POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Monitoring Human State in a Robotic Assistive Platform: Data Acquisition and Person Detection Systems

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Matteo Matteucci
Correlatore: Prof. Alexandre Bernardino

Tesi di Laurea di:
Christian Vismara, matricola 804763

Anno Accademico 2014-2015

Ai miei genitori.

Abstract

During the last decade, problems related to a sedentary lifestyle increased significantly and became one of the greatest burden of today's society. Robotic technologies are possibly one of the main resources we have to tackle the problem due to their versatility and adjustment capabilities.
The AHA (Augmented Human Assistance) project has the final aim of building a robotic assistive platform that could be employed to raise motivation in patients while monitoring their progresses and assessing their health conditions. In this thesis work the acquisition and Pedestrian Detection module of the final platform have been designed and developed.
In order to provide a satisfying human-robot interaction experience, awareness of the position of people is a crucial aspect in Robotic Mobile Platforms. Vision is one of the richest and most employed sensor in today's Robotic Systems due to the diversity of features that can be extracted and used to analyse the surrounding environment, people included. Omnidirectional cameras, in particular, give us the possibility to exploit and increased field of view at the price of an increased distortion in acquired images.
This work explores different ways to process omnidirectional images to obtain versions of them more suitable to be used together with state-of-the art pedestrian detectors and to assess their performance by using different training and testing combinations. Results showed that better performances can be achieved using unwrapped or rectified images and by training the detector with *ad Hoc* datasets, but there is still a big margin for further improvements.
In addition, due to the critical role acted by datasets, I present a system to accelerate the labelling task using skeletal data provided by Microsoft Kinect V2.
Finally, I introduce an architecture to acquire, store and visualize the different information obtainable from the different sensors and cameras connected to the system.

Acknowledgements

I would like to thank my main advisor, Prof. Alexandre Bernardino, for his precious guidance during my time in Portugal, his support has been essential for the success of this experience.
I want also to thank Prof. Jose Gaspar for his valuable advices and Prof. Matteo Matteucci for the confidence he placed in me, allowing me to carry out this priceless experience.
I wish to thank all the people from VisLab for their warm reception, in particular a big thanks goes to Matteo Taiana for being always available to answer my questions.
Finally, I also would like to thank Patricia Keleher for the patience and dedication she showed while inspecting my thesis work.

Chapter 0

Introduzione

0.1 Formulazione del problema

Nella società di oggigiorno i problemi e le malattie causate da uno stile di vita sedentario sono uno dei più grandi pesi sociali ed economici. La promozione di uno stile di vita attivo, durante gli ultimi anni, ha iniziato a mostrare i suoi effetti positivi e ad invertire lentamente il trend. Il problema, però, è ancora ben lontano dall'essere risolto e l'ictus è ancora una delle principali cause di morte, specialmente nella società occidentale (quinta causa negli Stati Uniti [34]).
Il progetto Augmented Human Assistance (AHA) tenta di contrastare questo scenario promuovendo uno stile di vita salutare, sostenendo l'invecchiamento attivo e supportando le persone con deficit motori. Lo scopo di questo progetto è quello di costruire una nuova piattaforma assistenziale robotica per la gestione di programmi d'allenamento salutari, capaci di attirare gli utenti, monitorare l'esecuzione di esercizi e supportare i medici durante la riabilitazione.

Lo scopo di questa tesi è stato quello di sviluppare una parte del sistema finale, in particolare sono stati prodotti tre risultati:

- il sistema di acquisizione, utilizzato per acquisire, archiviare e visualizzare immagini e dati ottenuti dai vari dispositivi;
- il sistema di rilevamento in grado di individuare le persone attorno al robot utilizzando algoritmi di Pedestrian Detection. Verrà utilizzato dalla piattaforma mobile per indirizzarsi automaticamente verso il paziente;
- un sistema di etichettatura non supervisionato il quale utilizza i dati

scheletrici ottenuti dal Kinect per tracciare automaticamente le Bounding Boxes. Questo sistema è importante per la creazione automatica di dataset da utilizzare successivamente per il training personalizzato dei Detector basato sul modello di camera.

Il rilevamento di persone è un'istanza canonica nell'area del rilevamento oggetti. Grazie alle sue applicazioni dirette nella sicurezza automobilistica, sorveglianza e robotica, ha attirato una crescente attenzione negli ultimi anni [10]. Il suo scopo è quello di apprendere le features in grado di descrivere una persona in posizione eretta e quindi ricercare altri pedoni in immagini e sequenze video. Le performance di questi detector sono strettamente correlate con i set di dati utilizzati durante la fase di training [53], e il rischio di compromettere la funzione di classificazione, adattandola eccessivamente ai dati del set utilizzato per il training, è sempre dietro l'angolo.

In molti sistemi la conoscenza dell'ambiente circostante è cruciale. Lo si ottiene utilizzando sensori in grado di trasformare informazioni dall'ambiente circostante in segnali che possano essere interpretati dal sistema. Una classe di sensori normalmente utilizzata per questo scopo è quella delle videocamere, dispositivi che proiettano la luce irradiata dall'ambiente su di un sensore, un piano di celle CCD o CMOS, che successivamente converte queste informazioni in cariche elettriche, costruendo l'immagine digitale. Una sottoclasse di videocamere diventata recentemente popolare è quella delle camere omnidirezionali, per vie del loro campo visivo allargato, al prezzo di una maggiore distorsione nell'immagine finale [45].
I detector standard non sono stati pensati per funzionare con immagini omnidirezionali, questo rende necessario uno sforzo addizionale per trovare quale sia la soluzione migliore di combinare il detector con questo tipo di immagini.

In questo lavoro di tesi ho inizialmente sviluppato il sistema di acquisizione affrontando alcune sfide tra cui, in particolare, problemi di disconnessione e e sincronizzazione.
Quindi ho esplorato la possibilità di utilizzare camere fish-eye per fornire un campo di visione di circa 360° alla piattaforma mobile. Ho esaminato varie soluzioni proposte in letteratura [43] per la calibrazione di camere diottriche e catadiottriche e varie tecniche per la trattazione di immagini omnidirezionali [28, 11], anzi che applicare direttamente il detector sulle immagini originali. Prima, per comprendere a fondo il comportamento di questi modelli, ho dovuto studiare i modelli di camere prospettiche presenti in letteratura.
Ho eseguito diversi test sui set di dati acquisiti utilizzando una camera diot-

trica omnidirezionale, così da poter verificare l'efficacia dei metodi.
Infine, ho scritto uno script per sfruttare i dati scheletrici ottenuti dal Kinect per poter etichettare automaticamente i dataset.
I test sui dataset sono stati eseguiti utilizzando un'implementazione del *LCDF/ACF-Caltech+ Detector* proposto in [39].

Tutto il codice prodotto durante la mia esperienza è stato scritto utilizzando C++ e MATLAB, sfruttando le librerie MongoDB, YARP e OpenCV.

Ho effettuato la calibrazione delle camere montate sul sistema e ho eseguito vari esperimenti valutando le performance del detector quando applicato ad immagini omnidirezionali, unwrapped e rectified dopo aver effettuato la fase di training con dataset *ad Hoc*.
In futuro ho intenzione di estendere i dataset utilizzati per gli esperimenti, testare il detector su sequenze di immagini ottenute unendo le immagini omnidirezionali, frontale e posteriore. Possibile direzione di ricerca potrebbe anche essere quella di tentare di individuare altre possibili vie per migliorare le performance del detector, ed esempio adattando le feature al tipo di immagine utilizzato.

0.2 Descrizione del sistema

Ho implementato un sistema di rilevamento di persone per robot autonomi fornito di due camere omnidirezionali diottriche, una per la visione frontale e una per la visione posteriore.
Ho utilizzato un detector a "finestra scorrevole" basato sull'algoritmo di machine learning AdaBoost [26] e Aggregate Channel Features [16]. In questa tesi considero solo l'individuazione di persone in posizione eretta in quanto la maggior parte dei dataset disponibili si concentrano su questa categoria e anche perché il contesto consente questo tipo di ipotesi, essendo pensato per individuare persone in posizione eretta di fronte al sistema pronte per eseguire gli esercizi.
I tipi algoritmi utilizzati per individuare pedoni, normalmente, non sono pensati per funzionare con immagini altamente distort quindi ho dovuto effettuare alcuni studi per capire come applicare il detector su immagini omnidirezionali. In particolare, ho descritto la relazione che intercorre tra la degradazione delle performance e la distanza della persona dal centro dell'immagine e dalla camera.
Ho anche esplorato varie modalità per ripetere la fase di training e migliorare le performance quando il detector viene applicato ad immagini fish-eye e ho

verificato la propensione ad apprendere gli artefatti e la distorsione introdotti da questo tipo di camere e dalla conseguente manipolazione delle immagini effettuata.

Inoltre, ho sviluppato il sistema di acquisizione per la piattaforma robotica finale in grado di acquisire, archiviare e visualizzare le informazioni acquisite dai sensori connessi al robot.
In particolare, la presenza di vari dispositivi come camere omnidirezionali, Microsoft Kinect V2 e biosensori, ha richiesto una particolare attenzione durante la fase di definizione della struttura dati e un'attenta gestione della concorrenza e sincronizzazione.

Infine, ho sviluppato un etichettatore automatico di dataset in grado di sfruttare i dati scheletrici forniti dal Kinect per creare autonomamente dei dataset con le immagini acquisite. Il sistema richiede una fase di calibrazione per calcolare la matrice di rotazione e traslazione che permetta di mappare i punti dallo spazio tridimensionale del Kinect allo spazio bidimensionale della camera omnidirezionale. La calibrazione può essere eseguita manualmente, aggiustando alcuni parametri, a risolvendo un problema di ottimizzazione.
Una volta che la matrice è stata calcolata, il sistema è in grado di disegnare le Bounding Boxes all'interno dell'area dell'immagine omnidirezionale corrispondente al campo visivo del Kinect, dove gli scheletri sono stati individuati.

0.3 Struttura della tesi

Nel capitolo 2 descrivo lo stato dell'arte nel campo dei detector di persone e nel campo della calibrazione di camere omnidirezionali. Nel capitolo 3 descrivo la struttura del detector a finestra scorrevole utilizzato per il progetto. Nel capitolo 4 presento una visione d'insieme dei modelli di camera, a partire dalla camera Pinhole fino ai modelli di camera omnidirezionale utilizzati per descrivere le videocamere impiegate nel sistema, dedicando una sezione alla calibrazione e manipolazione di immagini omnidirezionali. Nel capitolo 5 propongo una soluzione per la creazione di un sistema per etichettare set di dati senza supervisione sfruttando i dati scheletrici forniti dal Kinect. Nel capitolo 6 espongo in dettaglio la struttura del sistema proposto. Nel capitolo 7 espongo i risultati sperimentali ottenuti, mentre nel capitolo 9 illustro le conclusioni ricavate e delineo gli sviluppi futuri di questo lavoro.

Contents

Chapter 1

Introduction

"It's showtime, folks."

Saul Goodman, Better Call Saul.

1.1 Problem formulation

In today's society the problems and diseases caused by a sedentary lifestyle are one of the greatest social and economical burden. The promotion of an active lifestyle during the past few years is starting to give positive effects and slowly inverting the trend but the problem is still far away from being solved as stroke is still one of the top leading causes of death, especially in the western society (fifth in the United States [34]).
The Augmented Human Assistance (AHA) Project tries to tackle this challenging scenario promoting healthy lifestyle, sustaining active aging and supporting those with motor deficits. The scope of this project is to build a novel robotic assistive platforms for health exercise program management, able to engage users, monitor execution, and support therapists in rehabilitation.

The purpose of this thesis has been to develop part of the final system, in particular three main outcomes were produced:

- the acquisition system, used to acquire, store and display images from the various attached devices;
- the detection module able to locate people around the robot using Pedestrian Detection algorithms. This will be used by the mobile platform to automatically point itself toward the patient;

- an unsupervised dataset labeller that uses skeletal data from the Kinect to autonomously draw Bounding Boxes. This is important for the automatic creation of datasets for training Pedestrian Detectors customized to the given cameras.

Pedestrian detection is a canonical instance of object detection. Because of its direct applications in car safety, surveillance, and robotics, it has attracted much attention in the last years [10]. Its scope is to learn the features describing a person in a standing position and then seek images and video sequences to detect them. Pedestrian detector performances are strictly related to the dataset used during the training phase [53], and the risk to compromise the classification function with bias is always around the corner.

In many system the awareness of the surrounding environment is crucial. This is attained by using sensors able to transform information from the surrounding environment into signals that can be interpreted by the system. One class of sensors which is commonly used is cameras. Devices that project light rays radiating from the environment onto the image sensor, a plane of CCD or CMOS cells, which in turn convert this information to electrical charges, forming the digital image. One subclass of cameras that gained popularity is the omnidirectional camera, due to the fact it offers an increased field of view, at the price of getting distorted images [45].
Standard PD detectors are not conceived to work with omnidirectional images, so an additional effort is needed to find what is the best way to combine detectors with this kind of images.

In this thesis work I started developing the acquisition system facing some challenges especially due to disconnection and synchronization issues.
Then I explored the possibility of using fisheye cameras to provide an almost 360°-wide field of view to the mobile platform. I examined the various solutions [43] for dioptric and catadioptric camera calibration proposed in literature and the different techniques to unwrap omnidirectional images [28, 11] rather than applying the detector over unwrapped and highly-distorted ones. Before, to fully understand the behaviour of these models, I had to study the perspective and non-perspective camera models available in literature.
I ran several tests on datasets acquired using a dioptric omnidirectional camera in order to verify the effectiveness of the method.
Finally, I coded the script to exploit skeletal data from Kinect in order to autonomously label the acquired datasets.
Tests on the datasets were performed by running an implementation of the

LCDF/ACF-Caltech+ Detector proposed in [39].

All the code produced during my experience has been written using C++ and MATLAB technologies, relying on MongoDB, YARP and OpenCV libraries.

I performed camera calibration on the omnidirectional cameras mounted on the system and I ran several experiments assessing the performances of the detector applied to omnidirectional, unwrapped and rectified images when trained with standard and *ad Hoc* datasets.
In future work I plan to extend the acquired datasets, test the detector on sequences obtained by joining the frontal ad rear fish-eye images, and explore new ways to improve detector performances in these particular situations, e.g. adapting the features to the kind of images used.

1.2 Description of the system

I implemented a Pedestrian Detection system for autonomous robots equipped with two omnidirectional dioptric cameras, one for the frontal and one for the rear view.
I use a sliding window detector relying on the AdaBoost [26] machine learning algorithm and Aggregate Channel Features [16]. In this thesis I considered only the detection of standing people since the majority of the most successful datasets used in Pedestrian Detection are also focused on standing pedestrians and also because the context implies the presence of a standing person waiting for the robot to start the exercises.
Typical Pedestrian Detection algorithm, normally, are not conceived to work with highly-distorted images so I had to make some studies to understand how to apply the detectors over omnidirectional images. In particular, I depicted the correlation that exists between loss of performances and distance of the imaged pedestrian from the image center.
I also explored different ways to retrain the Detector in order to improve its performances when operating over fish-eye images and verify its ability to learn the distortion and artefacts introduced by the imaging system and the subsequent processing applied to the acquired images.

In addition I implemented the acquisition module of the Robotic assistive platforms able to acquire, store and display the data acquired by the sensors connected to the robot.
In particular, the presence of very diverse kinds of devices such as omnidi-

rectional cameras, Microsoft Kinect V2 and biosensors, required a particular attention when shaping the data structure and a careful management of concurrency and timestamps.

Finally, I coded a dataset labeller able to exploit skeletal data returned by Kinect to autonomously label the acquired datasets. This system requires a calibration to find the rotation and translation matrix that permits to map points from the Kinect 3D space to the omnidirectional 2D camera space. Calibration can be performed manually, by tweaking a few parameters, or by solving an optimization problem.
Once the rotation and translation matrix is provided, the system is able to draw Bounding Boxes inside the area of the Omnidirectional Images corresponding to the Kinect field of view where skeletons have been detected.

1.3 Structure of the thesis

In Chapter 2 I describe the state of the art on Pedestrian Detection and Omnidirectional Camera Calibration. In Chapter 3 I describe the structure of the Sliding Window Detector used in the project. In Chapter 4 I present a survey of camera models, from the Pinhole camera to the Omnidirectional Camera Model used to describe the two fish-eye cameras, dedicating a section to delineate how the rectification and unwrapping of omnidirectional images is performed. In chapter 5 I propose a way to build an unsupervised dataset labeller using Kinect skeleton data. In Chapter 6 I detail the structure of the proposed system. In Chapter 7 I present the experimental results obtained, while in Chapter 8 I draw conclusions and present future developments of this work.

Chapter 2

Survey and State of the Art

"Have no fear of perfection - you'll never reach it."

Salvador Dalí.

