming freely around the scene without any "rules" that could restrain them from acting naturally. These circumstances lead to results like the one portrayed in Figure 6.13, that can interfere with the quality and accuracy of the re-identification process.

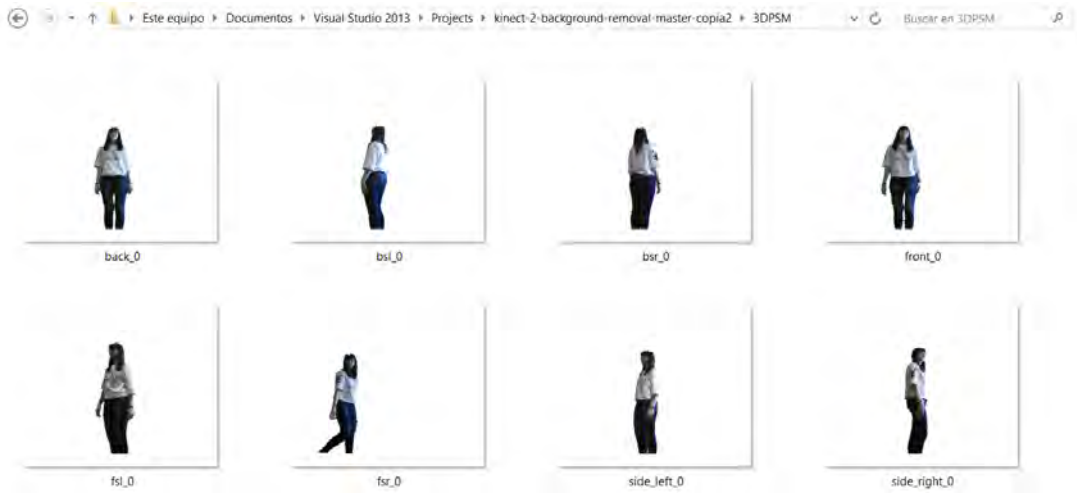


FIGURE 6.12: Full 3D PSM with eight partial scans for User 0. The file names comprise a partial scan pose followed by the user's ID number.

The robustness of 3D PSMs relies heavily on a high rate of capture from the partial scans. Meaning that, the higher the amount of partial scans retrieved when identifying a user, the higher the chances to successfully re-identify this user later on. Saving new poses from known users to complete their 3D PSM, might fail if there are not enough complementary views to compare with. This is specially the case for passive users interacting with the system, since they do not engage with the system for a sufficient amount of time.

Interestingly, it is quite possible to get a successful re-identification with only a few partial scans, as long as the system is able to capture a sharp and clear image from the user. Unlike other works in 2D image recognition, our approach provides background subtracted profiles from our human-body targets, which in turn reduces false positive detections in the matching stage and boosts the accuracy of the similarity score.

A people recognition session starts when a person is detected in the scene by Kinect. While on scene, Kinect keeps track of this person and our system captures as many partial scans as possible, silently building a 3D PSM for that individual. When the person leaves the area in the field of vision of the sensor, Kinect loses tracking. The `body.IsTracked` and `body.TrackingId` properties change, thus, the system is aware that the next person it encounters must be analyzed again, so as to determine whether a known user is back on the scene or if a new ID should be assigned instead. In the latter case, we start over again, collecting the new user's partial scans discreetly. Otherwise, we face a situation like the one depicted in Figure 6.14.

The system detects this user is currently standing with a Side Left pose and goes to the directory which contains the files from that particular partial scan, analyzing through the existing images from known users (filename = user ID) and comparing them to a file

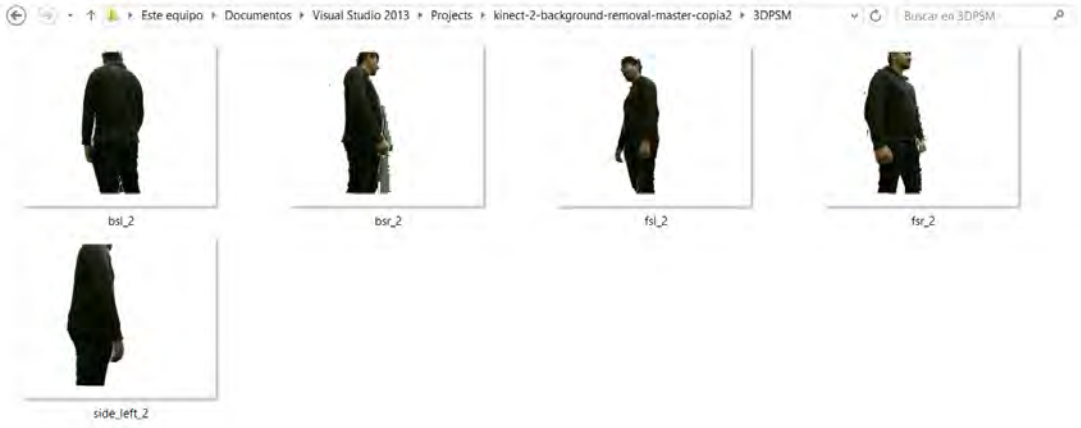


FIGURE 6.13: Incomplete 3D PSM with five partial scans for User 4. Incorrect *Side Left* pose estimation, as the user is too close to the sensor and joint tracking becomes inferred. Not the best captures for partial scans either, making it difficult to find a correct match in the re-identification procedure.

called *sr0* which is a copy of the current frame in the real-time streaming. The output is shown in a purple rectangle on the lower-left corner of Figure 6.14, the file path of each processed image followed by their similarity score is printed on screen. The system successfully re-identifies the person in the scene and says: "Welcome back, User 0. Great to see you again!".

The *sr0* file is actually the reference image we previously mentioned in Subsection 6.4.4 and is the one we use to measure similarity between the existing images from the database. The re-identification stage is quite accurate. Even if the Side Left partial scan of User 1 is not quite as good as its Front Side Right capture in Figure 6.15, the algorithm is able to find those similarity features in the image and find the correct match for each user. Therewith, scores provide valuable information which can certainly generate satisfactory evaluations.

On the other hand, we have noticed the RGB-D sensor is highly sensitive. The range intervals sometimes fail on account of fast motion or rapid change in position from the individuals, capturing and misclassifying their partial scans (see Figure 6.16). Moreover, sensor positioning also influences the accuracy of pose estimation: if the sensor was at a certain distance and height on a previous experiment, it is the best to place the sensor in the same way for the next experiment, otherwise, it is very likely to obtain a totally different performance.

