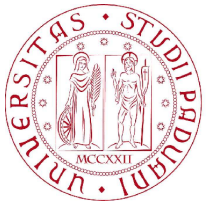
UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

Dipartimento di Ingegneria dell'Informazione
Scuola di Dottorato di Ricerca in Ingegneria dell'Informazione
Indirizzo: Ingegneria informatica ed elettronica industriali
XXI ciclo

Visual-SLAM for Humanoid Robots

Ph.D. School Head: Ch.mo Prof. Matteo Bertocco

Advisor: Ch.mo Prof. Enrico Pagello

Ph.D. Student: Alberto Pretto

Sommario

Nell'ambito della robotica, il Simultaneous Localization and Mapping (SLAM) è il processo grazie al quale un robot autonomo è in grado di creare una mappa dell'ambiente circostante e allo stesso tempo di localizzarsi avvalendosi di tale mappa. Negli ultimi anni un considerevole numero di ricercatori ha sviluppato nuove famiglie di algoritmi di SLAM, basati su vari sensori e utilizzando varie piattaforme robotiche.
Uno degli ambiti più complessi nella ricerca sullo SLAM è il cosiddetto Visual-SLAM, che prevede l'utilizzo di vari tipi di telecamera come sensore per la navigazione. Le telecamere sono sensori economici che raccolgono molte informazioni sull'ambiente circostante. D'altro canto, la complessità degli algoritmi di visione artificiale e la forte dipendenza degli approcci attualmente realizzati dalle caratteristiche dell'ambiente, rendono il Visual-SLAM un problema lontano dal poter essere considerato risolto.

Molti degli algoritmi di SLAM sono solitamente testati usando robot dotati di ruote. Sebbene tali piattaforme siano ormai robuste e stabili, la ricerca sulla progettazione di nuove piattaforme robotiche sta in parte migrando verso la robotica umanoide. Proprio come gli esseri umani, i robot umanoidi sono in grado di adattarsi ai cambiamenti dell'ambiente per raggiungere efficacemente i propri obiettivi.
Nonostante ciò, solo pochi ricercatori hanno focalizzato i loro sforzi su implementazioni stabili di algoritmi di SLAM e Visual-SLAM adatti ai robot umanoidi. Tali piattaforme robotiche introducono nuove problematiche che possono compromettere la stabilità degli algoritmi di navigazione convenzionali, specie se basati sulla visione. I robot umanoidi sono dotati di un alto grado di libertà

di movimento, con la possibilità di effettuare velocemente movimenti complessi: tali caratteristiche introducono negli spostamenti vibrazioni non deterministiche in grado di compromettere l'affidabilità dei dati sensoriali acquisiti, per esempio introducendo nei flussi video effetti indesiderati quali il motion blur. A causa dei vincoli imposti dal bilanciamento del corpo, inoltre, tali robot non sempre possono essere dotati di unità di elaborazione molto performanti che spesso sono ingombranti e dal peso elevato: ciò limita l'utilizzo di algoritmi complessi e computazionalmente gravosi. Infine, al contrario di quanto accade per i robot dotati di ruote, la complessa cinematica di un robot umanoide impedisce di ricostruire il movimento basandosi sulle informazioni provenienti dagli encoder posti sui motori.

In questa tesi ci si è focalizzati sullo studio e sullo sviluppo di nuove metodologie per affrontare il problema del Visual-SLAM, ponendo particolare enfasi ai problemi legati all'utilizzo di piccoli robot umanoidi dotati di una singola telecamera come piattaforme per gli esperimenti.
I maggiori sforzi nell'ambito della ricerca sullo SLAM e sul Visual-SLAM si sono concentrati nel campo del *processo di stima dello stato del robot*, ad esempio la stima della propria posizione e della mappa dell'ambiente. D'altra parte, la maggior parte delle problematiche incontrate nella ricerca sul Visual-SLAM sono legate al *processo di percezione*, ovvero all'interpretazione dei dati provenienti dai sensori. In questa tesi ci si è perciò concentrati sul *miglioramento dei processi percettivi* da un punto di vista della visione artificiale.
Sono stati affrontati i problemi che scaturiscono dall'utilizzo di piccoli robot umanoidi come piattaforme sperimentali, come ad esempio la bassa capacità di calcolo, la bassa qualità dei dati sensoriali e l'elevato numero di gradi di libertà nei movimenti. La bassa capacità di calcolo ha portato alla creazione di un nuovo metodo per misurare la *similarità* tra le immagini, che fa uso di una descrizione dell'immagine compatta, utilizzabile in applicazioni di SLAM topologico. Il problema del motion blur è stato affrontato proponendo una nuova tecnica di rilevamento di feature visive, unitamente ad un nuovo schema di tracking, robusto anche in caso di motion blur non uniforme. E' stato altresì sviluppato un framework per l'odometria basata sulle immagini, che fa uso delle feature visive presentate.

Si propone infine un approccio al Visual-SLAM basato sulle omografie, che sfrutta le informazioni ottenute da una singola telecamera montata su un robot umanoide. Tale approccio si basa sull'assunzione che il robot si muove su una superficie piana.
Tutti i metodi proposti sono stati validati con esperimenti e studi comparativi, usando sia dataset standard che immagini acquisite dalle telecamere installate su piccoli robot umanoidi.

Abstract

In robotics the Simultaneous Localization and Mapping (SLAM) is the problem in which an autonomous robots acquires a map of the surrounding environment while at the same time localizes itself inside this map. In the last years a lot of researchers have spent a great effort in developing new families of algorithms, using several sensors and robotic platforms.
One of the most challenging field of research in SLAM is the so called Visual-SLAM problem, in which various types of cameras are used as sensor for the navigation. Cameras are inexpensive sensors and can provide rich information about the surrounding environment, on the other hand the complexity of the computer vision tasks and the strong dependence on the characteristics of the environment in current approaches makes the Visual-SLAM far to be considered a closed problem.

Most of the SLAM algorithm are usually tested on wheeled robot. These platforms have become robust and stable, on the other hand the research in robot design moves toward a new family of robot platforms, the humanoid robots. Just like humans, a humanoid robot can adapt itself to changes in the environment in order to efficiently reach its goals.
Despite that, only a few roboticists focused theirs research on stable implementation of SLAM and Visual SLAM algorithms well suited for humanoid robots. Humanoid platforms raise issues which can compromise the stability of the conventional navigation algorithms, especially for vision-based approaches. A humanoid robot can move in 3D without the usual planar motion assumption that constraint the movement in 2D, usually with quick and complex movements combined with unpredictable vibrations, compromising the reliability of the acquired

sensors data, for example introducing in the images grabbed by the camera an undesired motion blur effect. Due to the strong balance constraints, a humanoid robot usually can't be equipped with powerfull but hefty computer boards: this limits the implementation of complex and computational expensive algorithms. Moreover, unlike wheeled robots, its complex kinematics usually forbids a reliable reconstruction of the motion from the servo-motor encoders.

In this thesis, we focus on studying and developing new techniques addressing the Visual-SLAM problem, with particular attention to the issues related to using as experimental platform small humanoid robots equipped with a single perspective camera.
The main efforts in SLAM and Visual SLAM research areas have been put into the *estimation functionality*. However, most of the functionalities involved in Visual SLAM are in *perception processes*. In this thesis we therefore focus on the *improvement of the perceptual processes*, from a computer vision point-of-view.
We faced small humanoid robot related issues like low-computational capability, the low quality of the sensor data and the high degrees of freedom of the motion. We cope with the low computational resources presenting a new similarity measure for images based on a compact signature to be used in image-based topological SLAM problem. The motion blur problem is faced proposing a new feature detection and tracking scheme that is robust even to non-uniform motion blur. We develop a framework for visual odometry based on features robust to motion blur.
We finally propose an homography-based approach to 3D visual SLAM, using the information provided by a single camera mounted on a humanoid robot, based on the assumption that the robot moves on a planar environment.
All proposed methods have been validated with experiments and comparative validation using both standard datasets and images taken by the cameras mounted on walking small humanoid robots.

Contents

List of Figures

Chapter 1

Introduction

One of the most fundamental feature of an autonomous mobile robot is the capability to localize itself inside the environments where it moves. Without the knowledge of its own position, a robot can't perform complex tasks as rescue, surveillance, or fetch and carry.

In order to provide a robot with localization capabilities, the programmer must give it a representation of the environment (map) where it will move. For many reasons, this representation is not always available, for example because of the working area is not known *a priori* (as in the case of a rescue robot).

Generating incrementally consistent maps of the environment while locating itself within this map is therefore another fundamental task of mobile robots, more general than the localization, and obviously more challenging. In robotics this capability is commonly referred as the *Simultaneous Localization and Mapping* (*SLAM*) problem [29], and in the last years it has received much attention within the research community.

SLAM has been formulated and solved as a theoretical problem in a number of different ways and many researchers presented several implementations using different robotic platforms and sensors. However, SLAM still remains an open problem, due to the strong dependency of almost all current implementations on the specific environment and the specific sensors used.

Moreover, the capability of autonomously navigate in a unknown environment becomes critically important in indoor application, where no global positioning systems as GPS are available.

1.1 Solving the SLAM problem: a brief introduction

A mobile robot is commonly equipped with *proprioceptive* sensors, (e.g., motor's encoders) and *exteroceptive* sensors (e.g., range finders or color cameras)[94]. When it moves it acquires observations from both these types of sensor, collecting data, unavoidably corrupted by noise, about its internal state and the external state. SLAM aims to reconstruct a map of the world and the actual position of the robot using only these data streams. The map is usually represented through a vector of landmark positions or through an occupancy grid of the environment. The map together with the robot's pose represents the *state* that should be estimated during the SLAM process.

Most of the methods proposed to solve the SLAM problem belong to the family of the Probabilistic (Bayesian) algorithms. The essential idea of this class of algorithms is to attempt to estimate at time t the posterior probability distribution over all possible states (maps and robot positions) given all the exteroceptive sensors measurements (usually referred as the *observations model*) and all the noisy predictions of robot's motion (usually referred as the *control model* or *motion model*), derived from the measurements of the *proprioceptive* sensors as motor's encoders.

The most influential SLAM algorithm is based on the *Extended Kalman Filter* (*EKF*) and was introduced by Smith and Cheeseman [97]. *EKF-SLAM* assumes a *Gaussian* noise for both *observations* and *controls*, while maps are represented through a vector of landmark positions. It calculates a solution for the *online* problem, that is the state is updated for every incoming measurement (observations and controls). This approach suffers from some drawbacks that limits the application in large environments: the sensitivity to failures in data association and the quadratic complexity. Moreover, EKF-SLAM employs linearized models: usually observations and controls models are non-linear.

An alternative algorithm to solve the SLAM problem is the so called *GraphSLAM*, based on a seminal paper by Lu and Milios [59]. It solves the offline *full* SLAM problem, that is it calculates a solution taking into account all measurements together. For this reason, GraphSLAM isn't suitable for real-time and

incrementally map building. GraphSLAM is based on a *sparse graph* that holds robot poses and landmarks location (*nodes*) and the constraints given from the controls and the observations (*links*). A maximum likelihood map and the corresponding robot poses are then obtained through an optimization process over all constraints.
A similar approach to GraphSLAM is the *Sparse Extended Information Filter* (*SEIF*) [101], based on the *Information Filter* (that is the dual of the Kalman Filter). SEIF only maintains a posterior over the present state (robot position and the actual map) but, just like GraphSLAM, it maintains an information representation of all knowledge. SEIF has to make a number of approximation to comply with the real-time constraints of an online SLAM approach: this results in less accurate solutions than EKF-SLAM and GraphSLAM.
Recently, Montemerlo and Thrun introduced the *FastSLAM* [71], a new family of algorithms for SLAM. FastSLAM is an instance of the *Rao-Blackwellized Particle Filters* and it use particles to represent the posterior over the complete robot's path and Gaussians to represent the posterior over landmark positions. FastSLAM is based on the statement that the knowledge of the robot's true path allows to estimate independently the positions of every map landmark. FastSLAM is a very efficient solution to the SLAM problem, however ignoring correlation information cause to underestimate the covariance of landmarks. Moreover, in loop closing problem the number of FastSLAM particles must grow as a function of the size of the loop: this decreases the efficiency of this approach for large loops. The methods presented above deal with the construction of metric maps: another approach to SLAM problem is to construct a *topological representations* of the environment (*Topological SLAM*) [19]. Topological maps attempt to capture spatial connectivity of the environment by representing it as a graph in which nodes hold significant places in the environment.

1.2 SLAM with vision: the Visual SLAM

The most successful SLAM systems required the use of range-finder sensors (e.g., laser and sonar) and usually aimed to build 2-D maps of planar environments. Vision has been introduced recently in SLAM ([92, 25]), anyway a lot of researcher

have spent a great effort in developing vision-based approach to simultaneous localization and mapping problem (*Visual SLAM*). Solving SLAM with vision include a lot of challenging issue such as robust feature detection, data association, and computationally efficient large-scale state estimation [74]. Despite that, there are many interests to use vision for SLAM: vision offers the benefit of perceiving the environment in a 3D volume, and the benefit of providing plenty of information (geometric and photometric) relevant to analyze the perceived scenes [52].

Current visual SLAM systems use perspective [25, 30, 31, 95], stereo [48, 32] or panoramic cameras [2], looking for points [25], lines [31] or planar features [95] as visual landmarks and most of these are based on probabilistic filtering approaches as Extended Kalman Filters and Rao-Blackwellized Particle Filters [71].
Most of the vision SLAM methods are based on a points features detection and extraction step. Davison *et al.* [25] proposed a feature-based SLAM approach using a single perspective camera and EKFs (Extended Kalman Filter), where a 3D map of the features are built using the bearing only information provided by the camera. A similar approach, but based on FastSLAM-type particle, was presented in [30]. In [48] a high resolution digital elevation maps is built from a sequence of stereovision image pairs where interest points are detected and matched between consecutive frames. A visual motion estimation algorithm is used to predict the movements, an extended Kalman filter is used to estimate both the position parameters and the map. The system presented in [2] uses Lowe's Scale Invariant Feature Transform (SIFT)[58] to compute the similarity between omnidirectional images. Links between the robot poses are established based on odometry and image similarity, then a relaxation algorithm is used to generate the map. In [32] a dense metric map of 3D point landmarks for large cyclic environments are built using the Rao-Blackwellised Particle Filter, where SIFT features are extracted from stereo vision and motion estimates are based on sparse optical flow. Eade *et al.* [31] presents a monocular visual SLAM approach using line features, where an efficient algorithm for selecting such landmarks is defined. Higher level landmarks are exploited in [95], where 3D camera

displacement and the scene structure are computed directly from image intensity discrepancies using an efficient second-order optimization procedure for tracking planar patches. Nistér *et al.* [78] presented a robust *visual odometry* system (i.e., the Visual SLAM sub-problem of estimating the robot's ego-motion using vision) based on features tracking and on the five-point algorithm, able to estimate the motion of a stereo camera or a perspective camera in large trajectory.

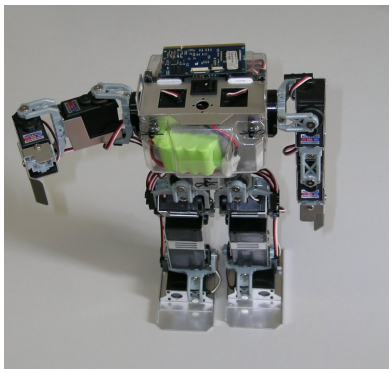
1.2.1 Visual SLAM and Humanoids Robots

Most of the proposed SLAM algorithms are tested using wheeled robots provided with stable exteroceptive sensors and with robust odometry information from its motors. However, odometry is not always available: this is the case for humanoid robots, where the complex kinematics combined with the unpredictable body movements prohibit a reliable reconstruction of the motion from the servo-motor encoders. Moreover, image processing in the humanoids robots domain is a very complex task: during walking, turning, and squatting movements, the camera of a humanoid robot moves in jerky and sometimes unpredictable way. For example, Bennewitz *et al.* highlight in [11] that due to unstable motion of the humanoid platforms, missing odometers, severe body vibrations, and shaking of the camera, standard localization techniques are less robust on a humanoid robot compared to a wheeled robot. As noted by Berthoz in his plenary talk at ICRA 2007 the gait of a humanoid robot should be designed in order to stabilize the head and so to simplify perception as it is done by animals and humans. However, this is not simple for the current humanoid technology.

The first successfull validation of a Visual SLAM algorithm in a humanoid robot was presented by Stasse *et al.* only in 2006 [98]. They presented a 3D SLAM application for humanoid robots based on a standard EKF framework, using the big and very expansive HRP-2 humanoid. However their system is working on the HRP-2 robot that performs a very stable slow gait. The system will not work on small walking robots.

1.3 Objectives of the Thesis

(a)



(b)

(c)

Figure 1.1: The humanoid robots used in the experiments: (a) The VStone Robovie-M; (b) The Kondo KHR; (c) The NimbRo Team Humanoid Robot.

The final aim of this thesis is to study and to develop new techniques addressing the Visual SLAM problem from a computer vision point-of-view, by directly addressing to the issues and the limitations coming into view using as experimental platforms small and cheap humanoid robots (Fig. 1.1) equipped with single perspective cameras.

The Visual SLAM problem is addressed starting from the image-based topological approach via image similarity. Then we focus on the Visual Odometry problem, with a deep investigation of the issues related to the application of such estimation techniques in the humanoid robots' domain. The mapping step of the visual SLAM problem is hence addressed from an homography-based point-of-view.

Beyond the experimentation and the validation, we would to provide clean and effective implementations of the proposed methods in order to support the usage of these in real-time and real world applications.
This thesis has been inspired by some works of the author in the field of the vision-based mobile robot localization [63, 66, 65] in which the omnidirectinal

vision has been exploited to accurately localize a mobile robot.

1.4 Thesis Overview

The thesis is organized as follows:

In Chapter 2 we introduce the most important algorithms for the solution from a probabilistic point-of-view of the SLAM problem.

In Chapter 3 some recent approaches to solve SLAM with vision are presented. We start describing an effective single camera SLAM approach based on point features, then we describe the most recent algorithms used in image based topological SLAM and the state-of-the-art techniques used to solve the visual odometry problem. The computer vision algorithms used in these contexts are presented as well.

In Chapter 4 we presented a new similarity measure for images to be used in image-based topological SLAM for robots with low computational resources. We propose there a compact signature to be extracted from the image and to be stored in memory. We also proposed a loop closure strategy based on this signature.
The contributions proposed in this chapter have been presented in [86, 85].

In Chapter 5 we present a solution to the Visual Odometry problem: we propose here a visual odometry framework based on monocular images designed to address the specific problem of motion estimation robust to motion blur. The system is based on a new feature detection and tracking scheme that is robust even to non-uniform motion blur.
The presented approaches are tested on small humanoid robots.
The contributions proposed in this chapter have been presented in [84, 88, 87].

In Chapter 6 we propose an homography-based approach to 3D visual SLAM,

using only the information provided by a single camera mounted on a humanoid robot. The floor plane is extracted and tracked in consecutive frames, the motion of the robot and an intensity map of the floor plane is estimetd using an efficient tracking method inside a probabilistic SLAM framework.

In Chapter 7 we present conclusions and some ideas to extended the presented methods.

Chapter 2

The SLAM Problem

This chapter presents an overview of some of the most influential Simultaneous Localization and Mapping algorithms.

In SLAM, the robot acquires a map m of its environment while at the same time localizing itself, using the sequence of the observations (i.e., sensors measurements) $z_{0:t} = \{z_0, \dots z_t\}$ and the sequence of the movements (also called actions or controls) $u_{0:t} = \{u_0, \dots u_t\}$, performed until time t (Fig. 2.1). The map m is a list of objects in the environment along with their properties, $m = \{m_1, m_2, \dots, m_N\}$. In *feature-based* maps, each m_i represents an environment landmark and its value contains the properties of this landmark (e.g., a distinctive signature used during the *data association* problem) along with its Cartesian location. In *location-based* maps, each m_i represents a location and its value is the property of this specific location (e.g., a value that specifies whether or not a location is occupied with an object).

In SLAM the *state* s_t of the system that should be estimated is, for every time t, the robot pose x_t together with the map m, $s_t = \{x_t, m\}$. From a probabilistic point of view, this means that we have to estimate a *posterior density function* (PDF) over the state:

$$p[x_t, m | z_{0:t}, u_{0:t}] \tag{2.1}$$

This is called the *online SLAM* problem since it requires the computation of a posterior over the current state. The PDF can be estimated recursively over time

using incoming observations and actions by the Recursive Bayes Filter [100]:

$$p[x_t, m|z_{0:t}, u_{0:t}] = \eta\, p[z_t|x_t, m_i] \int p[x_t, m|u_t, x_{t-1}] p[x_{t-1}, m|z_{0:t-1}, u_{0:t-1}] dx_{t-1} \tag{2.2}$$

The Recursive Bayes Filter exploits the Markov assumption of the stochastic SLAM process, that is the next state s_t depends *only* on the previous state s_{t-1} and on the current control u_t. The probability density $p[x_t, m|u_t, x_{t-1}]$ is called *motion model*, while the probability density $P[z_t|x_t, m_i]$ is called *sensor model*, η is a normalization factor.

On the other hand, the *full SLAM* problem involves the computation of a posterior over the entire path $x_{0:t} = \{x_0, \ldots, x_t\}$ along with the map:

$$p[x_{0:t}, m|z_{0:t}, u_{0:t}] \tag{2.3}$$

2.1 SLAM with Extended Kalman Filter

The earliest SLAM algorithm is based on the *Extended Kalman Filter* (*EKF SLAM*) [97, 54]: it recursively solves the online SLAM problem where the map is feature-based.

The Kalman filter is an efficient and optimal filter that estimates the state of a *linear* dynamic system: to cope with non-linear systems, it was introduced the Extended Kalman filter, that is an approximation of the original filter. For a complete derivation of the Kalman filter, see [100]. The EKF essentially linearizes the non-linear functions around the current state before computing the Kalman filter equations.

As in conventional Kalman filter, EKF SLAM estimates the state from a series of noisy measurements (movements and observations), where the noise is assumed to be always *Gaussian*. The PDF of the estimated state is Gaussian, as well:

$$p[x_t, m|z_{0:t}, u_{0:t}] = \mathcal{N}(\{x_t, m\}, \mu_t, \Sigma_t) = \mathcal{N}(s_t, \mu_t, \Sigma_t) \tag{2.4}$$

where $\mathcal{N}$ is a multivariate Gaussian probability density with mean μ_t and covariance matrix Σ_t. Therefore, for every iteration of the EKF filter, the uncertainty

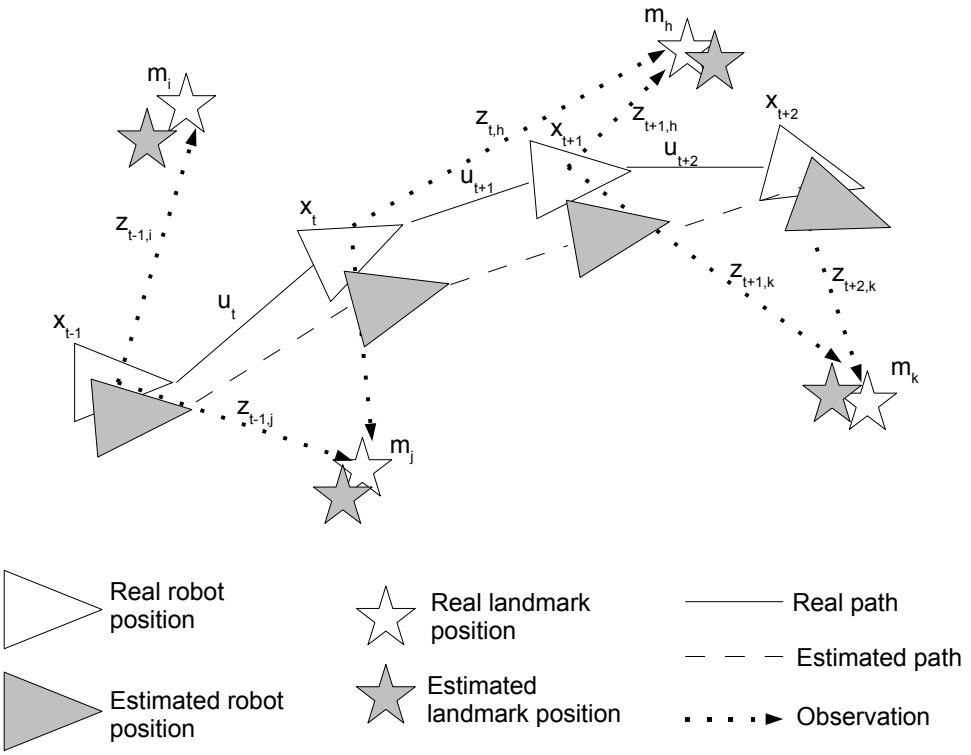


Figure 2.1: In SLAM, robot and landmark positions are estimated concurrently. Observations are made between true robot and true landmark positions.

of the state s_t will be represented through a column vector μ_t of size n and a covariance matrix Σ_t of size $n \times n$, where n is the dimension of the state.
In the EKF SLAM, the probability density of the state transition $p[s_t|s_{t-1}, u_t]$ (i.e., the *motion model*) must be Gaussian, this means that given the state s_{t-1} at time $t-1$ and the movement u_t at time t, the transition of the state can be written as:

$$s_t = f(u_t, s_{t-1}) + v_t \tag{2.5}$$

where v_t is an additive Gaussian noise with zero mean and covariance matrix Q_t that models the uncertain in the state transition, i.e. $p(v_t) \sim \mathcal{N}(0, Q_t)$. Since the motion model depends only on the previous robot pose and on the current movement u_t, we can omit in the equation the map vector, i.e.:

$$x_t = f(u_t, x_{t-1}) + v_t \tag{2.6}$$

The function f doesn't modify the map.
The probability density function of the observation z_t (i.e., the *observation model*) must be Gaussian as well, that is:

$$\hat{z}_t = h\left(s_t, m_i\right) + w_t \tag{2.7}$$

where h is a function that maps the current state in an expected observation $\hat{z}_t$ given the landmark m_i associated to z_t, w_t is an additive Gaussian noise with zero mean and covariance matrix R_t that models the observation noise, i.e. $p(w_t) \sim \mathcal{N}\left(0, R_t\right)$.
In the 2D case, the state s_t at time t can be represented by the following column vector:

$$s_t = \left(p_{x,t}, p_{y,t}, p_{\theta,t}, m_{x,1}, m_{y,1}, m_{s,1}, \ldots, m_{x,N}, m_{y,N}, m_{s,N}\right)^T \tag{2.8}$$

where $x_t = \left(p_{x,t}, p_{x,t}, p_{\theta,t}\right)$ denotes the robot's coordinate, $m_{x,i}, m_{y,i}$ are the coordinates of the $i-th$ landmark m_i with $m_{s,i}$ its distinctive signature, $i = 1, \ldots, N$. The signatures are used during the *data association* step of the algorithm, that is the problem to find the right correspondence (if exits) between an observation z and a landmark m_i in the current map.
The Kalman filter assumes that the function f of Eq. 2.6 and the function h of Eq. 2.7 are linear: this in general is not true. In the Extended Kalman filter these functions are linearized using first order Taylor Expansion around the most-likely state of the system (i.e., the *mean* μ_t) in order to apply the Kalman equations.
In the case of the state transition function f we can write:

$$f\left(u_t, s_{t-1}\right) \backsimeq f\left(u_t, \mu_{t-1}\right) + F_t\ \left(s_{t-1} - \mu_{t-1}\right) \tag{2.9}$$

where F_t is the $n \times n$ *jacobian* matrix of the function f (n is the dimension of the state). The jacobian usually depends on u_t and μ_{t-1}:

$$F_t = \nabla_{s_{t-1}} f\left(u_t, s_{t-1}\right)|_{s_{t-1}=\mu_{t-1}, u_t=u_t} \tag{2.10}$$

The Kalman filter is divided in two phases: prediction and correction [100]. In the SLAM problem, during the prediction phase the state (i.e., its mean and covariance) at time t is updated according to the motion model:

$$\mu_t^- = f\left(u_t, \mu_{t-1}\right) \tag{2.11}$$

$$\Sigma_t^- = F_t \Sigma_{t-1} F_t^T + Q_t \tag{2.12}$$

where μ_{t-1} and Σ_{t-1} are the mean and covariance of the state at time $t-1$, respectively.

During the correction step, for every observation z_t associated with the landmark m_i, it is computed an expected observation $\hat{z}_t = h\left(\mu_t^-, m_i\right)$ and a corresponding *Kalman gain* (Eq. 2.14) that specifies the degree to which the incoming observation corrects the current state estimation (Eq. 2.15 and 2.16):

$$
\begin{aligned}
S_t &= H_t \Sigma_t^- H_t^T + R_t && (2.13)\\
K_t &= \Sigma_t^- H_t^T S_t^{-1} && (2.14)\\
\mu_t &= \mu_t^- + K_t\left(z_t - h\left(\mu_t^-, m_i\right)\right) && (2.15)\\
\Sigma_t &= \left(I - K_t H_t\right)\Sigma_t^- && (2.16)
\end{aligned}
$$

where H_t is the jacobian of the function h:

$$
H_t = \nabla_{s_t} h\left(s_t, m_i\right)|_{s_t=\mu_t^-, m_i=m_i} \tag{2.17}
$$

Before fusing data into the map, new measurements are associated with existing map landmarks. If no existing landmarks in the map $m = \{m_1, m_2, \ldots, m_N\}$ are associated with the new observation, a new landmark m_{N+1} is initialized and added to the map.

The standard formulation of the EKF-SLAM solution is not robust to incorrect association of observations to landmarks: an accurate data association is then desirable. An important advance in the data association task is the concept of batch validation, where *multiple associations* are considered *simultaneously*, instead of searching the single associations using the maximum likelihood rule as in early SLAM implementations. Well known batch validation techniques are the *Joint Compatibility Branch and Bound* (JCBB) [75] method, which is based on a tree-search, and *Combined Constraint Data Association* (CCDA) [4], which is based on a graph search.

The quadratic complexity of EKF-SLAM limits the application in large environments, moreover EKF-SLAM employs linearized models where usually observations and controls models are non-linear.