This chapter lists and describes some of the most popular and top performing Pedestrian Detection algorithms, mostly following the comparison made in [21], after giving an overview on how the benchmarks on these detectors are performed, together with a survey of the most successful datasets used to test and train.
Finally, in section 2.3, some of the models used to describe omnidirectional cameras are introduced.

2.1 Benchmarking and Dataset Overview

During the last decade the forest of pedestrian detectors has grown significantly making necessary the definition of a common tool to test and compare detector performances. I will refer to the current standard benchmark used by the PD community, proposed by Dollar et al. in 2009 [20] and refined in 2012 [21], named Caltech. Another less popular but still noteworthy benchmark is KITTI [29], mostly aimed at evaluating autonomous driving systems.

2.1.1 Role of the Dataset

Dataset are a critical aspect for systems relying on Machine Learning since the classification function resulting from the learning process depends mostly on the images used to train the detector.
Furthermore, datasets are also used to test the detector performances and,

despite it has been shown that results are reasonably consistent over different datasets [21], outcomes may vary a lot depending on some dataset properties, especially when testing the detector on a dataset different from the one used to perform the training part.
In the context of PD there are many aspects that should be taken into account while choosing the dataset to train and test:

- Dataset size: providing a greater variety of images we are simultaneously supplying more information to the training algorithm, allowing it to learn a more fine tuned function.
- Imaging set-up: while mobile set-ups provide a variety of settings and backgrounds, video-surveillance videos have a high bias due to the fixed position of the cameras. Datasets made of collections of pictures, like INRIA [14], are likely to suffer from a *selection bias* introduced by the author [53].
- Data purity: data contained in the dataset can be *pure* if clearly representing the class of interest (non occluded, reasonably sized and standing pedestrian) or *impure* if not respecting the previous conditions. A good balance between these two classes should be kept while training the dataset.
- Labelling technique: It is not always clear what should be considered as positive, that is why some alternative labelling technique were introduced during the years to solve ambiguities arising when using only ground truth bounding boxes. For instance, in [21], the authors proposed an improved labelling introducing the so-called "ignore" regions.

2.1.2 Dataset Overview

In this section I will present the main dataset used to train and test the state-of-the-art detectors, briefly summarizing their main characteristics, plus a dataset introduced by the research group I am working with aimed at showing the impact of high-definition images on detector performances.

- INRIA [14]. It has been gathered from photos and it is one of the oldest PD dataset and small sized if compared to newer ones but the variety of pedestrian imaged makes it still a popular choice, especially when a fast training process is needed.
- Caltech [21]. Built with videos acquired on a vehicle in regular traffic, it is probably the most popular dataset for PD training and testing, due to its size (it contains about 10^5 pedestrian labelled in about

10^6 frames), its labelling completeness and also because the benchmarks made using this dataset are the most popular in PD community. The authors offer three versions of the dataset: *Training*, *Testing* and *Japan*, where the latter offers the same characteristics of the *Training* dataset but has been filmed in Japan.

- ETH [23], TUD-Brussels [56] and Daimler [22]. These datasets are similar to the Caltech one described above but their lower size and other peculiarities (e.g. Daimler lacks in colour channels) limited their popularity. Currently they are mostly used as a secondary detector benchmark, after Caltech.
- KITTI [29]. Gathered from a mobile set-up and mainly aimed at autonomous driver tasks, it is one of the predominant benchmark for PD. Even if not popular as the Caltech dataset, it is appreciated by the PD community because of its size and diversity.
- HDA [40] and then HDA+ after the improvememnts presented in [24]. Acquired by a network camera system to test Pedestrian Detection and Re-Identification system, it is used to test algorithms in video surveillance environments and to test the impact of high resolution images.

2.1.3 Evaluation Methodology

Evaluations is performed by running each detector over a testing dataset. Each frame is given to the detector, which returns a Bounding Box (BB), i.e. a rectangle on the image, paired with a confidence value for each detection. To match, a detected BB (BB_{dt}) and a ground truth BB (BB_{gt}) have to overlap more than 50%, in particular:

$$a_o = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} > 0.5 \tag{2.1}$$

Matches are counted as True Positives, where unmatched BB_{dt} count as False Positives and unmatched BB_{gt} as False Negatives (also called Missed Detections).

To compare detectors it is common to log-log plot miss detection rate against False Positives per image, by varying the threshold on detection confidence. In addition, to summarize detector performance in one value, it is common to use the *log-average miss rate*. It is computed by averaging miss rate at nine False Positive per image (FPPI) rates evenly log-spaced between 10^{-2}

and 10^0.

Evaluation is usually performed using a few different datasets but, since in [21] the authors showed that the performance of detectors is quite consistent across datasets, I will discuss mostly the results obtained with Caltech Pedestrian Dataset. In order to explore detector performances in various circumstances, evaluations on this dataset have been performed considering different conditions, rather than just running the detectors on the entire annotated dataset (see Fig. 2.1). Benchmarks have been performed considering different pedestrian scales and different cases of occlusion. The "reasonable" subset used by the authors includes only pedestrians taller than 50 pixels and under no or partial occlusion.

Due to the fact that applications for detectors include real-time tasks, another aspect that must be taken into account is the number of frames per second that each detector is able to process. To easily visualize and compare computational performances of algorithms it is common to plot the log-average miss rate versus the frames per second values achieved by each detector when run on a specific computer.

Some other little tweaks are performed to assure a fair comparison, e.g. ground truth filtering and aspect ratio standardization. For all the details please refer to [21].

2.2 Pedestrian Detection

Interest in Pedestrian Detection (PD) is constantly growing as proven by the increasing number of publications about it during the past years. Many new detectors have been presented, reaching an accuracy in detecting people that was unimaginable only a few years ago.
Although some attempts building detectors using segmentation [31] or keypoint [35] approaches have been made, the sliding window approach is the *de facto* standard approach in Pedestrian Detection so I will focus on detectors of this kind.

2.2.1 Milestone Detectors

Some pioneering detectors are nowadays outdated but still deserve to be cited, at least as a baseline to track PD evolution during the years. These detectors are Viola and Jones [54], known for the usage of AdaBoost with an improved cascade (see Section 3.3), and the HOG detector by Dalal and Triggs [14], that popularized the histogram of oriented gradient features (see Section 3.2).

2.2.2 Top-Performing Detectors

By evaluating benchmark results plotted in Figure 2.1, it can clearly be noticed that in every case there are always three detectors alternating in the top 3 positions:

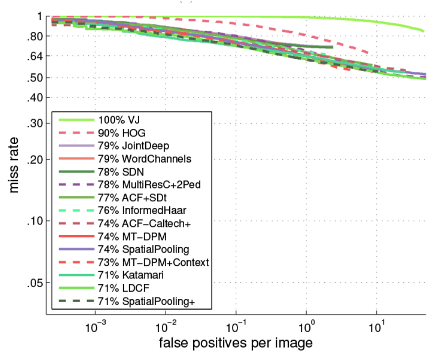
- *LCDF/ACF-Caltech+*, [39]. To avoid the usage of computational expensive oblique decision trees [38], the authors used the Aggregate Channel Feature (ACF) detector [16] as their baseline and introduced a technique that suits better orthogonal decision trees by locally decorrelating features, instead of decorrelating them globally as in [32].

- *Katamari*, [10]. This approach was used to show how, by combining complementary features, it is possible to build better performing methods.

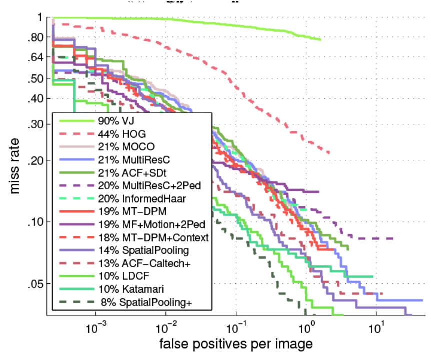
- *SpatialPooling+*, [41]. In this work they showed how to successfully apply spatial pooling, a technique to compute local features by grouping data from the selected region and extracting a statistic value out of it, to PD tasks.

Instead, when the aim is to build a real time system (or *frame-rate*, i.e. at 30 fps or higher, as defined in [17]), results depicted in Figure 2.2, where Log-average miss rate is plotted versus frames per second, become more relevant.
Here the two detectors that stand out for their speed over a satisfactory miss rate are: (i) *Crosstalk* [17], that exploits communication between cascade evaluations at nearby locations, and (ii) *FPDW* [18], where feature responses computed at a single scale are used to approximate features responses at nearby scales. The *MultiFtr+Motion* detector [55], in spite of having the lowest log-average miss rate, is not suited for real-time applications because its runtime speed is clearly inadequate.

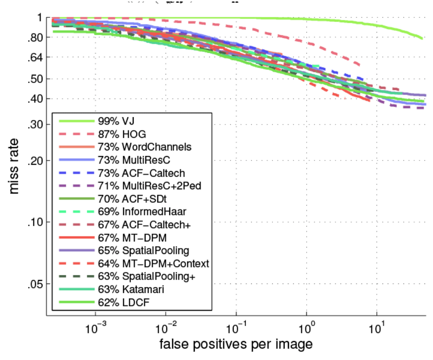
A complete list of all the available detectors and their respective benchmark results is stored and kept updated by the Caltech Vision Group [1].



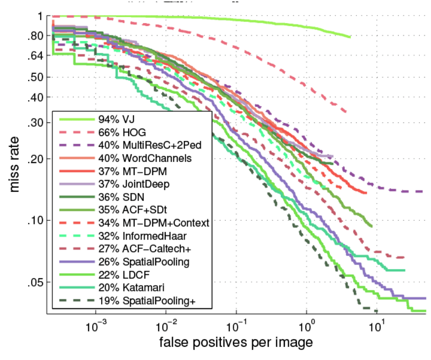
(a) Overall



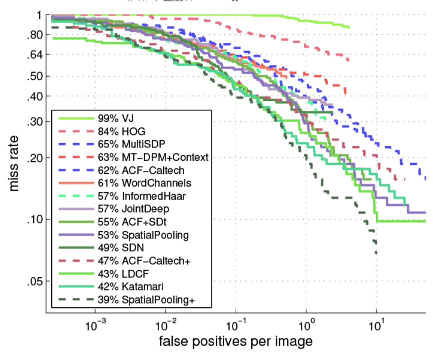
(b) Near scale



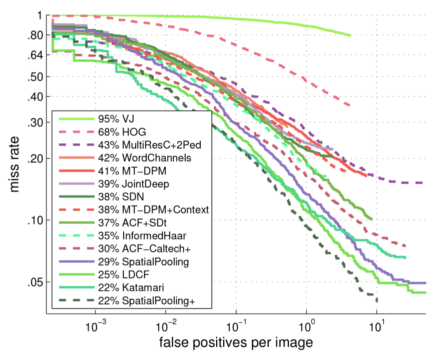
(c) Medium scale



(d) No occlusion

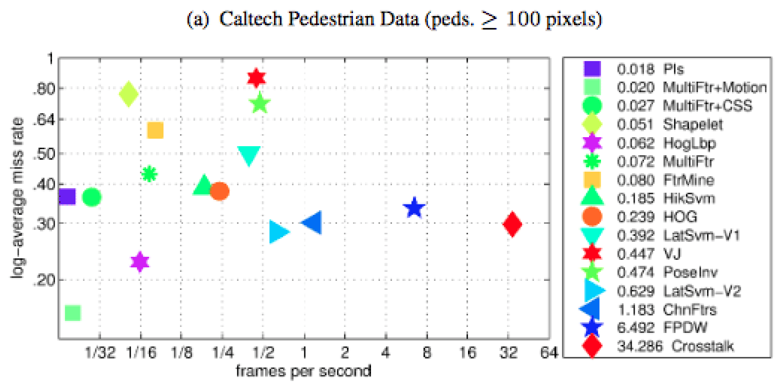


(e) Partial occlusion

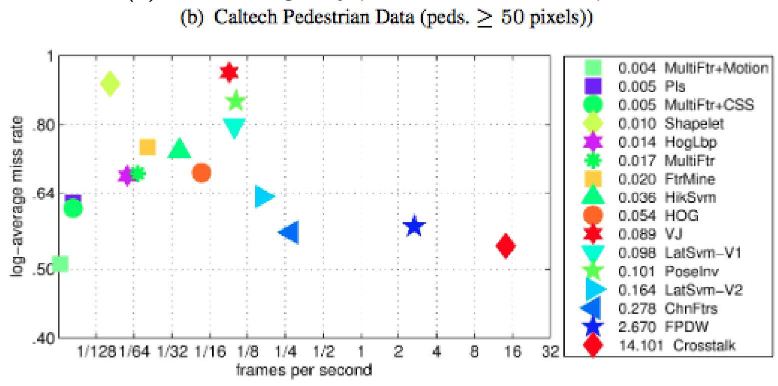


(f) Reasonable

Figure 2.1: Evaluation results under six different conditions on the Caltech Pedestrian Dataset, plots taken from [21]



(a) Considering only pedestrians over 100 pixels



(b) Considering only pedestrians over 50 pixels

Figure 2.2: Log-average miss rate versus frames per second plot obtained running detectors on the Caltech Pedestrian Dataset, using a specific machine and in two different conditions. plots taken from [21]

(a) Vivotek FE8174 dioptric camera, image taken from [3]
.

(b) Catadioptric camera
.

(c) Point Grey Ladybug5 polydioptric camera, image taken from [2]
.

Figure 2.3: Examples of omnidirectional cameras. Dioptric cameras offer a field of view of 180° over the vertical and horizontal plane. Catadioptric cameras offer a 360° field of view over the horizontal plane but must be placed on top of the robot at a reasonable height. Polydioptric cameras are the only ones able to acquire the entire sphere around the camera.

2.3 Omnidirectional Camera Models

Omnidirectional cameras offers a more extended field of view with respect to the standard cameras and, as illustrated in [45], they can be manufactured in many different ways:

1. Dioptric or fish-eye cameras (see Fig. 2.3a) use a combination of shaped lenses and can reach a field of view even bigger than 180° on the horizontal plane, if placed parallel to the floor.

2. Catadioptric cameras (see Fig. 2.3b) combine a standard perspective camera with a shaped mirror, normally with a parabolic shape but there are also cases with hyperbolic or elliptical mirror, that are, as shown in [4], the only shapes having the useful property of maintaining a single projection centre. This kind of camera succeed in providing a 360° field of view in the horizontal plane and more than 100° in elevation.

3. Polydioptric cameras (see Fig. 2.3c) use multiple cameras with overlapping field of view, the only omnidirectional camera model able to provide a 360°spherical field of view.

Usually, in robotic and computer vision fields, the options 1 and 2 are the most used due to their good quality/price ratio. In my case, since I needed

to detect people from a mobile robotic platform lower than a standing pedestrian, I preferred to use two dioptric cameras, one on the front and one on the back, instead of having a camera on top of the robot. This solution allows us to have almost 360° field of view on the vertical and the horizontal planes. It was not possible to obtain the full sphere due to the hidden area between the two cameras. The solution 3, instead, has been discarded due to its unaffordable cost.
For all the details about the system architecture, please refer to Chapter 6 and Figure 6.1.

There are projection systems whose geometry cannot be described using the conventional pinhole model because of the very high distortion introduced by the imaging device. Some of these systems are omnidirectional cameras [45].
Many models have been developed to describe the omnidirectional cameras and, to compare them, I followed the work presented in [43] where the performances of four calibration methods based on the respective models have been compared. These have been chosen because of their popularity, due to the fact that they work quite well, and because the authors provided open source Matlab toolboxes simplifying a lot the work of the people interested in omnidirectional camera calibration.

The above-mentioned models are:

1. Model by Mei and Rives [37] which uses the sphere camera model depicted in [30] and [6] and requires several images of a 2D pattern. This is the better performing one at the price of a complex model;

2. Model by Puig et al. [42], it also uses the sphere camera model and obtains a solution in closed form requiring a set of 3D-2D correspondences;

3. Model by Barreto and Araujo [7], that also uses the sphere camera model, it requires a single omnidirectional image containing a minimum of three lines. In the study I followed to evaluate model performances, the authors showed that this model fails in representing fisheye cameras;

4. Scaramuzza et al. [46] which does not exploit the spherical model but assumes that the image projection function can be described by a polynomial, based on Taylor series expansion. In spite of its simplicity,

this model achieved performance results close to the top-performing one.

Another well-known method not cited in the comparison I followed is the Division Model [25]. This method is more general: it can be applied to both omnidirectional and non-omnidirectional images and allows to correct distortion without having access to the original lens. This method successfully corrects distortion even using pictures with lack of straight lines but, in general, the performances are lower than the ones obtained with *ad Hoc* methods [37].

Chapter 3

Sliding Windows Detector

"All truly great thoughts are conceived while walking."

Friedrich Nietzsche.