Re-identification from users with incomplete 3D PSMs was also very effective. Figure 6.17 shows how the system detects the user is facing away from Kinect, goes through the

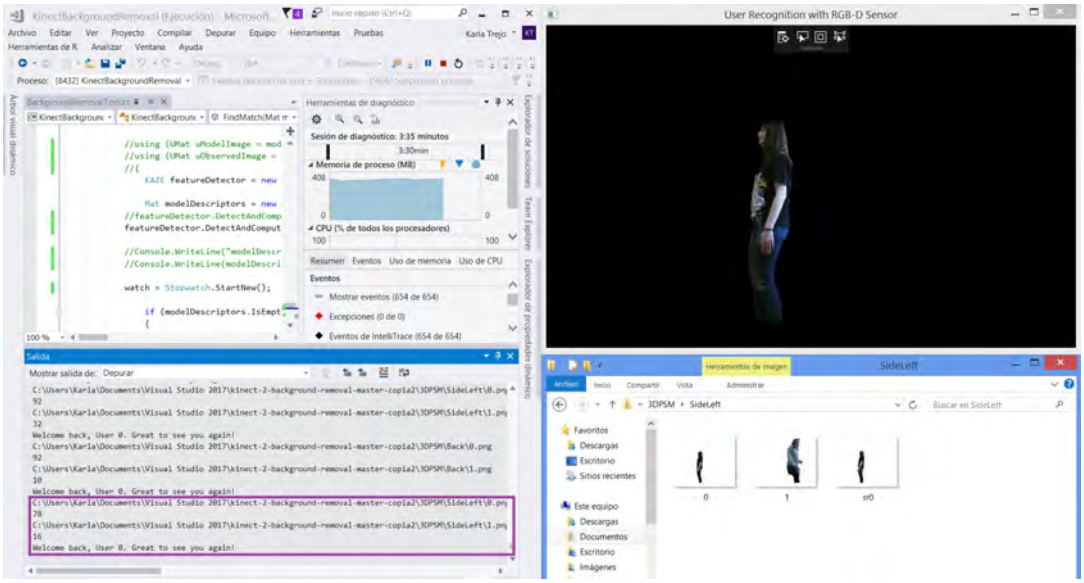


FIGURE 6.14: Re-identification of User 0. Upper-left: Diagnostics Tools providing elapsed time and memory consumption. Upper-right: Real-time streaming. Lower-left: Console output. Lower-right: 3D PSMs database, SideLeft directory.

Back directory of the 3D PSMs image database and analyzes the existing images –from User 0 and User 1– but it finds an image with higher similarity in the complementary view, finally retrieving a "Welcome back, User 3. Great to see you again!".

The upper-left corner of Figure 6.17 demonstrates the veracity of these results, since user ID information from the output message –provided by the system to address and greet the re-identified user– is extracted directly from the filename of the partial scan with the highest similarity.

To quantitatively sustain the previous statements, a two-day experiment has been conducted with three users: User A, User K and User Y. All possible permutations and configurations in the users' order of appearance are explored. As such, the percentage of accuracy for each experimental run is depicted in Table 6.2.

The algorithm has an average accuracy of 63% when computed on the total frame amount rather than giving the same importance to runs of different length. In terms of average processing time, the system performs at 242 miliseconds per frame.

We have noticed that incorrect user re-identification also relies on *faulty saves*, since the ratio of saved images for a user appears to decay with time. Thus, it is suspected that a memory overload is occurring. Parallel computing may optimize this task and boost the algorithm's general accuracy.

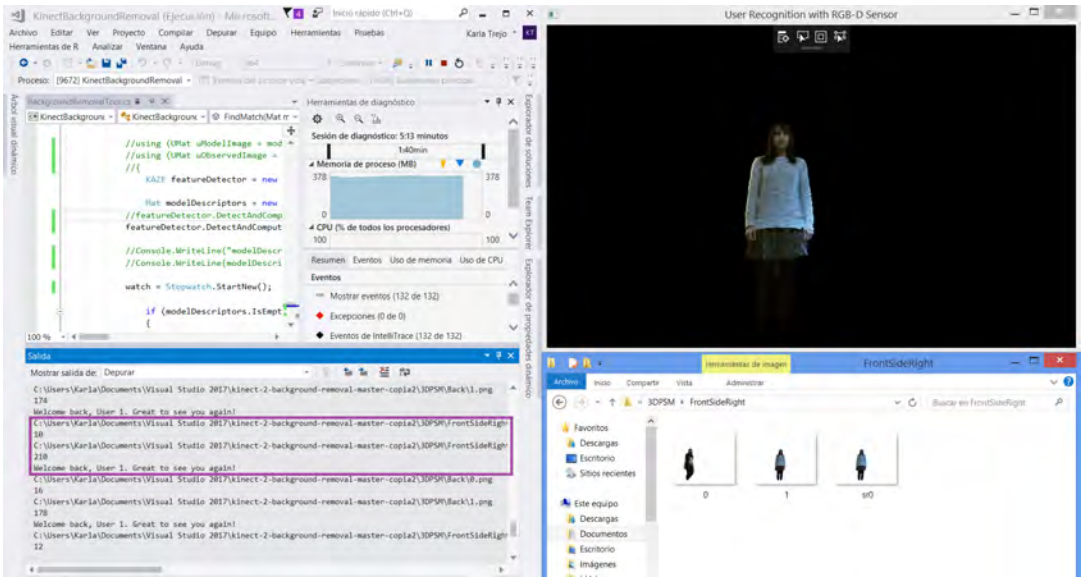


FIGURE 6.15: Re-identification of User 1. Even though the purple rectangle shows results corresponding to the FrontSideRight directory, positive results also obtained on the Back partial scan from User 1 can be seen in the output feed.

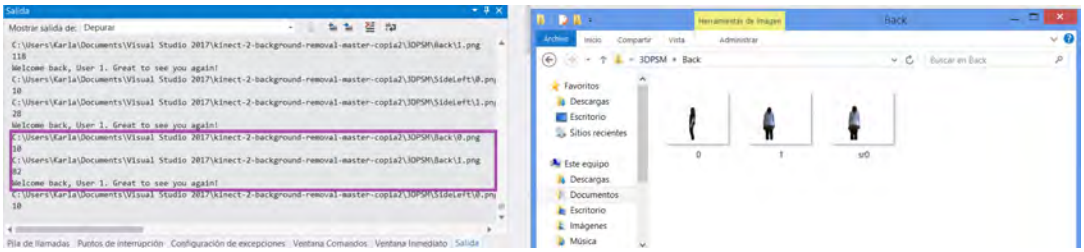


FIGURE 6.16: Re-identification of User 1. Left: Console output. Right: 3D PSMs database, Back directory. User 1 is successfully re-identified. However, partial scan from User 0 seems to belong to a Side Left view, rather than to a Back view.