2.2 Graph-based SLAM

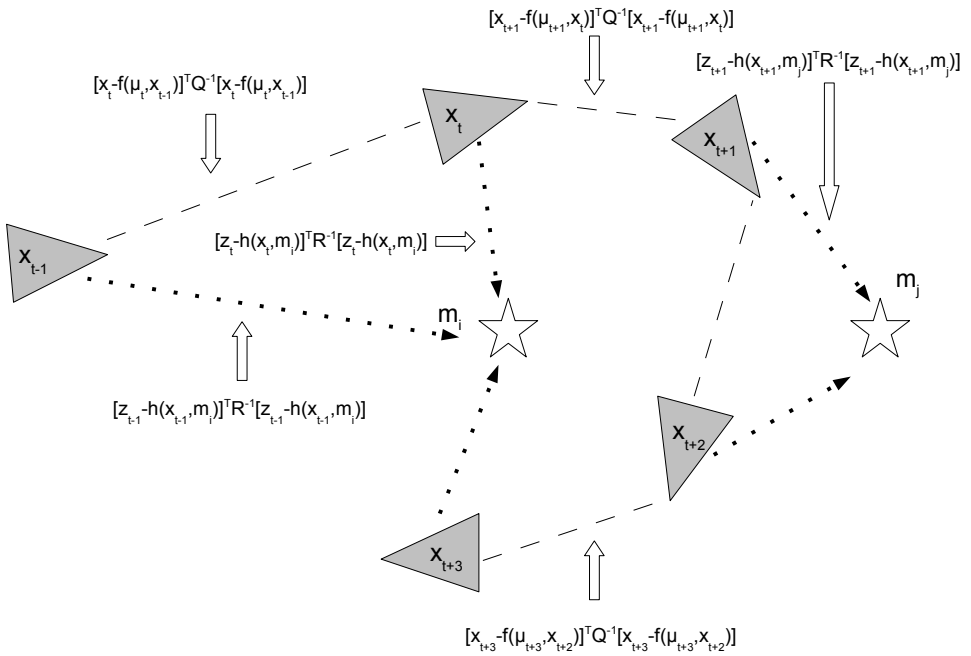


Figure 2.2: An example of Graph-based SLAM with five robot poses and two landmarks. Each link is a non-linear quadratic constraint. The triangles are robot poses, while the stars are landmarks. In the illustration only a subset of the constraint is reported.

In the graph-based SLAM framework, the SLAM problem is represented as a graph in which nodes are environment landmarks or robot poses, and edges are the measurements. Edges are modeled as rigid-body constraints between nodes. The goal of the graph-based SLAM algorithms is to find the configuration of nodes that maximizes the likelihood of the measurements. All nodes (robot locations and landmarks) are involved in the optimization process, this means that graph-based SLAM algorithms solve the *full SLAM* problem (Eq. 2.3).
Given a sequence of observations $z_{0:t}$ and a sequence of actions (i.e., movements)

$u_{0:t}$, Graph-based SLAM turns this data into a graph. The nodes are robot poses $x_0, \dots x_t$ (triangles in Fig 2.2) and landmarks $m_1, m_2, \dots, m_N$ (stars in Fig 2.2). Each edge in the graph corresponds to a movement (robot pose to robot pose link) or feature observation (robot pose to landmark link).
Lu and Milios [59] propose to apply a brute-force nonlinear least square error minimization techniques based on graph constraints: this approach was called *GraphSLAM*. First consider a control u_t, that provides information about the relative movement from the pose x_{t-1} to the pose x_t. This information induces a constraint in the graph between node x_{t-1} and node x_t. We can therefore define the error $e_{t-1,t}$ introduced by the constraint:

$$e_{t-1,t} = x_t - f\left(u_t, x_{t-1}\right) \tag{2.18}$$

where f is the function of Eq. 2.6 that maps the two robot pose given the control u_t. If the movement perfectly matches the current configuration of the nodes, $e_{t-1,t}$ is equal to 0. Assuming a *Gaussian noise* in the motion model, the negative logarithmic likelihood of the state transition $f\left(u_t, x_{t-1}\right)$ can be written (dashed lines in Fig. 2.2):

$$\left[x_t - f\left(u_t, x_{t-1}\right)\right]^T Q_t^{-1} \left[x_t - f\left(u_t, x_{t-1}\right)\right] \tag{2.19}$$

Q_t is the covariance matrix that models the uncertainty in the motion model (see Sec. 2.1), its inverse is called *information matrix*. We can think edges as "springs" in a spring-mass model.
In the same way, we can define the constraint introduced by a landmark observation z_t (dashed arrows in Fig. 2.2):

$$\left[z_t - h\left(x_t, m_i\right)\right]^T R_t^{-1} \left[z_t - h\left(x_t, m_i\right)\right] \tag{2.20}$$

R_t is the covariance matrix that models the uncertain in the sensor model and m_i is the landmark in the map associated to the current observation z_t. After incorporating all measurement and actions, we obtain a sparse graph of soft constraints. The sum of all constraints in the graph will be of form:

$$\begin{aligned} J_{GraphSLAM} = \Sigma_t \left[x_t - f\left(\mu_t, x_{t-1}\right)\right]^T Q^{-1} \left[x_t - f\left(\mu_t, x_{t-1}\right)\right] + \\ \Sigma_t \left[z_t - h\left(x_t, m_i\right)\right]^T R^{-1} \left[z_t - h\left(x_t, m_i\right)\right] \end{aligned} \tag{2.21}$$

In Lu and Milios [59] approach, the maximum likelihood map is obtained searching for the node configuration $x^* = \{x_0^*, \ldots, x_n^*\}$ that minimize Eq. 2.21 This approach seeks to optimize the *whole* graph at once: the brute-force implementation of this algorithm makes it impractical in real-time applications.

A similar approach to GraphSLAM is the *Sparse Extended Information Filter* (*SEIF*) [101], based on the *Information Filter*, that is the the dual of the Kalman Filter. Just like EKF SLAM, SEIF only maintains a posterior over the present state (robot position and the current map, Eq. 2.1) but, similar to GraphSLAM, it maintains an information representation of all knowledge (i.e., a global information matrix over the robot pose and the map that is modified for every incoming constraint). In SEIF, the motion update differs from GraphSLAM, since it eliminates past pose estimates. This information is not entirely lost: some of it is mapped into information links between pairs of features. Moreover SEIF employs a *sparsification step*, where some features are deactivated by eliminating its link to the robot. To compensate for this change in information state, links between active features are also updated.
SEIF is a computationally efficient SLAM algorithm but, due to the number of approximations, it provides a less accurate solutions than EKF-SLAM and GraphSLAM.

Other solutions to the graph-based SLAM have been recently proposed. Folkesson and Christensen [36] presents an algorithm based on a gradient descent optimization procedure. Dellaert proposed a smoothing method called Square Root Smoothing and Mapping [26] that applies smoothing to correct the poses of the robot and feature locations. Recently, Olson *et al.* [82] presented a fast non-linear optimization algorithm that rapidly recovers the robot trajectory using a variant of Stochastic Gradient Descent on an alternative state-space representation.

2.3 SLAM with Rao-Blackwellized Particle Filters

In the last years, Montemerlo and Thrun introduced the *FastSLAM* method, [71] (Factored Solution to the SLAM problem). FastSLAM is an instance of the *Rao-Blackwellized Particle Filters* (RBPF) and it uses particles to represent the posterior over the complete robot path and Gaussians to represent the posterior over landmark positions.

The Particle filter is an approximated solution of the Bayes Filter: the key idea is to represent the posterior density functions of the state by means of a set of M particles $\{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$. Each particle represents an *hypothesis* of the state (e.g., the robot path): the denser a subregion of the state space is populated by particles, the more likely it is the true state falls into this region [100].

FastSLAM is based on the statement that the knowledge of the robot's true path allows to estimate independently the positions of every map landmark. FastSLAM solves the *full SLAM* problem estimating a posterior over the entire path $x_{0:t}$ along with the map (Eq. 2.3). The intuition of the FastSLAM is to use a *factorization* of this posterior:

$$p[x_{0:t}, m|z_{0:t}, u_{0:t}] = p[x_{0:t}|z_{0:t}, u_{0:t}] \prod_{i=1}^{N} p[m_i|x_{0:t}, z_{0:t}, u_{0:t}] \tag{2.22}$$

The proof of the factorization can be found in [71].

FastSLAM estimates the first term in Eq. 6.20 using a particle filter: a particle is a hypothesis of a complete path. For each particle, N conditional landmark posteriors $p[m_i|x_{0:t}, z_{0:t}, u_{0:t}]$ are estimated using the Extended Kalman Filter. Given N the number of landmarks and M the number of particles, in total there are $N * N$ EKFs.

Each FastSLAM particle is of the form:

$$S_t^{[i]} = \{x_{0:t}^{[i]}, \mu_{m_1,t}^{[i]}, \Sigma_{m_1,t}^{[i]}, \ldots, \mu_{m_N,t}^{[i]}, \Sigma_{m_N,t}^{[i]}\} \tag{2.23}$$

where $[i]$ indicates the index of the particles, $x_{0:t}^{[i]}$ is the particle's robot path estimate and $\mu_{m_j,t}^{[i]}$ and $\Sigma_{m_j,t}^{[i]}$ are the mean and the covariance of the Gaussian representing the location of the $j-th$ landmark m_j conditioned on the path $x_{0:t}^{[i]}$.

The particle set $S_t = \{S_t^{[1]}, \ldots, S_t^{[M]}\}$ is calculated incrementally from the set S_{t-1}, the current observation z_t and the control u_t. At time t for each particle $S_{t-1}^{[i]}$, $i = 1, \ldots, M$, a new robot pose is sampled from the *motion model*:

$$x_t^{[i]} \sim p[x_t | u_t, S_{t-1}^{[i]}] \tag{2.24}$$

where u_t is the last movement. A new temporary particles set $\hat{S}_t^{[i]}$ is therefore generated adding the new poses $x_t^{[i]}$ to the robot path of each particle $S_{t-1}^{[i]}$, i.e. $\hat{S}_t^{[i]} = \{x_t^{[i]} \cup S_{t-1}^{[i]}\}$, with $i = 1, \ldots, M$.
The current observation z_t is then associated for each particle $\hat{S}_t^{[i]}$ with a landmark m_{j_i}: using the correction step of the EKF (Sec. 2.1), the means and the covariances $\hat{\mu}_{m_j,t}^{[i]}$ and $\hat{\Sigma}_{m_j,t}^{[i]}$ of this landmark are updated, with $i = 1, \ldots, M$.
The particles $\hat{S}_t^{[i]}$ are then *weighted* based on the *sensor model* $p[z_t | x_t^{[i]}, m_{j_i}]$.
Finally, a new particle set $S_t = \{S_t^{[1]}, \ldots, S_t^{[M]}\}$ is generated by selecting particles from the temporary population $\hat{S}_t$ with probability proportional to the weight of each one [100]. Some initial particles may be forgotten and some may be duplicated.
FastSLAM is a very efficient solution to the SLAM problem, however ignoring correlation information cause to underestimate the covariance of landmarks. Moreover, in loop closing problem the number of FastSlam particles grows as a function of the size of the loop: this decreases the efficiency of this approach for large loops.

An improved FastSLAM algorithm called FastSLAM 2.0 has been presented in [70]: it incorporates the current observation into the proposal distribution of the particle filter and consequently produces more accurate results than the FastSLAM original algorithm when motion noise is high relative to sensor noise. Recently, Grisetti *et al.* [40] presented novel adaptive techniques for reducing the number of particle in RBPF.

2.4 Topological SLAM

The methods presented above deal with the construction of metric maps: topological SLAM is an alternative approach to the SLAM problem that generates a map that provide a *topological representation* of the environment.

Choset and Nagatani [19] proposed a SLAM approach that use a generalized Voronoi graph (GVG) as topological map: this approach is well-suited for robot equipped with range sensor like laser and sonars.
The GVG is a set of curves that captures the topology of the robots environment. In the planar case, GVG edges are the set of points at equal distance from two obstacles, while nodes can be *boundary points* (both the distances are 0) or meet points (points equidistant to three or more obstacles).
To build the GVG, the robot initially moves away from the nearest obstacle until it is equidistant to two obstacles. When the robot reach a meet point, it puts into the map a *topological symbol* (node). While building the GVG, the robot locates itself on the meet points: the nodes of the GVG are indeed used as "virtual" landmarks. A specific control law is also provided to construct the GVG.
In order to reliably disambiguate the graph, the proposed approach suggests to look at the neighboring nodes of a particular meet point and to look for stable feature with unique *sensor signature* (i.e., a sensor reading that univocally identify a node). Lisien *et al.* [56] proposed a hierarchical approach where a GVG based topological map is used to decompose the space into regions where it is built a feature-based map.

Chapter 3

Vision in SLAM

Visual SLAM is the process of build maps of the sourrounding environment and in the same time estimate the robot ego-motion using mainly visual information. Conventional SLAM approaches commonly use information provided by *range finder* sensors as lasers or sonar rings. Range finder sensors provide easily interpreted outputs that can be directly used in the SLAM state estimation problem. On the other hand, vision-based sensors provides the robot with a large amount of information that should be properly interpreted before the estimation process. The process of understanding of the sensory information coming from vision is called *visual perception*. Generally visual perception is a complex task and it involves various scientific subjects as signal processing, geometry and pattern recognition. Often usefull information, for example visual landmark positions, are difficult to extract from images due to the sensor noise and the illumination changes, additionally 3D positions are not observable given only a single frame. A lot of computer vision techniques are involved in Visual SLAM systems, as visual features extraction and detection, features matching, image transformations and structure reconstruction. As introduced in the former chapters, current visual-SLAM systems use various types of cameras (perspective, stereo, panoramic, ...). Due to size and balance constraints, small humanoid robots are ususally equipped with a single, often low-cost, perspective camera. This is the case of the humanoids used as experimental platform in this thesis.

This chapter presents some topics related to the Visual SLAM problem faced

with a single perspective camera. After an introduction of the pinhole camera, the simple mathematical model mainly used to represent perspective cameras, it is described an effective and well-known method to build a 3D map of point features using a single camera. In the following sections, we introduce some recent applications of vision in the Topological SLAM problem and the state-of-the-art techniques used in the vision-based robot ego-motion estimation problem (visual odometry).
We further present an overview of the techniques used in current visual features detection and description schemes.

3.1 The camera

Digital cameras are the most common *vision-based sensors* mounted on mobile robots. Vision is the sense of capturing light from the sourrounding world. The light begins as rays emanating form a source (e.g., the sun, a lamp, ...). These rays strikes an object in the world, and much of the light is absorbed. We perceive the amount of the light that is not absorbed in the form of *color*: the reflected rays that strike our eye (or camera) is hence collected in our retina (or in the camera image sensor, as a CCD or a CMOS sensor, two of the most common image sensor.). Usually cameras mounted on mobile robot work with the light of the visible spectrum (e.g., Fig. 3.1(a,b,c)) but there are cameras that can work with other portions of the electromagnetic spectrum (e.g., Fig. 3.1(d)). A simple model used to describe what happens when a ray strikes a camera is the pinhole camera model [45]. In this model, light from a scene point passes through a single point (e.g., a small aperture) and projects an inverted image on a plane called *image plane*, i.e. the plane where it is located the image sensor. In the pinhole camera model, the small aperture is also the origin O of a 3D coordinate system whose Z axis is along the optical axis (i.e., the viewing direction of the camera). This coordinate system is called the standard coordinate system of the camera, the origin of this frame is called *focal point* or simply *camera center*. In Fig 3.3 this frame seen from the X axis. The image plane is parallel to axes X and Y and is located in the negative direction of the Z axis at distance f from the origin O: f is called the *focal length* of the pinhole camera. It is also defined the 2D

(a) (b)

(c) (d)

Figure 3.1: Some type of digital cameras : (a) A consumer CMOS photo camera; (b) A cheap CMOS webcam; (c) An industrial CCD camera; (d) An infrared thermal imaging camera.

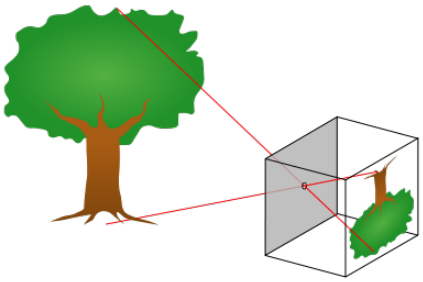


Figure 3.2: The pinhole camera model.

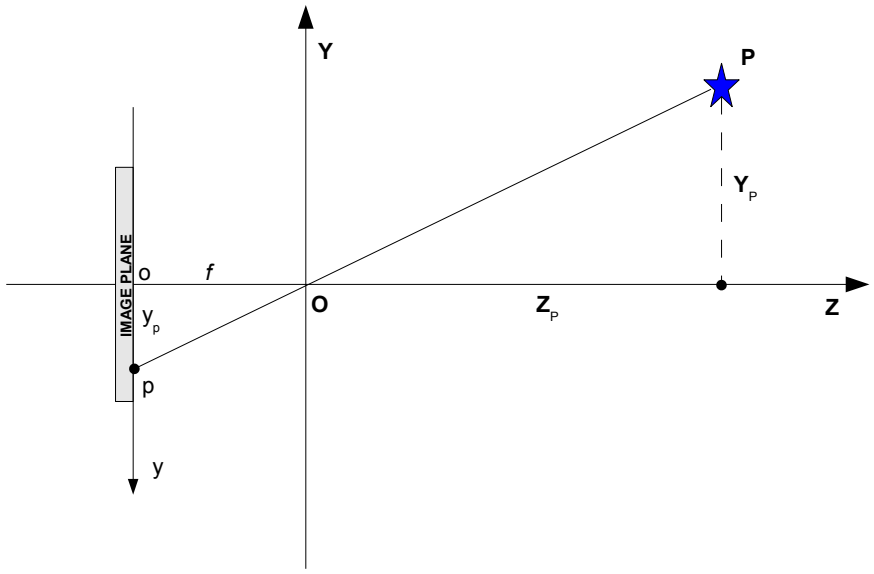


Figure 3.3: The standard coordinate system of the pinhole camera system seen from the X axis.

coordinate system of the image plane, whose origin (called *principal point*) is at the intersection of the optical axis with the image plane, and whose axes x and y are parallel and with opposite direction to the X and Y axes of the standard coordinate system.

A world 3D point $\mathbf{P}$ with coordinates (X_P, Y_P, Z_P) in the standard system will be projected at some point $\mathbf{p} = (x_p, y_p)$ in the image plane (Fig 3.3). The relationship between the two coordinate systems are:

$$x_p = \frac{X_P f}{Z_P} \quad , \quad y_p = \frac{Y_P f}{Z_P} \tag{3.1}$$

Using homogeneous coordinates, this projective transformations can be easily represented by a matrix multiplication:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{3.2}$$

where $s \neq 0$ is a scale factor.

In computer vision, the smallest item of information in an image is represented by the *pixel*. A pixel (u, v) is defined in a coordinate system whose origin is located in a corner of the image plane. The relationship between the two coordinate systems are:

$$u = u_c + \frac{x}{w_p} \quad , \quad v = v_c + \frac{y}{h_p} \tag{3.3}$$

where u_c and v_c are the coordinates of the principal point in pixels and w_p and h_p are the pixel width and height, respectively.

In the image coordinate system, the system in Eq. 3.2 become:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{3.4}$$

where $\alpha_u = \frac{f}{w_p}$ and $\alpha_v = \frac{f}{h_p}$. The 3×4 matrix $\mathbf{A}$ in Eq. 3.4 is called *perspective projection matrix*. In short hand notation, given $\tilde{\mathbf{u}} = [sv \ su \ s]^T$ and $\tilde{\mathbf{P}} = [X_P \ Y_P \ Z_P \ 1]^T$, we can write:

$$\tilde{\mathbf{u}} = \mathbf{A}\tilde{\mathbf{P}} \tag{3.5}$$

The parameters α_u, α_v, u_c and v_c are called the intrinsic parameters of the camera due to the fact they don't depend on the position and orientation of the camera in the environment.

In general the (homogeneous) coordinates of a 3D point $\tilde{\mathbf{P}}'$ are not specified in the actual standard coordinate system $(0, X, Y, Z)$ of the moving camera, but in a more convenient fixed frame $(0', X', Y', Z')$ often called *world frame*. Before projecting this 3D point in the image plane, we need to change its coordinates

from the world frame to the standard coordinate system. Given the 3×3 rotation matrix $\mathbf{R}$ that encodes the camera orientation with respect to the world frame and the 3×1 vector $\mathbf{t}$ that contains the camera displacement from the world frame, we can obtain the (homogeneous) coordinates of the point $\mathbf{P}'$ in the standard coordinate system as:

$$\tilde{\mathbf{P}} = \mathbf{T}\tilde{\mathbf{P}}' \quad , \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{3.6}$$

$\mathbf{R}$ and $\mathbf{t}$ are called *extrinsic parameters*. Finally, pluggin Eq. 3.6 in Eq. 3.5, we obtain:

$$\tilde{\mathbf{u}} = \mathbf{AT}\tilde{\mathbf{P}}' = \mathbf{C}\tilde{\mathbf{P}}' \tag{3.7}$$

The 3×4 matrix $\mathbf{C}$ is called *camera calibration matrix*. We can rewrite Eq. 3.7 in an alternative way as:

$$\tilde{\mathbf{u}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{P}}' \tag{3.8}$$

where $[\mathbf{R}|\mathbf{t}]$ is a 3×4 matrix composed by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ and $\mathbf{K}$ is a 3×3 matrix holding the intrinsic parameters:

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

In many cases, like in the wide-angles camera applications, the lens distortion should be taken into account in the perspective projection: distortion is modeled by nonlinear intrinsic parameters.
The process of finding the intrinsic parameters is called *camera calibration*: Zhang [108] proposed a new powerfull technique that allows to easily calibrate a camera observing a planar pattern (e.g., a chessboard) shown at a few (at least two) different orientations. Zhang's method is able to estimate radial lens distortion parameters, as well.

3.2 SLAM with a single camera

The first *real-time* full Visual SLAM approach using a single camera was presented by Davison in a seminal work in 2003 [24]. This approach is able to estimate the

complete camera trajectory and the full 3D map of all the observed features. The built map is *feature-based*, where features are 3D points associated with salient image patches extracted and tracked between images (Fig. 3.4). Davison's

Figure 3.4: SLAM with a single camera: on the left, the tracked pixel patches; on the right, the estimation of the camera position together with the locations of the 3D features [courtesy of Paul Smith].

approach assumes a Gaussian uncertainty for the whole state vector, therefore it maintains the mean and the full camera and feature covariance. The estimation of the posterior density function over the state is based on a traditional Extended Kalman Filter SLAM approach (see Sec. 2.1).
If we define the fixed world coordinate frame W, and a coordinate frame R fixed with respect to the camera, the camera state $\mathbf{x}_c$ is given by:

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r}^W \\ \mathbf{q}^{WR} \\ \mathbf{v}^W \\ \omega^W \end{bmatrix} \tag{3.10}$$

where $\mathbf{r}^W = [x\ y\ z]^T$ is the camera 3D position, $\mathbf{q}^{WR} = [q_0\ q_x\ q_y\ q_z]^T$ its orientation, while $\mathbf{v}^W$ and ω^W are the the linear and angular velocity of the camera, respectively.
Each feature $\mathbf{y}_i = [x_i\ y_i\ z_i]^T$ is a 3D position vector.
The posterior density function over the camera and feature state is modelled as

a single multi-variate Gaussian:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} \quad , \quad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{\mathbf{x}_c\mathbf{x}_c} & \mathbf{\Sigma}_{\mathbf{x}_c\mathbf{y}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{x}_c\mathbf{y}_N} \\ \mathbf{\Sigma}_{\mathbf{y}_1\mathbf{x}_c} & \mathbf{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{y}_1\mathbf{y}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Sigma}_{\mathbf{y}_N\mathbf{x}_c} & \mathbf{\Sigma}_{\mathbf{y}_N\mathbf{y}_1} & \cdots & \mathbf{\Sigma}_{\mathbf{y}_N\mathbf{y}_N} \end{bmatrix} \tag{3.11}$$

As we seen in Sec. 2.1, EKF SLAM is divided in two phases: prediction and correction. During the prediction step the state is updated according to the *motion model*, that depends only on the previous camera pose and on the actual movement, by means of the *state transition function* plus an additive Gaussian noise (Eq. 2.6). Davison's original approach uses the following state transition function:

$$\hat{\mathbf{x}}_c^{new} = \begin{bmatrix} \hat{\mathbf{r}}^{W,new} \\ \hat{\mathbf{q}}^{WR,new} \\ \hat{\mathbf{v}}^{W,new} \\ \hat{\omega}^{W,new} \end{bmatrix} = f_c\left(\mathbf{n}, \mathbf{x}\right) + \xi = \begin{bmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^W + \mathbf{\Omega}^W)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^W + \mathbf{\Omega}^W \end{bmatrix} \tag{3.12}$$

where ξ is a zero mean Gaussian noise and, assuming in each time step an unknown Gaussian zero mean acceleration $\mathbf{a}^W$ and angular acceleration α^W, $\mathbf{V}^W$ and $\mathbf{\Omega}^W$ are defined as:

$$\mathbf{n} = \begin{bmatrix} \mathbf{V}^W \\ \mathbf{\Omega}^W \end{bmatrix} = \begin{bmatrix} \mathbf{a}^W \Delta t \\ \alpha^W \Delta t \end{bmatrix} \tag{3.13}$$

During the prediction state, remembering Eq. 2.12, the covariance $\mathbf{\Sigma}_{\mathbf{x}_c\mathbf{x}_c}$ of the camera state is updated by:

$$\mathbf{\Sigma}_{\mathbf{x}_c\mathbf{x}_c}^{new} = (\nabla_{\mathbf{x}_c} f_c)\mathbf{\Sigma}_{\mathbf{x}_c\mathbf{x}_c}(\nabla_{\mathbf{x}_c} f_c)^T + (\nabla_{\mathbf{n}} f_c)\mathbf{P}_{\mathbf{n}}(\nabla_{\mathbf{n}} f_c)^T \tag{3.14}$$

where $\mathbf{P}_{\mathbf{n}}$ is the covariance of the noise vector $\mathbf{n}$ and $\nabla_{\mathbf{x}_c} f_c$ and $\nabla_{\mathbf{n}} f_c$ are the jacobians of the function f_c.

During the correction step, for every observation (i.e., a pixel location) $\mathbf{z}_i = [u_i \ v_i]^T$ associated with the 3D landmark $\mathbf{y}_i = [x_i \ y_i \ z_i]^T$, it is computed an *expected* observation $\hat{\mathbf{z}}_i$:

$$\hat{\mathbf{z}}_i = h\left(\mathbf{x}_c, \mathbf{y}_i\right) \tag{3.15}$$

The function h exploits the the standard pinhole camera model (Sec. 3.1) to project the 3D point $\mathbf{y}_i$ in an image pixel $\hat{\mathbf{z}}_i$. The association between 2D visual features $\mathbf{z}_i$ detected into the image and a 3D landmark $\mathbf{y}_i$ is performed as follows: the visual features are detected using the Harris corner detector ([43], see Sec. 3.5) as applied by Shi and Tomasi [93] to pixel patches of size 15×15. The information about the uncertainty presented in the actual map is projected into the image: the pixel patches associated to the projected landmark should lie with some desired probability inside the ellipse representing the uncertainty of this landmark . Matching within this region is achieved by an exhaustive correlation search.

The complete correction step is therefore given by:

$$\mathbf{S} = (\nabla_{\mathbf{x}} h)\boldsymbol{\Sigma}(\nabla_{\mathbf{x}} h)^T + \mathbf{R} \tag{3.16}$$

$$\mathbf{K} = \boldsymbol{\Sigma}\nabla_{\mathbf{x}} h^T \mathbf{S}^{-1} \tag{3.17}$$

$$\mathbf{x}^{new} = \mathbf{x} + \mathbf{K}\left(\mathbf{z}_i - h\left(\mathbf{x}_c, \mathbf{y}_i\right)\right) \tag{3.18}$$

$$\boldsymbol{\Sigma}^{new} = \left(I - \mathbf{K}\nabla_{\mathbf{x}} h\right)\boldsymbol{\Sigma} \tag{3.19}$$

where $\mathbf{R}$ is the noise covariance of the measurements and $\nabla_{\mathbf{x}} h$ is the jacobian of the function h.

The principal issue in this approach is the initialization of a *new* 3D elements in the map; in fact, from a single frame, we cannot estimate the depth of a 3D feature. In his original work, Davison proposed to performed a *delayed initialization* in which the uncertainty in the feature's depth is explicitly represented by means of a particle filter. The depth estimate is refined over several frames *outside* the actual state vector. When the standard deviation of the depth drops below a threshold, the new 3D feature is included in the EKF state vector.
Recently Civera *et al.* [20] proposed an undelayed initialization strategy, based on a direct parametrization of the inverse depth of features relative to the camera locations from which they were first viewed, that significantly improved the Davison's approach. Despite that, the described single camera SLAM approach suffers from some drawbacks. It tracks only a small number of points and assumes to encounter the same points again and again in the future: this is not the case for

example of a humanoid robot's forward walk, where many new detected features remain in the camera field-of-view only for a short time.
Moreover, 3D feature-based maps are usually sparse and useless for real navigation tasks.

3.3 Image-Based Topological SLAM

As introduced in Sec. 2.4, the Topological SLAM aims to segment the environment into distinctive places that form the nodes of a graph (the topological map) given the robot's sensor measurements.
Topological SLAM approaches are usually *appearance-based*, that is the sensor readings are used to infer if the node to which the current measurement pertains represents a new or a previously visited location. In most of the vision-based Topological SLAM approaches the appearance measurement is represented by a *similarity distance* between images, i.e. very similar images are considered as grabbed from the same location and thus as corresponding to the same node [3]. In these cases, omnidirectional vision is often the better choice since an omnidirectional camera always observes a 360-degree field-of-view.
Image similarity between omnidirectional images is exploited for example in [89]: Ranganathan *et al.* presented here the concept of Probabilistic Topological Maps (PTM), a sample-based representation that approximates the posterior distribution over topologies given available sensor measurements . The PTM is obtained by performing Bayesian inference over the space of all possible topologies and provides a systematic solution to the correspondence problem in the domain of topological mapping. The algorithm is completely data-driven in the sense that it does not require a control algorithm for robot exploration that aids in mapping. The algorithm also does not compute localization information for the robot during the map inference. They do not provide landmark detection algorithms or other techniques for detecting signicant places, but assume that these are available. A significant place is labeled with the Fourier signature obtained from an omnidirectional image grabbed at that place. The Fourier signatures are calculated using a modication of the procedure given in [64]. Firstly, a single column image obtained by averaging the columns of the input image is calculated and

subsequently, the one-dimensional Fourier transform of this image is performed. It is to be noted that Fourier signatures do not comprise a robust source of measurements, in the sense that images from distinct physical locations often yield similar Fourier signatures. However, they have the advantage of being simple to compute and model.
In [3] Angeli *et al.* proposed an appearance-based topological SLAM approach that relies on the visual bag of words paradigm [34] to represent the images, and on a discrete Bayes filter to compute the probability of loop-closure (i.e., when a robot returns to reobserve landmarks after a large path [29]). For every incoming image, a loop-closures among the nodes of the topological map is attempted. If the loop-closure fails, a new node is added to the map. This approach uses a single monocular wide-angle camera, and it allows to build in real-time consistent topological maps of indoor environment also under strong perceptual aliasing.

3.4 Visual Odometry

Odometry is the process of estimating the ego-motion a of a mobile robot. In robotics, odometry plays an essential role: it is a precondition in most robot localization and SLAM approaches. Odometry is simple and reliable to obtain with wheeled robots, where it is given by the wheel encoders. However, sometimes odometry is not available: this is the case for flying robots and also for humanoid robots, where the complex kinematics combined with the unpredictable body movements prohibit a reliable reconstruction of the motion from the servo-motor encoders. In such cases, odometry has to be estimated in other ways. In the last few years, several researchers have proposed "visual odometry" systems, in which the ego-motion of the robot can be estimated using on-board cameras [61, 78, 22, 91]. In almost all visual odometry systems, one can identify the following steps:

1. point features are detected;
2. these features are tracked along the image sequence;
3. the odometry is recovered from the apparent motion in the image plane of the tracked features.