Sliding window detectors follow a one-way work flow composed by different steps aimed at scanning the entire image and detecting people inside it (see Fig. 3.1).
First, the image pyramid is created out of the original image. Then, the features are extracted from each level of the image pyramid and the detection is run inside the feature space by sliding a detection window, a rectangle inside which the detector seeks for people. Finally, the classifier returns its outcome, paired with a confidence value in case it is positive, and the detection is mapped back in the original image, merging the overlapping detections.
In the following section I will explain more in detail each step and, for each one, list some of the various alternatives present in the literature.

3.1 Sliding Window

To detect people that may be spread all over an image, a window of fixed size is passed along a grid of locations on the picture. Every scanned position is classified either as "person" or "not person". For the case of "person", the position is associated with a value that encodes the confidence of the classifier in the presence of a person.

Since there may be people in the image that are taller, shorter, closer, further away from the camera, etc. and appear in very diverse sizes, the sliding window itself is not enough to ensure that all the pedestrians will be

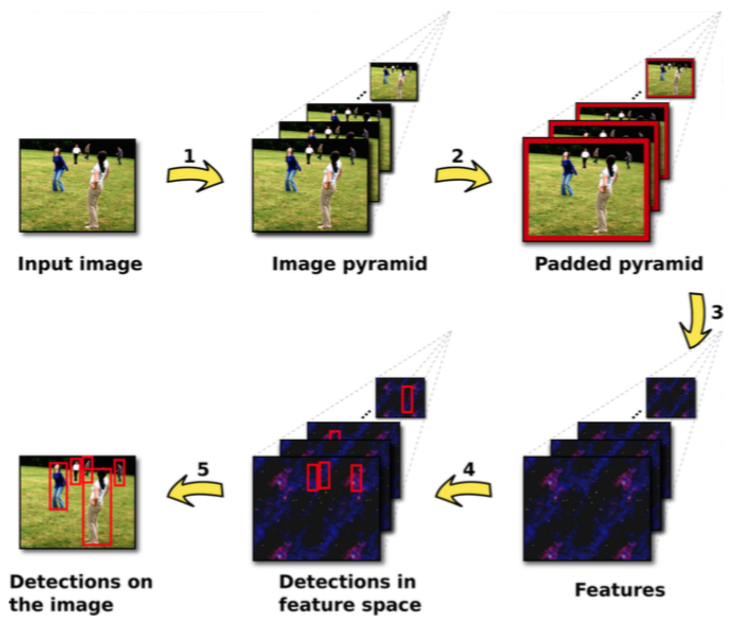


Figure 3.1: Sliding window detector work flow. Step 1: construction of the image pyramid. Step 2: padding of the image pyramid. Step 3: computation of the image features. Step 4: sliding-window-based detection. Step 5: Non-Maximum Suppression and mapping of the detections to the input image. Detection Windows are padded in order to exploit information from the context around pedestrians so, to keep the size of the window constant, the same amount of padding is applied to each pyramid level.

classified by a window of the correct size. To solve this problem it is common to run the sliding window over a set of scaled versions of the original image, named image pyramid (see Fig. 3.3a). Running the same window classifier on a shrunk version of the input image corresponds to running a larger window classifier on the input image. Normally 8-16 layers per octave (the size range that goes from one image height to its half or its double) are used and padding is performed after scaling.
An alternative way used by the Fastest Pedestrian Detector in the West (FPDW) [18] is to estimate the features of the layers between the different octaves instead of effectively sliding the window over them (see Fig. 3.3b). In addition, the authors of [9] proposed to calculate one model for each octave at training time and then compute features only at the base scale of the image (see Fig. 3.3d), providing even further temporal performance improvements.

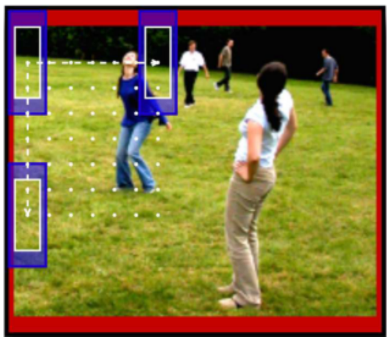


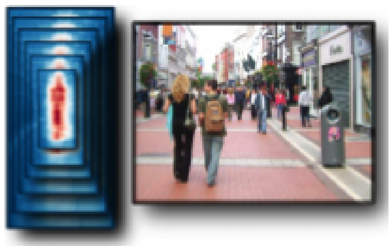
Figure 3.2: Sliding window running over the image allowing for the detection of pedestrians. The padding area of the detection window is marked with a blue shading.

One side effect of image pyramid I have to take into account is the possibility of generating more than one detection for a single pedestrian. To account for this, many Non-Maximum Suppression (NMS) techniques have been developed, like the one described in [21], which groups boxes overlapping for more than a certain threshold, and discards the ones with the lower confidence value.

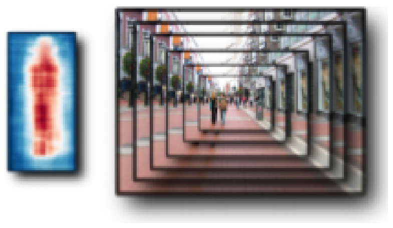
3.2 Feature Extraction

The feature extraction phase aims at extracting meaningful informations from the raw pixels enclosed by the detection window and the choice of these features often marks the line between success and failure in detecting pedestrians.

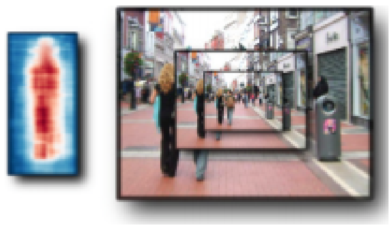
One of the first family of features used in PD was made of multi scale Haar-like wavelets (see Fig. 3.4a). These features are, essentially, rectangular areas applied on the detection window, inside which the difference between the sum of the pixels within two or more rectangular regions is computed. These features are very successful when detecting frontal faces, as shown in [54], but are unable to perform at the same level when used in PD task.
In [57] Wu and Nevatia presented a new family of features specifically developed for PD tasks, named *edgelets* (see Fig. 3.4b). Those feature describe a way to represent shapes locally by using a set of short lines and curves. This method can be considered as a variation of the well known Chamfer matching [8].
Big improvements of the detector performances arrived with the introduction of Histograms of Oriented Gradients (HOG) features (see Fig. 3.4c) in



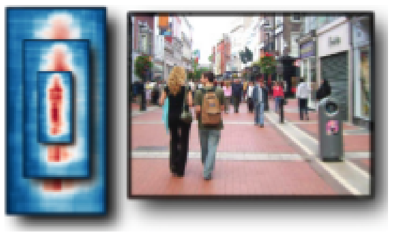
(a) N models, 1 image scale



(b) 1 model, N image scales



(c) 1 model, N/k image scales

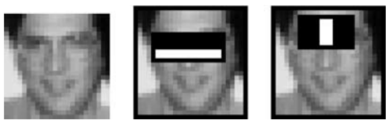


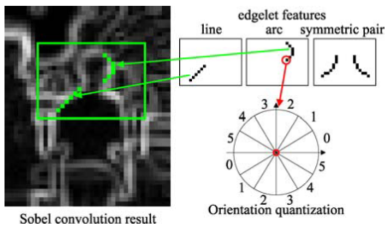
(d) N/k models, 1 image scale

Figure 3.3: Different image pyramid approaches. Method in 3.3a does not take in account the similarity between pedestrians imaged at similar scales and learns a model for each scale. Approach in 3.3b, instead, learns only one model and uses N different versions of the original image. Methods in 3.3c and 3.3d use, respectively, one model over $\frac{N}{k}$ images and $fracNk$ models over one image, exploiting the correlation between the levels instead of calculating all of them explicity. Images taken from [9]

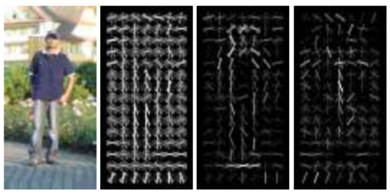
[14]. HOG compute descriptors over small square portions of image windows, named cells, pooling the gradient information. The gradient is computed for different orientations, each contributing to build the final histogram. In this way it is possible to represent both soft transitions and strong edges in a robust way with respect to small changes in orientation since the different gradient values are collected in angular bins grouping all the ones within a range.
After HOG came the Integral Channel Features (ICF) (see Fig. 3.4d) [19], employed in many modern detectors such as the Fastest Pedestrian Detector in the West (FPDW) [18]. This method assume the usage of different channels to describe a single window, in particular six channels are dedicated to different orientations of the gradient, one is for the gradient module and the last three encode the image with the LUV colour space, separating brightness from colour channels.
Finally, recent publications like [48], have shown a way to apply Convolutional Networks to PD tasks, inspired by neuroscience and biological processes. They showed how to reach the sate-of-the-art performance level by exploiting hierarchical feature extraction systems and unsupervised learning

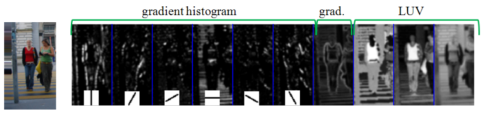
algorithms (see Fig. 3.4e).


(a) Haar-like features, image taken from [54]



(b) Edgelet, image taken from [57]


(c) Histogram of Oriented Gradients, image taken from [14]



(d) Integral Channel Features, image taken from [19]

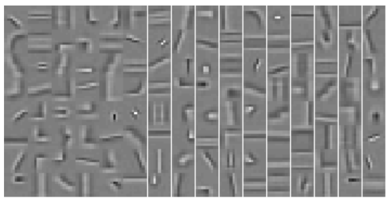

(e) Automatically trained filters, image taken from [48]

Figure 3.4: Examples of different features used in PD tasks.

3.3 Window Classifier

The window classifier is the fundamental module of the sliding window detector work flow. Its role is to get as input the set of features collected from an image window and to return a confidence value that expresses how confident the classifier is on the presence of a pedestrian inside the window.
The function in charge of mapping a point from the feature space to a score is learned by the classifier during an initial training phase by using a set of labelled images taken from a dataset and a machine-learning-based algorithm. Labelled datasets normally contain windows enclosing a pedestrian, labelled as positive, and windows that don't contain human being, labelled as negative, so that the machine learning algorithm can use the features extracted from these windows to learn the mapping function. The two most used ML algorithms in PD are the Support Vector Machines [13] and AdaBoost [26].

The *maximal margin classifier* compute the *optimal separating hyperplane* as the one that is farthest from the training observations. It is not always possible to find an hyperplane separating the two classes so, rather than seeking the largest possible margin so that every observation is on the correct side of the hyperplane, the *soft margin classifier* allows some observations to be on the incorrect side. Finally, since classes might be separable but not with a linear boundary, the concept has been extended defining the *support vector machine* (SVM), a class of classifiers able to separate classes with non-linear boundaries using the so-called "kernel trick". This technique intrinsically applies a linear boundary in a transformed version of the feature space which corresponds to a non-linear boundary in the original feature space.

The AdaBoost algorithm works by building a Strong Classifier (StC) as a combination of many Weak Classifiers (WkC). The basic version of WkC is the decision stump: the classification is based only on the value assumed by a feature, compared to a threshold. More advanced and better performing WkC classifiers, as shown in [19], are the depth-2 decision trees. They consist in three decision stumps arranged as a tree where the output of the first one decides which one will be the next decision stump to be used. Deeper trees are not used due the tendency they have to overfit the data.
To build the StC, the AdaBoost method starts by assigning a weight to the various examples of the training set and then by iterating over two phases: first, it learns the best possible WkC given the current weights; Second, chooses the classifier with the lowest error and updates the weights so that the examples that have been misclassified by best WkC have higher importance. This ensures that the following WkC will focus on those examples. Eventually, a Basic StC is built by summing the outputs of all the WkC weighted by their confidence value.
Viola and Jones presented in [54] an alternative to the Basic Strong Classifier, named Attentional Cascade. Here every example has to go through a cascade of strong classifiers, aimed at rejecting negative examples as fast as possible. Due to the fact that less computational effort is wasted with negative samples, it is much faster than the Basic Strong Classifier. In [58] a variant of the Attentional Cascade has been introduced, the so-called Soft Cascade. This method entails the usage of WkC instead of StC as building blocks of the cascade.
Normally, AdaBoost methods are slower than SVM during the training part but the fastest PD systems in the literature are mostly based on AdaBoost classifiers.

The last aspect that must be taken in account when training a classifier is the number of bootstrap epochs. Since the quantity of negative examples is much bigger than that of the positive ones, a technique to avoid an unnecessarily long training phase has been introduced: the classifier is trained multiple times alternating a training phase and a phase in which the detector is run over images that don't contain people. Every positive detection on those images is considered "hard negative" and added to the negative training set that will be used in the following iteration. It is common to use from 3 to 5 epochs of bootstrapping.

Chapter 4

Camera Models

"No one takes photographs of something they want to forget."

Sy Parrish, One Hour Photo.

Cameras are sensors that allow to acquire light rays coming from the surrounding environment. Every camera has its own characteristics that make it more suited for a task than another. Parameters that differentiate cameras can be the way the image is represented, digitally or analogically, the resolution, that is how close lines can be to each other in the picture and still be visibly resolved, and the number of frames that can be acquired. If more than one frame can be acquired in sequence, the maximum frequency of acquisition is also an important parameter. Nowadays cameras are employed in many different fields: movies we watch at the cinema, pictures posted on the internet, video streaming during video conferences, and so on. All of them are light rays acquired by a sensor. In our case cameras are the way we have to observe the world from a machine. In particular, since our scope is to continuously detect people in the area around the system by processing images on a computer, we will use a camera able to acquire multiple digital frames with an adequate speed and resolution.

Another critical aspect of cameras is the angle of view. This represents the angular portion of a scene that the camera is able to acquire, varying from 0 to $360°$. For instance, the human eye has an horizontal field of view of approximately $160°$, even if at the extremities it is just able to barely detect the movements. Because of this, eyes modelled as cameras are an imprecise model.

Finally, we also had to take in account the radial distortion, introduced by camera lenses. Radial distortion is responsible for line bending so, due to this phenomenon, what in world appears as a straight line, in images might look curved.

Camera models are functions with particular properties representing the mapping between 3D world and 2D images. The most common model for cameras is the perspective camera, whose model can be represented by a matrix operating in homogeneous coordinates. Most of the digital cameras we use can be represented using this model that offers a reduced angle of view but also low radial distortion, making images acquired by this kind of cameras more similar to the way we perceive the world.

In the next sections I will present the models of the previously listed cameras, following the overviews presented in [33] and [5], and then the chosen Omnidirectional Model proposed by Scaramuzza.

4.1 Perspective Cameras

In the following section I will go through the perspective camera model. Starting from the simplest and most specialized camera model that can be found in literature, the pinhole model, and progressively removing assumption in order to get a more generalized an versatile model.

4.1.1 The Pinhole Model

In pinhole cameras, all the light rays coming from the real world pass through a single point, with infinitesimal size, and form an image on a plane placed behind the hole, which is called *image plane* or *focal plane*. In this case, no lenses are employed so no radial distortion occurs.

The model can be specified by naming f the distance between the image plane and the hole (*focal length)*, and defining an euclidean system where the pinhole hole becomes the origin $\mathbf{C}$, called *camera centre* or *optical centre*, and z the axis perpendicular to the image plane passing by $\mathbf{C}$, also known as *principal axis* or *principal ray*. The point where the principal axis meets the focal plane is said *principal point* $\mathbf{p}$.
A point in the 3D world $\mathbf{X} = [X, Y, Z]^T$ is mapped to $\mathbf{x} = [x, y]^T$ on the

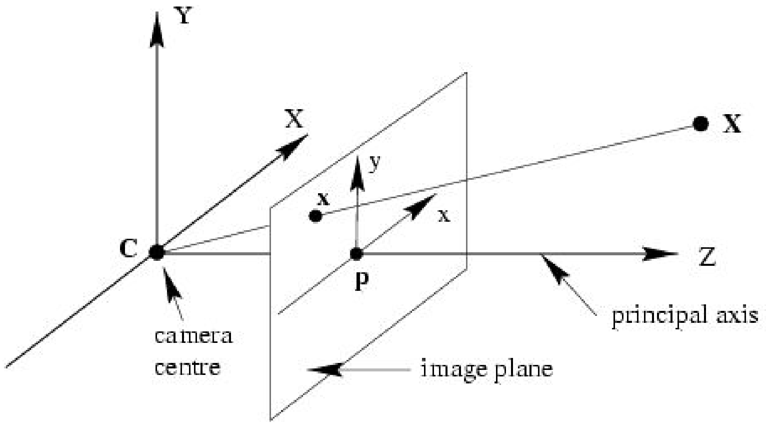


Figure 4.1: Illustration of the pinhole camera model having the focal plane in front of the camera centre. Notice that this way of placing the focal plane is commonly used to describe the horizontal and vertical flipping of the image, originally projected on a plane placed behind the camera centre at the same distance f. Without this transformation the image would appear mirrored and upside-down.

image plane according to the following relationship:

$$x = -f\frac{X}{Z} \qquad y = -f\frac{Y}{Z} \tag{4.1}$$

Notice that the minus sign is due to the fact that the image is flipped horizontally and vertically. Normally this minus sign is removed with a posteriori processing of the image. Removing this sign, for the sake of the model, means moving the image plane in front of the hole, exactly at the same distance f (see Fig. 4.1). By applying this to Equation 4.1 we obtain a new couple of equations:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{4.2}$$

The angle of view of the camera is strictly dependent on the size of the image plane. In particular, if $2r$ is the size of the plane, the field of view is equal to:

$$\theta = arctan(r/f) \tag{4.3}$$

with an upper limit of $180°$.