6.6 Conclusions

A new on-line approach for people recognition in identification and re-identification tasks with a RGB-D sensor was introduced in this chapter. It is a proposal that extols the virtues of 3D information and 2D processing, by handling highly-accurate data in real-time with less computationally intensive procedures.

Earlier works also reflected the need of designing NUIs actually obeying and following the natural interaction concepts and principles, which was an important fact to explore and fulfill here, since it is a steady way to foster and boost human-machine interaction. Considering the free interaction guidelines we adopted and the results that have been obtained under our experimental framework, *naturalness* is achieved.

Kinect's sensitivity to positioning (view-angle, height and distance) is an issue that could be solved in a real-scenario implementation. We should aim to create an smart

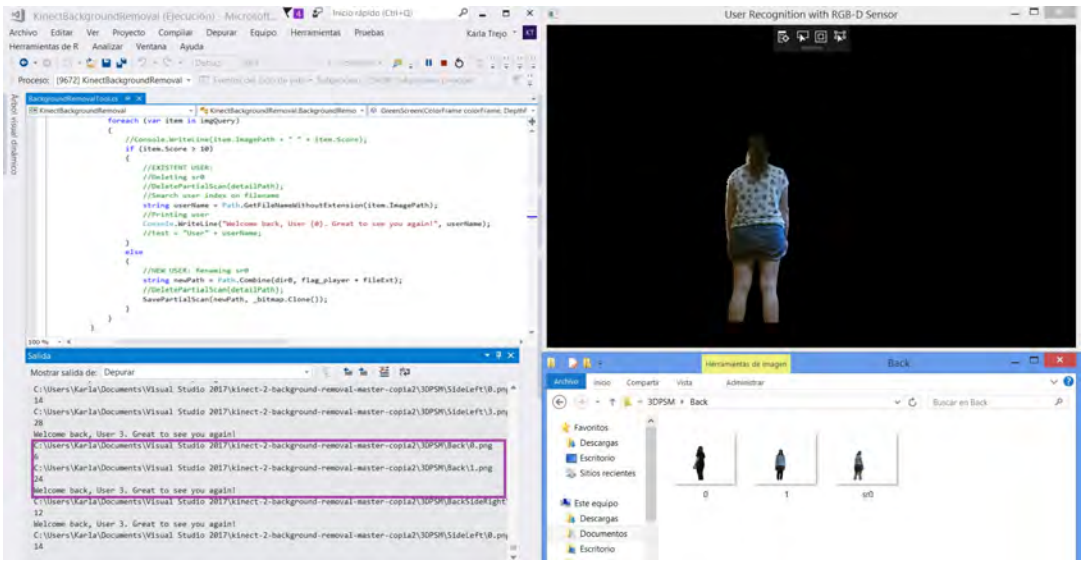


FIGURE 6.17: Re-identification of User 3. Successful and continuous user recognition in real-time despite the missing partial scans.

TABLE 6.2: 3D Partial Scans Models - Accuracy test

Day	Configuration	Frames	User Correct	User Incorrect	Accuracy (%)
1	AYK	794	529	275	67
	YAK	248	133	115	54
	YKA	287	197	90	69
2	KAY	669	397	323	60
	AKY	488	322	166	66
	KYA	598	353	245	60

environment where the RGB-D sensor can be embedded at a fixed and optimal position within the identification area of an ambient intelligence setting, or, endowed in a robotic companion that securely carries the sensor on its "head" (the higher the better).

Kinect for Windows v2 is a suitable device with which a developer or researcher can breach the world of AmI, robotics and create NUI solutions at a reasonable price. Kinect SDK contains truly powerful APIs, however, it is badly documented too. Insufficient and outdated documentation makes it hard to understand what is going on inside the APIs. It took a long time for us to properly interpret some data structures, protocols and object classes, which inevitably started with a trial and error process. However, nearing the end of this journey, Rahman published an exhaustive book about Kinect for Windows SDK 2.0 [104] that, even though it arrived very late, it became a valuable support to confirm everything we have learned so far.

As a final remark, the proposed system has yet to reach its full potential. Experiments conducted in this chapter constructed a database of up to four registered users, it would be interesting to include more individuals in future tests, albeit taking account of the memory consumption limitations, as we are still talking about processing in real-time. This also applies to processing more than one user in the scene and exploit Kinect's voice recognition capabilities, carrying it further towards the development of a more visible bond with the user.

Chapter 7

Conclusions and Future Work

This chapter discusses about the experimental results obtained as a whole, subject to the achievement of the thesis original objectives, and the contributions through the research work. Moreover, it also unveils some personal issues from the journey and the current perception we have obtained on people recognition after studying the problem from different points of view, whilst remaining true to the spirit of natural interaction and the future of humans and artificial intelligence. Final comments on the possible future work in this field according to the current stage and consistent with our technological reality is also addressed.

7.1 Conclusions

The research conducted in this thesis has made great strides in developing, improving and –in some extent– put into practice ideas for people recognition with more compact, direct and simple approaches that could become useful in real applications for security, surveillance and robotics. An ensemble of ideas with a common goal: challenging traditional standards of what it seems to be more robust and highly accurate instances due to their level of complexity both in number of resources and algorithmic structure.

From the beginning, our standpoint was not to compete on these terms but rather to provide solutions with a certain degree of novelty and a particular niche of applications, enabling systems to work in environments with very limited means. Quantitative comparisons in performance and accuracy with state-of-the-art methods would only bring us fairly obvious results, since we are deliberately putting ourselves in a technological disadvantage to achieve these goals.

Methodologies presented here forge new paths which will undoubtedly develop progressively in the future. They were not envisaged initially but there were always there, just waiting for the opportunity to be studied and which now, through experimentation, we have proven to perform at a sufficient speed and accuracy that could satisfy certain recognition activities, where sometimes "less is more".