Points (1) and (2) will be discussed in Sec. 3.5, in this section we focus on point (3). We present a brief introduction on how to recover the rigid-body transformation (rotation and translation) that relates two subsequent camera positions under an *epipolar geometry* point of view.

3.4.1 Relative pose estimation based on epipolar geometry

Epipolar geometry describes the relations between 3D points and their projections into the 2D image planes of two cameras (or one camera in two different positions) that view a scene from two distinct point-of-view. In the epipolar geometry, cameras are modeled with the pinhole cameras approximation (see Sec. 3.1).

Suppose we have two cameras, looking at the same 3D scene point $\mathbf{X}$ (Fig. 3.5).

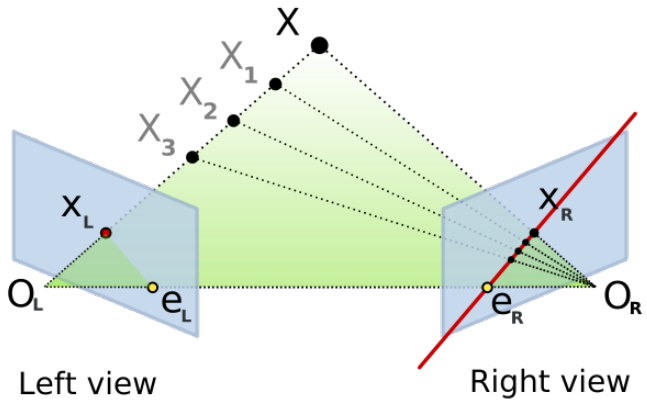


Figure 3.5: Two cameras observing the same scene point $\mathbf{X}$. $\mathbf{O}_L$ and $\mathbf{O}_R$ are the focal points of the cameras (i.e., the origin of theirs standard coordinate system, see Sec. 3.1.). Here the image plane is placed in front of the focal point at distance f: this is an alternative and equivalent way to represent the pinhole camera (Fig. 3.2).
$\mathbf{x}_L$ and $\mathbf{x}_R$ are the perspective projection of the scene point $\mathbf{X}$ into the two image planes, $\mathbf{e}_L$ and $\mathbf{e}_R$ are the epipoles while the red line $\{\mathbf{e}_R, \mathbf{x}_R\}$ is an epipolar line.

If we known the rotation matrix $\mathbf{R}_{LR}$ and the displacement $\mathbf{t}_{LR}$ that correlate

the two camera positions, we can define the *epipole* (or *epipolar point*) $\mathbf{e}_L$ as the projection of the camera center (i.e., its focal point) of the right camera $\mathbf{O}_R$ in the image plane of the left camera. We can define the epipole $\mathbf{e}_R$ as the projection of the camera center of the left camera $\mathbf{O}_L$ in the image plane of the right camera, as well. Obviously, both epipoles $\mathbf{e}_L$ and $\mathbf{e}_R$ and both focal points $\mathbf{O}_L$ and $\mathbf{O}_R$ lie in the same 3D line, the distance between $\mathbf{O}_L$ and $\mathbf{O}_R$ is called *baseline*.
A point $\mathbf{x}_L$ in the image plane of the left camera can be the projection of the scene point $\mathbf{X}$, but also the projection of all point $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots$ that lie in the line $\{\mathbf{O}_L\mathbf{X}\}$ (Fig. 3.5). All of these points will be projected in the image plane of the right camera in points (e.g., $\mathbf{x}_R$ in the case of the scene point $\mathbf{X}$) that lie in a line $\mathbf{l}_R$ passing through the epipole $\mathbf{e}_R$: this line is called *epipolar line*. This means that for each point observed in one image the same point must be observed in the other image on a known epipolar line. This provides an *epipolar constraint*. Mathematically, this fact can be described by means of the fundamental matrix $\mathbf{F}$ [45].
Given the two camera matrix $\mathbf{C}_L$ and $\mathbf{C}_R$ of the left and right camera, respectively, from Eq. 3.7 we can write:

$$\tilde{\mathbf{x}}_L = \mathbf{C}_L\tilde{\mathbf{X}} \tag{3.20}$$

$$\tilde{\mathbf{x}}_R = \mathbf{C}_R\tilde{\mathbf{X}} \tag{3.21}$$

where $\tilde{\mathbf{x}}_L$, $\tilde{\mathbf{x}}_R$ and $\tilde{\mathbf{X}}$ are the points $\mathbf{x}_L$, $\mathbf{x}_R$ and $\mathbf{X}$ in homogeneous coordinates, respectively.
Given the point $\tilde{\mathbf{x}}_L$, we determine the set of points in the 3D space that map to this point. This set of points lie in a ray passing through the focal point $\tilde{\mathbf{O}}_L$ (i.e., the frame origin in homogeneous coordinates for which $\mathbf{C}_L\tilde{\mathbf{O}}_L = \mathbf{0}$) and the point $\mathbf{C}_L^+\tilde{\mathbf{x}}_L$, where $\mathbf{C}_L^+$ is the pseudo-inverse of $\mathbf{C}_L$, $\mathbf{C}_L^+ = \mathbf{C}_L^T(\mathbf{C}_L\mathbf{C}_L^T)^{-1}$, for which $\mathbf{C}_L\mathbf{C}_L^+ = \mathbf{I}$. Point $\mathbf{C}_L^+\tilde{\mathbf{x}}_L$ lies in the ray since $\mathbf{C}_L(\mathbf{C}_L^+\tilde{\mathbf{x}}_L) = \mathbf{I}\tilde{\mathbf{x}}_L = \tilde{\mathbf{x}}_L$. The ray is the line formed by the join of these two points:

$$\tilde{\mathbf{X}}(\lambda) = \mathbf{C}_L^+\tilde{\mathbf{x}}_L + \lambda\tilde{\mathbf{O}}_L \tag{3.22}$$

parametrized by the scalar λ. Setting $\lambda = 0$ we obtain the point $\mathbf{C}_L^+\tilde{\mathbf{x}}_L$ and setting $\lambda = \infty$ we obtain the camera center $\tilde{\mathbf{O}}_L$ (remembering that $\tilde{\mathbf{O}}_L$ is expressed in 3D homogeneous coordinates). We can project these points in the image plane of the

right camera, i.e. $\mathbf{C}_R\mathbf{C}_L^+\tilde{\mathbf{x}}_L$ and $\mathbf{C}_R\tilde{\mathbf{O}}_L$. The epipolar line is the line joining these two projected points (the symbol $\times$ defines here the cross product operator):

$$\mathbf{l}_R = \mathbf{C}_R\tilde{\mathbf{O}}_L \times \mathbf{C}_R\mathbf{C}_L^+\tilde{\mathbf{x}}_L \tag{3.23}$$

the point $\mathbf{C}_R\tilde{\mathbf{O}}_L$ is the projection of the left camera center in the image plane of the right camera, i.e. it is the epipole $\tilde{\mathbf{e}}_R$ in homogeneous coordinates. Let $\mathbf{v} = [v_x\ x_y\ v_z]^T$, we define the *skew symmetric matrix* $[\mathbf{v}]_\times$:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{3.24}$$

so that $[\mathbf{v}]_\times\mathbf{x} = \mathbf{v} \times \mathbf{x}$ for all $\mathbf{x}$. Then we can write Eq. 3.23 as:

$$\mathbf{l}_R = [\tilde{\mathbf{e}}_R]_\times\mathbf{C}_R\mathbf{C}_L^+\tilde{\mathbf{x}}_L \tag{3.25}$$

The projections $\tilde{\mathbf{x}}_R$ in the right camera of all the 3D point that lie in the ray defined by Eq. 3.22 belong to the epipolar line $\mathbf{l}_R$, i.e. $\tilde{\mathbf{x}}_R^T\mathbf{l}_R = \mathbf{0}$. From Eq. 3.25 we obtain:

$$\tilde{\mathbf{x}}_R^T[\tilde{\mathbf{e}}_R]_\times\mathbf{C}_R\mathbf{C}_L^+\tilde{\mathbf{x}}_L = \mathbf{0} \tag{3.26}$$

where:

$$\mathbf{F} = [\tilde{\mathbf{e}}_R]_\times\mathbf{C}_R\mathbf{C}_L^+ \tag{3.27}$$

is the fundamental matrix that encodes the epipolar constraint:

$$\tilde{\mathbf{x}}_R^T\mathbf{F}\tilde{\mathbf{x}}_L \tag{3.28}$$

Since $[\tilde{\mathbf{e}}_R]_\times$ has rank 2 and $\mathbf{C}_R\mathbf{C}_L^+$ has rank 3, $\mathbf{F}$ is a matrix of rank 2: it is shown that any rank-2 matrix is a possible fundamental matrix [45]. Moreover, F is determined uniquely from the camera matrices, up to scale.

If we set the left camera's standard coordinate system as the world frame, from Eq. 3.8 we can write:

$$\tilde{\mathbf{x}}_L = \mathbf{C}_L\tilde{\mathbf{X}} = \mathbf{K}_L[\mathbf{I}|\mathbf{0}]\tilde{\mathbf{X}} \tag{3.29}$$

$$\tilde{\mathbf{x}}_R = \mathbf{C}_R\tilde{\mathbf{X}} = \mathbf{K}_R[\mathbf{R}_{LR}|\mathbf{t}_{LR}]\tilde{\mathbf{X}} \tag{3.30}$$

where $\mathbf{I}$ is the identity matrix. In this case Eq. 3.27 can be expressed [45] in the simplified form:

$$\mathbf{F} = \mathbf{K}_L^{-T}[\mathbf{t}_{RL}]_\times \mathbf{R}_{RL}\mathbf{K}_R^{-1} \tag{3.31}$$

where $\mathbf{K}_L$ and $\mathbf{K}_R$ are the matrices holding the intrinsic parameters of the two cameras. If we known these matrices, we can express the points $\tilde{\mathbf{x}}_L$ and $\tilde{\mathbf{x}}_R$ in *normalized coordinates*:

$$\begin{aligned} \tilde{\mathbf{x}'}_L &= \mathbf{K}_L^{-1}\tilde{\mathbf{x}}_L \\ \tilde{\mathbf{x}'}_R &= \mathbf{K}_R^{-1}\tilde{\mathbf{x}}_R \end{aligned} \tag{3.32}$$

Plugging Eq. 3.32 in Eq. 3.31 we obtain:

$$\tilde{\mathbf{x}'}_R^T[\mathbf{t}_{RL}]_\times \mathbf{R}_{RL}\tilde{\mathbf{x}'}_L = \mathbf{0} \tag{3.33}$$

where the matrix $\mathbf{E}$:

$$\mathbf{E} = [\mathbf{t}_{RL}]_\times \mathbf{R}_{RL} \tag{3.34}$$

encodes the epipolar constraint in a simplified form:

$$\tilde{\mathbf{x}'}_R^T \mathbf{E} \tilde{\mathbf{x}'}_L = \mathbf{0} \tag{3.35}$$

The matrix $\mathbf{E}$ is called *essential matrix*. The matrix $\mathbf{R}$ and the displacement $\mathbf{t}$ that encode the rigid transformation between two camera frames (i.e., the relative movement in the case of a single moving camera) can be recovered from the essential matrix on the basis of the following theorem [45]:

Theorem 3.1 *Let the singular value decomposition of the essential matrix be* $\mathbf{E} \sim \mathbf{U}diag(1,1,0)\mathbf{V}^T$*, where* $\mathbf{U}$ *and* $\mathbf{V}$ *are chosen such that* $det(\mathbf{U}) > 0$ *and* $det(\mathbf{V}) > 0$*. Then* $\mathbf{t} \sim \mathbf{t}_u \equiv [u_{13}\ u_{23}\ u_{33}]$ *and* $\mathbf{R}$ *is equal to* $\mathbf{R}_a \equiv \mathbf{U}\mathbf{D}\mathbf{V}^T$ *or* $\mathbf{R}_b \equiv \mathbf{U}\mathbf{D}^T\mathbf{V}^T$*.*

In order to determine which choice corresponds to the true configuration, the *cheirality constraint* is imposed, i.e. the constraint that the scene points should be in front of the camera: one point is sufficient to resolve the ambiguity.

Eight-Point algorithm

In a calibrated setting (i.e. we known the intrinsic parameters of the camera), the knownledge of the essential matrix or the essential matrix allows to estimate the relative movement between two camera poses. The problem is to find the possible solutions for these matrices given a number of corresponding points (projections of the same 3D points) in two images of the same scene grabbed from different point-of-views.
The best-known method is represented by the *Eight-Point algorithm* proposed by Longuet-Higgins [57] and improved by Hartley [44]. The Eight-Point algorithm allows to estimate the fundamental matrix F given eight corresponding points in two images.
Given a pair of matching points $\mathbf{y}$ and $\mathbf{y}'$ and the corresponding fundamental matrix F:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} \text{ and } \mathbf{y}' = \begin{bmatrix} y'_1 \\ y'_2 \\ 1 \end{bmatrix} \text{ with } \mathbf{F} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \tag{3.36}$$

from Eq. 3.28 we have:

$$\mathbf{y}'^T \mathbf{F} \mathbf{y} = \mathbf{0} \tag{3.37}$$

We can rewrite this constraint as:

$$y'_1 y_1 F_{11} + y'_1 y_2 F_{12} + y'_1 F_{13} + y'_2 y_1 F_{21} + y'_2 y_2 F_{22} + y'_2 F_{23} + y_1 F_{31} + y_2 F_{32} + F_{33} = 0 \tag{3.38}$$

This equation can be represented by a matrix multiplication $\mathbf{af} = 0$ where:

$$\mathbf{a} = [y'_1 y_1 \; y'_1 y_2 \; y'_1 \; y'_2 y_1 \; y'_2 y_2 \; y'_2 \; y_1 \; y_2 \; 1] \tag{3.39}$$

$$\mathbf{f} = [F_{11} \; F_{12} \; F_{13} \; F_{21} \; F_{22} \; F_{23} \; F_{31} \; F_{32} \; F_{33}]^T \tag{3.40}$$

Given a set of N point matches, we obtain a set of linear equations of the form:

$$\mathbf{Af} = 0 \text{ with } \mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_N \end{bmatrix} \tag{3.41}$$

The fundamental matrix F is defined only up to an scale: we can make the additional constraint $\| \mathbf{f} \| = \mathbf{f}^T \mathbf{f} = 1$: this also avoid the trivial solution $\mathbf{f} = \mathbf{0}$.

Having at least eight point matches, it is possible to find a solution to the system 3.41.
With more than eight matches, we have an overspecified system. In this case, the rank of $\mathbf{A}$ must be *at most* 8: with rank equals to 9, the system has a unique trivial solution $\mathbf{f} = \mathbf{0}$. Unfortunately this is the case of estimating $\mathbf{F}$ using real data: due to inaccuracies in the measurement, the matrix $\mathbf{A}$ will have rank 9, therefore we will not be able to find a non-zero solution to the system 3.41. Instead, we look for a least-squares solution of the system, i.e. we find a vector $\mathbf{f}$ that minimizes the $\| \mathbf{Af} \|$ taking into account the additional constraint $\| \mathbf{f} \| = 1$. The solution to this problem is the unit eigenvector of $\mathbf{A}^T\mathbf{A}$ corresponding to the smallest eigenvalue of $\mathbf{A}$ (see [45]). This can be found, for example, using the *Singular Value Decomposition*.

Hartley [44] proposed to precede the algorithm with a simple normalization (translation and scaling) of the coordinates of the matched points: this results in a notably improvement in the stability of the Eight-Point algorithm.

Five-Point algorithm

Recently Nistér proposed [81] a state-of-the-art algorithm to solve the Five-Point relative pose problem, that allows a robust estimation of the camera egomotion being at the same time computationally very efficient. We report here an overview of the Nistér's implementation presented in [79].
The Five-Point problem aims to estimate the essential matrix $\mathbf{E}$ given five corresponding points in two images. Given a pair of matching points $\mathbf{m}$ and $\mathbf{m}'$ (in this case expressed in *normalized coordinates*) and the corresponding essential matrix $\mathbf{E}$, from Eq. 3.35 we known that:

$$\mathbf{m}'^T\mathbf{Em} = \mathbf{0} \tag{3.42}$$

Since an essential matrix $\mathbf{E}$ is a representation of the *motion* (translation and rotation, up to a scale) of the camera (see 3.34), it has only five degrees of freedom. Consequently, to be a valid essential matrix, it must further satisfy more constraints, which are characterized by the following theorem [45] :

Theorem 3.2 *A real nonzero* 3×3 *matrix* $\mathbf{E}$ *is an essential matrix if and only if satisfies the equation:*

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0 \tag{3.43}$$

Given five point matches, we build a linear system in a similar way as in the Eight-Point algorithm derivation:

$$\mathbf{Q}\mathbf{e} = 0 \tag{3.44}$$

where $\mathbf{Q}$ is the 5×9 matrix built from the five point matches (similar to the $\mathbf{A}$ in the Eight-Point algorithm derivation) and $\mathbf{e}$ is represented by:

$$\mathbf{e} = [E_{11}\ E_{12}\ E_{13}\ E_{21}\ E_{22}\ E_{23}\ E_{31}\ E_{32}\ E_{33}]^T \tag{3.45}$$

From the system 3.44 it is possible to recover the following nullspace:

$$\bar{\mathbf{E}} = x\mathbf{E}_0 + y\mathbf{E}_1 + z\mathbf{E}_2 + w\mathbf{E}_3 \tag{3.46}$$

where $\mathbf{E}_i,\ i = 0, 1, 2, 3$ are the null-space bases: the essential matrix $\mathbf{E}$ must be of the the form reported in Eq. 3.46. The null-space bases $\mathbf{E}_i,\ i = 0, 1, 2, 3$ are extracted using a *QR-factorization*: this method is much more efficient than the conventional Singular value decomposition normally used for the nullspace extraction. Using the fact that $\mathbf{E}$ is homogeneous, without loss of generality, we set $w = 1$: given the null-space bases, the essential matrix $\mathbf{E}$ is hence defined by some (x, y, z).
By inserting Eq. 3.46 into the the nine equations of Eq. 3.43, we obtain a 9×20 coefficient matrix corresponding to the 20-dimensional vector:

$$[x^3\ y^3\ x^2y\ xy^2\ x^2z\ x^2\ y^2z\ y^2\ xyz\ xy\ xz^2\ xz\ x\ yz^2\ yz\ y\ z^3\ z^2\ z\ 1] \tag{3.47}$$

This matrix is hence reduced to an upper triangle form using Gaussian-Jordan elimination [83].
Looking for some relationships between the rows of the matrix, it is possible to define five equations in (x, y, z): these are arranged into two 4×4 matrices. Their determinant polynomials are arranged in a 10-th degree polynomial (both these determinant must vanish) from which 10 roots of z are obtained. For each root z

the variables x and y can be found using the equation system defined by the first 4×4 matrix.
The Five-Point algorithm find up to 10 solutions of possible essential matrices $\mathbf{E}$: the right solution is found using an additional point or using a *RANSAC*-like hypothesis-and-test framework (see next section).
The Five-Point algorithm somehow outperforms the Eight-Point algorithm: it's more efficient and obtains better results in the estimation of sideways motion. Moreover, the Five-Point method is essentially unaffected by the *planar degeneracy*.

The RANSAC algorithm

Normally for each coupe of consecutive images, the amount of corresponding points (i.e., point matches) is much larger than the minimal case required by the estimation algorithms introduced above, that is obviously five matches for the Five-point algorithm and eight matches for the Eight-Point algorithm. Indeed the standard procedures for the estimation of the camera motion include an hypothesis-and-test framework that exploits all the available point matches in order to improve the estimation accuracy and to detect completely wrong matches (usually referred as *outliers*).
The *RANdom SAmple Consensus* (RANSAC) algorithm [35] is the best-known method to estimate some parameters (e.g., the fundamental matrix or the essential matrix) from a set of observed data which contains outliers. The idea is very simple: a minimal set of matches are selected randomly (e.g., five in the case of the Five-point algorithm). This points define some hypothetical solutions (e.g., a set of essential matrix extracted by the Five-point algorithm). The support for this solution is measured by the number of points that lie within a distance threshold. In the case of the essential matrix, given two corresponding points, this distance could be the euclidean distance between the second point and epipolar line defined by the essential matrix applied to the first point. The random selection is repeated a number of times, the best model is taken as solution of the estimation problem.

RANSAC original method has been improved in a great number of contribu-

tions. Torr and Zisserman proposed the MLESAC estimator [102]: MLESAC uses the same sampling strategy as RANSAC where minimal sets of correspondences are used to derive hypothesized solutions. The remaining correspondences are used to evaluate the quality of each hypothesis. Unlike RANSAC, that count the number of inliers, MLESAC evaluates the likelihood of the hypothesis by representing the error distribution as a mixture model. Recently Nistér [80] proposed random a sample consensus framework well-suited for the ego-motion estimation that performs a preemptive scoring of the motion hypotheses.

3.5 Local Invariant Features

Local invariant features are a relatively new paradigm in computer vision. In the last years, they are widely used in recognition of specific objects or specific object classes, image mosaicking, 3D reconstruction and in mobile robots visual navigation, as well.
Local invariant features detection and description is not just a method to select interesting locations in the image, but rather a new image representation that allows to describe objects or image regions.
A feature in computer vision can be defined as a *meaningful, detectable parts of the image.*

The best-known and widely used feature detector and descriptor scheme was introduced by Lowe [58] and it is called SIFT (Scale-invariant feature transform). SIFT efficiently detects interest points by using a Difference-of-Gaussian (DoG) operator, at each extracted point is associated a 128-dimensional vector. As shown in [68], SIFT features outperformed previous features detectors-descriptor schemes (e.g. shape context [9], steerable filters [37], differential invariants [50]). Ke *et al.* [49] proposed a variation of the SIFT features, called PCA-SIFT: applying PCA in the gradient images, the descriptor is reduced to a 36-dimensional vector, and matching step is faster. PCA-SIFT are robust to focus-blur noise, but are less discriminative compared with SIFT [68]. Mikolajczyk *et al.* proposed a novel approach for detecting interest points invariant to scale and affine transformation [67]. Interest points are chosen by detecting the local maxima of the Harris

function of the image over the location, and the local maxima of the Laplacian-of-Gaussian (LoG) over the scale. The affine shape of a point neighborhood is then estimated based on second moment matrix. In [68] is presented a novel descriptor called GLOH (Gradient location-orientation histogram), an extension of the SIFT descriptor designed to increase its robustness and distinctiveness: it also uses PCA to reduce the dimension of the descriptor. Recently, Bay *et al.* [5] presented a novel and computationally efficient invariant feature detector-descriptor scheme called SURF (Speeded Up Robust Features). Repeatability and distinctiveness performance are similar to previous proposed schemes, but SURF features can be computed much faster.

An ideal feature should be [103]:

Invariant to same image transformation or distortion

Robust to noise, blur, discretization, compression

Distinctive : individual features can be matched to a large database of objects

Accurate : precise localization

Efficient : close to real-time performance

In every local invariant features extraction method, we can distinguish two sequential processes: the detection of the features, and the description.

3.5.1 Detecting local features

Detecting features means selecting a suitable set of points or regions (i.e., the *features*) that should be robust, stable and well-defined. This process is called *features detection* or *interest points detection*, and in computer vision it is a well known task and several methods have been presented for corners, blobs and edges detection.
The classical methodology used in most systems is to initially convolve the image with some kernel and/or apply to it some mathematical operator that responses strongly for some kind of features. Then the interest points are extracted out

from the filtered images through, for example, a non-maxima suppression process.

Here's are reported the most common features detector. In all of these operator, the input image $f(x, y)$ is initially convolved by a circular symmetric Gaussian kernel $g(x, y, \sigma)$ with zero-mean and standard deviation σ:

$$L(x, y, \sigma) = g(x, y, \sigma) * f(x, y)$$

and hence the operators are applied directly to the convolved image $L(x, y, \sigma)$

Laplacian of Gaussian and Difference of Gaussians

Laplacian of the Gaussian (LoG) is one of the most common blob detectors. Laplacian represents the second derivative of the Gaussian smoothed image L:

$$\nabla^2 L = L_{xx} + L_{yy}$$

where L_{xx} and L_{yy} are the second derivatives along x and y. Applying the Laplacian of Gaussian usually results in strong responses for blobs of extent σ. Interest points are isolated taking the local maxima of $\nabla^2 L$, greater than a certain threshold.
The Difference of Gaussians [58] is an approximation of the Laplacian of Gaussian operator, more computationally efficient (Figure 3.6):

$$\nabla^2 L \approx \frac{1}{2\Delta t}(L(x, y, \sigma + \Delta t) - L(x, y, \sigma - \Delta t))$$

Hessian detector

The Hessian detector [6] is defined as:

$$det(H(x, y, \sigma)) = det \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

where L_{xx}, L_{yy}, L_{xy} are the second derivatives of the convolved images $L(x, y, \sigma)$. The resulting kernels used by the Hessian detector are shown in Figure 3.7. Interest points are isolated taking the local maxima of $det(H(x, y, \sigma))$, greater than a certain threshold. Hessian detector responses mainly to corners and highly textured points (Figure 3.8).

Figure 3.6: The interest point selected by the Difference of Gaussians detector [courtesy of Tinne Tuytelaars].

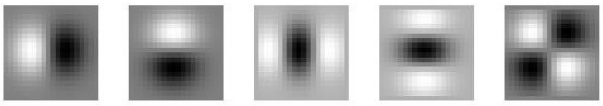


Figure 3.7: Gaussian derivatives up to second order ($\sigma = 3.6$).

Harris detector

Given the 2×2 matrix M:

$$M(x, y, \sigma) = \begin{bmatrix} L_x^2(x, y, \sigma) & L_x(x, y, \sigma)L_y(x, y, \sigma) \\ L_x(x, y, \sigma)L_y(x, y, \sigma) & L_y^2(x, y, \sigma) \end{bmatrix}$$

where L_x and L_y are the first derivatives of the convolved images $L(x, y, \sigma)$, we can compute a measure of corner response called *Harris detector* [43]:

$$R = det(M(x, y, \sigma)) - k(trace(M(x, y, \sigma)))^2$$

Figure 3.8: The interest point selected by the Hessian detector [courtesy of Tinne Tuytelaars].

where k is an empirical constant, $k = 0.04 - 0.06$. R is large for corners and small for flat regions: corners in the images are isolated taking the points of local maxima of R, greater than a certain threshold (Figure 3.9).

Automatic scale selection

The scale-space theory of Lindeberg [55] aims to represent the input image at different scales and it is at the base of the latest scale-invariant feature detectors and descriptors schemes such as SIFT [58] and SURF [5]. Scale-space representation is obtained convolving the original images $f(x, y)$ with a set of Gaussian filters with zero-mean $g(x, y, \sigma)$ and with increasing standard deviations σ (normally referred to as the *scale* of the smoothed image):

$$l(x, y, \sigma) = g(x, y, \sigma) * f(x, y)$$

In order to extract scale-space invariant features, it is applied the detector (e.g., the Laplacian of Gaussian) *for each* "scaled" image. Interest points are then

Figure 3.9: The interest point selected by the Harris detector [courtesy of Tinne Tuytelaars].

detected searching for local maxima over scale *and* location space in a $3 \times 3 \times 3$ neighborhood of each point: only local maxima than a threshold are selected.

3.5.2 Feature descriptors

An ideal feature descriptors should be repeatable, distinctive, compact and efficient [103]. A good example of feature descriptor is represented by the SIFT descriptor [58]. SIFT descriptor is invariant to image scale and rotation, and is quite robust in matching across affine transformations and changing of viewpoint. We report here an overview of SIFT descriptor.

Orientation assignment

SIFT assigns a distinctive orientation for each detected interest point. SIFT begins computing the gradients orientations and magnitudes of 16×16 sample points regularly spaced into a square window centered around the interest point.

Gradients orientations and magnitudes of sample points are computed using first Gaussian derivatives in the smoothed image with scale (see Sec. 3.5.1) closed to the characteristic scale of the interest point:

$$m(x,y,\sigma) = \sqrt{L_x(x,y,\sigma)^2 + L_y(x,y,\sigma)^2}$$
$$\theta(x,y,\sigma) = tan^{-1}\left(\frac{L_y(x,y,\sigma)}{L_x(x,y,\sigma)}\right) \tag{3.48}$$

The magnitudes are Gaussian-weighted with a circular bivariate Gaussian centered in the interest point. Magnitudes are then accumulated into an orientation histogram representing the discretized orientations of the gradients. After an histogram-smoothing step, the bins with values greater than a threshold are selected: multiple interest points are created with the initial location and scale but with these different orientations (interpolated with histogram neighborhood).

Descriptor assignment

After the orientation assignment, SIFT computes an 128-entry descriptor for each selected interest point.
It is selected a square window centered around the interest point and oriented according with its orientation. This region is regularly divided into 4×4 smaller sub-regions, each containing 4×4 regularly spaced sample points. The gradients orientations and magnitudes of the sample points are computed as in Eq. 3.48. Magnitudes are then Gaussian-weighted with a circular bivariate Gaussian in order to increase stability of the descriptor towards small affine transformation and localization errors. Each sample point's gradient is rotated according to the interest point orientation, then its magnitude is accumulated in a orientation histogram with 8 bins (i.e., eight discretized orientations) characteristic of the sub-region. The 4×4 8-bins histograms form the 128-entry descriptor of the selected interest point. The descriptor is finally normalized to an unit vector in order to obtain invariance toward contrast variations.

3.5.3 Features matching

After the computation of the descriptors, it is possible to match features between images. Matching is the process to find a one-to-one association between visual

features detected in a couple of images: the matched features should represent the same scene point seen from different views.

The simplest matching method is called *threshold-based*: two features are matched if the distance between their descriptors is below a threshold. In this case, a descriptor could have more than one match.
In *nearest neighbor-based* matching (NN), two features A and B are matched if the descriptor D_A is the nearest neighbor of the descriptor D_B and if this distance is below a threshold: in this case, a descriptor has at least one match.
Nearest neighbor distance ratio matching strategy (NNDR) is similar to NN, except that the thresholding is applied to the distance ratio between the first and the second nearest neighbor. This means that two features are matched if $\frac{\|D_A - D_B\|}{\|D_A - D_C\|} < t$, where D_B is the first nearest neighbor and D_C is the second nearest neighbor.

Usually the precision in matching is higher for the nearest neighbor-based strategies, this is because the nearest neighbor is mostly correct, although the distance between similar descriptors varies significantly due to image transformations [68].

Computational efficiency in features matching is often a critical issue: matching between hundred descriptors each one composed, for example, by more than one hundred bins is a very computationally expensive task. To cope such a problem, Beis *et al.* [8] proposed an approximate and very efficient algorithm for the nearest neighbor search, called Best-Bin-First (BBF). The BBF algorithm is based on the *kd-tree algorithm*, that organize k-dimensional points in a tree in order to speed-up the nearest neighbor search: the tree is explored searching for the node that is closer to the input point. The BBF algorithm only search m candidates, in the order of their closest distance from the query location, and returns the nearest-neighbor for a subset of queries.