4.1.2 Projection Using Homogeneous Coordinates

Instead of considering the point in camera reference system coordinates, we express it in world coordinates, getting $\mathbf{X}_w = [X_w, Y_w, Z_w]^T$. The mapping between a point in world coordinates and the same point in camera coordinates is defined as follows:

$$\mathbf{X}_c = R\mathbf{X}_w + \mathbf{T} \qquad where \qquad R \in \mathbb{R}^{3x3} \qquad and \qquad \mathbf{T} \in \mathbb{R}^{3x1} \tag{4.4}$$

where $\mathbf{X}_c = [X_c, Y_c, Z_c]^T$ is the point expressed in camera coordinate system, R is the rotation matrix and $\mathbf{T}$ is the translation vector describing the transformation to pass from one reference system to the other.
Remembering Equation 4.2, we have that $\mathbf{X}_c$ is projected onto the image plane in point $\mathbf{x}$, defined as follows:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix} \tag{4.5}$$

that can be rewritten using homogeneous coordinates as:

$$Z_C \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{4.6}$$

In addition, the $3x4$ matrix can be decomposed as following:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = K_f \Pi_0 \tag{4.7}$$

Where Π_0 is canonical projection matrix.
From here on I will indicate that I am expressing $\mathbf{x}$ in homogeneous coordinates by calling it $\widetilde{\mathbf{x}}$. The Z_c factor preceding vector $\widetilde{\mathbf{x}}$ in Equation 4.6 represents the depth and is usually unknown, so it can be better expressed calling it $\lambda \in \mathbb{R}^+$. Composing Equation 4.6 with Equations 4.7 and 4.4, and substituting Z_c with λ, we get the geometric model of an ideal camera, defined as follows:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.8}$$

where $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = S_w^c$ is the rotation and translation matrix between world and camera reference systems.
Finally, rewriting everything with matrix notation, we obtain:

$$\lambda \widetilde{\mathbf{x}} = K_f \Pi_0 S_w^c \widetilde{\mathbf{X}}_w \tag{4.9}$$

4.1.3 Perspective Cameras with Intrinsics Parameters

The model depicted in Equation 4.8 assumes the principal point as origin of the coordinates in image plane and metric units. In real cameras, images are expressed as pixels and the origin is normally defined in a corner, usually in the upper-left one.
We start by defining the relation that exists between the two units of measure, in particular, if x and y are defined in metres and x_s and y_s are the equivalent in pixels, x_s and y_s are a scaled version of the coordinates x and y. The relation can be expressed as:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.10}$$

Moreover, s_x and s_y may be different between each other, defining a rectangular pixel. The most general case would be the one of a trapezoidal pixel but, since in no modern camera it is used, we will omit this case assuming s_θ, the second element of the fist row of the scale matrix, equal to 0.
Now we still have to take in account the coordinate system origin problem: to solve it we will need to apply the following translation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{4.11}$$

where $[x_c, y_c]^T$ are the coordinates in pixels of the principal point and $[x', y']^T$ the coordinates in pixel of the point $[x_s, y_s]^T$ according to the new coordinate system.
By merging the previous equations and writing the result using homogeneous coordinates we obtain:

$$\tilde{\mathbf{x}}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_c \\ 0 & s_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.12}$$

or, using matrix notation:

$$\tilde{\mathbf{x}}' = K_s \tilde{\mathbf{x}} \tag{4.13}$$

Now, by mixing Equation 4.8 and Equation 4.12, we can get the perspective camera complete geometrical model, defined as:

$$\lambda\tilde{\mathbf{x}}' = \lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_c \\ 0 & s_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

or, equivalently:

$$\lambda \tilde{\mathbf{x}}' = K_s K_f \Pi_0 S_w^c \widetilde{\mathbf{X}}_w \tag{4.14}$$

Parameters defined in K_s and K_f are normally unknown and can be estimated just through a calibration procedure. This process is not able to retrieve all the single matrix coefficients but just the product between the two matrices, for this reason could be more convenient to express them as a single matrix $K = K_s K_f$ in the following way:

$$K = K_s K_f = \begin{bmatrix} f s_x & 0 & x_c \\ 0 & f s_x & x_c \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

Matrix K is named intrinsics parameters matrix.

4.1.4 The Projective Camera

The perspective camera model analysed during all this section is just a special case of the Projective Camera, the most general way to describe the mapping of a point from the 3D world to the 2D one. In particular we have the following general equation describing the relation between $\mathcal{P}^3$ and $\mathcal{P}^2$:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} \tag{4.16}$$

where $\mathbf{S} = [S_{ij}]$ is the transformation matrix and, since only the ratios of elements S_{ij} are important, it has 11 degrees of freedom. $(x_1, x_2, x_3)^T$ and $(X_1, X_2, X_3, X_4)^T$ are homogeneous coordinates related to $\mathbf{x}$ and $\mathbf{X}$ by:

$$\begin{array}{rcl} (x, y) & = & (x_1/x_3, x_1/x_3) \\ (X, Y, Z) & = & (X_1/X_4, X_2/X_4, X_3/X_4) \end{array} \tag{4.17}$$

The previous model can still be specialized, in different ways with respect to the perspective camera, by adding different constraints, obtaining other models suitable to describe other kinds of cameras. More details and examples are given in Section 4.2.

4.1.5 Distortion in Perspective Cameras

Pinhole camera model is widely used in camera theory even if, in reality, there are no cameras corresponding to that model because the light energy reaching the sensor from a single infinitesimal point is not enough. To get

enough light we would need to lengthen the exposition, and this might be acceptable for still scenes but is bad for general purposes because even small movements of the camera or of the pictured object will compromise the image, or enlarge the hole but, in this case, there will be more than one light ray coming in corresponding to a single point, compromising the image focus.
These problems are currently avoided by adding a series of lenses able to convolve many rays to a an adequately small hole, allowing the sensor to get enough light energy. However, by adding lenses, we also introduce some distortion. The Brown-Conrady mathematical model comes in handy to describe the introduced distortion that can be divided in two terms: the radial distortion and tangential distortion, even if the second one is not very relevant in case of perspective cameras.
By calling x_d and y_d the coordinates of a point $\mathbf{x}_d$ on the image affected by distortion and x_u, y_u the coordinates of the point $\mathbf{x}_u$ obtained after distortion removal, called rectification, the distortion model is defined as follows:

$$x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^2 + \cdots) + (P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3 r^2 + \cdots) \tag{4.18}$$

$$y_u = y_d + \underbrace{(y_d - y_c)(K_1 r^2 + K_2 r^2 + \cdots)}_{\text{Radial distortion}} + \underbrace{(P_1(r^2 + 2(y_d - y_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3 r^2 + \cdots)}_{\text{Tangential distortion}} \tag{4.19}$$

Where:

- x_c and y_c are the coordinates of the principal point
- K_i is the i-th coefficient of radial distortion
- P_i is the i-th coefficient of tangential distortion
- $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$ is the distance between x_d and the principal point.

In case of perspective cameras, normally K_1 and K_2 coefficients are enough to describe radial distortion.

4.2 Cameras Which Do Not Capture the Perspective Effect

As stated at the end of the previous section, the general projective camera model can be specialized by adding different constraints. In this section we will present a brief overview about some different special cases of cameras derived from the projective model but defined in a way that does not preserve the perspective effect obtained with the perspective camera.

4.2.1 The Affine Camera

The first special case of projective camera we can have is the affine camera, obtained by constraining $\mathbf{T}$ such that $T_{31} = T_{32} = T_{33} = 0$:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ 0 & 0 & 0 & T_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} \tag{4.20}$$

By defining $\mathbf{M}$ as a 2×3 matrix with elements $M_{ij} = T_{ij}/T_{34}$ and $\mathbf{t} = [T_{14}/T_{34}, T_{24}/T_{34}]T$ as a 2-vector giving the projection of $\mathbf{X} = \mathbf{0}$, in terms of image and scene coordinates, the mapping written in inhomogeneous coordinates becomes:

$$\mathbf{x} = \mathbf{MX} + \mathbf{t} \tag{4.21}$$

A key property of the affine camera is that it *preserves parallelism* so lines that are parallel in the world remain parallel in the image.

4.2.2 The Orthographic Camera

Orthographic cameras are a specialization of the affine camera model. In this case light rays, instead of coming in all from the optical centre as in the perspective camera model, travel and reach the image plane in a parallel fashion as shown in Figure 4.2.
This kind of camera is formally represented by choosing a matrix $\mathbf{M}$ that represents the first two rows of a rotation matrix. In particular, the simplest form is:

$$\mathbf{M}_{orth} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.22}$$

so:

$$\mathbf{T}_{orth} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.23}$$

and

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} \tag{4.24}$$

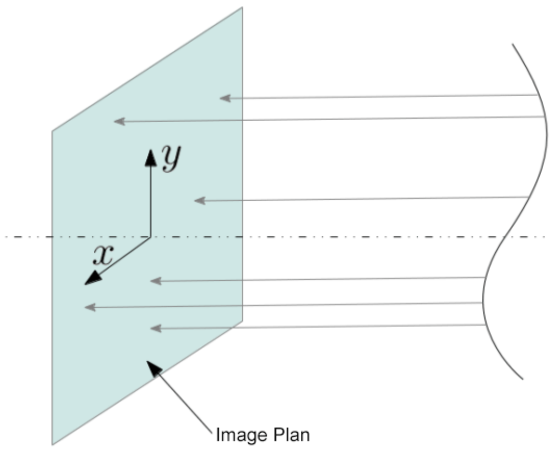


Figure 4.2: The orthographic camera model. Notice that light rays, instead of coming in all from the optical centre as in the pinhole camera model, travel and reach the image plane in a parallel fashion.

4.2.3 The Weak-Perspective Camera

When the camera field of view is small and the depth variation of pictured objects is small compared to the distance of the objects from the camera Z^c_{ave} ([52] suggests a 1:10 ratio), the individual depth Z^c_i can be approximated with the average Z^c_{ave}, leading to:

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z_{ave}} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{4.25}$$

So the weak-perspective camera combines orthographic and perspective projections because points are orthographically projected onto the average plane and then perspectively onto the image.
the weak-perspective mapping written in inhomogeneous coordinates can be written as:

$$\mathbf{x} = \mathbf{M_{wp}}\mathbf{X} + \mathbf{t_{wp}} \tag{4.26}$$

where $\mathbf{M_{wp}}$ is a $2x3$ matrix whose rows are the uniformly scaled rows of a rotation matrix and $\mathbf{t_{wp}}$ is a 2-vector. The simplest form is:

$$\mathbf{M}_{wp} = \frac{f}{Z_{ave}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.27}$$

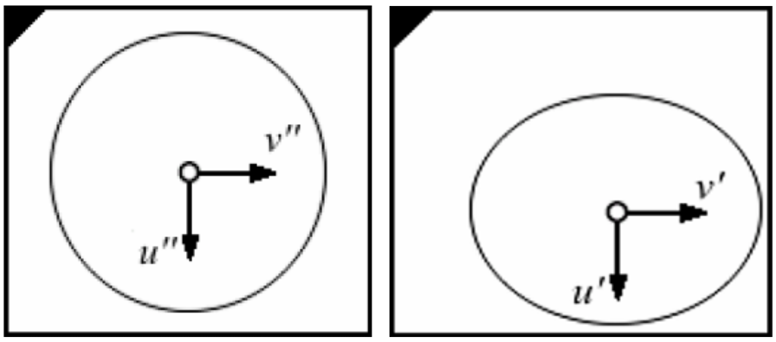
and so:

$$\mathbf{T}_{wp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & Z_{ave}/f \end{bmatrix} \tag{4.28}$$

4.3 Omnidirectional Camera

In case of detection tasks would be more desirable to use cameras with a wider angle of view, in order to be able to scan a bigger portion of reality. A category of cameras that offers this property is the omnidirectional cameras, sensors able to acquire images with an high angle of view, up to 360°. Unfortunately, this convenient property comes along with side effects: due to the lens used to get an higher field of view, the resulting image suffers of a huge radial distortion that doesn't allow us to model this kind of cameras with any of the previous model.
The usefulness of omnidirectional cameras is significantly reduced without a procedure to remove distortion as much as possible: in the following sections I will also depict the calibration procedure followed to compute the camera parameters, and then how I used these parameters to perform omnidirectional image rectification and unwrapping to get images in which the Pedestrian Detector could perform better.

4.3.1 The Model Proposed by Scaramuzza



(a) Sensor Plane *(b) Image Plane*

Figure 4.3: Camera Sensor Plane and Image Plane. If X is a scene point, $\mathbf{u}'' = [u'', v'']^T$ is the projection of X onto the sensor plane and $\mathbf{u}' = [u', v']^T$ its image in the camera plane.

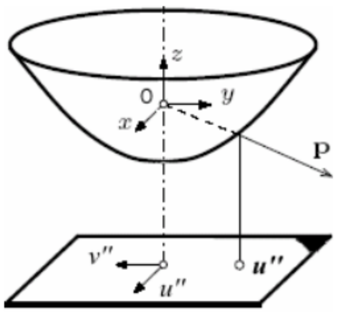


Figure 4.4: The camera image plane and the sensor plane in the case of a catadioptric camera. In the dioptric case, the sign of u'' would be reversed because of the absence of a reflective surface. All coordinates are expressed in the coordinate system placed in O, with the z axis aligned with the sensor axis.

We start by identifying two distinct reference planes: the camera image plane (u', v') (see Fig. 4.3b) and the sensor plane (u'', v'') (see Fig. 4.3a). In Figure 4.4 the two planes are shown in the catadioptric case. In case, instead, the camera is dioptric, the sign of u'' would be reversed because of the absence of a reflective surface. The coordinate system will be placed in O, with the z-axis aligned with the sensor axis. The two systems are related by an affine transformation:

$$\mathbf{u}'' = A\mathbf{u}' + \mathbf{t} \tag{4.29}$$

where $\mathbf{u}'' = [u'', v'']^T$ is the projection of a scene point $\mathbf{X}$ onto the sensor plane, $\mathbf{u}' = [u', v']^T$ is the respective point onto the image plane, $A \in \mathbb{R}^{2x2}$ and $t \in \mathbb{R}^{2x1}$.
Then, we have to introduce image projection function $\mathbf{g}$, used to describe the relationship between a point u'' and the vector $\mathbf{p}$ that goes from the viewpoint O to a scene point $\mathbf{X}$ (see Fig. 4.4), allowing us to write the model of an omnidirectional camera as:

$$\lambda\mathbf{p} = \lambda\mathbf{g}(\mathbf{u}'') = \lambda\mathbf{g}(A\mathbf{u}' + \mathbf{t}) = P\widetilde{\mathbf{X}}, \qquad \lambda > 0 \tag{4.30}$$

where $\widetilde{\mathbf{X}} \in \mathbb{R}^4$ is expressed in homogeneous coordinates and $P \in \mathbb{R}^{3x4}$ is the projection matrix.
Once the omnidirectional camera will be calibrated, we will be able to reconstruct, from each pixel, the direction of the corresponding scene point in the real world. To calibrate the camera we need to estimate A, t and the non-linear function $\mathbf{g}$ so all the vectors $\mathbf{g}(A\mathbf{u}' + \mathbf{t})$ satisfy the Equation 4.30.