The future of intelligent environments can benefit from robust and integrated person re-identification, since they could offer enhanced levels of personalized services via

natural interaction. However, the limited accuracy on person re-identification remains a significant obstacle in the realization of these kind of environments. Most previously published work in re-identification has been performed with the use of 2D cameras, and with the aim of reducing the search time of a target from the hundreds of people in a gallery set. Only recent work has benefited from the use of 3D data and information from RGB-D cameras, and only very recently higher resolution depth-sensing cameras, such as the Kinect v2, became easily available.

Our research deals with this issue by using an RGB-D sensor to bring accuracy to pose estimation and tackle the problem with minimum computational resources. It takes 3D information to the 2D terrain and builds an image processing stage with traditional feature descriptors, facilitating the system maintenance and applicability. It was a challenging attempt, since Kinect v2 sensor was poorly documented at the time and we had to keep track of Q&A from developers at Microsoft's forums and follow the tutorials from dedicated professionals contributing to the technical community. Just a few months ago, a beginner's guide book was properly released, revealing some features unknown to us and that could optimize the results of our recent experiments. Unfortunately, this book came far too late as we were closing that chapter in our research.

Despite not producing the desired accuracy in our re-identification approach, it is a people recognition problem that will be in trend for a long time, since research in this field is just getting started. It brings new possibilities, applications and challenges, like the rest of our proposals in this dissertation: a single-camera automatic landmarking approach for body shapes, a 3D body contour tracking for on-line people detection tasks and group categorization with leader tracking in public spaces.

It is difficult to compare some of our results with respect to other works of research, since the datasets we employ are quite new and still unexplored by the rest of the community. However, the quality of the results suggest it would be interesting for our colleagues to evaluate them against other methodologies in the near future.

To visualize all the results obtained from the work in this dissertation, the reader can access our Youtube channel[1] and make free use of the source codes on Github[2]. There are many opportunities for further research in the area of people recognition. The technological revolution is already happening, which makes it an imminent milestone to accomplish interactive environments that work robustly and reliably in the real world to deliver personalized services, as it still poses many interesting challenges.

7.1.1 Objectives

The general objective has been achieved with the creation of *3D Partial Scans Models*, a NUI system which identifies and re-identifies people interacting in the environment. By

[1] https://www.youtube.com/channel/UCCLRO9LyWkaL_0G9fpH2j-A
[2] https://github.com/karla3jo

using joint orientation information retrieved in real-time from a RGB-D sensor and later processed as 2D pose images, we were able to find a similarity score that compares body shape and appearance between all the users collected in the database. The similarity score determines if the person detected on scene is a known or a new user, recovering or assigning an ID correspondingly.

We were also able to fulfill the specific objectives at different scales: First, by delivering practical and effective solutions with less resources, boosting them with either state-of-the-art approaches or the latest technology available, tackling new problems and addressing them from a different perspective in the people recognition field of study. Second, by promoting the transfer of knowledge within the scientific community in the form of open-source code, giving greater visibility to the projects through conferences, publications, networking events, public videos and free access to developers. And third, by exposing the particular importance of human body shapes, which possess a considerable and hitherto untapped potential that empowers the natural interaction between smart agents and humans. Human body shapes has won its spurs as a remarkably useful, indeed essential, target to the vast field of people recognition as we know it today.

7.1.2 Contributions and Findings

The main contribution of this thesis is the proposed 3D Partial Scan Models (3D PSMs) technique for person re-identification described in Chapter 6. 3D PSMs in itself makes new contributions to the re-identification field, which include: (i) providing a direct method to estimate body orientation with one Kinect RGB-D sensor and the usage of this information to enable person re-identification with an open-source feature descriptor, (ii) the implementation of a NUI system that follows the basics towards a more natural interaction in intelligent spaces, fostering a sense of freedom form the users actively or passively involved. Nevertheless, other several contributions were derived in the process to reach this solution.

Chapter 3 holds the extrapolation of a face alignment algorithm into body shapes. To prove the feasibility of this hypothesis was a great endeavor, since body shape datasets were limited and scarce and we had to create our own dataset. However, this drawback was actually the perfect opportunity to demonstrate we could obtain satisfactory results despite having a small dataset for training. It is wake-up call to research in face recognition. There is absolutely no reason why body shapes can not be targeted more often, as they are a great source of relevant information that could support and improve the accuracy of many applications that depend solely in facial recognition.

Conditional Contour Detector (CCD) kernel presented in Chapter 5 is a minor contribution that, without it, it would not have been possible to realize a major problem we were facing at the time. CCD kernel's development helped us to better understand the issues related to a 3D AAM approach in re-identification and how they did not match with

our specific objectives. An interesting turn of events that allowed us to redirect the real-world implementation of 3D contour tracking under this scope, but more importantly, the original vision and nuance we wanted to deliver on the people recognition area.

Chapter 4 contains fascinating discoveries on recognition of groups of people: by establishing social relations from spatial interaction and motion logic, individuals are categorized based on their group role, which is directly related to the leader figure and the human concept of what it means to be in this position. A behavioural analysis which is based on the way we as humans think and cognitively perceive the associations and dis-associations between a group of observed people, and that we could convey into social robots as an intelligence paradigm.

7.2 Future Work

On 25 October 2017, Alex Kipman (creator of the Kinect) and Matthew Lapsen (XBOX Marketing) announced that Microsoft stopped manufacturing the Kinect sensor.

Kinect for Windows v2 is not going to end right away. Hardware does not just disappear. Even Kinect for Windows v1 is still available, four years after it was replaced by Kinect v2 and one year after it was discontinued. There are still numerous and different Kinect projects out there, dedicated to a variety of industries: healthcare, fitness, retail, advertising and gaming.

Kinect is no longer a moonshot technology, it is a *mature* technology. Turns out, a lot of people are still using Kinect to create motion applications for exhibition booths, events, or shows. Indeed, Kinect is still alive in this niche. Both Kinect sensors are still widely used in enterprise, even when the current crop of IoT and AR/VR systems are discontinued so that companies can sell marginally higher-spec hardware.

On the other hand, the developer community is also very active and new companies have emerged with several alternatives to Kinect. One of the most interesting projects that has drawn much of our attention is OpenPose. OpenPose is currently being developed by the Perception Computing Lab of Carnegie Mellon University and it is not, actually, a device. OpenPose uses any plain webcam to track the human body, face, and fingers with remarkable accuracy. We truly believe it is going to dominate the market within the next few years.