Chapter 4

Image Similarity for Image-Based Topological SLAM

4.1 Introduction

This chapter describes a similarity measure for images which can be used in image-based topological localization and topological SLAM problems by autonomous robots with low computational resources. Instead of storing the images in the robot memory, we propose a compact signature to be extracted from the images. The signature is based on the calculation of the 2D Haar Wavelet Transform of the gray-level image and weights 170 bytes only. We called this signature DWT-signature. We exploit the frequency and space localization property of the wavelet transform to match the images grabbed by the perspective camera mounted on board of the robot and the reference panoramic images built using an automatic image stitching procedure. The proposed signature allows, at the same time, memory saving and fast and efficient similarity calculation. For the topological SLAM problem we also present a simple implementation of a loop-closure detection based on the proposed signature.

We report experiments showing the effectiveness of the proposed image similarity measure using two kinds of small robots: an AIBO ERS-7 robot of the RoboCup Araibo Team of the University of Tokyo and a Kondo KHR-1HV humanoid robot of the IAS-Lab of the University of Padua.

The image-based localization approaches are composed of two stages: the setup stage and the running stage. In the setup stage the robot moves to different points in the environment (called reference locations) and collect a large set of images (called reference images) to be stored, with their positions, in the robot's memory and to be used as reference to calculate the correct position at the running stage. In the running stage the robot is moving autonomously in the environment. The robot can calculate its position by grabbing a picture and comparing it with the reference images stored in its memory, to find the most similar reference image. This will give a topological localization of the robot, because the robot will be more close to the position of this reference image than to any other reference position. The image comparison is done using an image similarity measure, which quantifies the degree of similarity between two images.

In topological SLAM, on the other hand, a map is not provided. The reference images are collected at the running stage. While a robot moves in the environment, if the current image doesn't match any of the previously collected reference image, a new node (i.e., a new location) is added to the map and the current image is saved as a new reference image (see Sec. 3.3).

Three are the main problems to be solved by any techniques of image-based localization or topological SLAM one can develop: (i) how to reduce the number of images necessary to fully describe the environment in which the robot is working; (ii) how to efficiently store a large data set of reference images without filling-up the robot's memory (it is common to have several hundred reference images for typical environments); (iii) how to calculate in a fast and efficient way the similarity of the input image against all the reference images in the data set.

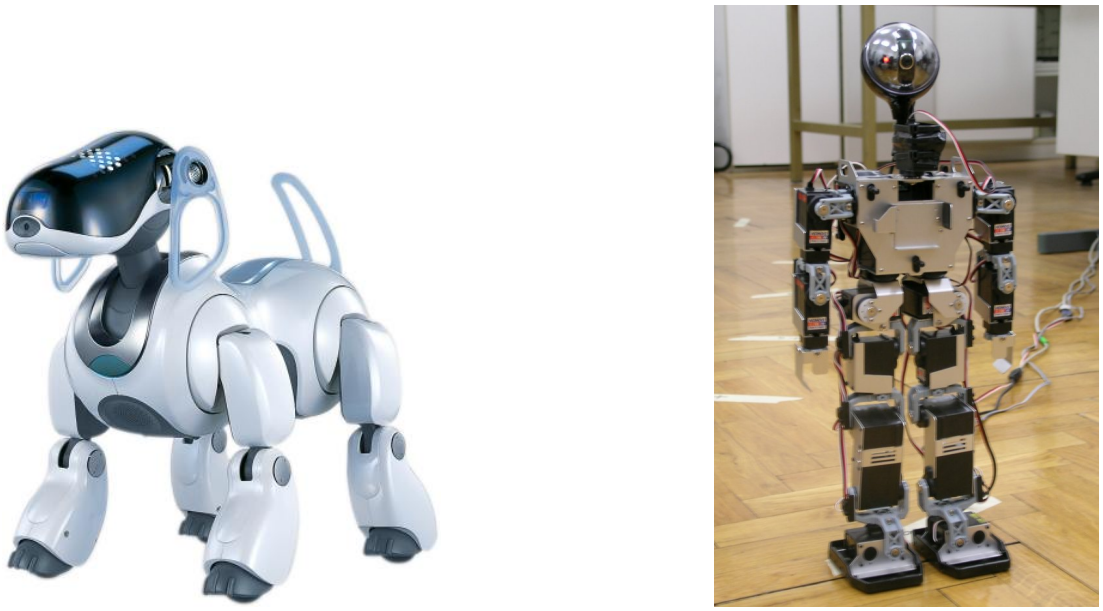
4.2 Related Works and Motivations

Several works have been published that use the image-based localization approach (among the others [73, 28]) or the image-based topological SLAM approach (among the others [89, 3]). Each work tried to solve these problems in a different way. One of the most effective approaches to reduce the number of im-

ages needed to describe the environment is to mount an omnidirectional camera on the robot. In fact, an omnidirectional camera can acquire a complete view of the surroundings in one shot: a single location is hence completely described by a single panoramic image, regardless of the current robot orientation. The most popular technique to reduce the memory consumption of the reference data set, is to extract a set of eigenimages from the set of reference images and to project the images into eigenspaces. The drawback of such systems is that they need to further preprocess the panoramic cylinder images they created from the omnidirectional image in order to obtain the rotational invariance, as in [1], in [47] and in [39] or to constrain the heading of the sensor as in [51]. An approach that exploit the natural rotational invariance of the omnidirectional images is to create a signature for the image based on the color histograms of vertical subwindows of the panoramic image, as in [41] or in [38]. However, this approach based on colors might not be very effective in a general environment with poor color information. An alternative approach to preserve the rotational invariance of omnidirectional image is the one presented in [64], which exploit the properties of the Fourier signature of the omnidirectional images.
Visual invariant features like SIFT [58] are widely exploited in topological localization and mapping tasks. In [13] is proposed a topological mapping strategy where SIFT features are matched between omnidirectional images and the number of correspondences that agree with the epipolar geometry and planar floor constraint are used as a similarity measure of the two images.

Despite the effectiveness of the approaches based on the omnidirectional cameras, it is not always possible to mount an omnidirectional camera on the robot. A solution can be to constrain the movements of the robot in order to keep the camera pointing at the same location [18], but this greatly limits the motion of the robot. An alternative solution can be to extract from the perspective images some features that reduce the amount of required memory while retaining a rich description of the image. A good example of this is reported in [106], where 936 images were stored in less than 4MB by extracting features invariant to translation and to some amount of scaling. However, to extract such a large amount of images is time consuming, even if an automatic procedure is available.

Recently, several methods of global localization and loop-closure detection based on bag of words methods has been presented. Bag of words methods [96, 77] aims to represents images with a set of clusterized visual descriptors (e.g. SIFT features) taken from a visual vocabulary generated offline. In [76] bag of words methods are exploited to detect loop-closure using an appearance based retrieval system within a 3D metrical SLAM framework. In [33] is presented a visual localization and map-learning system based on bag of visual words methods able to recognize the robot location after a short training. This approach is similar to [3], in which visual bag of words paradigm is used to build appearance-based topological maps using a discrete Bayes filter to estimate the probability of loop-closure.



(a) (b)

Figure 4.1: The robots used in the experiments: (a) The AIBO ERS-7; (b) The Kondo KHR-2HV.

Even though topological navigation algorithms based on bag of words methods exhibit exciting results in global localization and loop-closure detection tasks,

bag of words methods suffer from some drawbacks. The detection, description and clusterization of the visual features still remain computationally expensive, especially for robots equipped with low computational resources, as the platforms used in our experiments (Fig. 4.1). Moreover, bag of words methods rely on scale-space invariant features extracted from the images. Images grabbed in different environments (e.g., different rooms) often exhibit distinctive set of features, but images taken from different places inside the same room usually share a lot of common visual features. In some way, bag of words methods provide a *large scale* characterization of the environment, usefull to detect, for example, if an image corresponds to a previously visited room, but not to a different location in the same room.
The similarity measure we propose provides a robust characterization of each single image grabbed by a perspective camera: images grabbed in nearby locations, own similar signatures, also in presence of illumination changes. Unlike bag of words methods, images taken from different location inside the same environment obtain considerable different signatures. In some way, our methods provide a *small scale* characterization of the environment. Therefore, bag of words methods and our method can be combined in the same topological framework, the formers to provided a high level estimation of the robot position (for example, in which room the robot is), and the latter to provide a low level estimation (for example, in which corner the robot is).

As said above, a single omnidirectional image can completely define a reference location. However, small 4-legged robots or small humanoid robots usually are not equipped with omnidirectional cameras. However, if the camera can be panned by moving the neck of the robot, it is possible to create panoramic (or quasi-panoramic) images by stitching the images together. We implemented an automatic image collection and stitching procedure in order to automatically built the reference panoramic images using the perspective cameras that equip the AIBO and the Kondo humanoid robot used in the experiments.
We use these stitched panoramic images as reference images in topological localization and topological SLAM experiments. Incoming images grabbed by the robot's perspective camera are hence matched with the reference panoramic im-

ages using a simple and efficient sliding window matching strategy that exploits the frequency-space localization property of the 2D Haar Discrete Wavelet Transform (e.g., the black sliding window in Fig. 4.2(c)). The assumption in this matching strategy is the fact that the images are grabbed almost at the same height and with horizontal axis parallel to the ground. Image grabbed in different poses simply are not used in the process, they can be used for other purposes (e.g. tracking, object recognition, etc.).

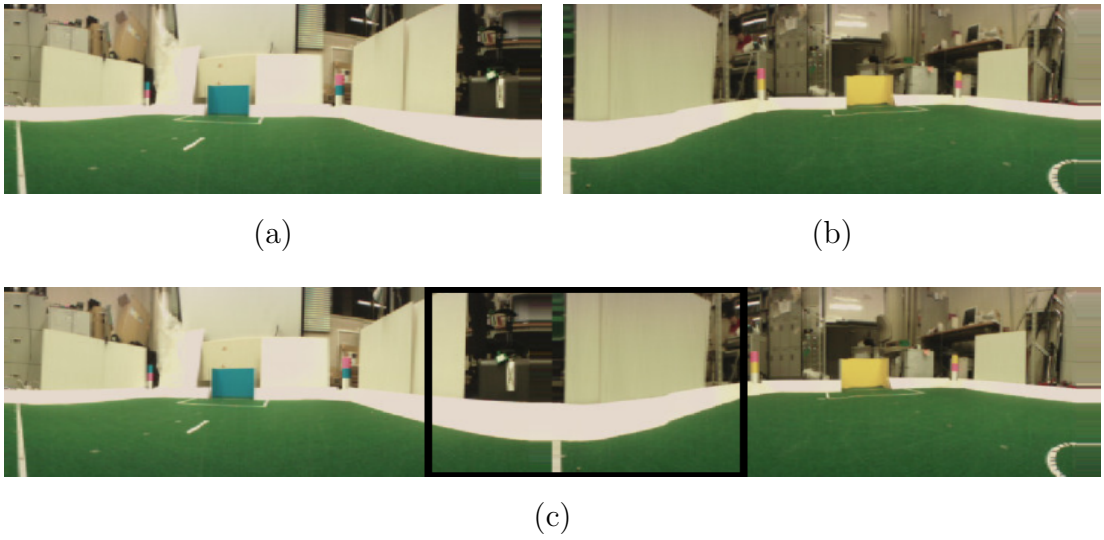
4.3 Reference Image Collection

As we said in the introduction, one of the crucial point in image-based localization and in topological SLAM is how to store in a memory-saving way the reference images and how to efficiently compare them with the input images. This is particularly true when using a robot with limited storage memory and limited computational resources, as the standard AIBO ERS-7 (Fig. 4.1(a)) or the small and lightweight humanoid robots (Fig. 4.1(b)) used in our experiments. AIBO ERS-7 is provided with a low-resolution CMOS perspective camera: with the following procedure, the robot can autonomously build 180-degrees panorama images of the environment stitching a sequence of perspective images. The ensemble of two 180 panorama images taken in the same location with opposite heading give the 360-degrees panoramic view. In the case of topological SLAM, the procedure is performed if the current grabbed image doesn't match any previously collected reference panoramic image.

The stitching procedures for the AIBO ERS-7 is the following:

1. the robot swings its color CMOS camera from right to left parallel to the ground. The process takes 8 seconds: a sequence of 208×160 pixels images are obtained, with a $40ms$ cycle;

2. 180 of the grabbed images, those obtained at angles $\alpha, -90\,\text{deg} \leq \alpha < 90\,\text{deg}$, are chosen as sources of the final 180-degrees image (Fig. 4.2(a)).



(a) (b)

(c)

Figure 4.2: In (a) and (b) the two 180-degrees panoramic images, both taken by the AIBO ERS-7 at the same reference position, but with opposite heading. In (c) their composed 360-degrees panoramic image, with depicted an horizontal-sliding window that highlights a portion suitable for input images matching. In the localization problem, the position of the window is related to the returned bearing angle.

We take the two central columns of these images (i.e., strips of 2×160 pixels) and we stitch them in a 180-degrees panorama image of 360×160 pixels;

3. the robot turns on the spot of 180-degrees and the procedure is repeated and a second 180-degrees view of the surrounding environment is taken (Fig. 4.2(b));

4. the two 180-degrees panorama images, representing a new reference location, are stitched together, regardless of the inevitable discontinuity at the boundaries between the two images, Fig. 4.2(c);

For the Kondo KHR-1HV humanoid robot (Fig. 4.1(b)) the procedure is slightly different. Our KHR-1HV is equipped with a low-cost pan-tilt camera that allow a horizontal movement (pan) in the range of $\pm$ 60 degrees. The camera is mounted on the robot's neck, that is composed by a DC servo motor with vertical axis: this allow a horizontal movement in the range of $\pm$ 100 degrees. Combining the two

Figure 4.3: The 360-degrees panoramic image composed with the automatically stitching procedure by the Kondo KHR-1HV.

movements and exploiting the complete angle of view of the camera (42 degree), the robot can build a complete 360-degrees panoramic image without turning on the spot (e.g, Fig.4.3).

4.4 DWT Image Signature

For an effective vision-based localization and mapping strategy, we need to store the visual memory of the environment (i.e., the images corresponding to reference locations) in a compact and effective way. Images should be represented using specific *signatures* that characterize the content and some useful features of each image in the database. Signatures must have very small size compared with image sizes. The signature of a query image will be directly compared with the signatures of *all* the reference images through a specific metric called *similarity measurement*. A small similarity value between two images means that the two images have been grabbed one close to the other. As presented in [64], one of the possible approaches could be to use the Fourier signatures of the 360-degrees reference images that results from the automatic stitching procedure. However, this approach didn't produce satisfying results. In fact the Fourier Transform gives the spectral content of the whole signal, but it gives no explicit information regarding *where* in space those spectral components appear (Fig. 4.4). This is a good feature when it is used to match omnidirectional images, but not when the input image has to match only a slice of the reference, as in Figure 4.7(a). A better tool for non-stationary signal analysis (whose frequency response varies in time, like in the images) is the Wavelet Transform [104]: it gives information

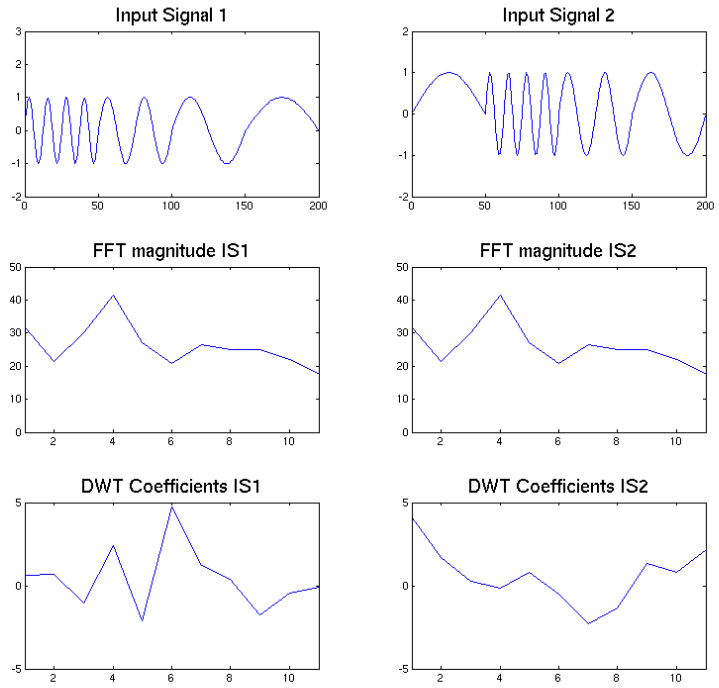


Figure 4.4: The magnitude components of the Discrete Fourier transform (FFT) and the Haar Wavelet coefficients magnitude (DWT) of a periodic signal (input signal 1) and of a shifted version of this signal (input signal 2). Unlike the Fourier transform that provides equal components for both signals, Wavelet transform gives information about where frequency components appear.

about which frequency components exist and where these components appear. Wavelet features are successfully exploited in the image coding algorithms; for instance, the image compression standard JPEG-2000 [21] is based on Wavelet Transform. As well, wavelet signatures are successfully used in image retrieval algorithms, e.g. [46, 72], and texture retrieval algorithm, e.g. [27]. We exploited these properties of the Wavelet Transform using the Discrete Wavelet Transform (DWT) coefficients in order to represent images in a compact way, without losing information about location of the image discontinuity, shapes and texture [72].

4.4.1 The Discrete Wavelet Transform

Discrete Wavelet Transform are used to analyze signals at different scale, $scale = 1/frequency$. In single level discrete 1-D Wavelet Transform (Figure 4.5), the signal is decomposed into a coarse approximation and a detail information (Eq. 4.4.1,4.4.1). Decomposition is performed convolving the input signal with a low-pass filter and an high-pass filter. After filtering, according to the Nyquist's rule, it is possible to eliminate half of the samples. $g()$ and $h()$ low and high-pass filter depend on chosen wavelet type.

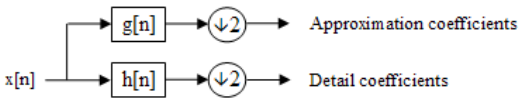


Figure 4.5: Single-level discrete 1-D Wavelet Transform.

$$y_{low}(k) = \sum_n x(n) * g(2k - n) \tag{4.1}$$

$$y_{high}(k) = \sum_n x(n) * h(2k - n) \tag{4.2}$$

The single-level discrete Wavelet Transform can be recursively repeated for further decomposition of the previously y_{low}.

In the 2-D case, the 1-D Wavelet Transform is applied first on each row of the image. The process results in two new matrices with half columns than the input image. A further 1-D DWT is applied to the columns of the resulting matrices. At the end of the one-level 2-D decomposition, $m \times m$ input matrix is decomposed in 4 $m/2 \times m/2$ matrices.

In Figure 4.5 is shown a multilevel 2-D Wavelet decomposition: I is the input image, C_i are the approximation coefficients, H_i,V_i and D_i are respectively the horizontal, vertical and diagonal detailed coefficients, $i = 1, 2, \ldots, n$ represent the recursion level of wavelet decomposition.

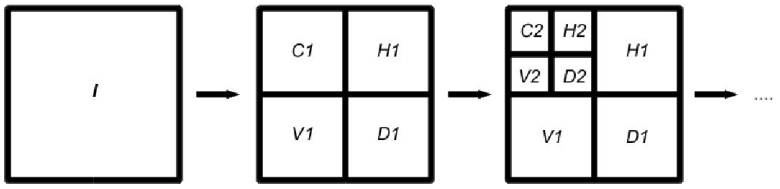


Figure 4.6: Multilevel 2-D Wavelet decomposition.

4.4.2 The proposed signature

We use as image signature a 2-D Haar Wavelet Transform of the grey-level values of the image. Given the resolution of the images grabbed by our robots, we decide to stop at 4-th level decomposition, and to characterize images by the detailed coefficients (horizontal, vertical and diagonal) of this level. Thanks to the great properties of frequency localization given by the DWT, it is possible to store only a few of Discrete Wavelet Signatures (DWT-signatures) for the references panoramic images: the subset of coefficients required for similarity measurement can be extracted using a simple sliding-window strategy.

Wavelet type

Haar Wavelet is chosen as wavelet type because of it is very effective in detecting the exact instants when a signal changes: image discontinuity are one of the most important features chosen in image-based localization. Haar Wavelet can be easily implemented and they are very fast to compute, for example using integral images based techniques [23]. If one is interested in image reconstruction phase, the Haar Wavelets are not the good choice, because they tend to produce a lot of squared artifacts in the reconstructed image. However, we are not interested in the reconstruction phase, we exploit the DWT coefficients to calculate the similarity.

Other wavelet type was taken into account: Daubechies' Wavelets [104] family, commonly used in image coding, were tested. Surprisingly, growing the vanishing moments of the wavelets (i.e. the Daubechies' Wavelets order) performances decade. Those Wavelets are better suitable than Haar to detect a rupture in high-order derivative, but we are interested on detecting discontinuity and features

directly in the signal.

Wavelet level

With the automatically stitching procedure described above we obtain 360-degrees panoramic 720×160 pixels images with the AIBO robot, and 1830×160 pixels with the Kondo robot. By applying recursively 2-D Haar Wavelet Transform, we can reduce a lot the signature size. On the other hand, high level decomposition discard some features in the images, like edge and texture, useful for environment characterization. Given the resolution of our images, we choose decomposition level 4 as a trade-off between a compactness representation and a reliability similarity computation.
In order to cope with the image noise and with the inaccuracy in the height at which the images are grabbed, before the computation of the 2-D Haar Wavelet Transform, images are convolved with a 2-D symmetric Gaussian kernel with standard deviation $\sigma = 2^{\text{wavelet level}}$.

4.4.3 Quantization of detailed coefficients

In our experience, the approximation coefficients are not well suitable for image similarity computation. Considering Haar Wavelet, those coefficients represent only the mean of the intensity of the pixels composing the macro-squares (16×16 pixels in our case), therefore they are strongly sensitive to illumination changes. On the other hand, detailed coefficients can be used to well detected and highlight image discontinuities, shapes and patterns. Our image signature is based on those coefficients computed at level 4: approximation coefficients are simply discarded. As shown in [46], a coarse quantization of these coefficients doesn't affect the effectiveness of the Haar Wavelet coefficients in the image retrieval field. We tested a similar approach for our scope obtaining very good experimental results. We simply represent detailed coefficients d_i as -1 if $d_i < 0$ and as 1 if $d_i \geq 0$. In this way, it is possible to storage every detailed coefficient in only a single bit.

4.4.4 The image similarity metric

Signatures are hence composed by the 3 $w/2^{\text{wavelet level}} \times h/2^{\text{wavelet level}}$ matrix holding the horizontal, vertical and diagonal quantized detailed coefficients, where w is the image width and h are is images height, in our case wavelet level is 4.
Given the signatures of two images with equal size, we compute our similarity measure as:

$$Sim = \frac{1}{Diss} \tag{4.3}$$

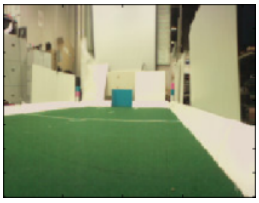
where:

$$\begin{aligned} Diss = w_h * \sum_m \sum_n |H_i(m,n) - H_r(m,n)| \\ + w_v * \sum_m \sum_n |V_i(m,n) - V_r(m,n)| \\ + w_d * \sum_m \sum_n |D_i(m,n) - D_r(m,n)| \end{aligned} \tag{4.4}$$

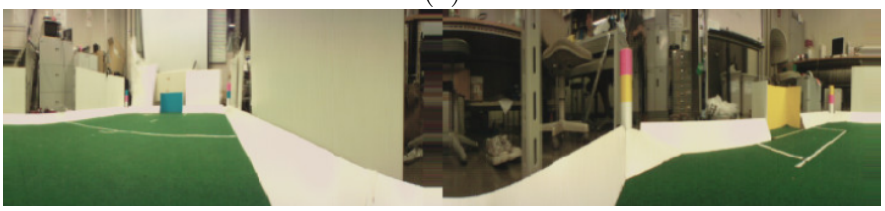
Where m, n represent rows and columns of the detailed coefficients matrices, H_i and H_r, V_i and V_r, D_i and D_r represent the horizontal, vertical and diagonal detailed coefficients respectively the two images. w_h, w_v, w_d are weights usefull to move importance through the three different set of coefficients in the matching process. Our default value are $w_h = 0.25, w_v = 0.5, w_d = 0.25$, because of the large amount of vertical characteristic in indoor environment. The memory saving of our approach is considerable: a gray-scale omnidirectional reference image $720 \times 160 = 115.2$ Kbyte of memory can be represented by our DWT signature with only 170 byte. Experimental results will show the reliability of our approach.

Sliding window wavelet

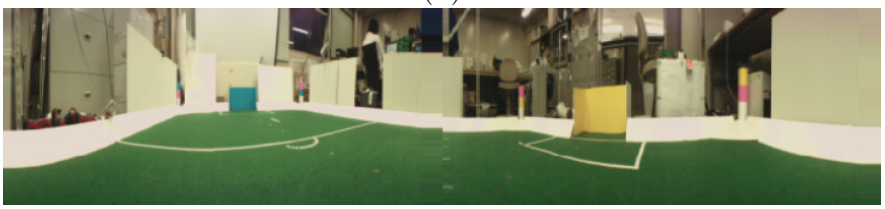
We match between perspective and panoramic images using a sliding window strategy in the signature domain.
Given the $m \times n$ detailed coefficients matrices H_I, V_I and D_I representing the signature of a perspective image I and the $m' \times n$ matrices H_P, V_P and D_P, representing the signature of a panoramic image P, we define $H'_P(i)$, $V'_P(i)$ and $D'_P(i)$, $i = \{1, \ldots, m' - m\}$ the $m \times n$ sub-matrices of H_P, V_P and D_P holding



(a)



(b)



(c)

(d)

Figure 4.7: In (a) is represented an input image; (b),(c),(d) are panoramic references images matched against (a) using the proposed DWT signature. (b) is the references image that best matches (a). Say 100% the similarity of (a) vs (b), similarity (a) against (c) and (d) are respectively 61% and 55%.

m contiguous columns, starting from the i-th columns. We extensively compute the similarity measures between H_I, V_I and D_I and $H'_P(i)$, $V'_P(i)$ and $D'_P(i)$ for all i. The highest value represents the similarity between the image I and the panoramic image P.

Coefficient of Haar Wavelet at level 4 pertains to 16×16 pixels square. In order to achieve higher accuracy in matching, before matching an incoming image with the panoramic reference images, we need to calculate more than one global Discrete Wavelet Signature for every perspective image. Given the an input image, we compute 8 global discrete wavelet signatures, starting from pixels with $x = \{0, 2, \dots \text{ to } 14\}$: for each signature, we compute the similarity using the sliding window strategy explained above. The highest value represents the similarity between the two images, as well.

4.5 Loop-Closure Detection

The DWT-signature can be used also for loop-closure detection similarly to what described in [3] where a topological map is built using visual bag of words paradigm.
We built a graph (e.g, Fig. 4.11) where nodes represents locations in which a complete 360-degrees panoramic reference image is acquired and links represent consecutive reached reference positions. Loops in the graph represents previously visited places (red circles Fig. 4.11).
As described in [3], while the robot moves it checks for a loop closing for every incoming (perspective) image. If the loop-closure is not detected, in our approach a new reference panoramic image is acquired and hence it is associated to a new node added to the graph. The process for the loop closing detection is the following:

1. A new perspective image is acquired.

2. If the similarity (Eq. 4.4.4) between the current perspective image and the last panoramic image added to the graph is over a threshold th, return to point (1), otherwise proceed to point (3).

3. A loop-closure between the current image and all the reference panoramic images (except the last visited) is attempt (see below for a detailed explanation).

4. If the loop-closure is detected, a link between the last visited node in the graph and the node associated with the matched reference image is added.

5. If the loop-closure fails, a new reference panoramic image is acquired with the method described in Sec. 4.3 and hence it is associated to a new node added to the graph.

6. The process restart from the point (1).

The loop-closure detection of point (3) is performed with the following procedure. At time t, given an incoming image I_t and the the current set of nodes in the graph $M_{t-1} = N_0, \ldots, N_n$, the loop-closure detection is performed searching for the node N_j of the actual graph that satisfies:

$$j = \text{argmax}_{i=-1,\ldots,n}\ p(S_t = i | I_t; M_{t-1}) \tag{4.5}$$

where $S_t = i$ is the event that image current image I_t is grabbed in the location represented by the node N_i, $S_t = -1$ is the event that current image is not grabbed in a previously visited location (i.e., no loop-closure detected). Following the recursive Bayesian approach in the discrete case, the posterior density function $p(S_t = i | I_t; M_{t-1})$ of Eq. 4.5 becomes:

$$p(S_t | I_t; M_{t-1}) = \eta\ p(I_t | S_t; M_{t-1}) \sum_{j=-1}^{n} p(S_t | S_{t-1} = j, M_{t-1}) p(S_{t-1} | I_{t-1}; M_{t-2}) \tag{4.6}$$

where η is a normalization factor.
M_{t-1} is obtained at time $t-1$ updating M_{t-2} according to Eq. 4.5 (for example, adding a new node at time M_{t-1} Eq. 4.5 gives -1).
We simply assume an uniform transition probability $p(S_t | S_{t-1} = j, M_{t-1})$. The probability $p(I_t | S_t; M_{t-1})$ is considered as a likelihood function $L(S_t | I_t; M_{t-1})$ that is obtained computing the similarity Eq. 4.4.4 between the perspective image I_t and every reference panoramic images associated with the nodes $M_{t-1} = N_0, \ldots, N_n$ (except the last visited).
One must select the value to be associated to $L(S_t = -1 | I_t; M_{t-1})$, that represents the likelihood of the event "no loop-closure". The node -1 usually it is a node with high probability (i.e., the event "loop-closure" is less likely than "add a new

node"). We set this likelihood at the same value as the threshold th used for determining if the current perspective image is matching or not the last visited node. In this way, we have just one magic number for these two thresholds. It is sensible to choose the same number, because if the similarity of the current image must be over a certain threshold to match the last visited reference image, the same threshold should be used also to check if the current image is matching one of the previously visited reference images (i.e. to check for loop-closure).

If Eq. 4.5 gives a $j \neq -1$, the last visited is linked with the node N_j, while if $j = -1$ a new reference panoramic image is acquired with the method described in Sec. 4.3 and a new node associated to this reference image is added to the graph.