To define better the model we also assume that the function $\mathbf{g}$ has the following form:

$$\mathbf{g}(u'', v'') = (u'', v'', f(\rho'')^T), \qquad \rho'' = \sqrt[2]{u''^2 + v''^2} \tag{4.31}$$

where f can be thought as rotationally symmetric with respect to the sensor axis. This assumption makes sense because, with nowadays technologies, mirrors and lenses can be shaped with high precision allowing us to suppose them symmetric.
The novel part introduced by the Scaramuzza's model is the way function f is defined, in particular, the following polynomial form has been presented:

$$f(\rho'') = a_0 + a_1\rho'' + a_2\rho''^2 + \cdots + a_N\rho''^N \tag{4.32}$$

coefficients a_i, $i = 0, 1, 2, \cdots N$ and the degree N are parameters that will be determined by the calibration process.
Putting everything together, we can rewrite the Equation 4.30 as:

$$\lambda \begin{bmatrix} u'' \\ v'' \\ w'' \end{bmatrix} = \lambda \mathbf{g}(A\mathbf{u}' + t) = \lambda \begin{bmatrix} (A\mathbf{u}' + t) \\ f(u'', v'') \end{bmatrix} = P\widetilde{\mathbf{X}}, \qquad \lambda > 0 \tag{4.33}$$

4.3.2 The Calibration Procedure

First of all, the image view field (see Fig. 4.3) is transformed from an ellipse to a circle centred on the ellipse centre allowing us to reduce the number of parameters by computing the matrices A and t up to a scale factor α. This transformation is performed automatically by following the procedure described in [47]. Now an image point $\mathbf{u}'$ is related to the corresponding point on the sensor plane $\mathbf{u}''$ by $\mathbf{u}'' = \alpha\mathbf{u}'$. By substituting it in Equation 4.33 we obtain:

$$\lambda \begin{bmatrix} u'' \\ v'' \\ w'' \end{bmatrix} = \lambda \mathbf{g}(\alpha\mathbf{u}') = \lambda \begin{bmatrix} (\alpha u') \\ (\alpha v') \\ f(\alpha\rho') \end{bmatrix} =$$

$$= \lambda\alpha \begin{bmatrix} (u') \\ (v') \\ a_0 + \cdots + a_N\rho'^N \end{bmatrix} = P\widetilde{\mathbf{X}}, \qquad \alpha, \lambda > 0 \tag{4.34}$$

where $\rho' = \sqrt[2]{u'^2 + v'^2}$. We can further lower the number of parameters $(a_0, a_1, \cdots, a_N)$ to be estimated by joining the α and the λ factors, reducing them to $N + 1$.

As said before, to calibrate the camera, images of a planar calibration pattern are acquired in different positions. Those positions are related to the camera by a rotation matrix $R = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ and a translation $\mathbf{t}$, called the extrinsic parameters. By defining I^i as an observed image of the calibration pattern, $\mathbf{M}_{ij} = [X_{ij}, Y_{ij}, Z_{ij}]^T$ as the 3D coordinates of each point in the pattern coordinate system (with $Z_{ij} = 0$ since the pattern is planar) and $\mathbf{m}_{ij} = [u_{ij}, v_{ij}]^T$ as the relative coordinates on the image plane expressed in pixel, we can transform Equation 4.34 in:

$$\lambda_{ij} \cdot \mathbf{p}_{ij} = \lambda_{ij} \begin{bmatrix} u_{ij} \\ v_{ij} \\ a_0 + \cdots + a_N \rho_{ij}^N \end{bmatrix} = P^i \widetilde{\mathbf{X}}_{ij} =$$

$$= \begin{bmatrix} \mathbf{r}_1^i & \mathbf{r}_2^i & \mathbf{r}_3^i & t^i \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^i & \mathbf{r}_2^i & t^i \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix} \qquad (4.35)$$

Then we can remove the dependence from λ_{ij} by performing vector multiplication by $\mathbf{p}_{ij}$ in both sides of the equation:

$$\lambda_{ij} \cdot \mathbf{p}_{ij} \times \mathbf{p}_{ij} = \mathbf{p}_{ij} \times \begin{bmatrix} \mathbf{r}_i^1 & \mathbf{r}_i^2 & \mathbf{r}_i^3 \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix} = 0 \Rightarrow$$

$$\Rightarrow \begin{bmatrix} u_{ij} \\ v_{ij} \\ a_0 + \cdots + a_N \rho_{ij}^N \end{bmatrix} \times \begin{bmatrix} \mathbf{r}_i^1 & \mathbf{r}_i^2 & \mathbf{r}_i^3 \end{bmatrix} \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix} = 0 \qquad (4.36)$$

and, by focusing on a single observation i, we obtain that each point of the pattern contributes to three equations:

$$v_j(r_{31}X_j + r_{32}Y_j + t_3) - f(\rho_j)(r_{21}X_j + r_{22}Y_j + t_2) = 0 \qquad (4.37)$$

$$f(\rho_j)(r_{11}X_j + r_{12}Y_j + t_1) - u_j(r_{31}X_j + r_{32}Y_j + t_3) = 0 \qquad (4.38)$$

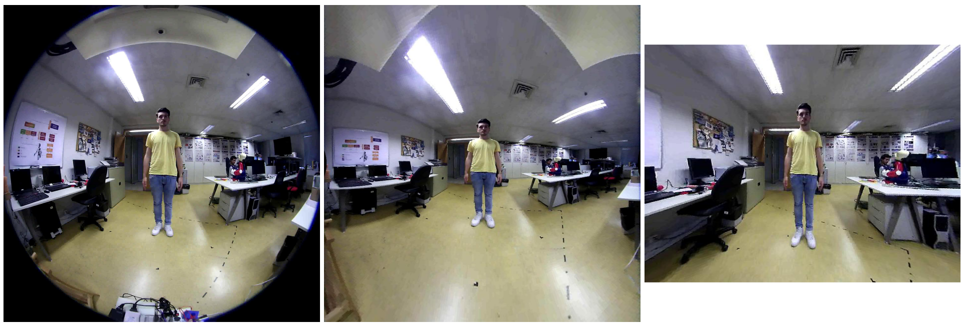
$$u_j(r_{21}X_j + r_{22}Y_j + t_2) - v_j(r_{11}X_j + r_{12}Y_j + t_1) = 0 \qquad (4.39)$$

One of the three equations we got (Equation 4.39) is linear in the unknown extrinsic parameters so, if X_j, Y_j, Z_j, u_j and v_j are all known, it is possible to get a linear estimation of all the extrinsic parameters, t_3 excluded.
The procedure to get the values of X_j, Y_j, Z_j, u_j and v_j expects the usage of a calibration pattern of a known shape and the graphical interface offered by Scaramuzza. Then, by clicking on some given points, we can retrieve the coordinates of the points in both coordinate systems: image plane and

pattern coordinate system.
Once we managed to estimate the extrinsic parameters, we can go back to Equations 4.38 and 4.39, substitute the values we got and find the least-squares solution for the camera intrinsic parameters $a_0, a_1, a_2, \cdots, a_N$ and the missing extrinsic one, t_3.
The only missing parameter is the polynomial degree N and we have to estimate it by starting from $N = 2$ and increasing it by unitary steps, computing each time the re-projection error and choosing the one with the lowest value. For all the details about the calculations, refer to [46].

4.3.3 Omnidirectional Image Unwrapping

Now, once obtained the camera parameters it is possible to perform image rectification and unwrapping. In particular I will follow these three common approaches [11, 28]:



(a) Original image *(b) Unwrapped Image* *(c) Rectified Image*

Figure 4.5: Examples of omnidirectional images unwrapping. Notice how the lines that were highly distorted in the Omnidirectional Image have been rectified in Figure 4.5c. The Unwrapped image, instead, still suffer of distortion, especially with horizontal lines but preserve the entire field of view.

- Raw Image (see Fig. 4.5a): applying the algorithm to an unwrapped omnidirectional image may seem a naive approach but, since the high distortion is not constant along the image, will be interesting to test how the algorithm performs when used in a real case.

- Cylinder Unwrapping (see Fig. 4.5b): it can be done using an open-ended cylinder plane wrapping around the sphere. The mapping between the coordinates is done so that the vertical coordinates on the resulting image are proportional to the tangent of the elevation angle

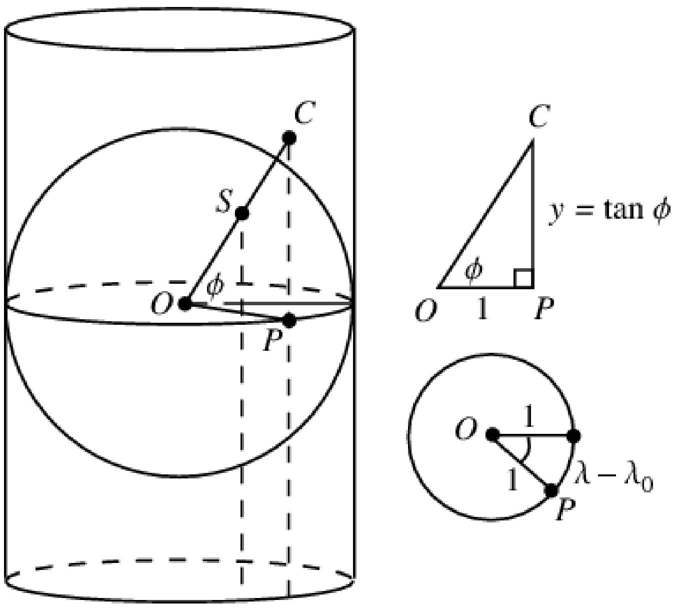


Figure 4.6: Cylindrical Projection of points on a unit sphere centered at O. Line OS is extended for each point S until it intersects a cylinder tangent to the sphere at its equator at a corresponding point C. Then Cartesian coordinates become $x = \lambda - \lambda_0$ and $y = \tan\phi$, where $\lambda - \lambda_0$ is the difference in longitude (azimuth) and ϕ is the latitude (elevation).

on the sphere and the horizontal ones are proportional to the azimuth (see Fig. 4.6).

- Camera Calibration (also called Bird's Eye View, see Fig. 4.5c): this approach allows us to look at the imaged scene as if it was done through a pinhole camera, maintaining the linearity of the imaged 3D straight lines. Drawback of this approach is that with one pinhole camera is not possible to see the entire scene so, in order to explore it entirely, I will have to apply some transformations to simulate camera rotation and then process multiple images.

Chapter 5

Unsupervised Labeller

"Once you label me you negate me."

Søren Kierkegaard.

I this chapter I will illustrate the approach I followed to build an unsupervised dataset labeller exploiting Kinect skeleton data. This comes in handy to accelerate the creation of *ad hoc* datasets and, subsequently, the detector training phase.
Notice that the scope of the Unsupervised Labeller was not to completely label the dataset but only to speed up that phase. A complete labelling with only one Kinect camera is not possible due to many technical limitations: first of all, the angle of view of the RGB camera from the Kinect is smaller than the Omnidirectional one, second, Kinect is able to track people only until a certain distance from the sensor and, finally, the tracking precision is not 100% so a final inspection by an external agent is still needed.

5.1 Point Projection

Microsoft Kinect V2 uses an anthropomorphic model to describe human skeleton, more in detail it tracks some specific points in the human body and returns their position inside the camera space (see Fig. 5.1).
I exploited these information to draw the Ground Truth Bounding Boxes around people depicted in the omnidirectional images but, in order to correctly map the points in the 3D Kinect space into the omnidirectional camera 2D space, I had to apply some transformations:

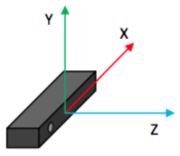
First of all, I had to map the Kinect 3D space with the Omnidirectional camera 3D space. This is realized by applying the following 4 × 4 rotation and translation matrix:

$$Rt = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

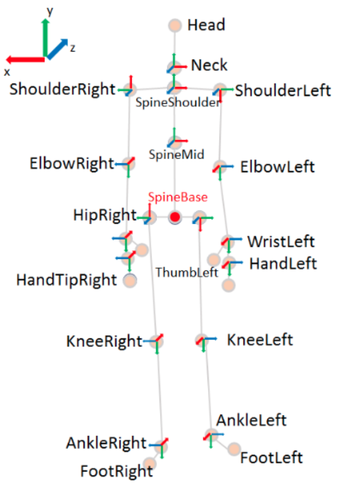
where the R_{ij} terms are the rotation components and t_i the translation ones.

Second, The 3D points from the Omnidirectional Camera 3D reference system have to be projected over the image plan. This step has been achieved using the parameters computed during the camera calibration procedure and the Scaramuzza's toolbox.

Finally, in case the labelling procedure was applied over unwrapped or rectified images, I had to transform the point coordinates according to the chosen unwrapping method (see Sections 4.3.2 and 4.3.3).



(a) Kinect Space



(b) Kinect Skeleton

Figure 5.1: Kinect Reference Systems

5.2 Kinect Calibration

Scope of the Kinect calibration phase is to compute the Rt matrix in charge of mapping points from the 3D Kinect space to 3D Omnidirectional space. In particular, the equation that must be satisfied is the following:

$$\mathbf{X}_o = Rt \cdot \mathbf{X}_k = \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ 1 \end{bmatrix} \tag{5.2}$$

where X_o are the 3D homogeneous coordinates on the Omnidirectional Camera reference system and X_k are the 3D homogeneous coordinates on the Kinect reference system.
The rotation matrix R can be built with a sequence of three translations on the three axis. In particular, the three separated rotation matrices are:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

$$R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

$$R_z = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

The general rotation matrix depends on the order of rotations, e.g. by multiplying the matrices in the order $R_z R_y R_x$ means that the rotation has been performed around the x, the y and the z axis.
In my case, since the Kinect is placed above the Omnidirectional Camera on a parallel horizontal plane (see Chapter 6 for all the details about the System Architecture), I could consider that there is no rotation around the z axis, so $\gamma = 0$, and also no translation along the z axis, so $t_3 = 0$. Consequently, the Rt matrix becomes:

$$Rt = \begin{bmatrix} \cos(\beta) & \sin(\alpha)\sin(\beta) & \cos(\alpha)\sin(\beta) & t_1 \\ 0 & \cos(\alpha) & -\sin(\alpha) & t_2 \\ -\sin(\beta) & \cos(\beta)\sin(\alpha) & \cos(\alpha)\sin(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

5.2.1 Manual Calibration

I built a graphical user interface that applies the current *Rt* matrix to some given 3D coordinates and then plots the results over a picture imaged with the Omnidirectional Camera. The GUI allows the user to tweak the four parameters (α, β, t_1 and t_2) and consequently match the plotted points with their correct position in the Omnidirectional Image in order to find the *Rt* matrix.

5.2.2 Function Minimization

This approach consider the minimization of the following function:

$$\min_{Rt} \quad \sum_{\mathbf{x}_{prj}} (\mathbf{x}_{clk} - \mathbf{x}_{prj})^2 \tag{5.7}$$

where $\mathbf{x}_{clk}$ are 2D coordinates in the Omnidirectional Camera Image Plan given by asking to the user to click on the parts of the body tracked by the Kinect skeleton, in a given order (see Fig. 5.2).
$\mathbf{x}_{prj}$ are the 2D coordinates obtained by applying the *Rt* transformation matrix to the Kinect skeletal body points, projecting them onto the omnidirectional 3D space ($\mathbf{X}_k = Rt \cdot \mathbf{X}_k$), and then multiplying the X_k with the Projection Matrix P obtained with the Camera Calibration (see Section 4.3.2).
I performed unconstrained non-linear optimization using the `fminunc` MATLAB function.

5.3 Dataset Labelling

Once I managed to obtain the complete mapping from Kinect 3D world to the Omnidirectional Camera Image Plan, I had to translate these coordinates into label coordinates, following the convention used by the Dataset Utility released by Piotr Dollar [15].
More in detail, to describe the position and the dimension of a window, four numbers are needed: the (x, y) coordinates of its upper-left corner, the width and the height.
To retrieve these values I iterate the positions of all the joints of the Kinect skeleton and, for both the x and the y coordinates, I retrieve the lowest and the highest values. The lowest ones are stored as coordinates of the upper-left corner while the differences between the hight and the lowest are used as width and height.
In the script I coded I also offered the possibility to define an amount of

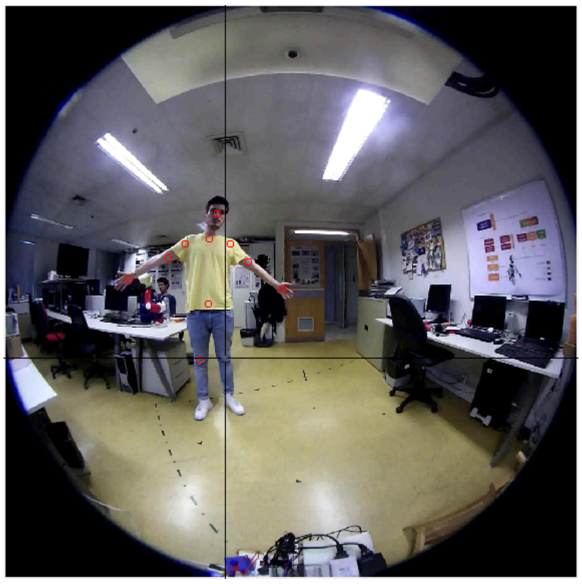


Figure 5.2: Example of Kinect auto calibration where the user to click on points on the image corresponding to Kinect skeleton joints, in a given order. These point coordinates will be later used to solve the optimization problem.

padding that will be added to all the detected windows.
In Figure 5.3 it is possible to see an example of the labeller output, notice that the two pedestrian on the left are outside from the Kinect field of view and so they have not been detected and labelled.