Right now, OpenPose is not market-ready, though. It requires a high-end PC to run and has been reported to be quite tricky to setup, since it lacks support for popular platforms. However, this open-source software will be the ideal successor in our opinion, more than suitable to keep working in people recognition tasks and further explore the topics discussed in this dissertation, considering its current state for R&D projects only.

Bibliography

[1] M. Agrawal, K. Konolige, and M. R. Blas. "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching". In: *Computer Vision – ECCV 2008*. Ed. by D. Forsyth, P. Torr, and A. Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 102–115.

[2] W. Aitfares et al. "Hybrid region and interest points-based active contour for object tracking". In: *Applied Mathematical Sciences* 7.117-120 (2013), pp. 5879–5899.

[3] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. "KAZE Features". In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*. ECCV'12. Florence, Italy: Springer-Verlag, 2012, pp. 214–227. ISBN: 978-3-642-33782-6.

[4] M-S. Allili and D. Ziou. "Active contours for video object tracking using region, boundary and shape information". In: *Signal, Image and Video Processing* 1.2 (June 2007), pp. 101–117. ISSN: 1863-1711.

[5] M. Andriluka, S. Roth, and B. Schiele. "People-tracking-by-detection and people-detection-by-tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2008*. June 2008, pp. 1–8.

[6] C. Angulo et al. "Evaluating the use of robots to enlarge AAL services". In: *Journal of Ambient Intelligence and Smart Environments* 7.3 (2015), pp. 301–313.

[7] A. R. Araujo, D. D. Caminhas, and G.A.S. Pereira. "An Architecture for Navigation of Service Robots in Human-Populated Office-like Environments". In: *IFAC-PapersOnLine* 48.19 (2015). 11th IFAC Symposium on Robot Control SYROCO 2015, pp. 189–194. ISSN: 2405-8963.

[8] N. Ayache and Olivier D. Faugeras. "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.1 (Jan. 1986), pp. 44–54.

[9] R. Bajcsy and S. Kovačič. "Multiresolution Elastic Matching". In: *Comput. Vision Graph. Image Process.* 46.1 (Apr. 1989), pp. 1–21. ISSN: 0734-189X.

[10] S. Baker, I. Matthews, and J. Schneider. "Automatic Construction of Active Appearance Models as an Image Coding Problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004), pp. 1380–1384.

[11] I. B. Barbosa et al. “Re-identification with RGB-D Sensors”. In: *Computer Vision – ECCV 2012. Workshops and Demonstrations*. Ed. by A. Fusiello, V. Murino, and R. Cucchiara. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 433–442. ISBN: 978-3-642-33863-2.

[12] H. Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142.

[13] A. Bedagkar-Gala and S. K. Shah. “A survey of approaches and trends in person re-identification”. In: *Image and Vision Computing* 32.4 (2014), pp. 270–286. ISSN: 0262-8856.

[14] S. Belongie, J. Malik, and J. Puzicha. “Shape matching and object recognition using shape contexts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.4 (Apr. 2002), pp. 509–522.

[15] T. O. Binford. “Visual Perception by Computer”. In: *IEEE Conference on Systems and Control*. 1971.

[16] M. J. Black, D. J. Fleet, and Y. Yacoob. “A framework for modeling appearance change in image sequences”. In: *IEEE International Conference on Computer Vision, 1998*. Jan. 1998, pp. 660–667.

[17] M. J. Black and Y. Yacoob. “Recognizing Facial Expressions in Image Sequences Using Local Parameterized Models of Image Motion”. In: *International Journal of Computer Vision* 25.1 (Oct. 1997), pp. 23–48. ISSN: 1573-1405.

[18] M. Blank et al. “Actions as space-time shapes”. In: *IEEE International Conference on Computer Vision, 2005*. Vol. 2. Oct. 2005, pp. 1395–1402.

[19] V. Blanz and T. Vetter. “Face recognition based on fitting a 3D Morphable Model”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.9 (Sept. 2003), pp. 1063–1074.

[20] E. Bondi et al. “Long Term Person Re-identification from Depth Cameras Using Facial and Skeleton Data”. In: *Understanding Human Activities Through 3D Sensors*. Ed. by H. Wannous et al. Cham: Springer International Publishing, 2018, pp. 29–41. ISBN: 978-3-319-91863-1.

[21] W. Brendel, A. Fern, and S. Todorovic. “Probabilistic event logic for interval-based event recognition”. In: *CVPR 2011*. June 2011, pp. 3329–3336.

[22] W. Burgard et al. “Experiences with an Interactive Museum Tour-guide Robot”. In: *Artif. Intell.* 114.1-2 (Oct. 1999), pp. 3–55. ISSN: 0004-3702.

[23] W. Burgard et al. "The Interactive Museum Tour-guide Robot". In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. AAAI '98/IAAI '98. Madison, Wisconsin, USA: American Association for Artificial Intelligence, 1998, pp. 11–18. ISBN: 0-262-51098-7.

[24] C. W. Chen and C. C. Wang. "3D active appearance model for aligning faces in 2D images". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2008, pp. 3133–3139.

[25] G.E. Christensen et al. "Topological Properties of Smooth Anatomic Maps". In: *Information Processing in Medical Imaging* (1995), pp. 101–112.

[26] R.T. Collins et al. "A system for video surveillance and monitoring: VSAM final report". In: *The Robotics Institute, Carnegie Mellon University* (2000).

[27] T. F. Cootes, G. J. Edwards, and C. J. Taylor. "Active Appearance Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.6 (June 2001), pp. 681–685.

[28] T. F. Cootes and C. J. Taylor. "Modelling Object Appearance Using The Grey-Level Surface". In: *5th British Machine Vision Conference*. Ed. by E. Hancock. BMVA Press, 1994, pp. 479–488.

[29] T.F. Cootes et al. "Active Shape Models-Their Training and Application". In: *Computer Vision and Image Understanding* 61.1 (1995), pp. 38–59. ISSN: 1077-3142.

[30] M. Covell. "Eigen-points: control-point location using principal component analyses". In: *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. Oct. 1996, pp. 122–127.

[31] F. Cupillard, F. Bremond, and M. Thonnat. "Tracking Groups of People for Video Surveillance". In: *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*. Ed. by P. Remagnino et al. Boston, MA: Springer US, 2002, pp. 89–100. ISBN: 978-1-4615-0913-4.

[32] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for human detection". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2005*. Vol. 1. June 2005, pp. 886–893.

[33] M. Danelljan et al. "Accurate Scale Estimation for Robust Visual Tracking". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[34] M. Dantone et al. "Human Pose Estimation Using Body Parts Dependent Joint Regressors". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2013*. June 2013, pp. 3041–3048.