4.6 Experiments on the AIBO ERS-7 Robot

We tested the system in a RoboCup Four-legged League 540 × 360 cm soccer field, using a grid of 13 by 9 reference images. The images have been grabbed in known poses regularly distributed all over the field. For every reference position two 180-degrees panoramic images were collected, using the technique explained in the previous sections. A set of input images, taken in distinct known positions and at different rotations, was used to test the proposed image similarity measure. The ground-truth position of the robot for every input image was measured by hand with an error less than 1 cm. We compared our DWT-based image signature with the Fourier signature proposed in [64] and used in a topological SLAM approach in [89]. In Figure 4.8, the corresponding similarity values against the reference images are plotted for both the proposed DWT signature and the Fourier signature. The similarity values have been interpolated to obtain a similarity value for every possible pose of the robot in the field. In the plot the dark red areas correspond to a higher similarity, while the dark blue areas correspond to a lower similarity. The white cross represents the actual pose of the robot.
The proposed DWT signature outperforms the Fourier signature for all the sample images, resulting in well-defined unimodal distributions around the real robot positions, where the Fourier signature results in a higher uncertainty with often multimodal distributions. This is due to the fact that Fourier Transform gives

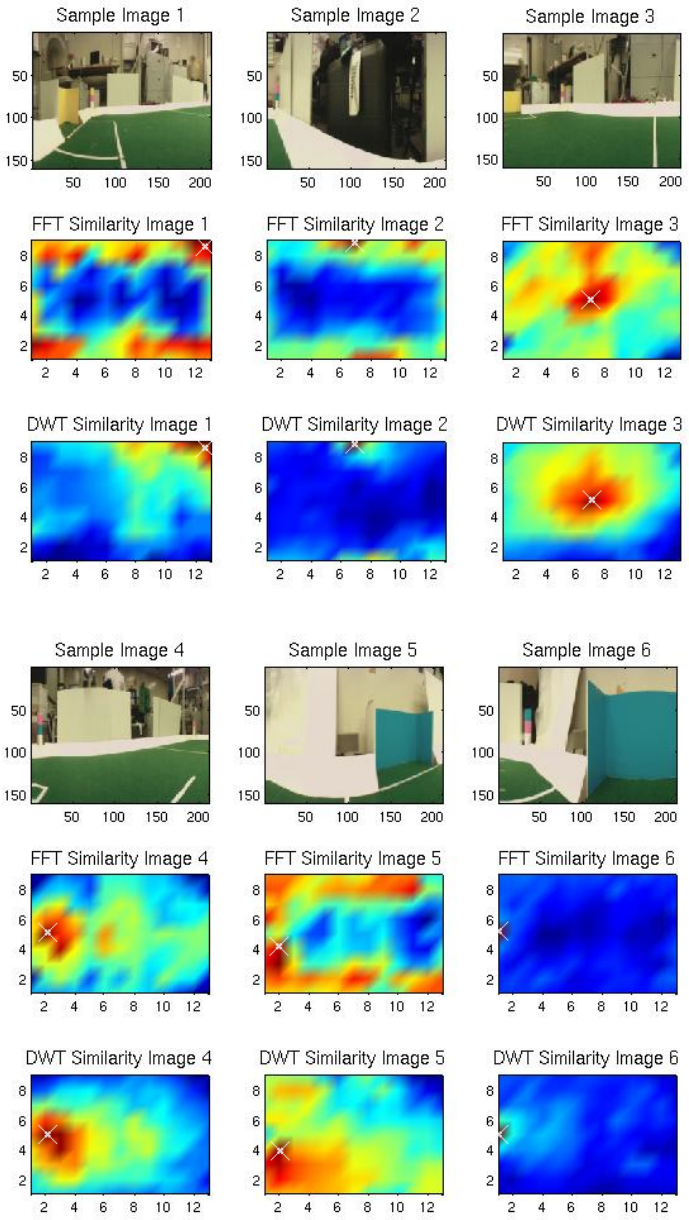
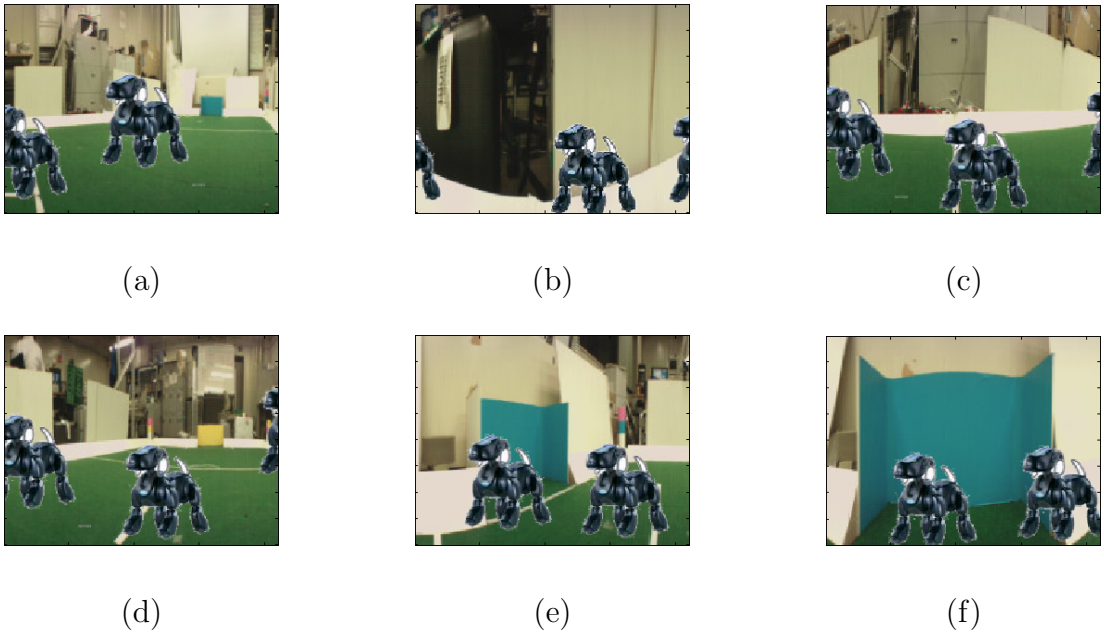


Figure 4.8: The proposed Discrete Wavelet Transform signature compared to Fourier signature. The interpolated similarity values are depicted for all possible poses of the robot in the field. Dark red areas correspond to a higher similarity, while dark blue areas correspond to a lower similarity. The crosses represent the current robot poses.

information regarding the whole spectral content of the images, but no information regarding the locations of the intensity patterns inside the images.

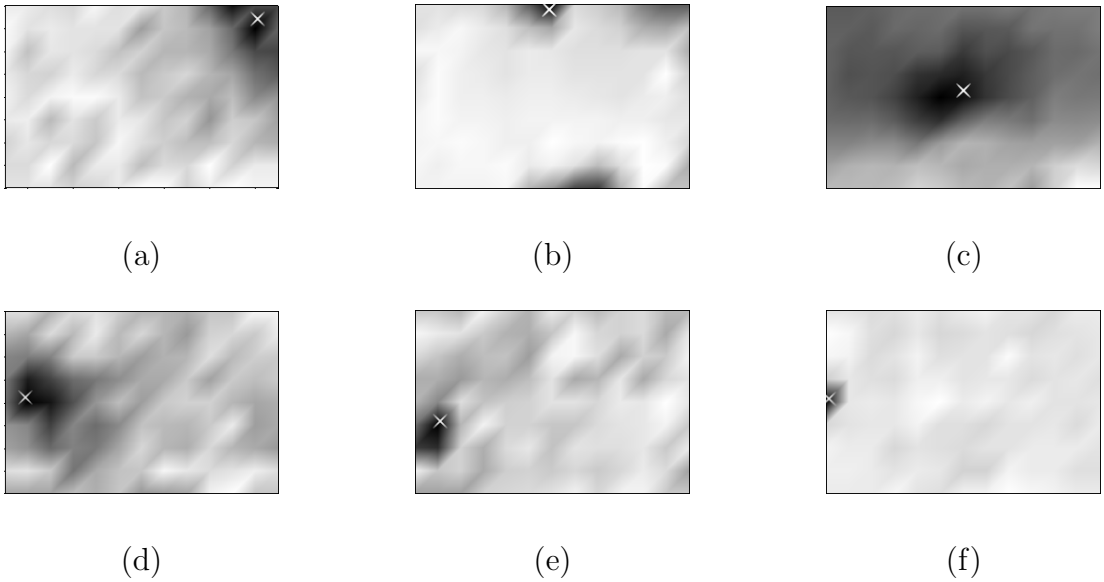


(a) (b) (c)

(d) (e) (f)

Figure 4.9: Sample images added with occlusions.

In order to prove the robustness of our approach to occlusion in the images,



(a) (b) (c)

(d) (e) (f)

Figure 4.10: Interpolated similarity values for all possible poses of the robot in the field, given the input images with occlusions of Figure 4.9. Darker areas correspond to a higher similarity. The crosses represent the current robot poses.

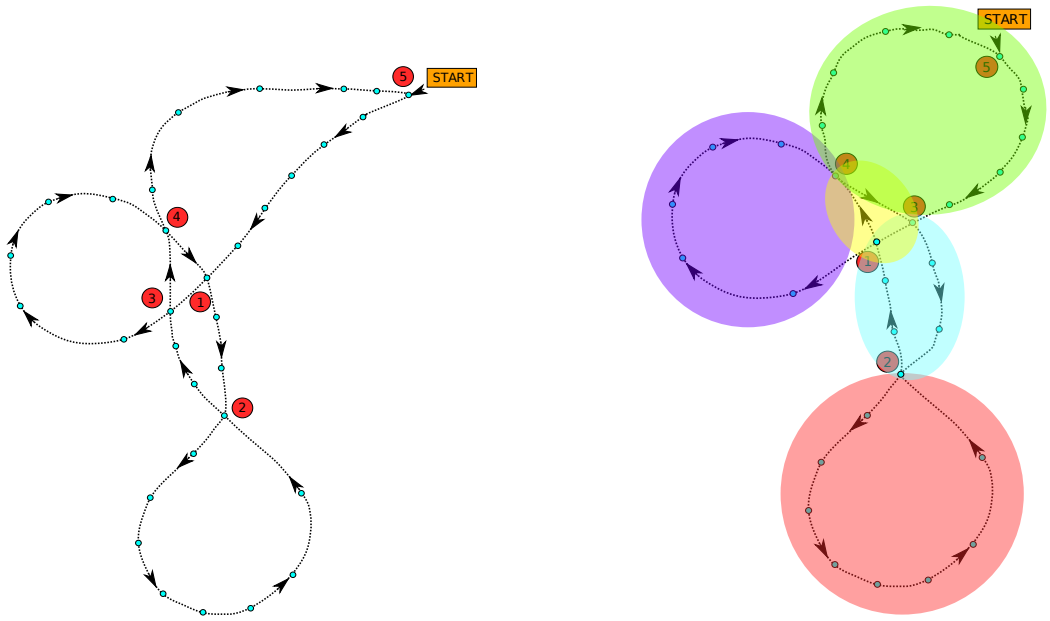
we generated some synthetic images, by overlapping at the original input images some pictures of AIBO robots (Fig. 4.9). The effect of image occlusion is to increase the similarity of the input image also with reference images close to the the correct reference image (Fig. 4.10). However, this effect is limited and it is only reducing a little bit the discriminative power of the DWT signature without impairing its effectiveness. Note that Fig. 4.9(c) is the one in which the similarity measure decrease most its discriminative power. This is because the preponderant feature constituted by the white line is occluded and the rest of the picture is very general with the green and the white smooth surfaces that can be found in any picture.

4.7 Experiments on the Kondo KHR-1HV Robot

In addition to the image similarity based on the DWT-signature, we tested also the loop-Closure detection strategy in a 12 meters trajectories walked by the Kondo KHR-1HV humanoid robot. At every step, a perspective image is acquired. If this does no longer match the last reference image, the loop-closure detection strategy presented in Sec. 4.5 is performed. If no loop-closure is detected, a new panoramic image is acquired, and a new node is added to the graph. In Fig. 4.11 (a) the ground-truth of the walked trajectory, in Fig. 4.11 (b) the built graph: all the loop-closure are correctly detected.
In Fig. 4.12 and 4.13 are depicted the posterior density functions (Eq. 4.6) computed in 4 locations in which a loop-closure is detected (see Fig. 4.11).

4.8 Summary

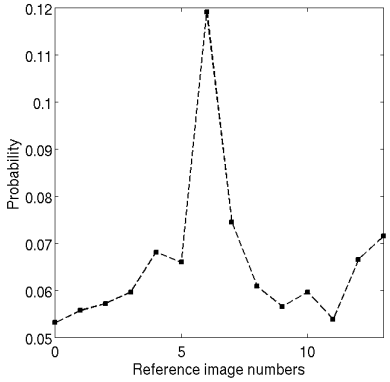
In this chapter we presented a new way to calculate the similarity between images to be used in the image-based topological localization and SLAM approaches on autonomous robot with low computational resources. We presented a technique that enables a quick visual characterization of the environment by building at run-time the panoramic reference images. Then, we proposed a new image signature that exploits the properties of the Haar Wavelet Transform and suitable for matching perspective images against panoramic images. We presented successful



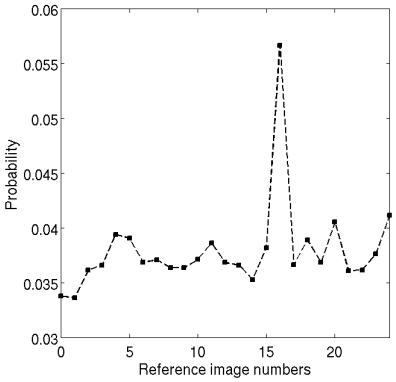
(a) (b)

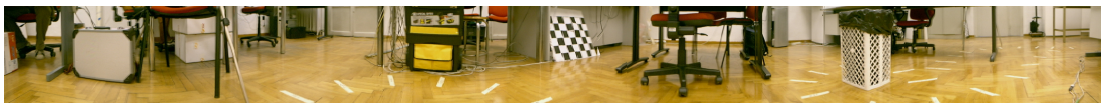
Figure 4.11: The result of the presented Loop-closure detection strategy. (a) The ground-truth of the walked trajectory, the small cyan circles represent the locations where a panoramic image is acquired, and a new node is added. Red circles represents the real loop-closure positions. (b) The graph built using the proposed signature with depicted the detected loop-closures.

experiments on the calculation of the image similarity of real images grabbed by a AIBO ERS-7 robot and an experiment in which multiple loop-closures events are correctly detected along a trajectories walked by the Kondo KHR-1HV small humanoid robot.



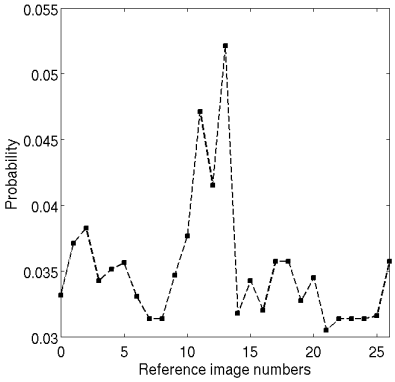
(a)



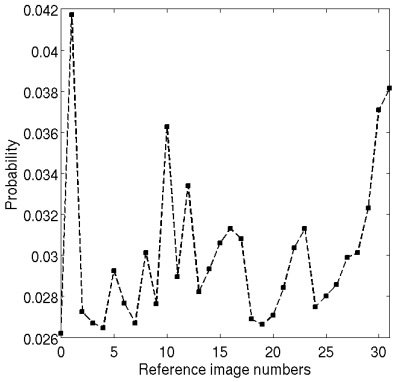


(b)

Figure 4.12: Input image, matched panoramic and posterior density functions of the loop-closure event for the locations locations 1 and 2 in which a loop-closure is detected.



(c)



(c)

Figure 4.13: Input image, matched panoramic and posterior density functions of the loop-closure event for the locations locations 4 and 5 in which a loop-closure is detected.

Chapter 5

Visual Odometry with Improved Local Invariant Features

5.1 Introduction

As introduced in Chapter 2, SLAM algorithms base their prediction steps on information given by the robot odometry. We also introduced that not always the odometry is available: this is the case for flying robots and also for humanoid robots. In such cases, odometry has to be estimated in other ways for example using a *visual odometry* strategy (Sec. 3.4), that estimates the robot position using only images.
Most of the proposed visual odometry approaches were developed for wheeled robots, but humanoid robots introduce novel challenges to visual odometry. When a humanoid robot is walking, turning, or squatting, its camera moves in a jerky and sometimes unpredictable way. This causes an undesired *motion blur* in the images grabbed by the robot's camera that negatively affects the performance of the feature detectors and especially of the feature tracking classic algorithms. A typical image affected by motion blur grabbed by a walking robot is depicted in Fig. 5.1.

Motion blur is a severe problem: standard feature extraction and tracking approaches (Sec. 3.5) typically fail when applied to sequences of images strongly affected by motion blur. In this chapter, we propose two new feature detection

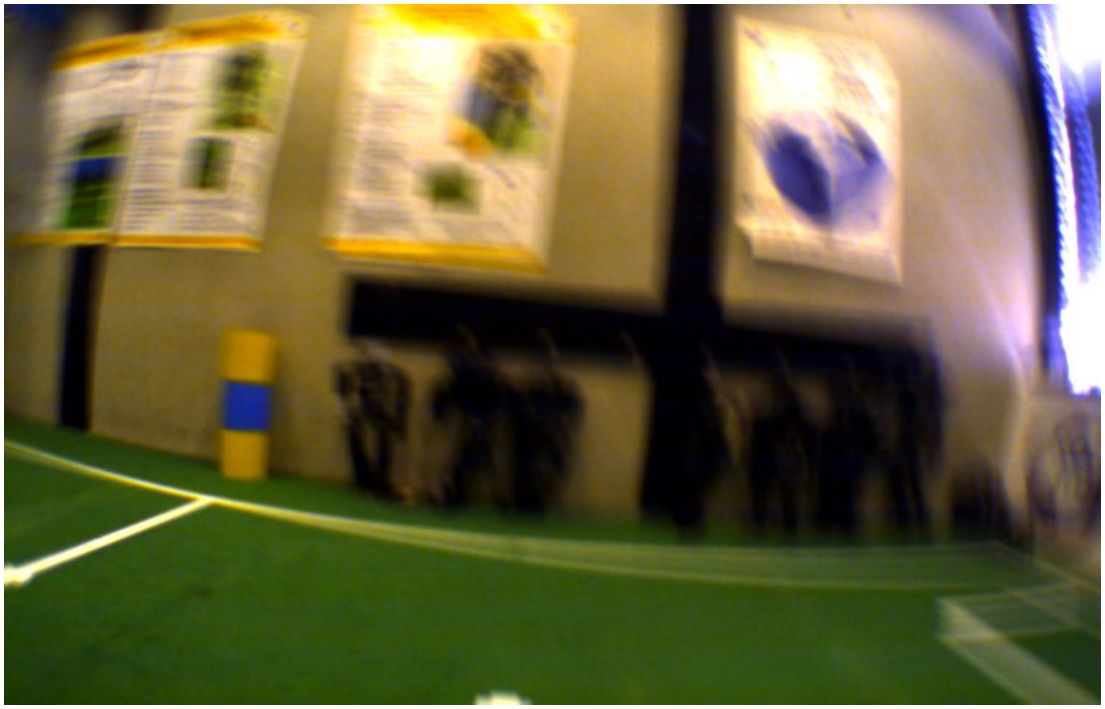


Figure 5.1: Typical image grabbed by a walking humanoid robot. As can be seen, the image is highly affected by motion blur.

and tracking schemes that are robust to motion blur.

The first approach is based on the introduction of a blind-deconvolution step before starting the invariant feature detection and description process. During this step, the parameters of the unknown blurring function (PSF, *Point Spread Function*) are estimated using a direct method. If a motion blur is detected (the PSF magnitude is over a preset threshold), the image is deblurred according to the estimate PSF using an efficient Wiener filter. The invariant features detection and description is then performed in the restored image.

The second approach overcomes the assumption of a single motion blur function for the whole image, so it is robust also to *non-uniform* motion blur. In this approach, before detecting interest points, an image preprocessing step estimates the *Point Spread Function* (PSF) of the motion blur in the image. We calculate

not a unique PSF in the whole image, but we segment each image on the basis of the local motion blur. The estimated PSFs are then used to build an *adapted scale-space representation* trying to minimize the undesired effect of the motion blur. The scale-space extrema are extracted based on the determinant of the Hessian, and a SIFT descriptor is calculated for each keypoint. Before matching features between images, features with less distinctive descriptors are discarded based on their entropy.

Furthermore, we developed a framework for visual odometry based on features extracted out of and matched in monocular image sequences, in which robot ego-motion is estimated from the matched features with a method based on the Five-point algorithm (similarly to the visual odometry strategy proposed by Nister *et al.* [78]).

We present experiments performed on standard datasets corrupted with motion blur and on images taken by a camera mounted on walking small humanoid robots to show the effectiveness of our approach. The experiments demonstrate that our techniques are able to reliably extract and match features even in presence of strong motion blur effects.
We also present experiments in which odometry could reliably be estimated from images grabbed by walking humanoid robots in the presence of strong motion blur effect. This is obtained without any global bundle adjustment process (a process which is too computationally expensive for the processing units on-board of small humanoids robots) and without the aid of any inertial measurement sensor.

5.2 First approach: restoring the image

Motion blur is the effect of the relative movements between the camera and the objects of the observed scene during the exposure time (i.e., the integration time of the grabbed image).
When an image is captured while the camera is moving during the exposure time, a certain number of scene point is projected at any single image pixel. All these points contribute to the final pixel value. This effect is called motion-blur, and

it depends on the relative movement between the camera and the objects of the observed scene during the exposure time. When this motion is linear and with uniform velocity, the blur can be determined by two parameters: the blur extent d and the direction θ. The observed image so can be obtained from the equation:

$$g(x,y) = f(x,y) * h(x,y) + n(x,y) \tag{5.1}$$

where $*$ is the convolution operator, $f(x,y)$ is the uncorrupted version of the observed image (i.e. the ideal image grabbed without relative motions), $n(x,y)$ is an additive noise function and $h(x,y)$ is the blurring function, called *PSF* (*Point Spread Function*). In the linear case:

$$h(x,y) = \begin{cases} \frac{1}{L} & \text{if } \sqrt{x^2+y^2} \leq \frac{L}{2}, \frac{x}{y} = -tan(\phi) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Our first approach is based on two assumptions:

- Motion is linear
- We consider only a PSF for the whole image

The first assumption is acceptable for images grabbed by a camera mounted in a simple humanoid robot as VStone Robovie-M (Fig. 5.4): with a shutter frequency of 25-30 Hz, relative motions can be approximated as linear motions. Moreover, this kind of robot doesn't perform quick rotation around the optical axis of the image. The second assumption means that we try to restore the image considering the PSF that globally best fit all motion blur functions into the image.

Using deconvolution techniques, the image can be partially deblurred. For example, the Richardson-Lucy algorithm [60] and the Expectation-Maximization [53] method are well known *iterative* deconvolution procedure, while Wiener filter [90] is a *non-iterative* image restoration technique that tries to build an "optimal" estimate of the unblurred image.

5.2.1 PSF estimation

In order to restore the image with deconvolution techniques, the motion blur parameters (direction and extent) must be known: if they are unknown, the image

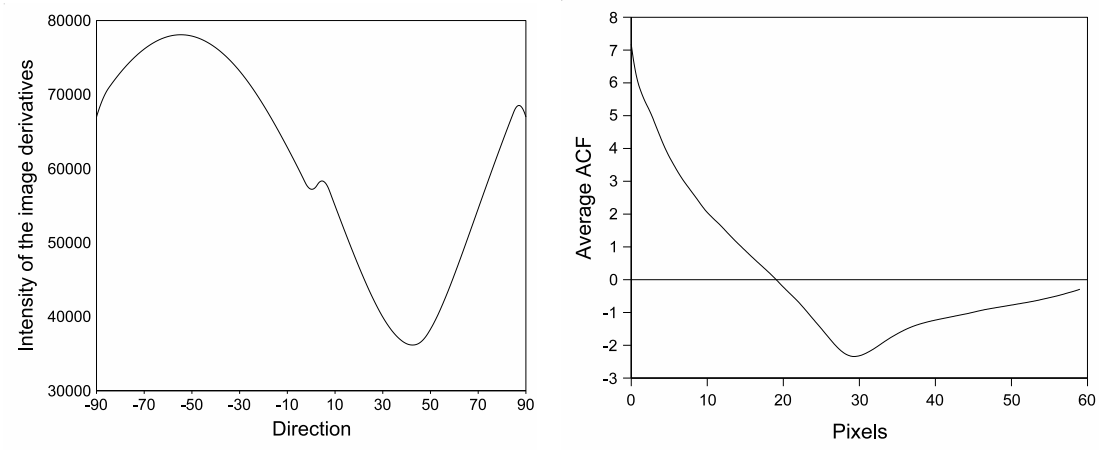
restoration process is called *blind deconvolution*. In [107] is presented the whitening method: this is a non-iterative method that identify the PSF by high-pass filtering the blurred image. The filtered image is characterized mostly by the correlation property of the blur function. In [99] the PSF is obtained searching for image moments that are invariant with respect to the motion blur. In [62] blur direction is determined by an inertia-like tensor, while the extent is determined finding zeros of the blur slice of the power spectrum or bispectrum in this direction. In [69] PSF is estimated by using Radon transform to find direction and fuzzy set concepts to find its extend.

For PSF estimation, we use an approximated version of *whitening method* [107]. Motion during exposure affects the image by decreasing its resolution mostly in the motion direction. We search for the direction in the image with the lowest resolution: this can be done high-pass filtering the image in all directions. The direction with the lowest responses represent the blur direction. The high-pass filter used is the absolute value of the derivatives in the candidate directions: we take the absolute value of the difference of two adjacent pixels along the direction. For better approximation, pixels are interpolated. In order to preserve the efficiency, we compute responses in 5 pixels regularly spaced sample points along 36 directions (every 5 degrees): responses are accumulated in a 36-bins histogram. The bin with the lowest value represent the blur direction: the final direction is obtained by an interpolation step with the histogram bins closed to the peak. In Figure 5.2(a) the responses for a 45-degrees blurred image. In order to estimate the blur extent, the PSF correlation properties along its direction are emphasized. For theoretical details, see [107]. An auto-correlation operation (ACF) in the image derivative lines along motion direction is performed:

$$\begin{aligned} R_l(j) = \frac{1}{M} \sum_{i=-M}^{M} l(i+j)l(i), \quad j \in [M, -M] \\ l(i) = 0 \quad \text{for} \quad i \notin [0, M] \end{aligned} \tag{5.3}$$

where $l(i)$ is the image derivatives line of index i in the motion direction. The operation is obtained rotating the image with an angle of $-\alpha$, where α is equals to the blur direction angle: the derivatives along X axis of the resulting image are

then calculated. The auto-correlations responses are accumulated in a histogram: the global minimum falls in the blur extent estimation. In Figure 5.2(b), the ACF for an image with motion-blur extent of 30 pixels. In our implementation, after a histogram-smoothing step, we search for negative peaks over a certain threshold (e.g., 2-3 pixels): the presence of noise in the images can introduce negative peaks in the auto-correlation function at very low value of extent.



(a) (b)

Figure 5.2: (a) The motion-blur direction identification: the global minimum falls in the blur direction estimation (in degrees). (b) The average ACF used for estimation of the extent. The global minimum falls in the blur extent estimation (in pixels).

5.2.2 Image restoration

If the estimated PSF extent is over a certain threshold (e.g., we use 6), an image restoration step is performed. We use as deconvolution filter an efficient implementation of the Wiener filter. In frequency domain and without noise, Eq. 5.1 can be write as:

$$G(u, v) = F(u, v)H(u, v) \tag{5.4}$$

In this case, with the knowledge of the PSF $h(x, y)$, the uncorrupted image $f(x, y)$ can be recovered by the Inverse Fourier Transform of $G(u, v)/H(u, v)$: this is

(a) (b)

Figure 5.3: (a) The Blurred image. (b) The image restored with Wiener filter.

commonly referred to as the *Inverse Filtering method*. Unfortunately Inverse Filtering method has some problems: $1/H(u,v)$ does not necessarily exist and the filter not performs well in presence of noise. Images grabbed by a CCD or CMOS camera are not noise-free: a better choice is the Wiener filter [90], a well-known and effective deconvolution filter, that is designed to minimize the error due to image noise:

$$W = \frac{H^*}{|H|^2 + \gamma} \tag{5.5}$$

where H is the 2D Fourier Transform of the PSF (Eq. 5.2), H^* its conjugate and γ a constant that depends on noise presents into the image (γ can be interpreted as the reciprocal of the signal-to-noise ratio). Restored image can be obtained by the Inverse Fourier Transform of:

$$F(u,v) = W(u,v)G(u,v) \tag{5.6}$$

where $G(u,v)$ is the 2D Fourier Transform of the corrupted image. In Figure 5.3 an example of blurred image restoration. Wiener filter is not the best deconvolution filter: iterative methods ([60, 53]) can produce better results in terms of quality of the restored image. However, these methods have a very high computational cost. Wiener filter is capable to restore very efficiently local structures of the image that can be well detected during features detection step.

5.2.3 Interest Point Detector

The proposed invariant features scheme takes advantage of two successfully approaches: we use a detector method similar to SURF features [5] and the descriptor proposed in SIFT features scheme[58].
The first step is to select a set of interest point that are invariant to scale transformation. This is performed searching for features in a scale space representation of the images [55], obtained convolving the original images $I(x, y)$ with Gaussian smoothing filters $G(x, y, \sigma)$ and increasing standard deviation values σ (normally referred as the *scale* of the smoothed image):

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{5.7}$$

The scale space is divided in *octaves* (i.e. the last smoothed image of the octave has twice the scale of the first). As in [58], each octaves is divided into an integer number s of intervals, with scales $\sigma_i = \sigma_{i-1} * 2^{\frac{1}{s}}$, where σ_0 is the initial scale chosen to be 1.6. We choose $s = 3$, so we compute Eq. (5.7) at scales $1.6, 2.0159, 2.5398, 3.2, 4.0317$. The latest scale is computed to detect local scale space maxima at the higher scale of the octave, i.e. 3.2. Once an octave is completed, the image is resampled to half its original size: this image has obviously twice the scale of the original image. A new octave is then processed on the resampled (smaller) image, using the same σ_i values. Normally the number of the octaves is 4, it can be reduced to obtain much faster computation of the detector. In order to detect interest points, the scaled images $L(x, y, \sigma)$ are convolved with filters that response mainly to invariant local features of the image. Harris and Hessian based detectors response to corners and highly textured points, whereas Difference-of-Gaussian (DoG) (used in [58]) and Laplacian-of-Gaussian (LoG) based detectors response mainly to blobs: the latters descriptors are less stable due to the possibility to detect points closed to contours of straight edges [67]. As in [5], we use the determinant of Hessian of the scaled image for selecting both location and characteristic scale of the interest points: the trace of Hessian is the LoG, taking the determinant points in witch the second derivative change in only one direction are penalized (e.g. straight edges):

$$det(\sigma^2 H(x,y,\sigma)) = det \begin{bmatrix} \sigma^2 L_{xx}(x,y,\sigma) & \sigma^2 L_{xy}(x,y,\sigma) \\ \sigma^2 L_{xy}(x,y,\sigma) & \sigma^2 L_{yy}(x,y,\sigma) \end{bmatrix} \tag{5.8}$$

where in Eq. (5.8) L_{xx}, L_{yy}, L_{xy} are the second derivatives of the scaled images $L(x,y,\sigma)$. The second derivatives are multiply with the square of the scale σ: this is due to the fact that the amplitude of spatial derivatives decreases with scale, so normalization is required for true scale invariance [55]). The implementation strategy is to convolve the initial image with Gaussian smoothing filter at different scales: at this scope, we use an efficient Gaussian smoothing algorithm provided with OpenCV library[1] [15]. First and second derivatives are then computed in scaled images: in Eq. 5.9 the first and second Gaussian derivatives in x direction are computed.

$$\begin{aligned} L_x(x,y,\sigma) &= L(x+1,y,\sigma) - L(x-1,y,\sigma) \\ L_{xx}(x,y,\sigma) &= L_x(x+1,y,\sigma) - L_x(x-1,y,\sigma) \end{aligned} \tag{5.9}$$

First derivatives are stored in memory for efficient computation of the descriptors (see Section 5.2.4), second derivatives are used to compute the determinant of Hessian.