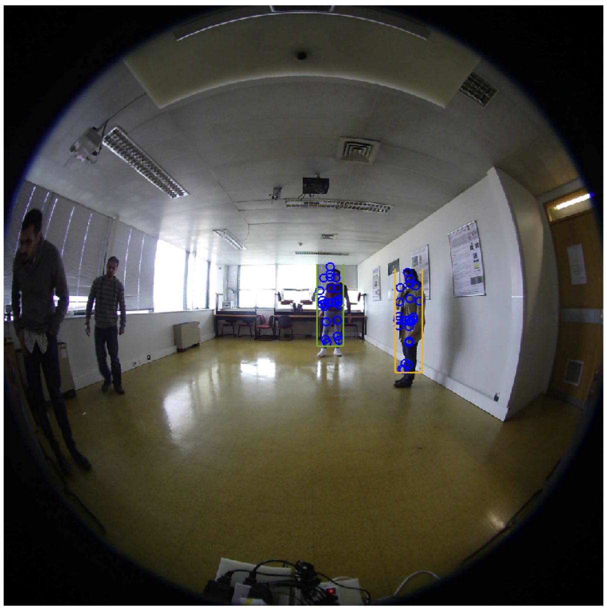


Figure 5.3: Example of image labelled using the Unsupervised Labeller with $Padding = 0$. *Notice that only the two people inside the Kinect field of view have been labelled whether the other two have been ignored and I had to label them manually.*

Chapter 6

System Architecture

"Being a robot's great, but we don't have emotions and sometimes that makes me very sad."

Bender, Futurama.

6.1 System Overview

The final version of the robotic assistive platforms will be composed of many different modules, in particular:

- Virtual Coach Module: Exploits artificial intelligence to enhance machine-human interaction performing virtual-therapist tasks;
- Monitoring Module: Is the part of the system in charge of monitoring the activity and estimating the physiological state of the patient providing an overview of the user conditions to the other modules and contingent specialists;
- Game Adaptive Training Module: Provides gamified and personalized training sessions to increase user engagement and motivation;
- Robotic Assistance Platform: Able to autonomously navigate inside a scene using human and object detection technologies and interact with the patient.

All the high-level modules mentioned above will strongly rely on the acquisition and Pedestrian Detection system I developed during my thesis work.

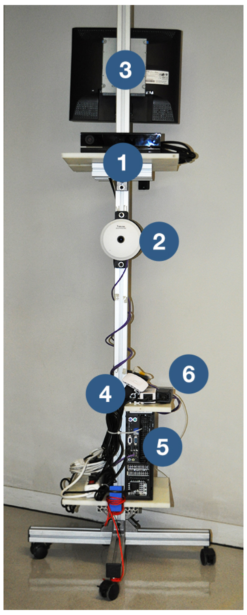


Figure 6.1: Current Version of the System. The peripherals are, in order: (1) Micorsoft Kinect V2; (2) Vivotek FE8174 Omnidirectional Cameras; (3) LCD Display and wireless keyboard; (4) Bitalino/Bioplux Biosensors; (5) Computer; (6) Eight port switch.

6.1.1 Current Version of the System

At the current version of the system the autonomous mobile platform is still missing so all the peripherals are mounted on a metal stand and the platform must be pointed by an agent toward the patient we want to monitor during the training session.
The devices connected to the system at its current state (see Fig. 6.1) are:

1. *Microsoft Kinect V2*: Motion sensing input device simultaneously acquiring RGB images, scene depth information and infra-red. Furthermore, the provided software is able to compute the body skeleton and the face expression data.

2. *Vivotek FE8174 Omnidirectional Cameras*: Sensors able to acquire images with a field of view of 180°, by using two of them it is possible to image almost the entire sphere around the system. For all the details about omnidirectional cameras, see Section 4.3.

3. *LCD Display and wireless keyboard*: Display the program interface allowing user interaction.

4. *Bitalino/Bioplux Biosensors*: Wireless real-time biosignal acquisition and transmission units, the first one is a low-cost hardware and open-source software toolkit whereas the second is a professional combinable platform. Currently the system is able to acquire the signals from both of them and we are currently performing some test to figure out the best set-up. The different sensors are placed according to the schema illustrated in Figure 6.2.

5. *Computer*: Currently all the modules are running on this machine. It is equipped with an ethernet port and a bluetooth dongle in order to acquire signals from the different peripheral devices.

6. *Eight port switch*: Allows the usage of multiple ip network cameras.

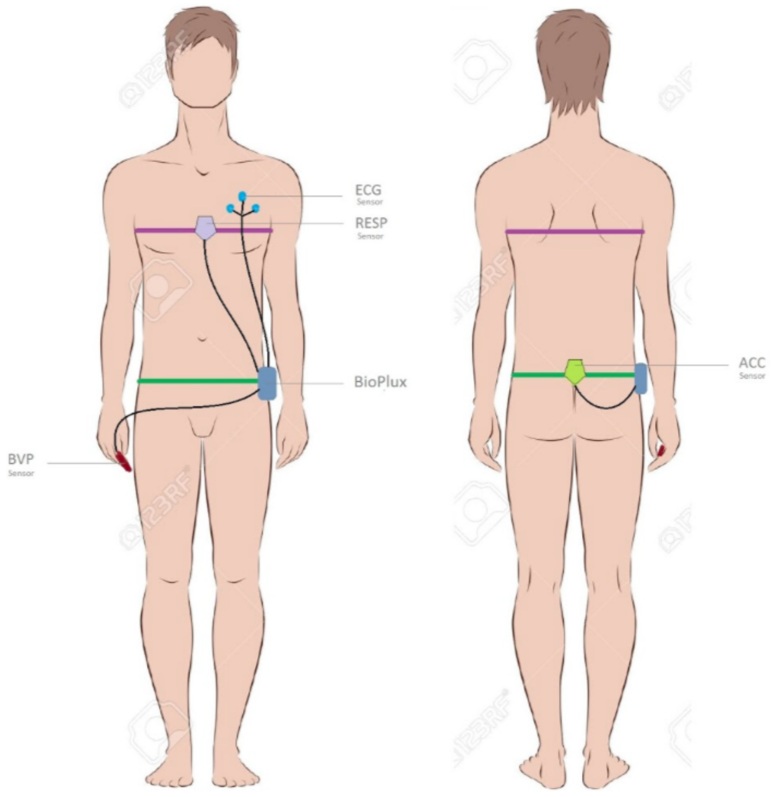


Figure 6.2: Biosensor placement schema. The available sensors are: ECG, for electrocardiography, RESP, measuring the respiration, BVP, that is for the Blood Volume Pulse and ACC, the accelerometer.

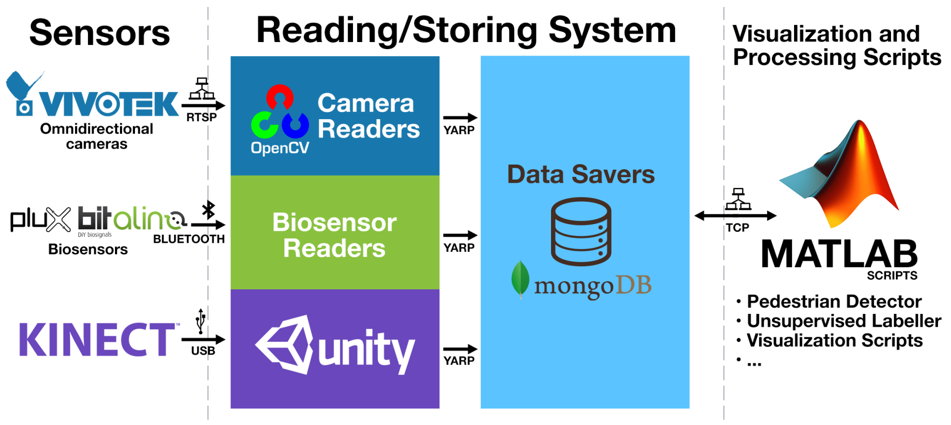


Figure 6.3: System Architecture Schema. Biosensors are connected via bluetooth whether Kinect uses and USB link and the omnidirectional cameras rely on RTSP protocol. Once acquired the signals are spread using YARP ports. Currently only the respective data savers are reading from these ports but this configuration supports the extension of the systems with new modules reading from them, in an effortless way. Aim of Reader Modules is to receive the signal, parse and store it in MongoDB database. Matlab scripts access the MongoDB dabase via TCP protocol to retrieve stored information and then visualize or process them.

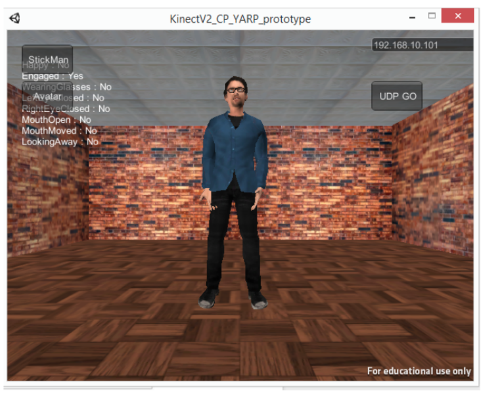
6.2 Acquisition Module

This part of the platform has fundamental role of acquiring signals and images of all the connected devices, store them in a database and provide an interface to visualize and process the acquired data.
To build the architecture I relied on YARP, a middle-ware framework that allows to build the robotic system as a collection of programs communicating using ports in a peer-to-peer way using different connection types. Thanks to YARP all the acquired data can be easily replicated and spread to different application, e.g. we can send one copy to the application in charge of storing data in a database and another copy to the application doing the real-time processing.

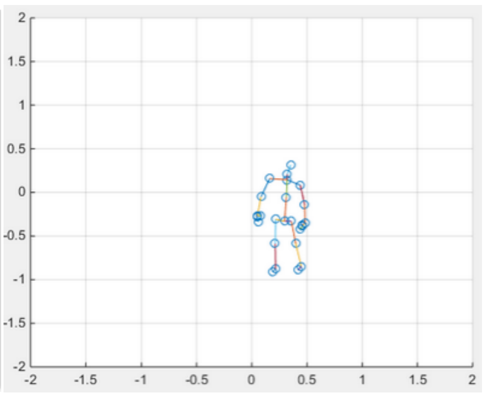
6.2.1 Software Architecture

The software architecture is currently composed by three main blocks: The acquisition applications, the storing applications, and the visualization/processing application.

The acquisition modules exploit the APIs provided by the sensors manu-

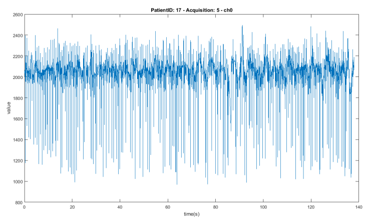


(a) Kinect Avatar

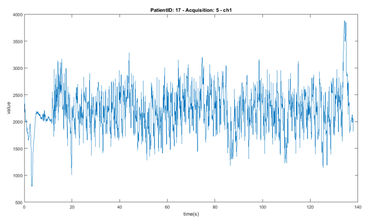


(b) Body plot

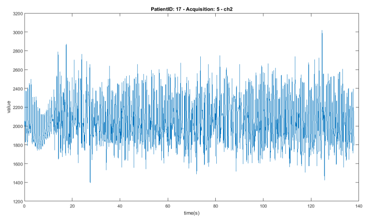
Figure 6.4: Visualization Examples. On the left there is the Kinect Avatar created with Unity whether on the right the skeletal joints connected with lines are plotted.



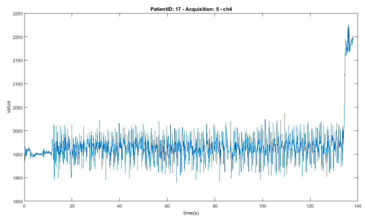
(a) Channel 0: Electrocardiogram



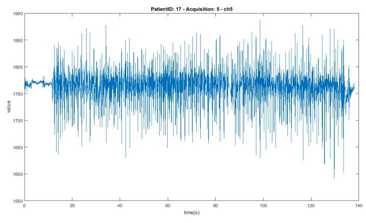
(b) Channel 1: Respiration



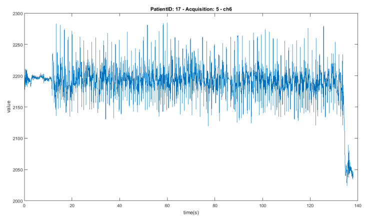
(c) Channel 2: Blood Volume Pulse



(d) Channel 4: Accelerometer on axis x



(e) Channel 5: Accelerometer on axis y



(f) Channel 6: Accelerometer on axis z

Figure 6.5: Example of data acquired using the bioplux sensor. Notice that there were no sensors connected to channels 3 and 7.

facturer to access their digital outputs. Kinect acquisition drivers represent an exception: we access Kinect data using Unity graphical engine in order to provide real time visualization of the avatar built using Kinect skeleton data (see Fig. 6.4a).
All the acquired signals are then sent out relying on YARP framework, without knowing who will receive them. YARP allows to connect many input ports to a single output port, currently this feature is not completely exploited but it will become in handy in the next stages of the project since will allow us to add new modules (e.g. on-line processing and real-time visualization) without any additional effort to manage the communication among the different agents.

The storing applications receive the signals broadcast by the acquisition modules and store them in a database. In more detail, I am using MongoDB, a NoSQL document-oriented database. Data such as biosignals and Kinect Skeleton already comes in a text format so it is stored as a document; images, instead, must be stored as binary data. More details about the structure of the database are described in Section 6.2.2.

The visualization and processing part is composed by a collection of MATLAB scripts. Currently they only allow off-line management of the stored data retrieved from the database. In the future versions we will add scripts to read data directly from the YARP ports and perform real-time operations. Some of the functions offered by the module are: Visualization and plotting of acquired data (see Fig. 6.5), detection of pedestrian over acquired images, unsupervised labelling, and image unwrapping. In Figure 6.4b it is depicted an example of Kinect skeletal data plotting.

6.2.2 Database Structure

MongoDB document data model allows to store data of any structure and dynamically modify the schema. Nevertheless, the data stored so far comply with the schema illustrated in Figure 6.6.
Bioplux and bitalino collections store the biosignals acquired from the respective device. CameraBack and cameraFront store the images from the omnidirectional cameras whether the collections which name start with "kinect" store the data acquired from Kinect, that could be the RGB images, the depth images, the infra-red images, the skeletal data or the data about the detected faces. All the previous collections have as attributes the ID of the patient the data is about, the number of the acquisition that patient is doing,

a progressively increasing number used to number the different samples part of the same acquisition, and the timestamp. The timestamps indicate the number of milliseconds since the 1st of January, 1970 (UTC).
A particular collection is the one dedicated to settings, used to store some choices made by the user until the next execution of the acquisition software.

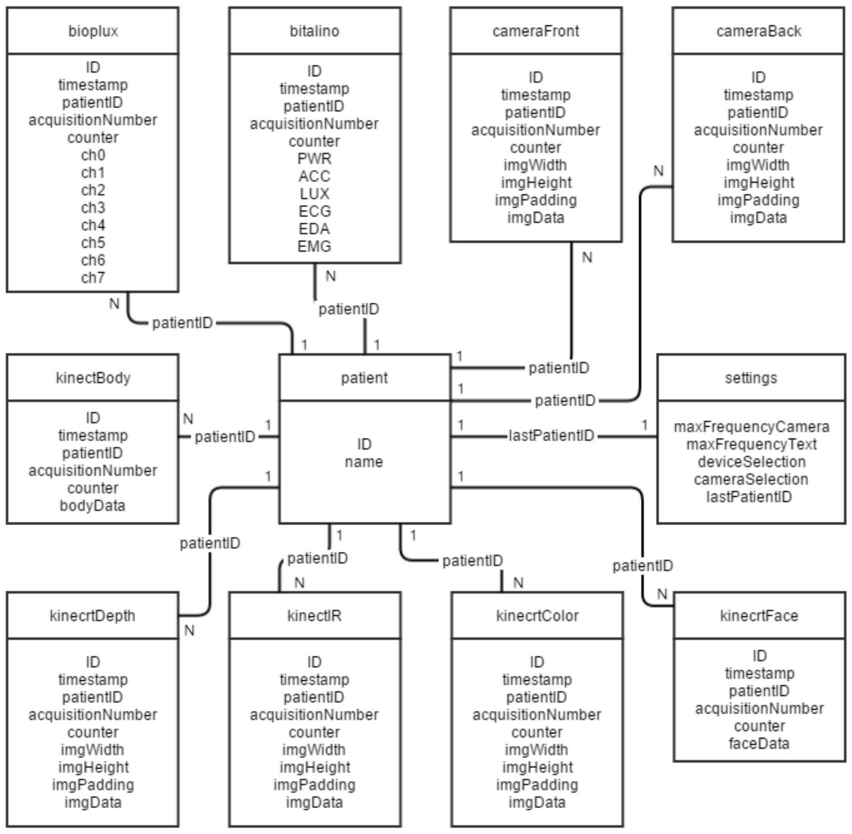


Figure 6.6: MongoDB database schema. The arrows represent 1 to N or 1 to 1 relationships. This is an indicative schema, since MongoDB is a NoSQL database, it does not guarantee that the store data follow the suggested schema.