[35] M. Dantone et al. "Real-time facial feature detection using conditional regression forests". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2012*. June 2012, pp. 2578–2585.

[36] M. Díaz-Boladeras et al. "Evaluating Group-Robot Interaction in Crowded Public Spaces: A Week-Long Exploratory Study in the Wild with a Humanoid Robot Guiding Visitors Through a Science Museum". In: *International Journal of Humanoid Robotics* 12.04 (2015), p. 1550022.

[37] M. Dimitrijevic, V. Lepetit, and P. Fua. "Human body pose detection using Bayesian spatio-temporal templates". In: *Computer Vision and Image Understanding* 104 (2006), pp. 127–139.

[38] A. Dopfer, H.H. Wang, and C.C. Wang. "3D Active Appearance Model alignment using intensity and range data". In: *Robotics and Autonomous Systems* 62.2 (Feb. 2014), pp. 168–176.

[39] G. J. Edwards, T. F. Cootes, and C. J. Taylor. "Face recognition using Active Appearance Models". In: *European Conference on Computer Vision, 1998*. Vol. 1407. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 581–595.

[40] G. J. Edwards, C. J. Taylor, and T. F. Cootes. "Learning to identify and track faces in image sequences". In: *IEEE International Conference on Automatic Face and Gesture Recognition, 1998*. Apr. 1998, pp. 260–265.

[41] G. J. Edwards et al. "Statistical Models of Face Images - Improving Specificity". In: *British Machine Vision Conference*. 1996, pp. 765–774.

[42] S. Escalera. "Human Behavior Analysis from Depth Maps". In: *Articulated Motion and Deformable Objects*. Ed. by F. J. Perales, R. B. Fisher, and T. B. Moeslund. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 282–292. ISBN: 978-3-642-31567-1.

[43] T. Ezzat and T. Poggio. "Facial analysis and synthesis using image-based models". In: *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. Oct. 1996, pp. 116–121.

[44] G. Fanelli, M. Dantone, and L. Van Gool. "Real time 3D face alignment with Random Forests-based Active Appearance Models". In: *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, 2013*. Vol. 0. 2013, pp. 1–8.

[45] G. Farnebäck. "Two-frame Motion Estimation Based on Polynomial Expansion". In: *Proceedings of the 13th Scandinavian Conference on Image Analysis*. SCIA'03. Halmstad, Sweden: Springer-Verlag, 2003, pp. 363–370. ISBN: 3-540-40601-8.

[46] A. Fossati et al. "Bridging the Gap between Detection and Tracking for 3D Monocular Video-Based Motion Capture". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2007*. June 2007, pp. 1–8.

[47] O. Freifeld et al. "Contour people: A parameterized model of 2D articulated human shape". In: *IEEE Conf. on Computer Vision and Pattern Recognition, (CVPR)*. IEEE, June 2010, pp. 639–646.

[48] A. Garrell and A. Sanfeliu. "Cooperative social robots to accompany groups of people". In: *The International Journal of Robotics Research* 31.13 (2012), pp. 1675–1701.

[49] M. Ghosh and H. Kuzuoka. "An Ethnomethodological Study of a Museum Guide Robot's Attempt at Engagement and Disengagement". In: *Journal of Robotics* (2014).

[50] D. Gowsikhaa, S. Abirami, and R. Baskaran. "Automated human behavior analysis from surveillance videos: a survey". In: *Artificial Intelligence Review* 42.4 (Dec. 2014), pp. 747–765. ISSN: 1573-7462.

[51] E. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. Cambridge, MA, USA: MIT Press, 1990.

[52] E. Hall. "Handbook of Proxemics Research". In: *Society for the Anthropology of Visual Communications* (1974).

[53] J. Hanbyul, S. Tomas, and S. Yaser. "Total Capture: A 3D Deformation Model for Tracking Faces, Hands, and Bodies". In: *CoRR* abs/1801.01615 (2018).

[54] D. D. Hoffman and W. Richards. "Parts Of Recognition". In: *Cognition* 18 (1983), pp. 65–96.

[55] E. Hunt. "Tay, Microsoft's AI chatbot, gets a crash course in racism from Twitter". In: *The Guardian* (Mar. 2016).

[56] I.T. Jolliffe. *Principal component analysis*. Springer series in statistics. Springer-Verlang, 1986. ISBN: 9780387962696.

[57] M. J. Jones and T. Poggio. "Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes". In: *International Journal of Computer Vision* 29.2 (Aug. 1998), pp. 107–131. ISSN: 1573-1405.

[58] M-H. Ju and H-B. Kang. "A fast 3D-AAM method using the estimated depth information". In: *2010 IEEE International Conference on Progress in Informatics and Computing*. Vol. 2. Dec. 2010, pp. 941–945.

[59] V. Kazemi and J. Sullivan. "One Millisecond Face Alignment with an Ensemble of Regression Trees". In: *IEEE Conference on Computer Vision and Pattern Recognition, 2014*. June 2014, pp. 1867–1874.

[60] H. Kobayashi and F. Hara. "A Basic Study of Dynamic Recognition of Human Facial Expressions". In: *IEEE International Workshop on Robot and Human Communication*. Vol. 11. 1993, pp. 271–275.

[61] K. Koide et al. "People tracking and re-identification by face recognition for RGB-D camera networks". In: *2017 European Conference on Mobile Robots (ECMR)*. Sept. 2017, pp. 1–7.

[62] M. Kuderer, H. Kretzschmar, and W. Burgard. "Teaching mobile robots to cooperatively navigate in populated environments". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2013, pp. 3138–3143.

[63] Y. Kuno et al. "Museum Guide Robot Based on Sociological Interaction Analysis". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, pp. 1191–1194. ISBN: 978-1-59593-593-9.

[64] Y. Kuno et al. "Museum Guide Robot with Communicative Head Motion". In: *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*. Sept. 2006, pp. 33–38.

[65] S. Kwak, B. Han, and J. H. Han. "Multi-agent Event Detection: Localization and Role Assignment". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. June 2013, pp. 2682–2689.

[66] S. Kwak, B. Han, and J. H. Han. "Scenario-based video event recognition by constraint flow". In: *CVPR 2011*. June 2011, pp. 3345–3352.

[67] M. Lades et al. "Distortion invariant object recognition in the dynamic link architecture". In: *IEEE Transactions on Computers* 42.3 (Mar. 1993), pp. 300–311. ISSN: 0018-9340.