To improve computation speed of the detector, it can be used more approximated Gaussian derivatives: in [5] discretized and cropped filter are used, computed using the *integral images* technique. We show only a slight degradation of the descriptor reliability using this fast method. Once computed the determinant of Hessian for each location of the multi-scaled image, interest point are detected searching for local maxima over scale and location space in a $3\times3\times3$ neighborhood of each point: only local maxima with determinant of Hessian greater than a threshold are selected as interest points. Finally the location and the scale (called *characteristic scale*) of the extracted points are interpolated [16] by fitting a 3D quadratic to the scale-space determinant of Hessian and taking the maxima of this quadratic. This step is useful to obtain a more accurate characteristic scale of the point (negatively affected by the discrete nature of the scale space) and to reduce the localization errors.

[1] http://sourceforge.net/projects/opencvlibrary/

5.2.4 Interest Point Descriptor

In our experience we note that the SIFT descriptor is slightly more stable than SURF descriptor. We decided to implement SIFT descriptor, tuning the parameters of the algorithm to improve reliability.

Orientation assignment

In order to assign orientations to detected interest points, we compute gradients orientations and magnitudes of 16×16 regularly spaced sample points into a square window centered around the interest point. The side length of the window is equal to $12 * scale$, where *scale* is the interpolated characteristic scale (see Section 5.2.3). Gradients magnitudes and orientation of sample points are computed using the stored first Gaussian derivatives in the discretized scale closed to the characteristic scale of the interest point:

$$\begin{aligned} m(x,y,\sigma) &= \sqrt{L_x(x,y,\sigma)^2 + L_y(x,y,\sigma)^2} \\ \theta(x,y,\sigma) &= tan^{-1}\left(\frac{L_y(x,y,\sigma)}{L_x(x,y,\sigma)}\right) \end{aligned} \tag{5.10}$$

The magnitudes are Gaussian-weighted with a circular bivariate Gaussian centered in the interest point with standard deviation equals to $2.5 * scale$. Magnitudes are then accumulated into an orientation histogram with 36 bins representing the discretized orientations of the gradients. After an histogram-smoothing step, the bins with values greater than 0.8 the global histogram maximum are selected: multiple interest points are created with the initial location and scale but with these different orientations (interpolated with histogram neighborhood).

Descriptor assignment

It is selected a square window centered around the interest point with side length of $20 * scale$ and oriented according with its orientation. This region is regularly divided into 4×4 smaller sub-regions, each containing 4×4 regularly spaced sample points. The gradients orientations and magnitudes of the sample points are computed as in Section 5.2.4. Magnitudes are then Gaussian-weighted with a

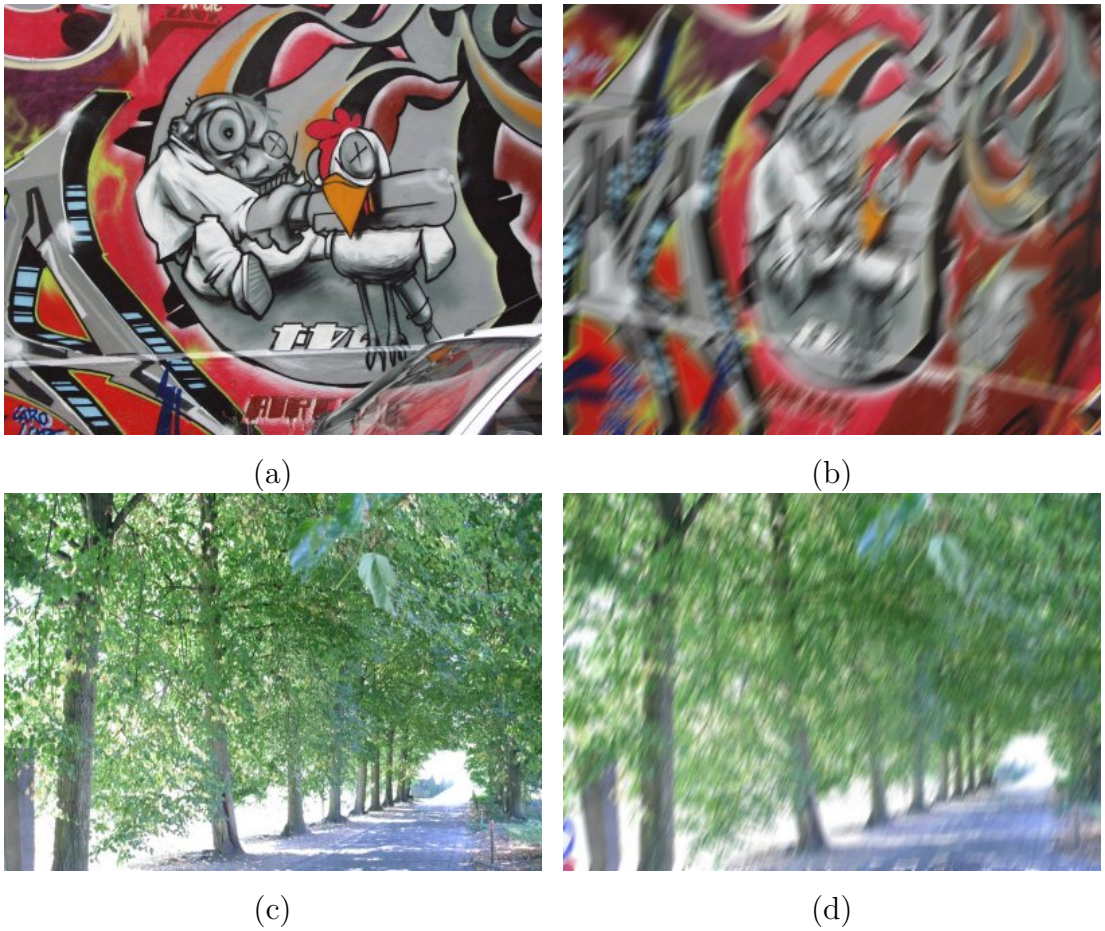
circular bivariate Gaussian with standard deviation equals to $6.7 * scale$ in order to increase stability of the descriptor towards small affine transformation and localization errors. To avoid high variations in the distribution of the gradients inside a sub-region caused by small pixels shift, magnitudes are further weighted with a weight of $1 - d$, where d is the distance of the sample point from the central value of the bin as measured in units of the histogram bin spacing [58]. Each sample point gradient is rotated according to the interest point orientation, then its magnitude is accumulated in a orientation histogram with 8 bins (i.e., 8 discretized orientations) characteristic of the sub-region. The 4×4 8-bins histograms form the 128-entry descriptor of the selected interest point. The descriptor is finally normalized to an unit vector in order to obtain invariance toward contrast variations.

5.2.5 Experiments

Figure 5.4: The VStone Robovie-M robot used in the experiments.

We tested the proposed technique using both a standard dataset [2] with added synthetic motion-blur (some example in Figure 5.5) and real images with motion-blur effect grabbed by the CMOS camera that equip our humanoid robot (Figure

[2] http://www.robots.ox.ac.uk/~vgg/research/affine/



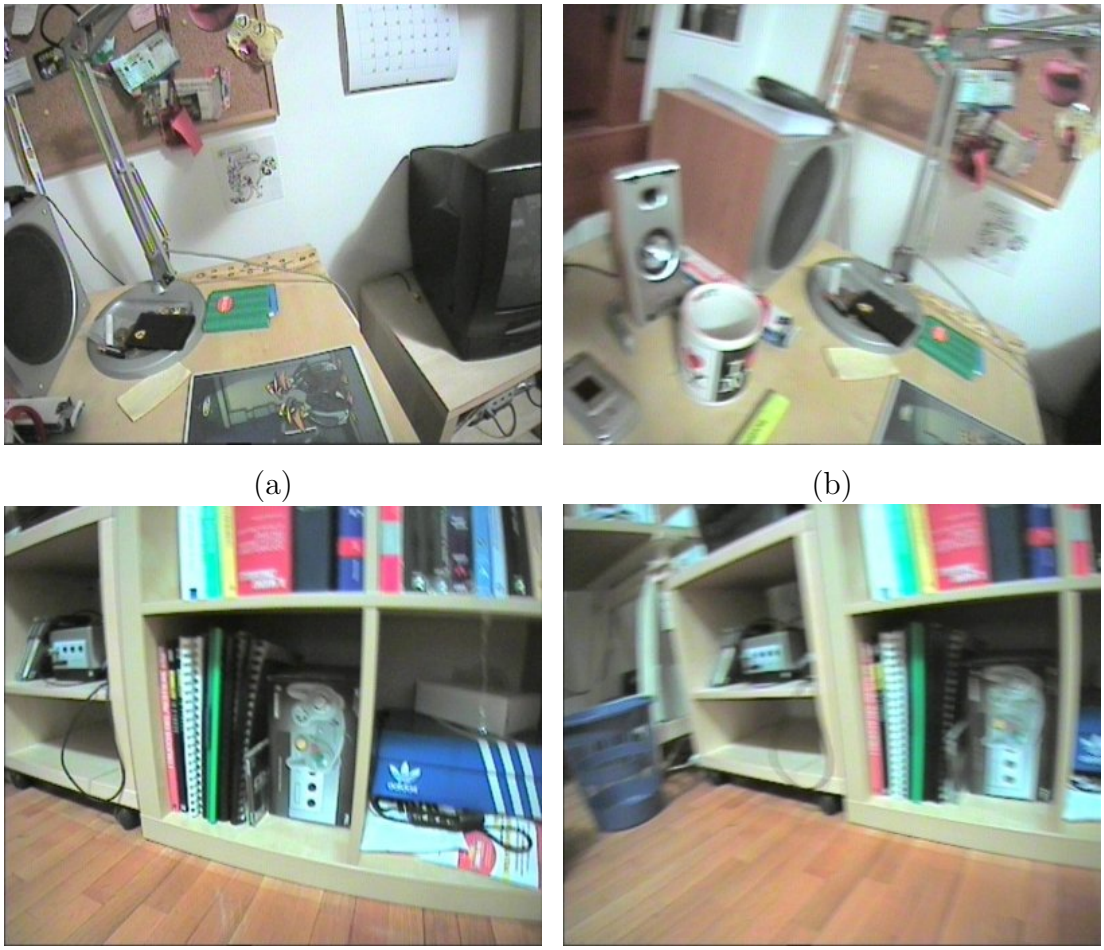
(a) (b)

(c) (d)

Figure 5.5: Some of the standard dataset images used in the experiments. Second image of every pair is blurred with a synthetic linear motion function with different directions and extents.

5.4, some images in Figure 5.6). Aim of our tests is to evaluate the effectiveness of the proposed method in matching images taken in a real humanoid robot scenario in presence of motion-blur phenomena.

Testing image pairs are composed by two images of the same scene taken from different viewpoint. The first image is still, the second image is affected by motion-blur effect. The standard dataset we used is provided with *homographies* (plane projective transformations) between images: the map between the two images is known, the exact correspondence of every point in one frame to the corresponding points in the other frame is known. We can determine in this



(a) (b)

(c) (d)

Figure 5.6: Some of the real images used in the experiments. In (a),(b) the image pair 1, in (c),(d) the image pair 2. Second image of every pair present some motion-blur phenomena, PSF parameters are not a-priori known.

case ground truth matches and also the accuracy (i.e. the localization error of the matches). For real images set , we manually identified the correct matches between frames (Figure 5.9). Our approach is compared to the SIFT features scheme [58] and to the SURF features scheme [5]. Comparisons are performed using well-known implementation of these methods[3] [4], without changing the algorithms standard parameters. SIFT features implementations usually double the

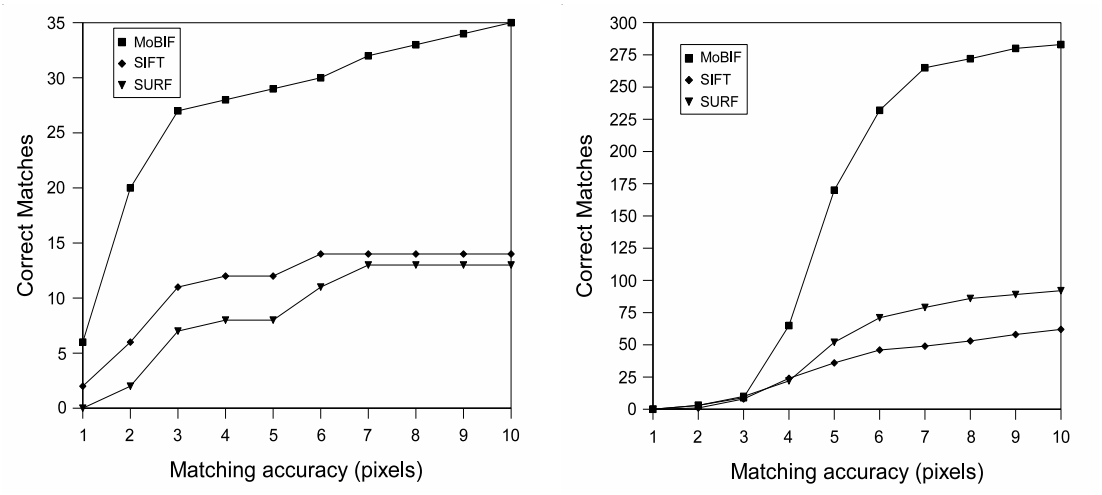
[3]http://web.engr.oregonstate.edu/~hess/index.html

[4]http://www.vision.ee.ethz.ch/~surf/

image size before starting features detection in order create more sample points. This step increase the computational cost and decrease robustness of matching in presence of motion-blur phenomena: for these reasons, we skip the resize step. We refer here to the proposed descriptor-detector scheme as *MoBIF* (Motion-Blur Invariant Features). For all tested approaches, we use the Nearest Neighbor Distance Ratio matching strategy (see [68]), with distance ratio equal to 0.5. In Figure 5.7 are presented matching results of the standard dataset image pairs in Figure 5.5. Images 5.5(b) and 5.5(d) are blurred with synthetic motion-blur function of directions -45 and 23 degrees and extents of 30 and 20, respectively. The matching accuracy is the distance in pixels between the ground truth match and the obtained match. MoBIF approach outperforms SIFT and SURF in both the number of correct matches and the localization accuracy. This is very important especially in visual odometry tasks, where the accuracy in matching affect significantly results in motion estimation. Results for some real images are presented in Figure 5.8: the X axis represent the image pairs used in matching process. Image pairs 1 and 2 are shown in Figure 5.6. Estimated motion-blur extents are in these cases 13, 14, 12, 20, 23 and 19, respectively. Also with real images MoBIF outperforms other approaches, with higher number of correct matches (Figure 5.8(a)) and a very high and stable correct matches ratio over all detected matches (Figure 5.8(b)). Especially with large motion-blur function extent (test image pair 5, estimated extent equals to 23) our approach preserves the reliability in matching (Figure 5.9) where SIFT and SURF techniques tend to fail.

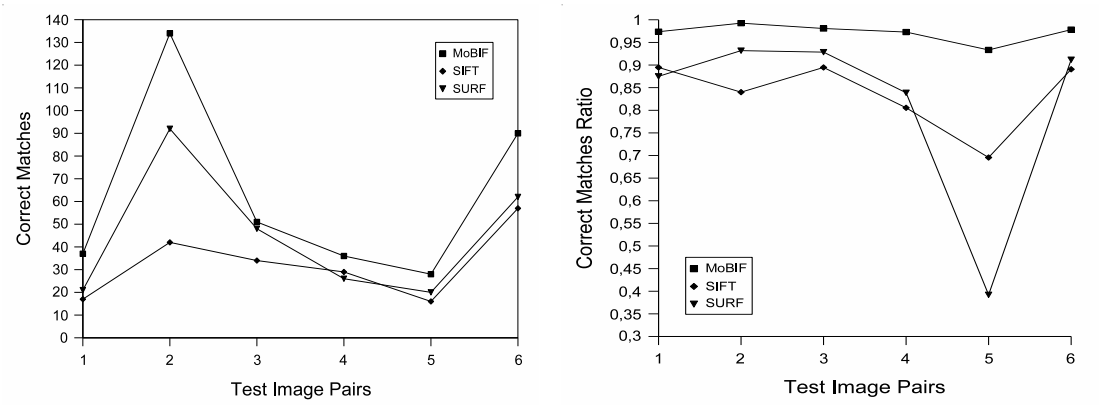
5.3 Second approach: adapting the scale-space representation

The first approach we propose is based on a deconvolution techniques (the Wiener filter) that aims to completely restore the images affected by motion blur.
Unfortunately the quality of the restored image strongly depends on the accuracy of the PSF estimation. Wrong PSF used for the deconvolution can produce unacceptable resulting images. Moreover, this method assumes a linear motion blur and the presence of PSF uniform in the whole image. Even if in simple small



(a) (b)

Figure 5.7: Correct matches for the standard dataset images of Figure 5.5. Accuracy is the distance in pixels between the ground truth match and the obtained match.



(a) (b)

Figure 5.8: (a) Correct matches for the real images set. In the x axis, the correspondent image pair. Estimated PSF extents are 13, 14, 12, 20, 23, 19, respectively. (b) Correct matches ratio over all detected matches

humanoid robots (e.g., Fig. 5.4 and Fig. 1.1(a,b)), these assumption can hold to a certain extent [87], from our experiments, however, we experienced that the cameras of robots with complex kinematics, like for instance the humanoid platform

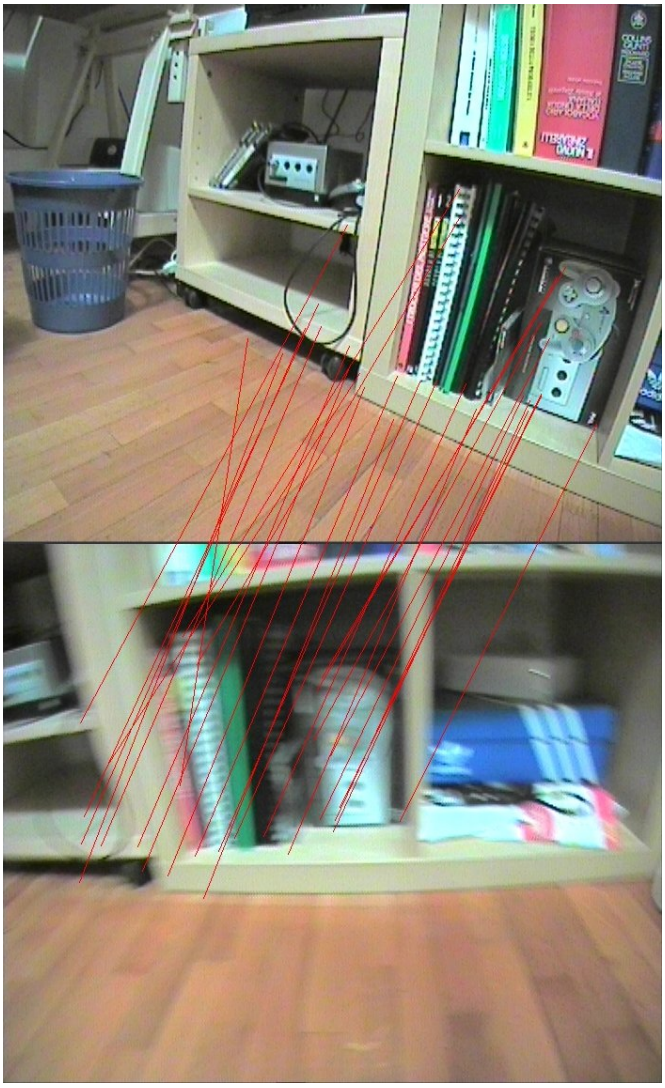


Figure 5.9: MoBIF detected matches for the test image pair 5.

of the NimbRo Robocup Team [7] (Fig. 1.1(c)), performs complex movements resulting in different translation and rotation of the image that can introduce non-linear and non-uniform motion blur effect. In these case, conventional deconvolution techniques can easily fail. Instead of trying to restore the original, unblurred images, we propose an adapted scale-space representation that tries to overcome the negative effect of the motion blur in the invariant features detection and description process. With respect to the method presented in Sec. 5.2, we improved the estimation of the PSF by relaxing the constraint of a uniform PSF

over the hole image and which leads to a better estimate than the simple Wiener filter deconvolution presented in our first approach.

The scale-space theory of Lindeberg [55] aims to represent the input image at different scales and it is at the base of the scale-invariant feature detectors and descriptors such as SIFT and SURF (Sec. 3.5). Scale-space representation is obtained convolving the original images $f(x, y)$ with a set of Gaussian filters with zero-mean $g(x, y, \sigma)$ and with increasing standard deviations σ (normally referred to as the *scale* of the smoothed image):

$$l(x, y, \sigma) = g(x, y, \sigma) * f(x, y) \tag{5.11}$$

If one uses the conventional scale-space representation for images affected by motion blur, it will blur with Gaussian noise the image $b(x, y)$ that is already blurred with the motion blur $h(x, y)$. Thus, the resulting filter is not the desired Gaussian filter, but the composition of a Gaussian filter plus the motion blur filter (applied to the uncorrupted image by the motion of the camera). The motion blur filter can be approximated to be Gaussian, but cannot be approximated to be with equal marginal standard deviations. Thus, the resulting filter is no longer circular symmetric. Therefore, we propose to compute the scale-space representation of an image corrupted by motion blur by finding an appropriate non-circular symmetric $g'(x, y)$, determined from the PSF of the actual motion blur in the image, that convolved with $h(x, y)$ approximates a Gaussian filter with equal marginal standard deviations. In other words, *we smooth less the image along the motion blur direction*. We obtain from Eq. 5.1 and Eq. 5.11 (omitting for simplicity the additive noise):

$$l'(x, y, \sigma) = g'(x, y) * h(x, y) * f(x, y) \tag{5.12}$$

where $g'(x, y)$ is a zero-mean Gaussian smoothing filter with different marginal standard deviations. The proposed strategy is to find a $g'(x, y)$ filter that minimize the sum of squared difference between l and l' over the whole image (here, w is image width and h is image height in pixels):

$$\sum_{x=1}^{w} \sum_{y=1}^{h} (l - l')^2 \tag{5.13}$$

For the distributivity and associativity properties of the convolution operator one can write:

$$l - l' = (g(x, y, \sigma) - g'(x, y) * h(x, y)) * f(x, y) \tag{5.14}$$

So, for an image $f(x, y)$, we have to find a filter $g'(x, y)$ that minimizes the difference:

$$g(x, y, \sigma) - g'(x, y) * h(x, y) \tag{5.15}$$

Let us define as σ (i.e. the scale), the marginal standard deviation of $g'(x, y)$ in the direction perpendicular to the PSF direction, and σ' the marginal standard deviation in the PSF direction. One might think that $g'(x, y)$ could be easily obtained in the frequency domain by a standard deconvolution techniques as Wiener filter, but, for the reason explained above, without an accurate estimation of the real PSF $h(x, y)$, results are very poor. Thus, we compute the value of σ' by minimizing the function (5.15) in the discrete domain (i.e., using discrete kernel's filter): we use the Levenberg-Marquardt algorithm (LMA) for the solution of least squares problems in non-linear case. For example, given the PSF $h_0(x, y)$ with extent d and direction $\theta = 0$ and given the $scale = \sigma$, we compute using LMA the σ' that minimize:

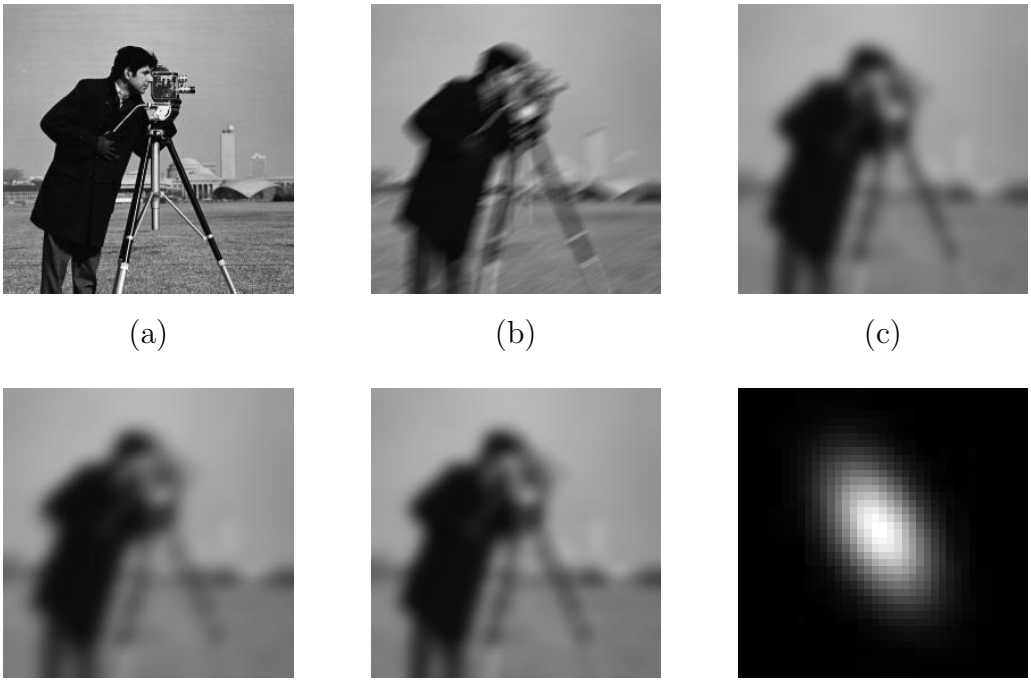
$$g(x, y, \sigma) - g'(x, y, \Sigma) * h_0(x, y), \;\; \Sigma = \begin{bmatrix} \sigma' & 0 \\ 0 & \sigma \end{bmatrix} \tag{5.16}$$

where Σ is the covariance matrix of the adapted Gaussian filter $g'(x, y, \Sigma)$. For a general PSF $h(x, y)$ with $\theta \neq 0$, we rotate the Gaussian kernel obtained for $h_0(x, y)$ according to θ (see Fig. 5.10).

5.3.1 PSF clustering

For PSF estimation, we use the approximated version of the *whitening method* [107] described in Sec. 5.2.1.

The proposed adapted scale-space representation assumes that the PSF is linear: this is, in general, an approximation of the real PSF, that also isn't usually uniform in the whole image. In order to take into account of the non-uniform nature of the PSF, we introduce a clustering step that aims to divide the image in sub-regions characterized by different PSF.

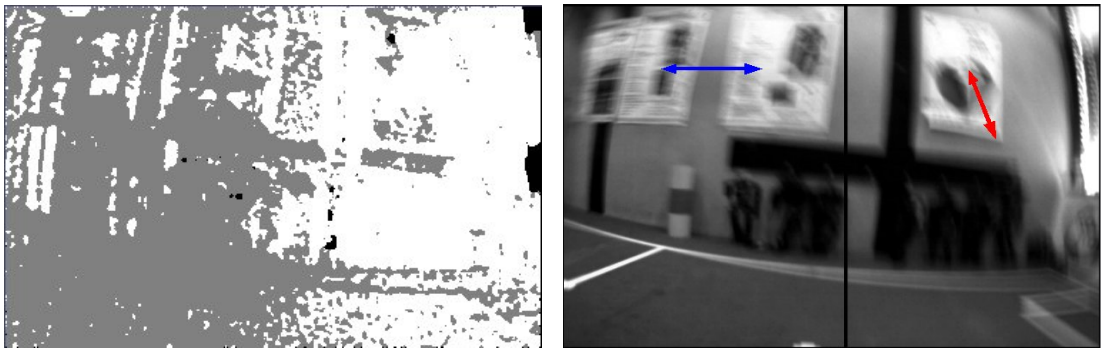


(a) (b) (c)

(d) (e) (f)

Figure 5.10: (a) The Cameraman original image. (b) Original image synthetically blurred with PSF extent $d = 18$ and direction $\theta = \frac{5}{6}\pi$. (c) Original image smoothed with circular symmetric bivariate Gaussian kernel with $\sigma = scale = 6.4$. (d) Motion blurred image (b) smoothed with circular symmetric bivariate Gaussian kernel with $\sigma = scale = 6.4$. (e) Motion blurred image (b) smoothed with non circular bivariate Gaussian kernel with marginal standard deviation $\sigma = scale = 6.4$ in the direction perpendicular to the PSF and in this case $\sigma' = 3.69$ (computed with the LMA algorithm) in the PSF direction. (f) The adapted Gaussian filter used to obtain (e). We can see that (e) tends to approximate the original smoothed image (c) better than (d).

The segmentation of the image is performed using a modified version of the K-means clustering algorithm [12]. Formally, we divide the image points in K clusters where the 3-dimensional vector $\mu_k = (x_{\mu_k}, y_{\mu_k}, \alpha_{\mu_k})$ (here, x_{μ_k}, y_{μ_k} are the image coordinates, α_{μ_k} is the PSF direction (discretized) and $k = 1, \ldots, K$) is a prototype associated with the k^{th} cluster. One can think of the μ_k as representing the centroids of the clusters with uniform PSF with direction α_{μ_k}. Given an



(a) (b)

Figure 5.11: (a) The result of the clustering process for the image Fig. 5.1: Using the K-means algorithm (here with $K = 2$), each point is assigned to a cluster that is characterized by an (approximated) uniform PSF in the region close to the cluster centroid. (b) Based on segmentation performed in (a), the image is divided in rectangular subregions with assigned uniform PSFs. Red and blue arrows represent the computed directions of the PSF in each single subregion: about 0° on the left side and around 80° on the right side.

image point X_i with coordinates x_i, y_i and H_i being the histogram (normalized to unit vector) that holds the absolute values of the derivatives in each discretized directions for this point (see section 5.2.1), we define the distance from a cluster centroid μ_k as the weighted Euclidean distance:

$$d(X_i, \mu_k) = H_i(\alpha_{\mu_k}) * \sqrt{(x_{\mu_k} - x_i)^2 + (y_{\mu_k} - y_i)^2} \tag{5.17}$$

where $H_i(\alpha)$ is the response of the absolute derivatives along the direction α for the sample X_i: This response tends to be a minimum in the motion blur direction. Here's the algorithm:

1. For each sample point i, compute the histogram H_i that holds the absolute values of the derivatives in each discretized direction, 18 in our case (see section 5.2.1). Each histogram is normalized to the unit vector.