6.2.3 Main Issues

One of the main requirements of the system was to keep all the data synchronized. To address the problem I attached to all the signals an absolute

timestamp immediately after their acquisition. This still leaves a little imprecision due to the different times the data take to go through the communication channel but the delay is small enough to not affect significantly the system synchronization.
Another problem that could affect the system is the access concurrency to the database. MongoDB already provides concurrency control measures such as locking to prevent multiple clients from modifying the same piece of data simultaneously.

Chapter 7

Experimental Results

"I never said he was a COMPLETELY successful experiment."

Dexter, Dexter's Laboratory.

In the following section I illustrate the results obtained from the experiments I made. In particular I show the output achieved by effectuating camera calibration, then I discuss the plots resulting from applying detectors using different combinations of training and testing datasets. Finally, I report the performances of the unsupervised dataset labeller.

7.1 Omnidirectional Camera Calibration

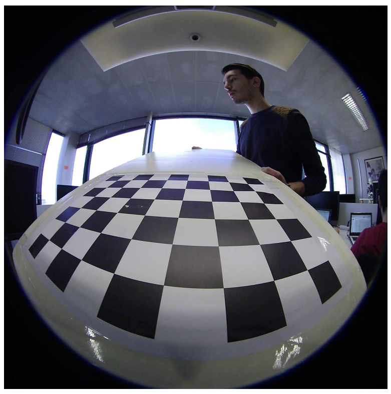


Figure 7.1: Example of image used to calibrate the Omnidirectional Camera. The calibration pattern is imaged in different positions and orientations.

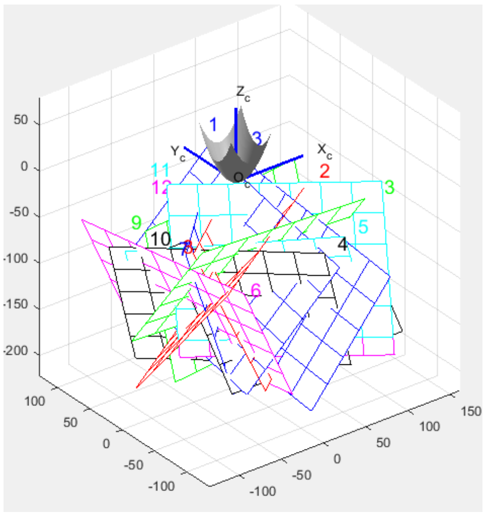


Figure 7.2: The extrinsic camera parameters. The plot shows the position of every checkerboard with respect to the reference frame of the omnidirectional camera.

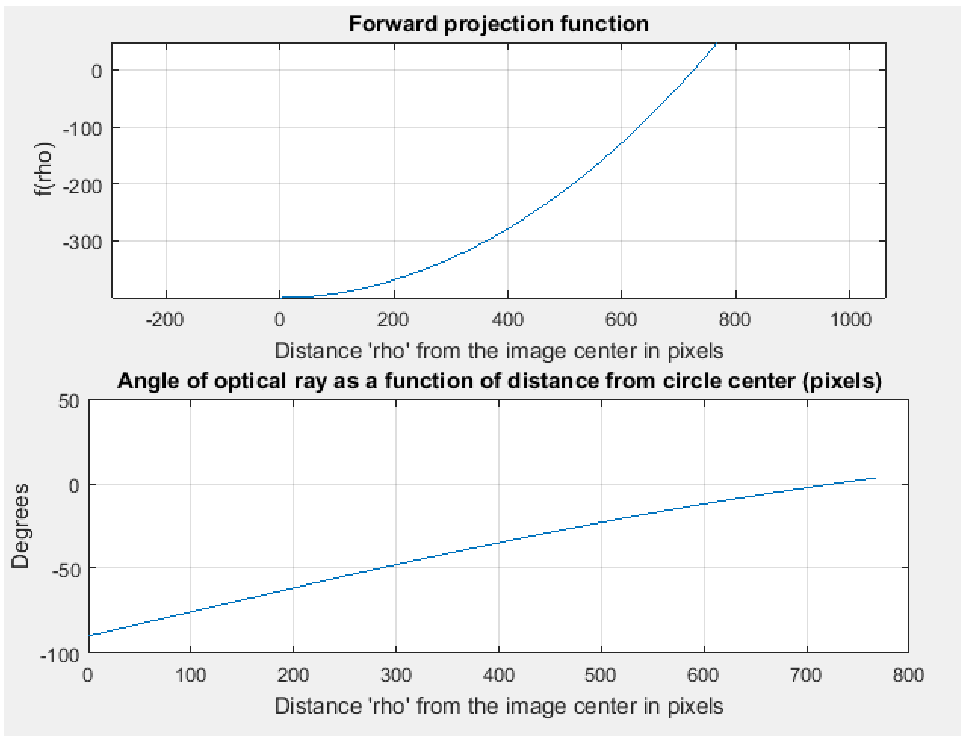


Figure 7.3: Plots of the polynomial function f, and the angle θ of the corresponding 3D vector with respect to the horizon.

To perform the Omnidirectional Camera calibration I used 13 calibration images, like the one shown in Figure 7.1, and a polynomial order of 4. For the details about Camera Calibration, please refer to Section 4.3.2.
In Figure 7.2 are represented the extrinsic parameters as a plot of every position used during calibration with the grid pattern. In Figure 7.3 are depicted the plot of the polynomial function f, and the plot of angle θ of the corresponding 3D vector with respect to the horizon.

7.2 Pedestrian Detection

Initially, I tried to apply the detector on the omnidirectional images and their unwrapped and rectified versions. Then I tried to train the detectors with custom datasets, to asses how much a different training is affecting the performances, and I ran them again over the acquired images, omnidirectional, unwrapped and rectified. From these results I also extracted some informations about detector performances in relation with position of the pedestrian inside the image and size of the window.

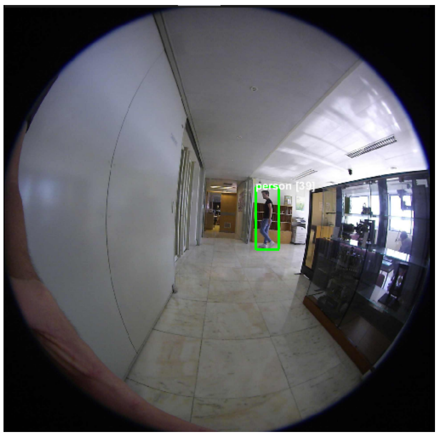
7.2.1 Dataset Description

I acquired the images in 6 different places imaging 14 different people and I flipped all the images horizontally to increase the size of the dataset. Then I randomly divided the sequence in two sets, one for training and the other one for testing. Finally I unwrapped and rectified the acquired images to obtain datasets also for these class of images. Rectification has been performed using focal distance of 3 because the images obtained using this value seemed the best compromise between the field of view width and the amount of artefacts introduced. The statistics of these datasets are illustrated in Tab. 7.1.

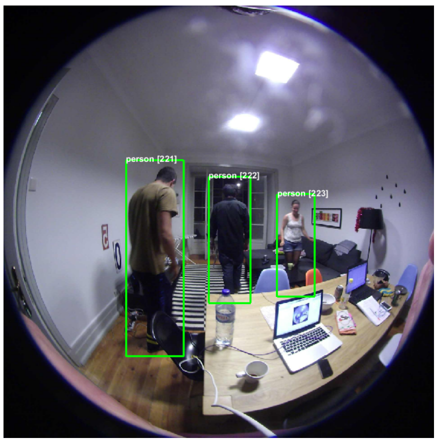
When training or testing the detectors, I used the full set of people labelled, without any kind of filtering based on size or occlusion (an example of dataset filtering is the one applied to Caltech dataset, described in [21]), making the detection more challenging.

7.2.2 Detection Results

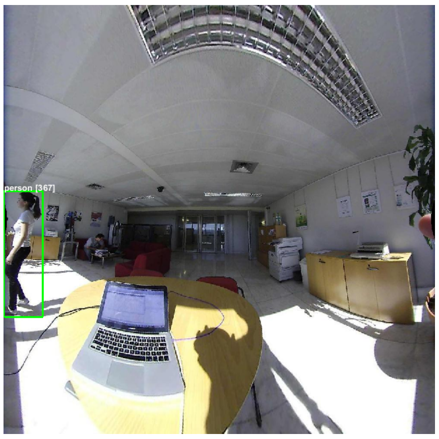
In this phase of the experiment I first applied the detector trained with the Inria and Caltech datasets, two of the most successful datasets use to train detectors by the PD community, over three datasets described in the previous



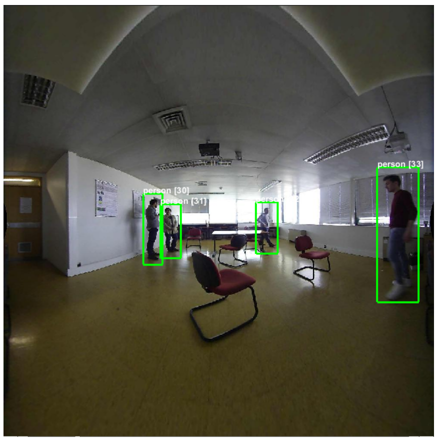
(a) Original dataset, sample number 1



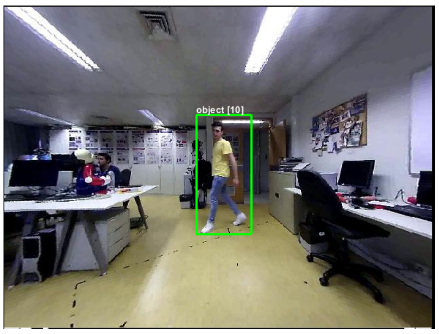
(b) Original dataset, sample number 2



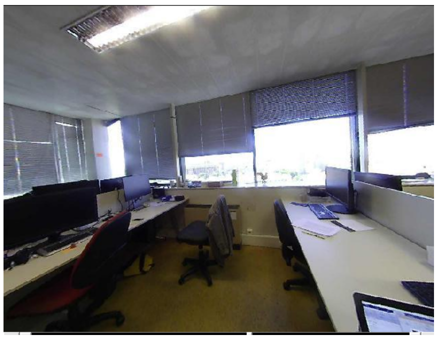
(c) Unwrapped dataset, sample number 1



(d) Unwrapped dataset, sample number 2



(e) Rectified dataset, sample number 1

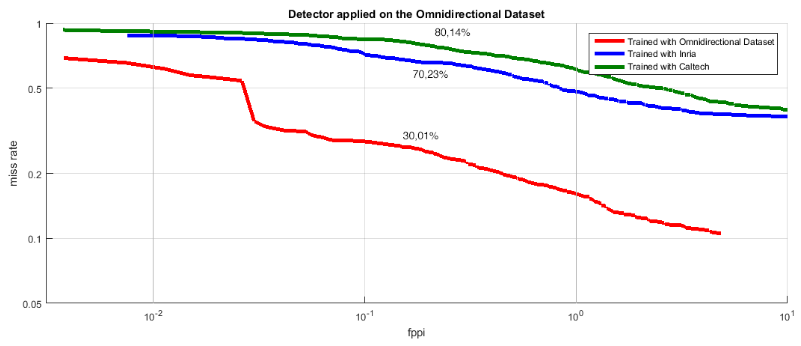


(f) Rectified dataset, sample number 2

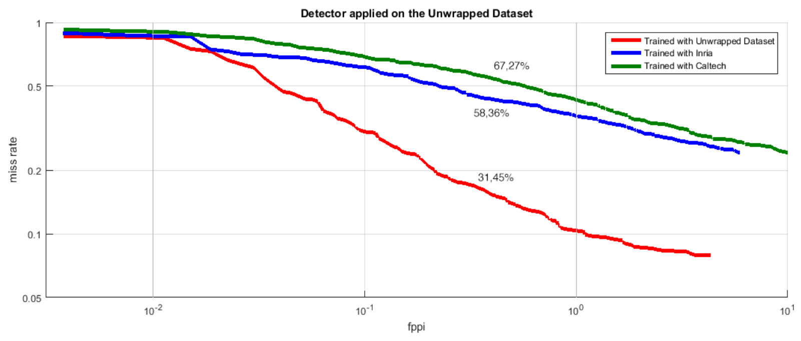
Figure 7.4: Sample images taken from the ad Hoc datasets created to train and test the detectors showing some of the different places and people imaged. Every green window represents a labelled person.

Dataset Name	Train Size	Test Size	Pedestrians in Train	Pedestrians in Test
Omnidirectional	261	267	607	679
Unwrapped	261	267	607	679
Rectified	261	267	508	547

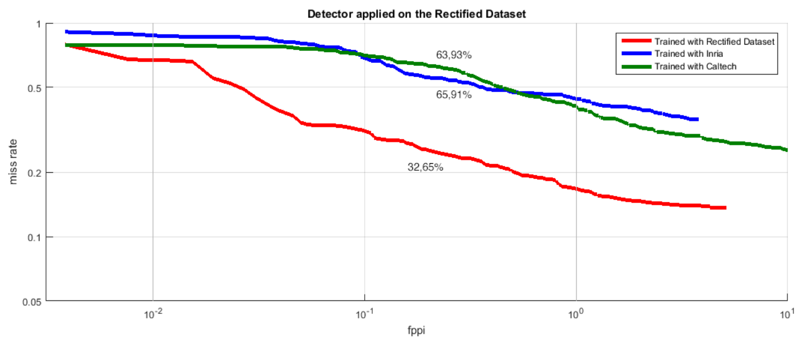
Table 7.1: Dataset Statistics



(a) Detector applied on the Omnidirectional Dataset



(b) Detector applied on the Unwrapped Dataset



(c) Detector applied on the Rectified Dataset

Figure 7.5: Detection Results comparing, for the three different cases, the performance of the detector when trained with the dataset of the same kind versus the performance reached with INRIA or Caltech trainings.

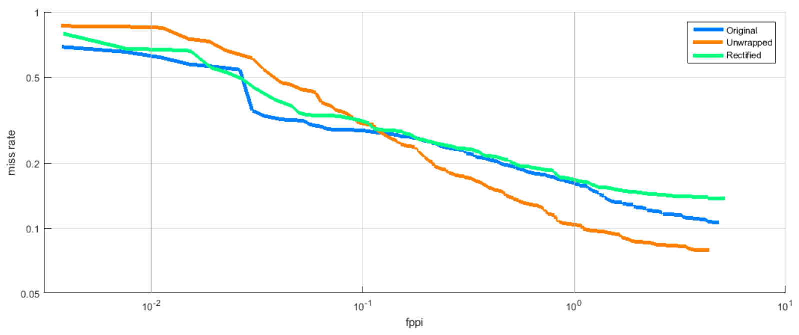


Figure 7.6: Comparison of detector performances for the three cases when it is trained and tested with datasets of the same kind.

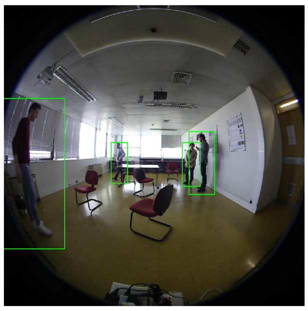
section in order to assess how standard PD tools perform in this particular situation.
From the plots in 7.5, it is quite clear that the detector performs better when applied to unwrapped or rectified images but, still, the performances are far from the State of the Art values.

Then I trained the detector with the new training sets and I applied it in the respective test set (e.g. I ran the detector trained using unwrapped images over the test set composed of the same kind of images) to assess the impact of a different training. The results of these experiments are illustrated in Figure 7.5.
On all the three datasets the performances of the *ad Hoc* detector improved significantly.

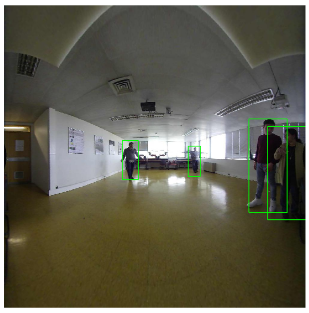
In Figure 7.6 the performances of the detector when trained and tested with sets of the same kind are showed, pointing out that the usage of unwrapped images outperform the usage of the original ones only when the number of False Positives per Image is higher than 10^{-1}. The performances of the detector that uses rectified images are generally worse than the one that uses original images, outperforming them only in few points.