[68] T. Lan, L. Sigal, and G. Mori. "Social roles in hierarchical models for human activity recognition". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 1354–1361.

[69] A. Lanitis, C. J. Taylor, and T. F. Cootes. "Automatic interpretation and coding of face images using flexible models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (July 1997), pp. 743–756. ISSN: 0162-8828.

[70] B. Lau, K. O. Arras, and W. Burgard. "Multi-model Hypothesis Group Tracking and Group Size Estimation". In: *International Journal of Social Robotics* 2.1 (Mar. 2010), pp. 19–30. ISSN: 1875-4805.

[71] B. Lau, K. O. Arras, and W. Burgard. "Tracking groups of people with a multi-model hypothesis tracker". In: *2009 IEEE International Conference on Robotics and Automation*. May 2009, pp. 3180–3185.

[72] B. Laxton, J. Lim, and D. Kriegman. "Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. June 2007, pp. 1–8.

[73] T. Linder and K. O. Arras. "Multi-model hypothesis tracking of groups of people in RGB-D data". In: *17th International Conference on Information Fusion (FUSION)*. July 2014, pp. 1–7.

[74] H. Liu, L. Hu, and L. Ma. "Online RGB-D person re-identification based on metric model update". In: *CAAI Transactions on Intelligence Technology* 2.1 (2017), pp. 48–55. ISSN: 2468-2322.

[75] M. Loper et al. "SMPL: A Skinned Multi-person Linear Model". In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 248:1–248:16. ISSN: 0730-0301.

[76] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405.

[77] D. G. Lowe. "Object recognition from local scale-invariant features". In: *IEEE International Conference on Computer Vision, 1999*. Vol. 2. 1999, pp. 1150–1157.

[78] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[79] K. Marino. "Real Time Human Pose Estimation for Boosted Random Forests and Pose Machines". In: *Robotics Institute Summer Scholars (RISS) Working Papers*. Vol. 2. Carnegie Mellon University, 2014, pp. 45–49.

[80] F. Martí Carrillo et al. ""Help Me Help You": A Human-assisted Social Robot in Pediatric Rehabilitation". In: *Proceedings of the 28th Australian Conference on Computer-Human Interaction*. OzCHI '16. Launceston, Tasmania, Australia: ACM, 2016, pp. 659–661. ISBN: 978-1-4503-4618-4.

[81] I. Matthews and S. Baker. "Active Appearance Models Revisited". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 135–164. ISSN: 1573-1405.

[82] S.J. McKenna et al. "Tracking Groups of People". In: *Computer Vision and Image Understanding* 80.1 (2000), pp. 42–56. ISSN: 1077-3142.

[83] C. Migniot, P. Bertolino, and J-M. Chassery. "Iterative Human Segmentation from Detection Windows Using Contour Segment Analysis". In: *VISAPP 2013 - 8th International Conference on Computer Vision Theory and Applications*. Barcelone, Spain, Feb. 2013.

[84] D. Minnen, I. Essa, and T. Starner. "Expectation grammars: leveraging high-level expectations for activity recognition". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. June 2003, pp. II–II.

[85] T. Moeslund, A. Hilton, and V. Krüger. "A Survey of Advances in Vision-based Human Motion Capture and Analysis". In: *Comput. Vis. Image Underst.* 104.2 (Nov. 2006), pp. 90–126. ISSN: 1077-3142.

[86] A. Møgelmose et al. "Tri-modal Person Re-identification with RGB, Depth and Thermal Features". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. June 2013, pp. 301–307.

[87] V. I. Morariu and L. S. Davis. "Multi-agent event recognition in structured scenarios". In: *CVPR 2011*. June 2011, pp. 3289–3296.

[88] R. Moreira, A. Magalhães, and H. P. Oliveira. "A Kinect-Based System for Upper-Body Function Assessment in Breast Cancer Patients". In: *Journal of Imaging* 1.1 (2015), pp. 134–155. ISSN: 2313-433X.

[89] G. Mori and J. Malik. "Estimating Human Body Configurations using Shape Context Matching". In: *European Conference on Computer Vision. LNCS 2352*. Vol. 3. 2002, pp. 666–680.

[90] M. Moussaïd et al. "The Walking Behaviour of Pedestrian Social Groups and Its Impact on Crowd Dynamics". In: *PLoS ONE* 5.4 (2010).

[91] M. Muja and D.G. Lowe. "Scalable Nearest Neighbor Algorithms for High Dimensional Data". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36 (2014).

[92] C. Nastar, B. Moghaddam, and A. Pentland. "Generalized image matching: Statistical learning of physically-based deformations". In: *Computer Vision — ECCV '96*. Ed. by B. Buxton and R. Cipolla. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 589–598. ISBN: 978-3-540-49949-7.

[93] S. K. Nayar, H. Murase, and S. A. Nene. "Parametric Appearance Representation". In: *Early Visual Learning*. 1996, pp. 131–160.

[94] A. Oikonomopoulos, M. Pantic, and I. Patras. "Sparse B-spline polynomial descriptors for human activity recognition". In: *Image and vision computing* 27.12 (Dec. 2009), pp. 1814–1825.

[95] D. Paillacho, C. Angulo, and M. Díaz. "An exploratory study of group-robot social interactions in a cultural center". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS 2015 Workshop on Designing and Evaluating Social Robots for Public Settings*. 2015, pp. 44–48.

[96] F. Pala et al. "Multimodal Person Reidentification Using RGB-D Cameras". In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.4 (Apr. 2016), pp. 788–799. ISSN: 1051-8215.

[97] X. Perez-Sala et al. "A Survey on Model Based Approaches for 2D and 3D Visual Human Pose Recovery". In: *Sensors* 14.3 (2014), pp. 4189–4210. ISSN: 1424-8220.

[98] P. Perona and Jitendra M. "Scale-space and edge detection using anisotropic diffusion". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (1990), pp. 629–639.

[99] R. Plaenkers and P. Fua. "Model-Based Silhouette Extraction for Accurate People Tracking". In: *Computer Vision — ECCV 2002*. Ed. by Anders Heyden et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 325–339. ISBN: 978-3-540-47967-3.

[100] E. Pot et al. "Person re-identification visualization tool for object tracking across non-overlapping cameras". In: *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Aug. 2015, pp. 1–5.