2. Compute, as accumulation of the histograms of point 1), the global histogram that, for each discretized direction, holds the sum of the responses in that direction for all the sample points. Extract the K directions with

minimum responses and assign them as initial choices for the direction α_{μ_k} of the the K cluster centroids μ_k. For all cluster centroids, assign at x_{μ_k}, y_{μ_k} the center of the image.

3. Using the distance (Eq. 5.17), assign each sample point to the closest cluster, i.e., the cluster with the closest centroid.

4. Re-assign each sample point to the mode of its 8-neighbors sample points, i.e., to the cluster that occurs most frequently in its 8 neighbors (Fig. 5.11 (a)).

5. For each cluster with centroid μ_k, compute as accumulation the histogram H_{μ_k} that holds the sum of the the responses of the single histograms of the sample points that fall in the cluster.

6. Re-compute the centroids μ_k with coordinates x_{μ_k}, y_{μ_k} equal the mean of the coordinates of the sample points assigned to the corresponding cluster and with α_{μ_k} corresponds to the directions with minimum response in histograms H_{μ_k}.

7. Repeat from point 3 until convergence.

8. For each cluster, compute the PSF extent as explained in section 5.2.1

9. Divide the image in rectangular subregions with uniform PSFs based on their cluster centroids (Fig. 5.11 (b)).

We experimentally found out that $k = 2$ yields good results.

5.3.2 Finding distinctive features

After the PSF clustering step, we have a set of rectangular subregions characterized by a local PSF (Fig. 5.11 (b)): For each subregion we can now easily compute the adapted scale-space representation, as explained at the beginning of section 5.3, using the local PSF.
The detection and the description of the interest points are performed following a method similar to that described in Sec. 5.2.4. The scaled images L are computed in this case according to the local PSF based on the adapted scale-space

representation explained before, i.e. in the Eq. 5.11 we use the adapted Gaussian filter $g'(x, y, \Sigma)$ of Eq. 5.16.

Motion blur effects tend to suppress the high frequency components, the resulting image so loses a lot of small details: It happens that some interest point extracted during detection step represents in reality very simple and not much distinctive features, that they can compromise the stability of the feature matching step producing more outliers. In order to avoid this issue, we introduce a discarding process based on the Shannon entropy of the normalized descriptor. If we take a normalized feature descriptor $s(i)$, we can see it as a probability mass function, with possible values $1, \ldots, n$ the indexes of the bins of the descriptor, in the case of SIFT descriptor $n = 128$. We can compute the entropy as:

$$H(s) = -\sum_{x=1}^{n} s(i)log(s(i)) \tag{5.18}$$

We notice that simple and not much distinctive features tends to obtain descriptors with low entropy. For each descriptor, we first compute the entropy, then we compute the mean μ_H and the standard deviation σ_H of all entropy values. We finally discard all the descriptors with entropy values less than $\mu_H - \sigma_H$. This step improves noticeably the stability of the following features matching precess.

5.3.3 Experiments

We implemented our detection-descriptor scheme in C++ using the efficient OpenCV image processing library[5] [15]: the whole process take on average 1 second for a 640X480 image on a 2Ghz core 2 PC.
As experimental platforms, we used the custom built NimbRo Robocup Team humanoid robot and the commercial Kondo KHR-1 HV humanoid robot (Fig. 1.1(b,c)).
We compared our detection and descriptor scheme to the SIFT features [58] and to the SURF-128 features [5] (i.e., the improved version of the SURF features). Comparisons are performed using well-known implementation of these methods[6,7]

[5] http://sourceforge.net/projects/opencvlibrary/
[6] http://www.cs.ubc.ca/~lowe/keypoints/
[7] http://www.vision.ee.ethz.ch/~surf/

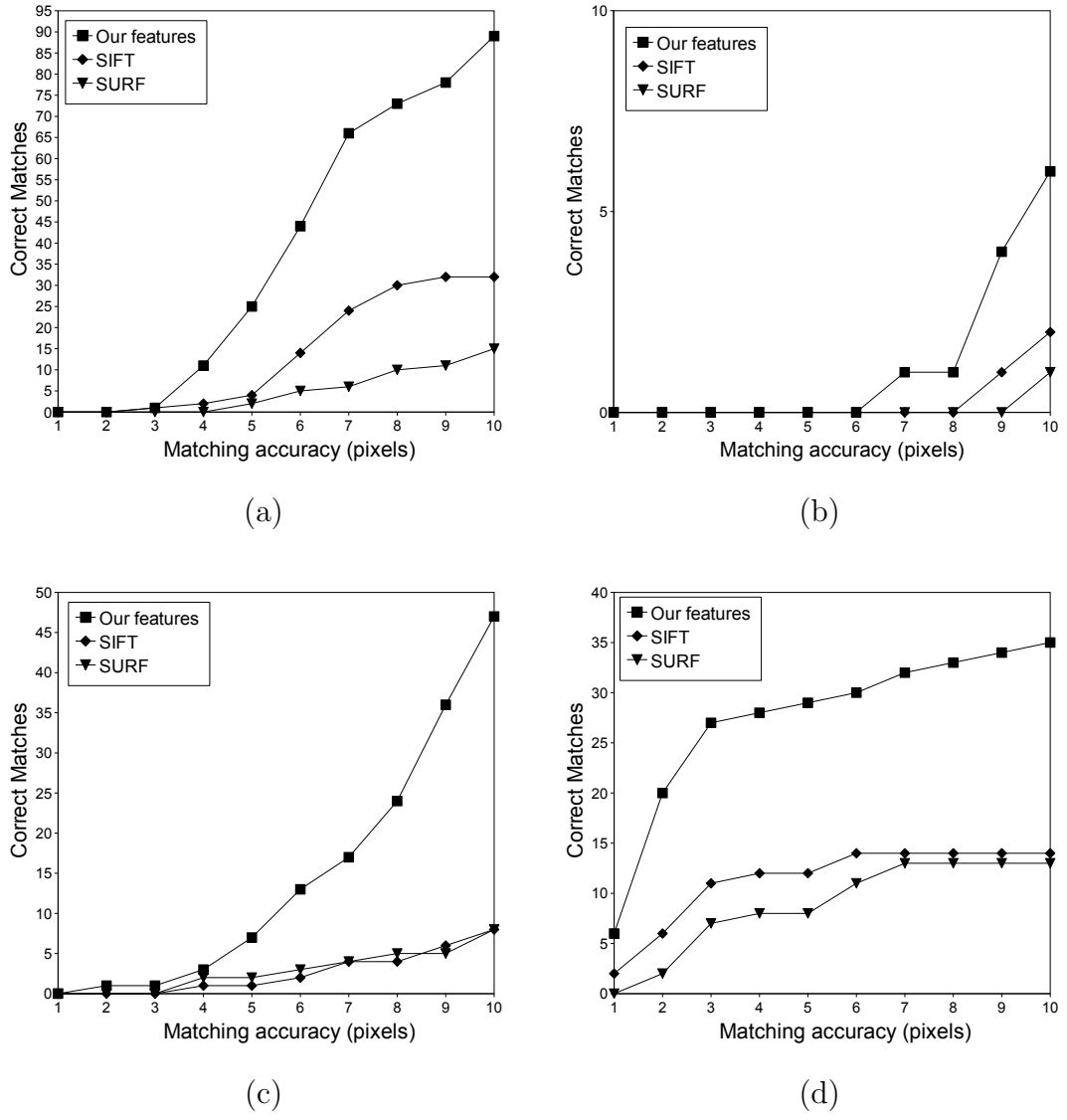


Figure 5.12: Correct matches for the standard dataset images. (a) Image pair 1 and 3 of the *graf* series. (b) Image pair 1 and 6 of the *boat* series. (c) Image pair 1 and 6 of the *trees* series. (d) Image pair 1 and 6 of the *leuven* series. Accuracy is the distance in pixels between the ground truth match and the obtained match.

without changing the standard parameters of the algorithms. The input data are from two sources: (i) a standard dataset[8] with added synthetic motion blur, (ii) sequences of images grabbed by the CMOS camera of a walking humanoid robot

[8]http://www.robots.ox.ac.uk/~vgg/research/affine/

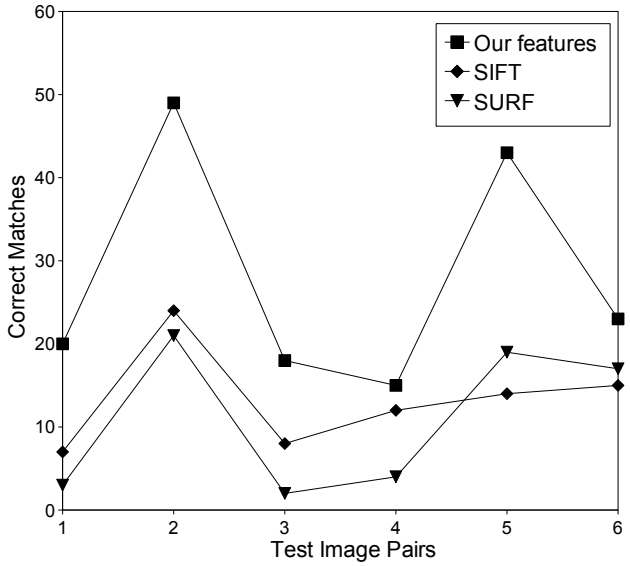


Figure 5.13: Correct matches for the real image sets. In the x-axis, the corresponding image pairs are denoted. The y-axis indicates the number of correct matches. Our approach finds much more feature correspondences compared to SIFT and SURF.

affected by real motion blur effect. Testing image pairs are composed by two images of the same scene taken from different viewpoint. One or both the images are affected by motion blur. The standard dataset we used is provided with *homographies* (plane projective transformations) between images: the map between the two images is known, the exact correspondence of every point in one frame to the corresponding points in the other frame is known. We can determine in this case ground truth matches and also the accuracy (i.e., the localization error of the matches). For the real images set, we manually identified the correct matches between frames. For all tested approaches, we use the Nearest Neighbor Distance Ratio matching strategy (see [68]), with distance ratio equal to 0.5. Fig. 5.12 presents matching results for image pairs of the standard dataset. One or both the image images of the pairs are blurred with synthetic motion-blur functions with different directions and extent variable between 10 an 40 pixels. The matching accuracy is the distance in pixels between the ground truth match and the obtained match. As can be seen, our approach outperforms SIFT and SURF-128

in both the number of correct matches and the localization accuracy. This is very important especially in visual odometry tasks, where the accuracy in matching affect significantly results in motion estimation. Results for some real images are presented in Fig. 5.13: Here, the x-axis represents the image pairs used in matching process. As the results demonstrate, also with real images our approach outperforms the others with respect to the higher number of correct matches.

5.4 Visual odometry

We match the described features between pairs of frames using an efficient Best Bin First (BBF) algorithm (Sec. 3.5.3, [8]) that finds an approximate solution to the nearest neighbor search problem. The algorithm is similar to the kd-tree search algorithm, where the tree is explored searching for the node that is closer to the input descriptor. The BBF algorithm only search M candidates, and returns the nearest-neighbor for a subset of queries.

Given five corresponding points, it's possible to recover the relative positions of the points and cameras, up to a scale. This is the minimum number of points needed for estimating the relative camera motion from two calibrated views, using the so called *five-point algorithm* (see Sec. 3.4). The Five-point algorithm offers many benefits compared with other relative pose problem solutions, as the well-known Eight-point algorithm. The Five-point algorithm needs fewer correspondences to find a solution. Moreover, it is essentially unaffected by the planar degeneracy and it still works for planar scenes where other methods fail. The estimation accuracy of the Five-point algorithm is also higher than other solutions to the relative pose problem. In our visual odometry approach, we use the efficient solution to the Five-point relative pose problem[9] proposed by Nister [81]. We assume that the camera used in the visual odometry is fully calibrated, i.e., intrinsic matrix K is given. For a static scene point projected in two views, we can write:

$$m'^{T} F m = 0 \tag{5.19}$$

[9]We use the Five-point algorithm implementation provided with the VW34 library by Oxford's Active Vision Lab.

where F is a fundamental matrix and m and m' are the image points expressed in homogeneous coordinates for the first and second view, respectively. If the camera is calibrated, the fundamental matrix is reduced to an essential matrix, denoted by E, and the relationship becomes;

$$q'^T E q = 0 \tag{5.20}$$

with $q = K^{-1}m$ and $q' = K^{-1}m'$. Using the Five-point algorithm with five correspondences q_i, q'_i, $i = 1, \ldots, 5$, one can obtain at most ten possible essential matrices (including complex ones) as solutions of the problem. For each essential matrix four combinations of possible relative rotation R and translation T of the camera can be easily extracted [81]. In order to determine which combination corresponds to the true relative movement, the constraint that the scene points should be in front of the camera for both the two views is imposed. The image points are triangulated into 3D points [45] using all the combination of R and T. The final solution is identified as configuration more compliant with the given constraints.

We use the Five-point algorithm in conjunction with MLESAC estimator [102]: MLESAC uses the same sampling strategy as RANSAC where minimal sets of correspondences (5 in our case) are used to derive hypothesized solutions. The remaining correspondences are used to evaluate the quality of each hypothesis. Unlike RANSAC, that count the number of inliers, MLESAC evaluates the likelihood of the hypothesis by representing the error distribution as a mixture model. Our mono-camera visual odometry scheme operates as follows:

1. Extract the features from the images using the features detection and descriptor scheme described in Sec. 5.3.

2. Track interest points over two frames using the BBF matching strategy.

3. Randomly chose a number of samples each composed of 5 matches between the first and the second frame. Using the Five-point algorithm generate a number of hypotheses for the essential matrix.

4. Search for the best hypotheses using MLESAC estimator and store the correspondent inliers. The error function is the distance between the epipolar line E_q associated with q and p',

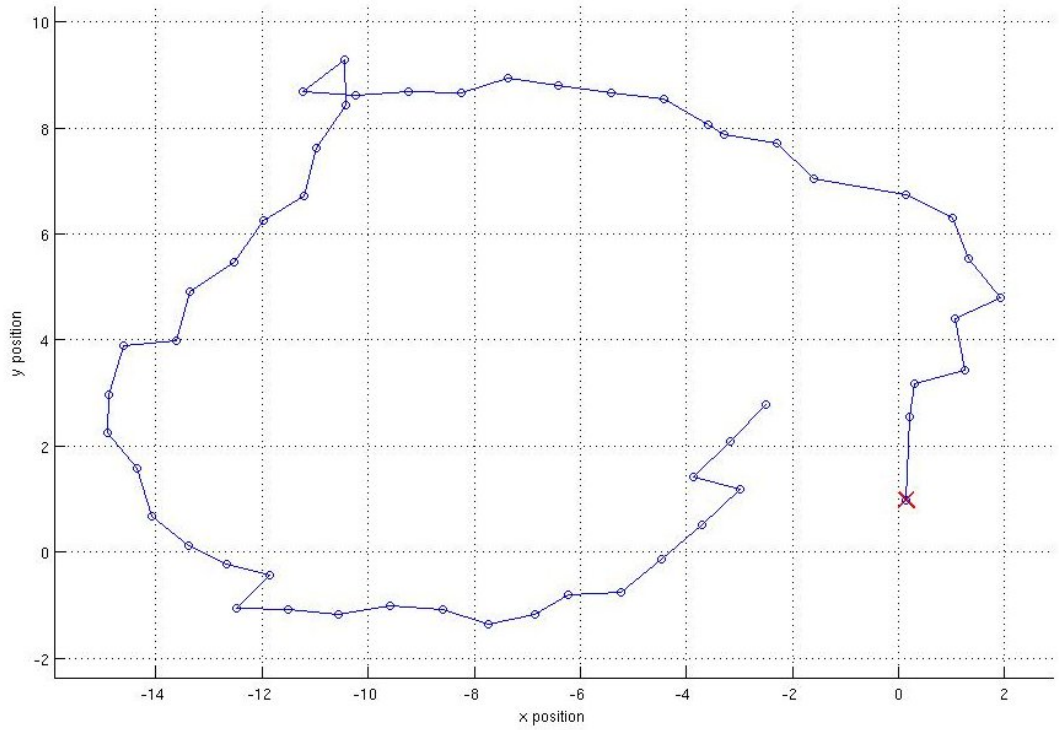
5. Extract from the resulting essential matrix E the four combinations of possible relative rotation R and translation T. Triangulate all the inlier correspondences for each combination. Take the configuration with more 3D points in front of both camera views.

6. If this is not the first time inside the loop, select the features tracked in the present reconstruction, that were tracked also in the previous one, and compute using triangulation the depth for both reconstruction. Use these information in conjunction with RANSAC to estimate the scale factor between the present reconstruction and the previous. Put the present reconstruction in the coordinate system of the first reconstruction.

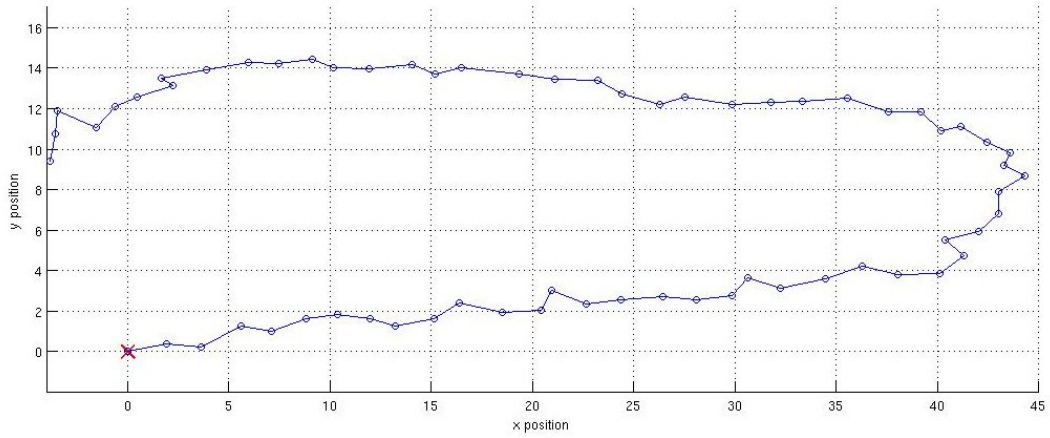
7. Repeat from Point 1.

5.4.1 Experiments

We tested our visual odometry framework with trajectories walked by humanoid robots. The accuracy is measured by checking the error between the start and the endpoints of the recovered trajectory (Fig. 5.14). The path of Fig. 5.14 (a) is a closed loop in which the starting point and the end point are the same point in the environment. The robot walked a loop of about 4-5 m in diameter in the cluttered environment of our laboratory. In the path of Fig. 5.14 (b) the robot walked down a corridor for 5 m, it turned around, and it walked back to almost the same position. In Fig. 5.15 and 5.16 is depicted the estimation of a path followed by the NimbRo humanoid robot where the grabbed images were affected by a very strong motion blur effect.
Unfortunately, it was not possible to record the ground-truth of the robot, but the robot path was closely surveilled the paths of Fig. 5.14 are calculated up to a scale, one can see that the proposed visual odometry can reliably estimate the motion of the robot, even it is not so accurate when the robot is turning. This is the reason why the start and end points do not overlap in Fig. 5.14 (a) and the mutual distance is a bit too large in Fig. 5.14 (b). In fact, the reconstructed paths are open-up because the robot rotation was underestimated. Unfortunately, we cannot report a comparison with SIFT and SURF approaches on this visual



(a)



(b)

Figure 5.14: Estimation of the robot motion using the proposed visual odometry framework for two closed trajectories. The red crosses are the start points of the trajectories.

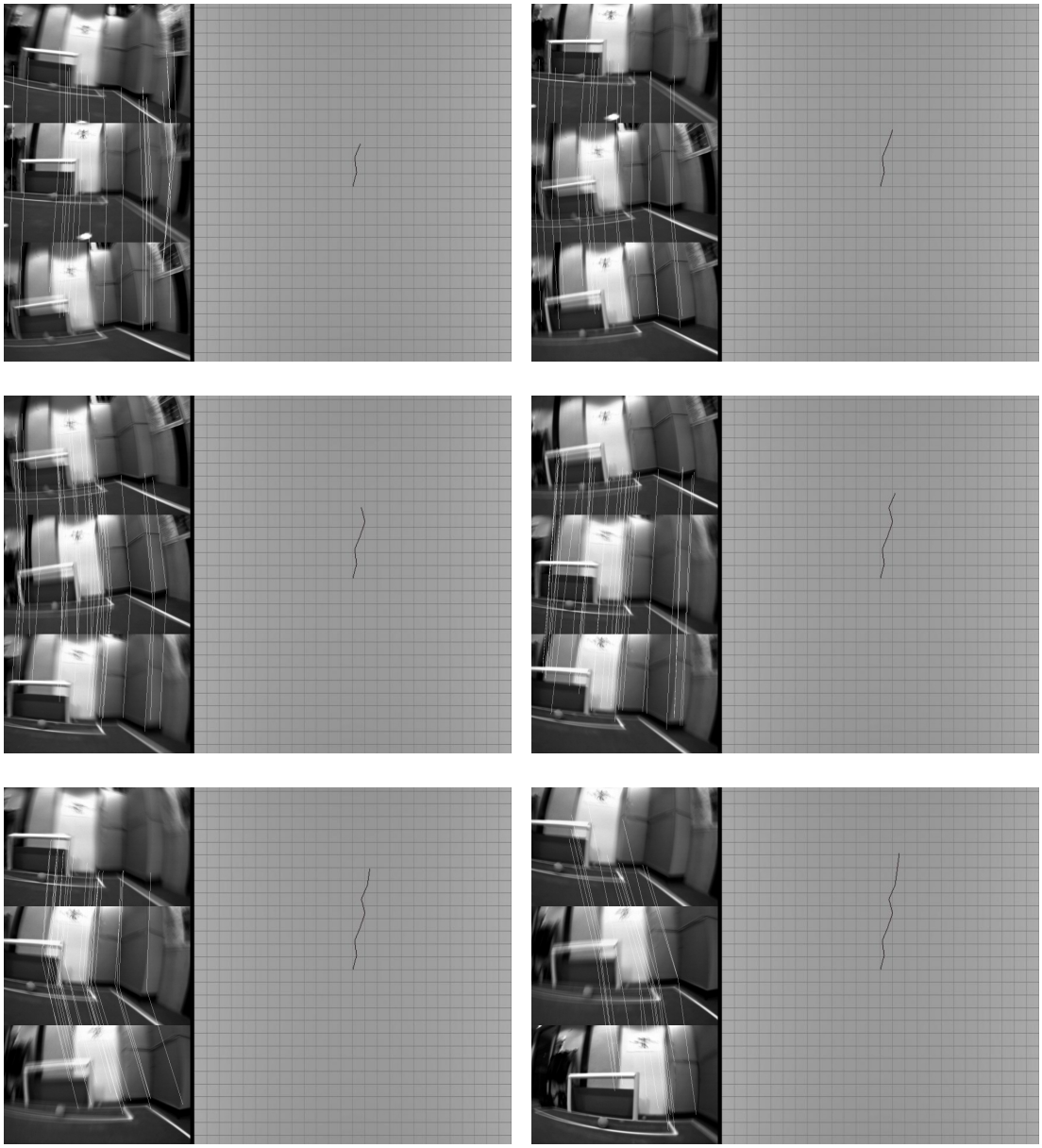


Figure 5.15: Estimation of a path followed by the NimbRo humanoid robot (first part). The images are affected by a very strong motion blur effect. On the left of every frame the matches detected using the approach proposed in Sec. 5.3. On the right, the current path estimation obtained with the proposed visual odometry strategy.

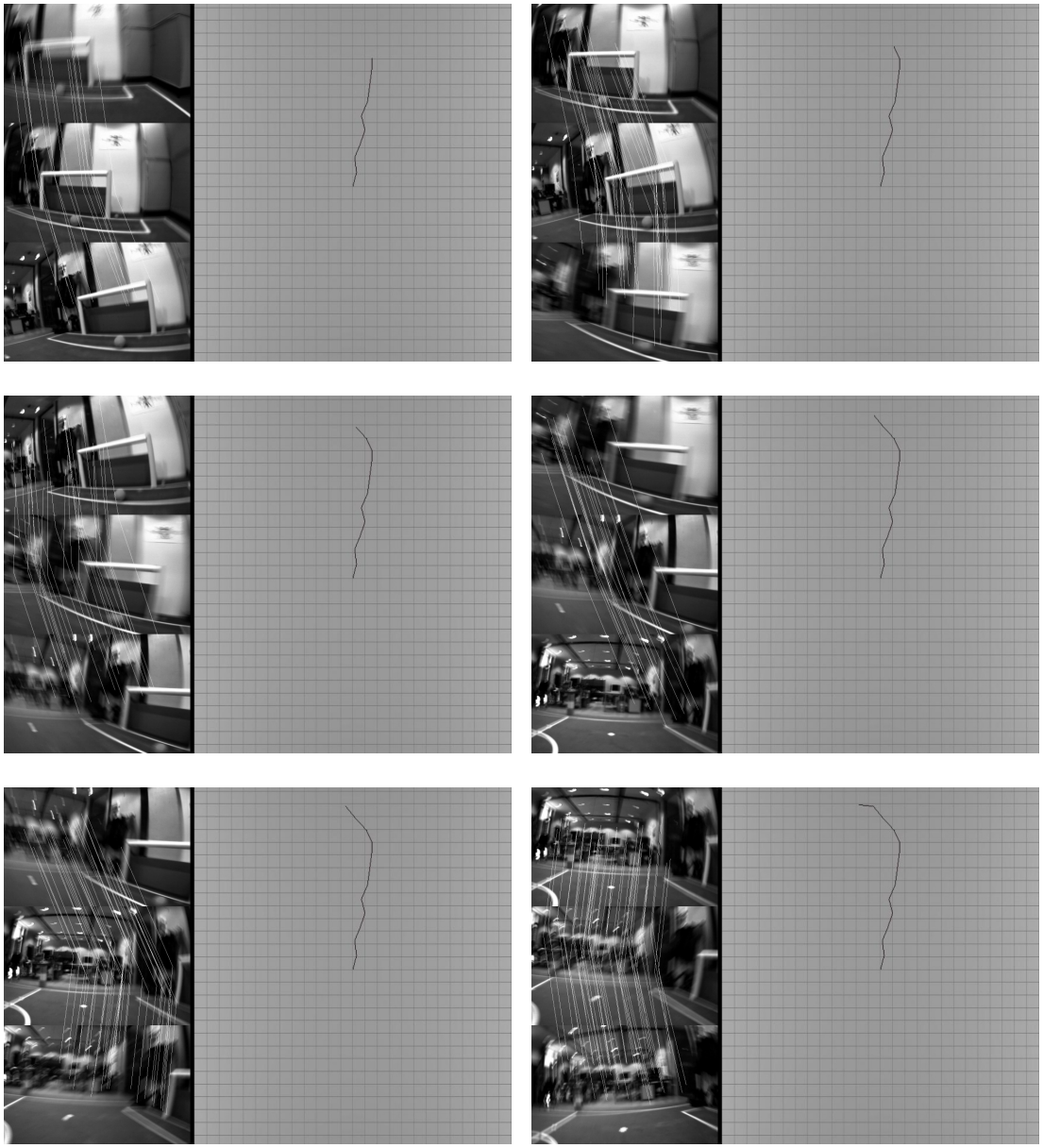


Figure 5.16: Estimation of a path followed by the NimbRo humanoid robot (second part). The images are affected by a very strong motion blur effect. On the left of every frame the matches detected using the approach proposed in Sec. 5.3. On the right, the current path estimation obtained with the proposed visual odometry strategy.

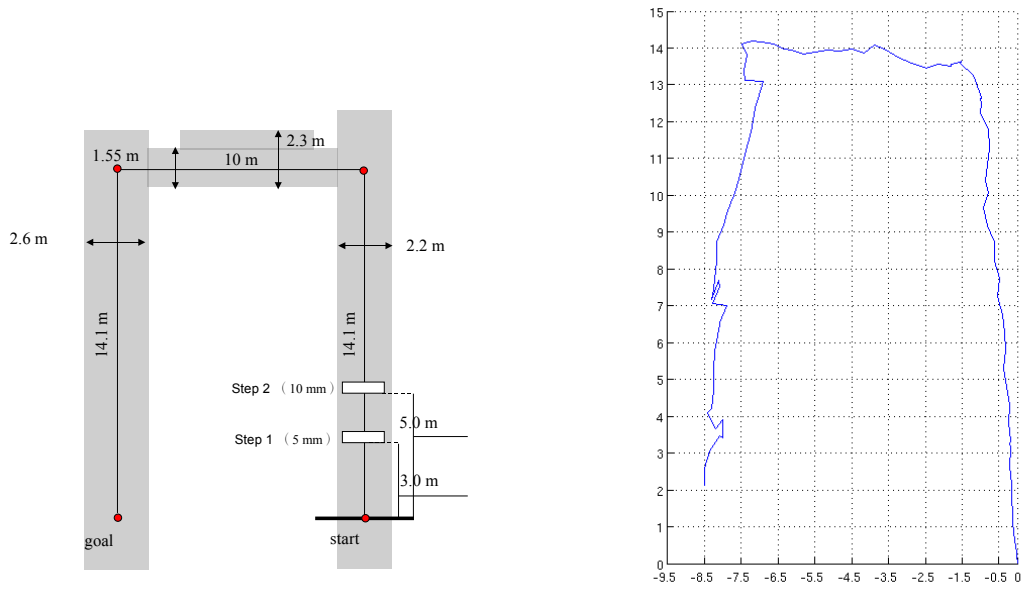
odometry experiment, because both approaches did not pick enough features in the image sequence to reliable reconstruct the path. Indeed, the unmodified environments in which we performed the experiments where quite dim and most of the surfaces did not have bright patterns. Moreover, as reported in Fig. 5.13, the motion blur affecting the images in the walking sequence lowered even more the number of feature detected by SURF and SIFT approaches.

5.4.2 Testing visual odometry on a wheeled robot

Our visual odometry strategy is also tested in a 40 meters trajectory covered by an omnidirectional wheeled robot. The accuracy is tested checking the error between the start and the endpoints of the recovered trajectory. No assumption about the planar movement of the robot nor about the camera initial configuration are taken into account: a full 3D visual odometry is performed. In Fig. 5.17(a) the map of the corridor where the robots performs the trajectory. The path followed by the robot is almost superimposed to the black line. In Fig. 5.17(b) the 2D projection of the estimated path. The pose estimation is more accurate in the first (right) corridor: this is because along this corridor there are more local features than others corridor, so with more (correct) matches between features the odometry is more reliable. At the end of the path the global drift is less than 3 meters. Moreover, from Fig. 5.17(c) (the 3D estimated path), we can see that the estimated 3D trajectory is essentially coplanar.