7.2.3 Performance Evaluation

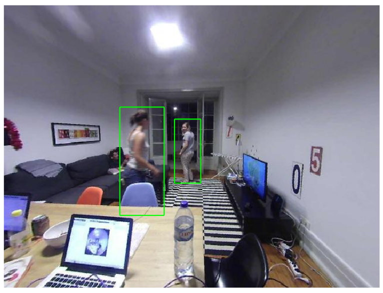
In Figure 7.7 some successful examples where the detector managed to correctly locate all the people in a scene are shown. Unfortunately, in some other cases, the detector fails in properly detect people or returns False Positive detections, windows that do not enclose people correctly or do not enclose



(a) Original

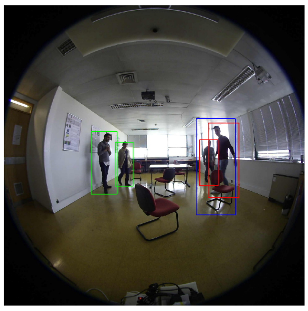


(b) Unwrapped

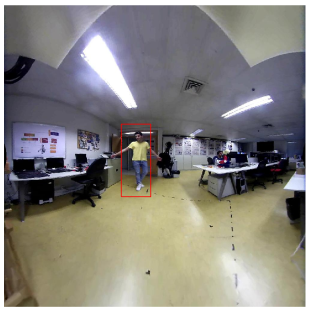


(c) Rectified

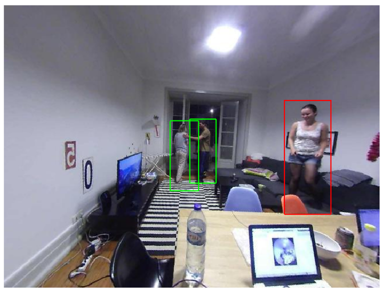
Figure 7.7: Sample images showing successful detection cases in chellenging situations. The green windows reprent a correct detection.



(a) Original



(b) Unwrapped



(c) Rectified

Figure 7.8: Sample images showing situations in which the detector fails. The green windows reprent a correct detection, the blue ones represent false positives and the red ones represent missed detections.

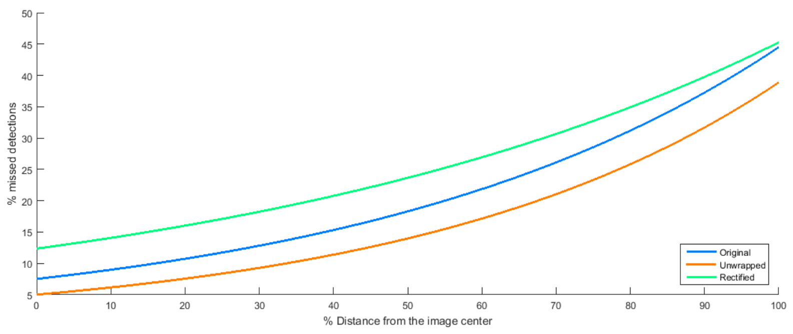


Figure 7.9: Plot representing the three non-linear functions fitting the experimental data about the missed detections over the euclidean distance from the image center, obtained from the previous experiments. Notice that the distance from the image center is rescaled over an interval from 0 *to* 100*.*

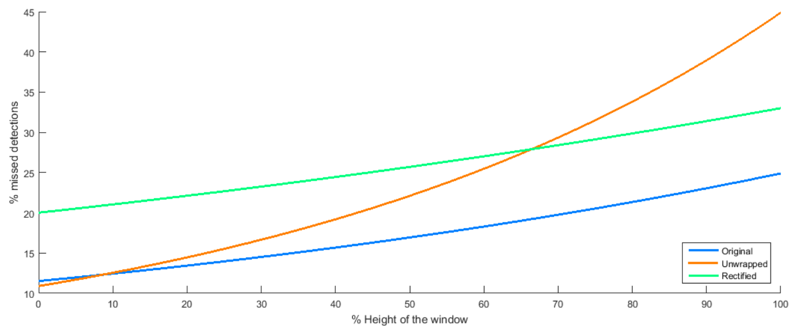


Figure 7.10: Plot representing the three non-linear functions fitting the experimental data about the missed detections over the window height, obtained from the previous experiments. Notice that the window height is rescaled over an interval from 0 *to* 100*.*

people at all.

To better explore and understand in which conditions the detector was failing, I coded a MATLAB script to analyse the results of the previous experiments. In particular, I extracted information about the amount of missed detections related to the distance of the person from the image center and the size of the Bounding Box.
After collecting data about the detections missed when applying the detector trained with the *ad Hoc* datasets on the respective ones, I ran the script described above.
More in detail, the *Distance from the Image Center* of a person is the euclidean distance of center of the the Bounding Box to the image center.
The condition to consider a detected window as a correct detection is described in Section 2.1.3. I considered as Missed Detections all the Ground Truth windows without a correspondent detection. As a threshold for the confidence value I considered the arithmetic mean computed over all the confidence values in the Ground Truth.
In Figures 7.9 and 7.10 the percentage of missed detections is put in relation with the distance from the image center and with the size of the Bounding Box, for all the three datasets.
To better compare the results I pooled the distance from the image centre and the window size values in 30 bins and then I rescaled the values from 1 to 100. Then I applied a non-linear regression to the resulting experimental data using the `fitnlm` MATLAB function. Beyond the data used to estimate

the non-linear model, this function requires three more parameters:

- The functional form of the model, I used the exponential function $\alpha e^{\beta x}$;
- The origin, I chose the $(0, 0)$ point;
- The weight attributed to each data value. This is an optional parameter and is used to give more importance to the values obtained from an higher number of observations. In my case I used as a weight vector the number of Ground Truth entries for each bin.

From the plot in Figure 7.9 we can observe that the curve corresponding to the original dataset is steeper than the ones corresponding to the unwrapped and rectified datasets, meaning that, in the original case, performances degrade more when people to be detected are close to the image border.
The plot in Figure 7.10, instead, shows us that, whether the original and the rectified datasets do not suffer that much people proximity, the unwrapped set is very sensitive to vicinity and the performances degrade quickly when the window size increases.

7.3 Unsupervised Labeller

To test the effectiveness of the Unsupervised Labeller I used subset of the original dataset and I compared the labelling obtained from the Unsupervised Labeller with the one I did manually, considering as correct detection all the detections satisfying the following rule:

$$a_o = \frac{area(BB_{kinect} \cap BB_{manual})}{area(BB_{kinect} \cup BB_{manual})} > 0.5 \tag{7.1}$$

In particular, I ran three different test using three different *Rt* matrices:

1. In the first case I obtained the *Rt* matrix by using the GUI and manually tweaking the parameters. The system correctly located the 64% of the labels inside the Kinect field of view.
2. In the second case I used the *Rt* matrix obtained by solving the optimization problem, using as initial guess the zero 4×4 matrix. In this case the successful detections were the 56%.
3. Finally, I obtained the *Rt* matrix by solving the optimization problem again but this time by using as initial guess the *Rt* matrix obtained at the point 1. With this settings the system correctly detected the 60% of the labels.

Chapter 8

Conclusions and future work

"End? No the journey doesn't end here."

Gandalf, The Return of the King.

In this thesis I developed a system able to acquire, visualize and store signals from multiple peripherals. The system supports the acquisition of biosignals, images from omnidirectional cameras and data from Microsoft Kinect V2 and it is robust to synchronization and concurrency problems.
Exploiting skeletal information provided by Kinect I could track movements of people and, by using the biosignals, it will be possible to assess their health condition. Omnidirectional cameras, instead, are used to detect people in the environment around the system and they will be essential for the human-robot interaction module of the system.
I also studied how imaging systems are modelled. In particular, I learnt how to model and calibrate omnidirectional cameras. I performed the calibration of the two Vivotek FE8174 cameras used in the system. The parameters obtained from this phase have been used to process the omnidirectional images and perform unwrapping and rectification.
I acquired a reasonable amount of omnidirectional images and I used them to build a train and a test dataset. Finally, I processed the two datasets to create train and test unwrapped and rectified datasets.
With the test sets mentioned above I evaluated the performance of one of the state-of-the-art Pedestrian Detectors when ran on these kinds of images. I found out that using the processed version of the omnidirectional images it is possible to improve the performances obtaining a log-average miss rate 10-20% lower. The best result was a log-average miss rate of 58,36%, obtained when using a detector trained with the INRIA dataset on the unwrapped test set. This value is still too high and unsatisfactory for uses in real cases.

To improve the performance of the detector I tried to train it with the train sets acquired beforehand and then I applied the detector on the respective test set. In all the three cases the performances improved considerably. In particular, the omnidirectional case was the one performing better in terms of log-average miss rate (30,01%) but, if considering more than 10^{-1} false positives per image, the unwrapped set is the one performing better.
A closer look to the results showed us the weak points of the different methods, in particular I found out that the original dataset is the most sensitive to the distance from the image center whether the cylindrical one fails more when people are too close to the camera. The rectified dataset did not show a particular sensitivity to these two aspects, meaning that we have to look somewhere else to improve the performances with this kind of images.
In light of the obtained results I opted for the detector trained with the unwrapped images. The performances of the unwrapped and the original case are close. The first one is performing a little bit worse in the image center, probably due to some artefacts introduced with the unwrapping procedure, but it is more reliable when people are close to the image boundary and, in general, its performances are more stable along the field of view. The degrade of performances related to people closeness is not a big deal since, in the project context, people will not be often so close to the camera.

Finally, I built an unsupervised dataset labeller that exploits skeleton data acquired by the Kinect to autonomously label datasets and accelerate the task. Experimental results showed that Kinect is not able to label 100% of the people in its field of view (it probably fail when they are too close or too far) but can still accelerate the task. I obtained the best results when using the manually calibrated transformation matrix.

In future work I intend to extend the *ad Hoc* datasets to alleviate the overfitting side effects and, in the case of rectified images, try different focal length. I also plan to test the detector on images composed by joining the frontal and the rear fish-eye images testing, in particular, the performance in the area corresponding to the two blind spots between the two cameras. Eventually, I also plan to explore different ways to improve the performance of the detector, e.g. testing modified features thought particularly to deal with omnidirectional highly-distorted images.
About the unsupervised labeller, I am planning to assess if it is possible to improve its performances when using the *Rt* matrix obtained by solving the optimization problem. Possible ways that can be explored are: (i) increasing the number of points provided by the user; (ii) try different minimization

techniques.

Bibliography

[1] Caltech Pedestrian Dataset: Evaluated Algorithms. pages 1–4.

[2] Point Grey Research, Product Catalogue.

[3] Vivotek, Product catalogue, 2015.

[4] Simon Baker and Shree K Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2):175–196, 1999.

[5] Subhashis Banerjee. Camera Models and Affine Multiple Views Geometry. 2001.

[6] Joao P Barreto and Helder Araujo. Issues on the geometry of central catadioptric image formation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–422. IEEE, 2001.

[7] Joao P Barreto and Helder Araujo. Geometric properties of central catadioptric line images and their application in calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1327–1333, 2005.

[8] Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, DTIC Document, 1977.

[9] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE, 2012.

[10] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *Computer Vision-ECCV 2014 Workshops*, pages 613–627. Springer, 2014.

[11] Nguan Soon Chong, Yau Hee Kho, and Mou Ling Dennis Wong. A closed form unwrapping method for a spherical omnidirectional view sensor. *EURASIP Journal on Image and Video Processing*, 2013(1):1–17, 2013.

[12] Ibrahim Cinaroglu and Yalin Bastanlar. A direct approach for human detection with catadioptric omnidirectional cameras. In *Signal Processing and Communications Applications Conference (SIU), 2014 22nd*, pages 2275–2279. IEEE, 2014.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[15] Piotr Dollár. Piotr's image and video matlab toolbox (pmt). *Software available at: http://vision. ucsd. edu/˜ pdollar/toolbox/doc/index. html*, 2013.

[16] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1532–1545, 2014.

[17] Piotr Dollár, Ron Appel, and Wolf Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision-ECCV 2012*, pages 645–659. Springer, 2012.

[18] Piotr Dollár, Serge Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *BMVC*, volume 2, page 7. Citeseer, 2010.

[19] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *BMVC*, volume 2, page 5, 2009.

[20] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE, 2009.

[21] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[22] Markus Enzweiler and Darieu M Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, 2009.

[23] Andreas Ess, Bastian Leibe, and Luc Van Gool. Depth and appearance for mobile scene analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[24] Dario Figueira, Matteo Taiana, Athira Nambiar, Jacinto Nascimento, and Alexandre Bernardino. The hda+ data set for research on fully automated re-identification systems. In *Computer Vision-ECCV 2014 Workshops*, pages 241–255. Springer, 2014.

[25] Andrew W Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–125. IEEE, 2001.

[26] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[27] Lufang Gao. Human detection by omni-directional camera. 2011.

[28] Jose Gaspar, Niall Winters, Etienne Grossmann, and Jose Santos-Victor. Toward robot perception through omnidirectional vision. In *Innovations in Intelligent Machines-1*, pages 223–270. Springer, 2007.

[29] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[30] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical implications. In *Computer Vision-ECCV 2000*, pages 445–461. Springer, 2000.

[31] Chunhui Gu, Jasmine J Lim, Pablo Arbeláez, and Jagannath Malik. Recognition using regions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1030–1037. IEEE, 2009.

[32] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *Computer Vision–ECCV 2012*, pages 459–472. Springer, 2012.

[33] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[34] Kenneth D Kochanek, Sherry L Murphy, Jiaquan Xu, and Elizabeth Arias. Mortality in the united states, 2013. *NCHS data brief*, 178:1–8, 2014.

[35] Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 878–885. IEEE, 2005.

[36] Miguel Lourenço. *Keypoint Detection, Matching, and Tracking in Images with Non-linear Distortion: Applications in Medical Endoscopy and Panoramic Vision*. PhD thesis, 2014.

[37] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3945–3950. IEEE, 2007.

[38] Bjoern H Menze, B Michael Kelm, Daniel N Splitthoff, Ullrich Koethe, and Fred A Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[39] Woonhyun Nam, Piotr Dollár, and Joon Hee Han. Local decorrelation for improved detection. *arXiv preprint arXiv:1406.1134*, 2014.

[40] Athira Nambiar, Matteo Taiana, Dario Figueira, Jacinto C Nascimento, and Alexandre Bernardino. A multi-camera video dataset for research on high-definition surveillance. *International Journal of Machine Intelligence and Sensory Signal Processing*, 1(3):267–286, 2014.

[41] Sakrapee Paisitkriangkrai, Chunhua Shen, and Anton van den Hengel. Pedestrian detection with spatially pooled features and structured ensemble learning. *arXiv preprint arXiv:1409.5209*, 2014.

[42] Luis Puig, Yalin Bastanlar, Peter Sturm, José Jesús Guerrero, and João Barreto. Calibration of central catadioptric cameras using a dlt-like approach. *International Journal of Computer Vision*, 93(1):101–114, 2011.

[43] Luis Puig, Jesús Bermúdez, Peter Sturm, and José Jesús Guerrero. Calibration of omnidirectional cameras in practice: A comparison of methods. *Computer Vision and Image Understanding*, 116(1):120–137, 2012.

[44] Martin Rufli, Davide Scaramuzza, and Roland Siegwart. Automatic detection of checkerboards on blurred and distorted images. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3121–3126. IEEE, 2008.

[45] Davide Scaramuzza. Omnidirectional camera. In *Computer Vision*, pages 552–560. Springer, 2014.

[46] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 45–45. IEEE, 2006.

[47] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5695–5701. IEEE, 2006.

[48] Pierre Sermanet, Koray Kavukcuoglu, Sandhya Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3626–3633. IEEE, 2013.

[49] Larry S Shapiro, Andrew Zisserman, and Michael Brady. 3d motion recovery via affine epipolar geometry. *International Journal of Computer Vision*, 16(2):147–182, 1995.

[50] Matteo Taiana, Jacinto Nascimento, and Alexandre Bernardino. On the purity of training and testing data for learning: The case of pedestrian detection. *Neurocomputing*, 150:214–226, 2015.

[51] Matteo Taiana, Jacinto C Nascimento, and Alexandre Bernardino. An improved labelling for the inria person data set for pedestrian detection. In *Pattern Recognition and Image Analysis*, pages 286–295. Springer, 2013.

[52] Daniel W Thompson and Joseph L Mundy. Three-dimensional model matching from an unconstrained viewpoint. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 208–220. IEEE, 1987.

[53] Antonio Torralba, Alexei Efros, et al. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[54] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[55] Stefan Walk, Nikodem Majer, Konrad Schindler, and Bernt Schiele. New features and insights for pedestrian detection. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 1030–1037. IEEE, 2010.

[56] Christian Wojek, Stefan Walk, and Bernt Schiele. Multi-cue onboard pedestrian detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 794–801. IEEE, 2009.

[57] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[58] Cha Zhang and Paul A Viola. Multiple-instance pruning for learning efficient cascade detectors. In *Advances in Neural Information Processing Systems*, pages 1681–1688, 2008.