[101] U. Prabhu, K. Seshadri, and M. Savvides. "Automatic Facial Landmark Tracking in Video Sequences Using Kalman Filter Assisted Active Shape Models". In: *Trends and Topics in Computer Vision*. Vol. 6553. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 86–99.

[102] V. Pterneas. *Background removal using Kinect2 (green screen effect)*. 2014. URL: `http://pterneas.com/2014/04/11/kinectbackground-removal/`.

[103] V. Pterneas. *Understanding Kinect Coordinate Mapping*. 2014. URL: `http://pterneas.com/2014/05/06/understanding-kinect-coordinate-mapping/`.

[104] M. Rahman. *Beginning Microsoft Kinect for Windows SDK 2.0: Motion and Depth Sensing for Natural User Interfaces*. 1st. Berkeley, CA, USA: Apress, 2017. ISBN: 9781484223154.

[105] V. Ramakrishna et al. "Pose Machines: Articulated Pose Estimation via Inference Machines". In: *European Conference on Computer Vision*. July 2014.

[106] D. Ramanan, D. A. Forsyth, and A. Zisserman. "Tracking People by Learning Their Appearance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.1 (Jan. 2007), pp. 65–81.

[107] L. Ren et al. "Multi-modal Uniform Deep Learning for RGB-D Person Re-identification". In: *Pattern Recogn.* 72.C (Dec. 2017), pp. 446–457. ISSN: 0031-3203.

[108] G. Riva, F. Vatalaro, and F. Davide. "Ambient Intelligence: from Vision to Reality". In: *IST Advisory Group (ISTAG)*. IOS Press, Mar. 2005.

[109] M. Rodriguez and M. Shah. "Detecting and segmenting humans in crowded scenes". In: *Proceedings of the 15th ACM international conference on Multimedia, 2007*. 2007, pp. 353–356.

[110] M-C. Roh et al. "Accurate object contour tracking based on boundary edge selection". In: *Pattern Recognition*. Vol. 40. 3. 2007, pp. 931–943.

[111] A. Ross. *Procrustes Analysis*. Tech. rep. Department of Computer Science and Engineering, University of South Carolina, SC 29208, 2004.

[112] M. S. Ryoo and J. K. Aggarwal. "Semantic Representation and Recognition of Continued and Recursive Human Activities". In: *Int. J. Comput. Vision* 82.1 (Apr. 2009), pp. 1–24. ISSN: 0920-5691.

[113] M. S. Ryoo and J. K. Aggarwal. "Stochastic representation and recognition of high-level group activities: Describing structural uncertainties in human activities". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. June 2009, pp. 11–11.

[114] S. Saxena et al. "Crowd Behavior Recognition for Video Surveillance". In: *Advanced Concepts for Intelligent Vision Systems*. Ed. by J. Blanc-Talon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 970–981. ISBN: 978-3-540-88458-3.

[115] J. Shi and C. Tomasi. "Good features to track". In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. June 1994, pp. 593–600.

[116] M. W. Tao et al. "SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm". In: *Computer Graphics Forum (Eurographics 2012)* 31.2 (May 2012).

[117] S. Thrun et al. "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva". In: *The International Journal of Robotics Research* 19.11 (2000), pp. 972–999.

[118] M. A. Turk and A. P. Pentland. "Face recognition using eigenfaces". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 1991, pp. 586–591.

[119] M. Turk and A. Pentland. "Eigenfaces for Recognition". In: *Cognitive Neuroscience* 3.1 (Jan. 1991), pp. 71–86.

[120] A. Valli. "Natural interaction". In: *White Paper* (2007).

[121] C. Verstraeten. *OpenCV Simple Motion Detection*. Kerberos.io: Open source video surveillance and motion detection. 2014.

[122] H. H. Wang, A. Dopfer, and C. C. Wang. "3D AAM based face alignment under wide angular variations using 2D and 3D data". In: *2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 4450–4455.

[123] J. Weickert. "Efficient image segmentation using partial differential equations and morphology". In: *Pattern Recognition* 34.9 (2001), pp. 1813–1824.

[124] J. Weickert, B.M.T.H. Romeny, and M. A. Viergever. *Efficient and Reliable Schemes for Nonlinear Diffusion Filtering*. 1998.

[125] P. Weinzaepfel et al. "DeepFlow: Large Displacement Optical Flow with Deep Matching". In: *2013 IEEE International Conference on Computer Vision*. Dec. 2013, pp. 1385–1392.

[126] K. Woolford. "Defining Accuracy in the Use of Kinect V2 for Exercise Monitoring". In: *Proceedings of the 2Nd International Workshop on Movement and Computing*. MOCO '15. Vancouver, British Columbia, Canada: ACM, 2015, pp. 112–119. ISBN: 978-1-4503-3457-0.

[127] A. Wu, W-S. Zheng, and J. Lai. "Robust Depth-based Person Re-identification". In: *CoRR* abs/1703.09474 (2017).

[128] J. Xiao et al. "Real-time combined 2D+3D active appearance models". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. June 2004.

[129] Y. Xu et al. "Locally adaptive combining colour and depth for human body contour tracking using level set method". In: *IET Computer Vision* 8.4 (Aug. 2014), pp. 316–328. ISSN: 1751-9632. DOI: 10.1049/iet-cvi.2013.0164.

[130] L. Yang et al. "Evaluating and Improving the Depth Accuracy of Kinect for Windows v2". In: *IEEE Sensors Journal* 15.8 (Aug. 2015), pp. 4275–4285. ISSN: 1530-437X.

[131] Y. Yang and D. Ramanan. "Articulated Human Detection with Flexible Mixtures of Parts". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.12 (Dec. 2013), pp. 2878–2890. ISSN: 0162-8828.

[132] Z. Zhang, T. Tan, and K. Huang. "An Extended Grammar System for Learning and Recognizing Complex Visual Events". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.2 (Feb. 2011), pp. 240–255.

[133] W. Zhao et al. "Face recognition: A literature survey". In: *ACM Computing Surveys* 35 (Dec. 2003), pp. 399–458.

[134] D. Zhou, D. Petrovska-Delacretaz, and B. Dorizzi. "Automatic landmark location with a Combined Active Shape Model". In: *IEEE International Conference on Biometrics: Theory, Applications, and Systems, 2009*. Sept. 2009, pp. 1–7.

[135] S. W. Zucker. "Computer Vision and Human Perception: An Essay on the Discovery of Constraints". In: *International Joint Conference on Artificial Intelligence, 1981 - Volume 2*. 1981, pp. 1102–1116.