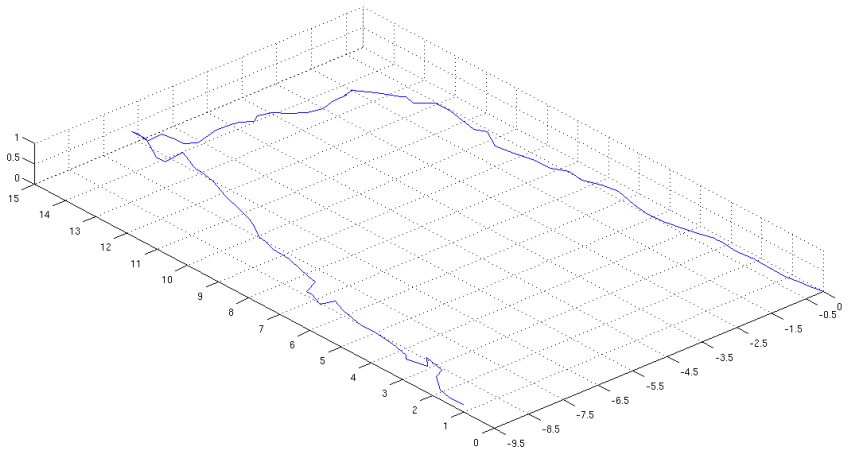
5.5 Summary

In this chapter, we presented a novel framework for visual odometry with a single camera robust even to non-uniform motion blur.
We developed two improved feature detector and descriptor schemes that can find good correspondences even in heavily blurred images, such as the ones grabbed by robots performing brisk movements. The proposed methods outperforms the SIFT and the SURF approach in detecting and matching corresponding features between two images in which one image or both of them are corrupted with motion blur. We evaluated our methods on images taken from standard datasets



(a)

(b)



(b)

Figure 5.17: Estimation of a wheeled robot motion using the proposed visual odometry framework for a 40 m trajectory. (a) The experiment configuration (the path followed by the robot is almost superimposed to the black line); (b) The 2D projection of the estimated path; (c) The 3D estimated path (the axes of the path are manually scaled to obtain a correct metrical result).

and on images grabbed by walking humanoid robots.
We also reported experiments for successful visual odometry estimation using both small humanoid robots and a omnidirectional wheeled robot.

Chapter 6

Visual SLAM Based on Floor Plane Homographies

6.1 Introduction

In this chapter, we propose an homography-based approach to Visual SLAM using only visual information provided by a perspective camera mounted on a humanoid robots.

This approach aims to reconstruct an intensity map of the floor plane on which a humanoid robot walks. This map can be used to localize the robot inside an indoor environment, to recognize previously visited locations, and to recognize the parts of the floor where there are no obstacles.

The proposed system is based on the assumption that the robot moves on a planar environment and that portions of this plane are projected on the image plane. The method is based on the extraction of the homographies that relate pixels in the camera image plane and points that lie in the floor plane, using an efficient tracking method [95, 10]. The map estimation is then performed using a Rao-Blackwellized Particle filter [71].

From a perception point-of-view, the floor plane is searched in the images extracting connected areas using an edge detection and a flood fill algorithm. These areas represent candidate planar patches of the 3D scene.

An Efficient Second-Order method [95, 10] is used to track over a sequence of images the extracted boundaries and therefore to obtain the homographies, that

represent the coplanar relation between different views of a plane. Outlier regions, i.e. connected areas that don't represents planar patches, are discarded during the tracking process as well.

The proposed system assumes that the camera is fully calibrated, then it is possible to recover the relative rotation and translation, and the normal to the tracked plane. Only planar patches with normal vector close to the normal of the floor plane measured when the humanoid robot is standing upright are taken into account. Pixels that belong to floor plane are therefore projected to a single image representing the map of the floor plane.

The SLAM problem is solved using a Rao-Blackwellized Particle Filter. For each particle (potential trajectory) an image (map) of the floor plane is held in memory and updated for every incoming plane projection. The next generation of particles are sampled at each step using the rigid transformation extracted by the homography that relate the points that lie in the floor plane.
The importance weight of each particle is updated according to the drawn pose and the last plane projection, the map of each particle is updated consequently.

We report some experiments performed using a Kondo KHR-1HV humanoid robot walking in an indoor environment that show the effectiveness of the proposed approach in building reliable intensities map of the floor plane.

6.2 Motivations

Currently, most of the Visual SLAM approaches are based on points features: features are detected and tracked between consecutive frames, the motion of the camera and the locations in 3D of the tracked point features are hence estimated concurrently using probabilistic estimation techniques.
A reported in Sec. 3.2, Davison *et al.* [25] proposed a point features-based SLAM approach using a single perspective camera and EKFs (Extended Kalman Filter). The 3D map of the points are built using the bearing only information provided the camera. A similar approach, but based on the FastSLAM framework particle,

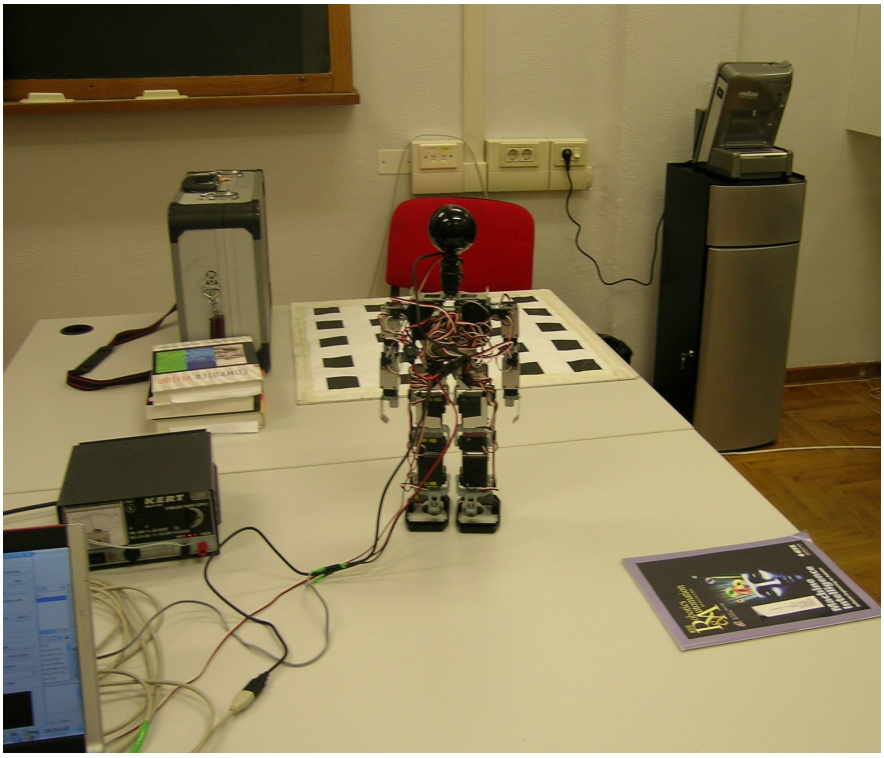


Figure 6.1: The Kondo KHR-1HV humanoid robot walking on a table during the experiments.

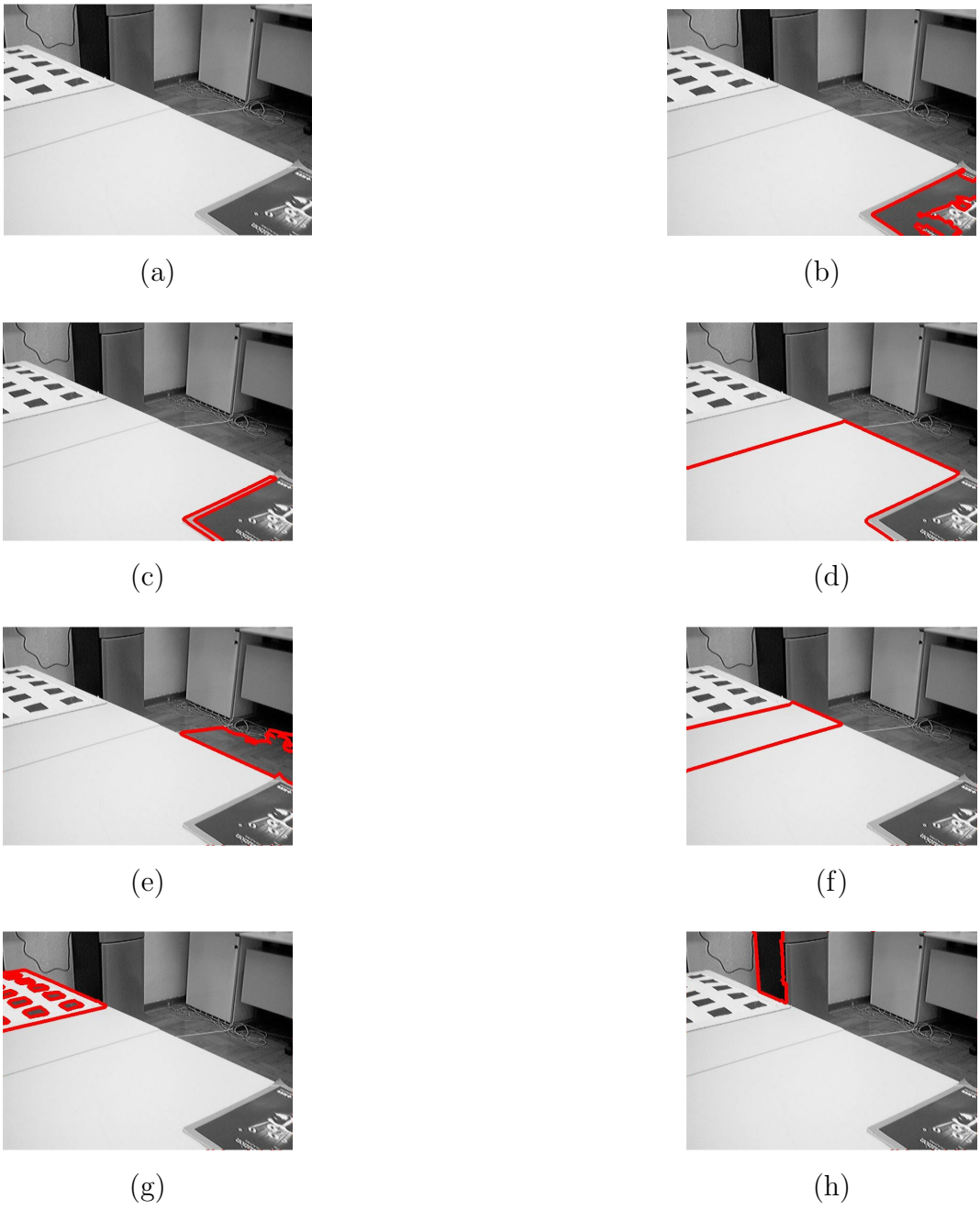
is presented in [30].
In [48] a high resolution digital elevation maps is built from a sequence of stereo-vision image pairs where interest points are detected and matched between consecutive frames. A visual motion estimation algorithm is used to predict the movements, an extended Kalman filter is used to estimate both the position parameters and the map.
In [32] a dense metric map of 3D point landmarks for large cyclic environments are built using the Rao-Blackwellised Particle Filter, where SIFT features are extracted from stereo vision and motion estimates are based on sparse optical flow. Eade [31] presents a monocular visual SLAM approach using line features, where an efficient algorithm for selecting such landmarks is defined.

In this thesis, we used point features to estimate the robot motion: in Chapter

5 we propose an invariant features robust even to non-uniform motion blur, the matched features are used to estimate the motion of the robot in 3D exploiting the epipolar geometry constraints. In this case, we *didn't* build a map of this points: the points features were tracked only only during the time they are inside the field-of.view of the camera. Once a point feature go outside the field-of.view, this is simply forgotten. On the other hand, when a wheeled or humanoid robot is moving quickly, usually points stay inside the field-of.view of the camera only for a short time: this is a critical issue for the conventional point-based Visual SLAM approaches, that assume to track points inside the image for long periods. Moreover a map of 3D points doesn't provide many usefull information about the environment: usually mapped points are sparse and provide the robot with an incomplete information about the appearance of the environment and the obstacles that it can encounter.

In our opinion, point features tracking techniques are powerfull tools for estimating the motion of the robot using only vision, but we believe that a map of the environment built using vision should holds more information than the locations of some hundreds points.
In order to learn with vision more usefull maps with exhaustive information about the appearance of the environment, and also the obstacles inside it, it is necessary to exploit more complex visual features.
Higher level landmarks are exploited in [95], where 3D camera displacement and the scene structure are computed directly from image intensity discrepancies using an efficient second-order optimization procedure for tracking planar patches. Due to complexity of the detecting and tracking higher level landmarks like planes, not many works of Visual SLAM approaches based on planes have been presented.

In this chapter, we aim at introducing a method that exploit a robust plane tracking techniques inside a probabilistic framework in order to reconstruct an appearance map of the floor plane.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.2: The red pixels in (b),...,(h) represent the boundaries of a set of putative planes extracted from the input image (a). Planar patches found in image (e) and image (h) will be discarded, because not belonging to the plane on which the robot is walking.

6.3 Extracting putative floor planes portions

Planes inside an indoor environment are often painted with an uniform color. The lack of chromatic discontinuities makes problematic the tracking task. We hence focus on the *shape* of the planes, to be more precise we decide to track planar surfaces considering their boundaries, i.e. the image edges (the peaks of the image derivatives).

First, we need to extract from the images grabbed by the robot's camera all the boundaries of the regions of uniform color. In order to correct the radial distortion of the camera [108], we use the algorithm provided by Bouguet [14]. Then we extract the edges into the image using the Canny operator [17]. The Canny operator first smooths the image with a Gaussian kernel: we chose to set the standard deviation of this filter to $\sigma = 1$. Then a first derivative operator is applied to the smoothed image to highlight regions with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. Then a non-maximal suppression process is performed in order to isolate those points that with a high probability lie to an edge. The Canny operator results in a binary image, where a pixel value $p(x, y) = 1$ means that this points lie in an edge.

In order to correctly detect connected regions in the images, we apply to the binary image a dilation operator followed by an erosion operator [42]. The effect of the dilation operator is to enlarge the boundaries of regions, while the erosion operator erodes away the boundaries of regions. The final effect of this couple of operator is to join partially disconnected edges.

Closed boundaries are hence highlighted using a flood fill operator, that fill closed regions with an uniform color. We select only regions with area greater than a fixed threshold: this is because small regions don't provide robust planarity constraints in the following tracking process.

All the pixels in the original images that lie in the boundaries of this regions are finally chosen as boundaries of putative planes. In Figure 6.2 the boundaries of a set of putative planes extracted from an input image are depicted.

In order to robustly track the planes with the technique described in the next section, we set the thickness of boundaries to 6 pixels.

6.4 Floor Plane Detection and Tracking

In our approach we assume that the robot moves on a planar environment, and that portions of this plane are projected on the image plane. A walking humanoid robot performs small trunk motions to stabilize its posture, and the camera mounted on its head moves accordingly to this body swing. So, given a camera-centered coordinate systems, the floor plane normal vector changes dynamically during the motion. Aim of this section is to provide a method to extract the image pixels that belong to the floor plane and therefore to calculate the current rotation, translation and the normal to the floor plane.

6.4.1 Theoretical background

Image pixels that belong to a planar surface in the 3D space are related between different views of the same scene by a homography [45]. Suppose that $\mathbf{X}$ is a 3D point in homogeneous coordinates that lies in a scene plane $\pi = (\mathbf{n}^{*\top}, -d^*)^\top$, where $\mathbf{n}^*$ is the normal to the plane in the reference frame, $\|\mathbf{n}^*\| = 1$, and d^* is the distance from the plane. Given the normalized camera matrix for the two views of the plane:

$$\mathbf{P}^* = [\mathbf{I}|\mathbf{0}] \quad , \; \mathbf{P} = [\mathbf{R}|\mathbf{t}] \tag{6.1}$$

where $\mathbf{R} \in \mathbb{SO}(3)$ and $\mathbf{t} \in \mathbb{R}(3)$ are the rotation and the translation matrices between the two frames, we can obtain the projections of the 3D point $\mathbf{X}$ in the two views, i.e. $\mathbf{x}^* = \mathbf{P}^*\mathbf{X}$ and $\mathbf{x} = \mathbf{P}\mathbf{X}$. For the first view, any point on the ray $\mathbf{X} = (\mathbf{x}^{*\top}, \rho)^\top$ projects to $\mathbf{x}^*$. If $\mathbf{X}$ lies in the plane π, then $\pi^\top\mathbf{X} = 0$, so we can determine ρ, $\mathbf{X} = (\mathbf{x}^{*\top}, \frac{\mathbf{n}^{*\top}\mathbf{x}^*}{d^*})^\top$.
For the second view we have:

$$\begin{aligned} \mathbf{x} = \mathbf{P}'\mathbf{X} = [\mathbf{R}|\mathbf{t}]\mathbf{X} \\ = \mathbf{R}\mathbf{x}^* + \frac{\mathbf{t}\mathbf{n}^{*\top}\mathbf{x}^*}{d^*} = \left(\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^{*\top}}{d^*}\right)\mathbf{x}^* \end{aligned} \tag{6.2}$$

Where:

$$\mathbf{H} = \mathbf{R} + \mathbf{t}\mathbf{n}_d^{*\top}, \quad \mathbf{n}_d^{*\top} = \frac{\mathbf{n}^{*\top}}{d} \tag{6.3}$$

is called *homography matrix*.
From the normalized coordinates of an image point $\mathbf{x}$, we can obtain the image coordinates in pixel $\mathbf{p}$ with:

$$\mathbf{p} = \mathbf{K}\mathbf{x} \tag{6.4}$$

where $\mathbf{K}$ is a camera intrinsic parameters matrix.

The Efficient Second-order Minimization proposed, among the other, in [95] aims to find the homography (i.e., the rotation matrix, translation and the normal vector, Eq. 6.3) that relates two images $\mathcal{I}^*$ and $\mathcal{I}$, starting from a set of q pixels in the reference images $\mathcal{I}^*$ and searching for the solution *iteratively updating* an initial guess until convergence.

We can define:

$$\mathbf{G}(\mathbf{T}, \mathbf{n}_d^*) = \mathbf{K}\mathbf{H}(\mathbf{T}, \mathbf{n}_d^*)\mathbf{K}^{-1} \tag{6.5}$$

$\mathbf{G} = g_{ij}$ defines a projective transformation in the image, i.e. it is the homography between images points. Moreover $\mathbf{G} \in \mathbb{SL}(3)$, where $\mathbb{SL}(3)$ is the Special Linear Group, i.e. the determinant of $\mathbf{G}$ is equal to 1.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.6}$$

is the matrix that hold the rotation matrix and the translation (see Eq. 6.1).

We define $\mathbf{w}[\mathbf{G}](\mathbf{p})$ as a $\mathbb{P}^2$ automorphism, $\mathbf{w}[\mathbf{G}] : \mathbb{P}^2 \rightarrow \mathbb{P}^2$, such that (remembering that $\mathbf{G} = g_{ij}$):

$$\mathbf{w}[\mathbf{G}](\mathbf{p}_i^*) = \left[\frac{g_{11}u_i^* + g_{12}v_i^* + g_{13}}{g_{31}u_i^* + g_{32}v_i^* + g_{33}}, \frac{g_{21}u_i^* + g_{22}v_i^* + g_{23}}{g_{31}u_i^* + g_{32}v_i^* + g_{33}}, 1 \right]^\top \tag{6.7}$$

where $\mathbf{p}_i^* = [u_i^* \ v_i^* \ 1]^\top$ are the coordinates of a pixel in the reference image $\mathcal{I}^*$.

We define now the basis $\mathbf{A}_i$, $i = 1, \ldots, 6$ of the Lie algebra $\mathfrak{se}(3)$ [105]. Let the (3×1) standard basis for $\mathbb{R}^3$, i.e. $\mathbf{b}_x = (1, 0, 0)$, $\mathbf{b}_y = (0, 1, 0)$, $\mathbf{b}_z = (0, 0, 1)$, then:

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{b}_x \\ \mathbf{0} & 0 \end{bmatrix}, \mathbf{A}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{b}_y \\ \mathbf{0} & 0 \end{bmatrix}, \mathbf{A}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{b}_z \\ \mathbf{0} & 0 \end{bmatrix} \tag{6.8}$$

$$\mathbf{A}_4 = \begin{bmatrix} [\mathbf{b}_x]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, \mathbf{A}_5 = \begin{bmatrix} [\mathbf{b}_y]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, \mathbf{A}_6 = \begin{bmatrix} [\mathbf{b}_z]_\times & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \tag{6.9}$$

where $[\mathbf{b}_i]_\times$ is the (3×3) anti-symmetric matrix that corresponds to the vector $\mathbf{b}_i = (\omega_x \ \omega_y \ \omega_z)$:

$$[\mathbf{b}_i]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{6.10}$$

It can be shown that for each $\mathbf{x} \in \mathbb{R}^6$ then:

$$\mathbf{T}(\mathbf{x}) = \exp\left(\sum_{i=1}^{6} x_i \mathbf{A}_i\right) = \sum_{i=0}^{\infty} \frac{1}{i!} \left(\sum_{i=1}^{6} x_i \mathbf{A}_i\right)^i \tag{6.11}$$

is a matrix of the type of Eq. 6.6. In other words, moving $\mathbb{R}^6$ with Eq. 6.11 we always obtain a valid matrix of the type defined in Eq. 6.6.

Finally, let $\mathbf{n}_d^* \in \mathbb{R}^3$ the plane normal vector in the reference camera frame scaled by its distance, i.e. $\mathbf{n}_d^* = \left[\frac{\mathbf{n}_1^*}{z^*}, \frac{\mathbf{n}_2^*}{z^*}, \frac{\mathbf{n}_3^*}{z^*}\right]^\top$. Given three noncollinear image points $\mathbf{p}_i^*, i = 1, 2, 3$, we can write:

$$\mathbf{n}_d^{*\top} \mathbf{K}^{-1} \mathbf{p}_i^* = \mathbf{n}_d^* \mathbf{p}_{i_n}^* = \frac{1}{z_i^*} \tag{6.12}$$

where $\mathbf{p}_{i_n}^*$ is the projection in the normalized image of the pixel $\mathbf{p}_i^*$ and z_i^* is the distance to the point $\mathbf{p}_i^*$ in the reference camera frame. Using the inverse depths of the three points, i.e. $\mathbf{z}^* = \left[\frac{1}{z_1^*}, \frac{1}{z_2^*}, \frac{1}{z_3^*}\right]^\top$, we can write:

$$\mathbf{n}_d^{*\top} \mathbf{K}^{-1} [\mathbf{p}_1^* \ \mathbf{p}_2^* \ \mathbf{p}_3^*] = \mathbf{z}^{*\top} \tag{6.13}$$

Then:

$$\mathbf{n}_d^{*\top} = \mathbf{z}^{*\top} [\mathbf{p}_1^* \ \mathbf{p}_2^* \ \mathbf{p}_3^*]^{-1} \mathbf{K} \Rightarrow \mathbf{n}_d^* = \mathbf{K}^\top [\mathbf{p}_1^* \ \mathbf{p}_2^* \ \mathbf{p}_3^*]^{-\top} \mathbf{z}^* \tag{6.14}$$

$$\mathbf{n}_d^* = \mathbf{n}_d^*(\mathbf{z}^*) = \mathbf{M}\mathbf{z}^*, \ \mathbf{M} = \mathbf{K}^\top [\mathbf{p}_1^* \ \mathbf{p}_2^* \ \mathbf{p}_3^*]^{-\top} \tag{6.15}$$

$$\mathbf{z}^*(\mathbf{y}) = \exp(\mathbf{y}) > 0, \ \mathbf{y} \in \mathbb{R}^3 \tag{6.16}$$

The Efficient Second-order Minimization (ESM) procedure aims to find the $\mathbf{T}(\mathbf{x})$ and $\mathbf{z}^*(\mathbf{y})$ (i.e., the $\mathbf{x} \in \mathbb{R}^6$ and the $\mathbf{y} \in \mathbb{R}^3$) that minimizes:

$$\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{q} \left[\mathcal{I} \left\{ \mathbf{w} \left[\mathbf{G} \left(\hat{\mathbf{T}} \mathbf{T}(\mathbf{x}), \mathbf{n}_d^*(\hat{\mathbf{z}}^* \cdot \mathbf{z}^*(\mathbf{y})) \right) \right] (\mathbf{p}_i^*) \right\} - \mathcal{I}^* \{\mathbf{p}_i^*\} \right]^2 \quad (6.17)$$

where "$\cdot$" is the element-wise multiplication, q is the number of tracked pixels that should lie in a planes, $\mathcal{I}^*$ is the reference image (i.e., the image for which its coordinates frame coincides with the world frame) and $\mathcal{I}$ is an image taken from a different point-of-view.

Setting $\theta = (\mathbf{x}, \mathbf{y})$ and $\mathbf{d}(\theta) = \phi(\mathbf{x}, \mathbf{y})$, during every update step we have to find the optimal value such that:

$$\theta^0 \; = \; \text{argmin}_\theta \; \frac{1}{2} \parallel \mathbf{d}(\theta) \parallel \quad (6.18)$$

it can be shown that an efficient second-order approximation of $\mathbf{d}(\theta)$ is:

$$\mathbf{d}(\theta) \backsimeq \mathbf{d}(\mathbf{0}) + \frac{1}{2} \left(\mathbf{J}(\theta) + \mathbf{J}(\mathbf{0}) \right) \theta \quad (6.19)$$

where $\mathbf{J}(\theta)$ is the jacobian of $\mathbf{d}(\theta)$ computed in θ: from 6.19 it is possible to efficiently compute θ^0 (Eq. 6.18).

6.4.2 Selecting the floor plane and estimating the motion

Given a pair of consecutive images grabbed by the robot, we set the first as the reference image. Here we extract putative floor planes portions as described in Sec. 6.3. For each set of pixels (one set for each putative plane) we perform an ESM (Efficient Second-order Minimization), in order to find the optimal $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{n}$ (i.e., the normal to the plane), given that points. The procedure is as follow:

1. Initialize in Eq. 6.17: $\hat{\mathbf{T}}$ to the identity matrix and $\hat{\mathbf{z}}^*$ to $[0\ 0\ 1]^T$
2. Calculate the Jacobian $\mathbf{J}$ and find θ^0 solving Eq. 6.19.
3. Given $\theta^0 = (\mathbf{x}^0, \mathbf{y}^0)$.
4. Update $\hat{\mathbf{T}}$ and $\hat{\mathbf{z}}^*$ as $\hat{\mathbf{T}} \leftarrow \hat{\mathbf{T}} \mathbf{T}(\mathbf{x}^0)$ and $\hat{\mathbf{z}}^* \leftarrow \hat{\mathbf{z}}^* \mathbf{z}^*(\mathbf{y}^0)$.

5. Stop the iteration if the increments becomes lower than a threshold, otherwise restart from point (2)

Some set of pixels don't lie in reality in a planar surface: in this case the ESM tends to diverge from a coherent solution. The procedure in these cases is stopped if the sum of differences between the value of the pixels in the reference image and the corresponding values in the other image becomes larger than a threshold. Only sets of pixels with resulting normal vector close to the normal of the floor plane measured when the humanoid robot is standing upright are taken into account. Therefore, all the selected pixels, together with the pixels that are enclose by the selected boundaries, are clustered in a single set. The ESM is performed again over this set: the estimated rotation matrix $\mathbf{R}$, translation $\mathbf{t}$ and normal $\mathbf{n}$, together with the pixel values, will be used in the next section inside a FastSLAM framework.

6.5 Updating the Intensities Map

The plane selection and tracking phase of the proposed algorithm provide the robot with an estimation of the relative motion between two positions from which two consecutive images are grabbed, with also the normal to the floor plane.
We exploit this information in a probabilistic SLAM framework based on *FastSLAM* method [71], that is an instance of the a Rao-Blackwellized Particle Filters (Sec. 2.3). The intuition of the FastSLAM is to use a factorization of the posterior over the estimated state:

$$p[x_{0:t}, m|z_{0:t}, u_{0:t}] = p[x_{0:t}|z_{0:t}, u_{0:t}] \prod_{i=1}^{N} p[m_i|x_{0:t}, z_{0:t}, u_{0:t}] \tag{6.20}$$

Fast SLAM represents the posterior density functions of the state by means at time t of a set of M particles $\{x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}\}$. In our case, each particle is an hypothesis of the 3D trajectory followed by the robot, together with its associated intensities map m_i. The map is an image in which are *accumulated* the intensities of the pixel that should belong to the floor plane, projected in order to undistort the perspective distortion. In other words, this map represent a *byrd's eye view* of the intensities value of the plane.

6.5.1 Sampling a new pose

At the beginning, we set all particles at the origin of the world frame (i.e., all represent a trajectory of only one step), while we set their intensities maps to zero.

At time t for each particle $S_{t-1}^{[i]}$, $i = 1, \ldots, M$, a new robot pose is sampled from the *motion model*:

$$x_t^{[i]} \sim p[x_t | u_t, S_{t-1}^{[i]}] \tag{6.21}$$

where u_t is the last movement.
The sample model we use derives from the relative rotation matrix $\mathbf{R}$ and translation $\mathbf{t}$ extracted during the plane tracking. From $\mathbf{R}$, we extract the Euler angles *yaw*, *pitch*, and *roll*. The proposal distribution $p[x_t | u_t, S_{t-1}^{[i]}]$ is a Gaussian centered at the estimated movement values (i.e., the translation $\mathbf{t}$ and the Euler angles). For each sample, we sample a relative movements from this distribution. A new temporary particles set $\hat{S}_t^{[i]}$ is therefore generated adding the new poses $x_t^{[i]}$ to the robot path of each particle $S_{t-1}^{[i]}$, i.e. $\hat{S}_t^{[i]} = \{x_t^{[i]} \cup S_{t-1}^{[i]}\}$, with $i = 1, \ldots, M$.

6.5.2 Update the map

The current observation z_t is represented by the set of pixel selected during the tracking phase. All of these pixels should be the perspective projections of points that lie in the floor plane.
For each particle, the position of the robot is provided by the particle's trajectory. while from the ESM tracking is provided the normal $\mathbf{n}$ to the floor plane.
Given these information, the homography matrix that relates the selected pixel and the floor plane can be easily computed, and hence we can projected this pixel into the intensities map that represents the floor plane. For each particle, these pixel values are *accumulated* in the intensities map according to the state (trajectory) of each particle itself.
Every map becomes with a mask of equal size, in which every pixel contains the occurrences that the corresponding pixel in the map has been updated. To recover the gray scale value of a pixel in the map, the (accumulated) value presents in the map should be divided by the corresponding value in the mask.
Before accumulating the projected pixel in the maps, the particles $\hat{S}_t^{[i]}$ are *weighted*

based on the *sensor model* $p[z_t|x_t^{[i]}, m_{j_i}]$.
In our case, the particle weight is the sum of the square difference between the current gray scale value of the pixel in the map and the value of the projected pixel in that point, i.e. given $\{p_0, \ldots, p_k\}$ the set of projected pixel, $f(p_i)$ the gray level value in the image and $\frac{map(h(p_i))}{mask(h(p_i))}$ the gray level value in the projected position (where $h(p_i)$ is the projection function), the particle weight is obtained as:

$$w^{[i]} = \Sigma_{i=0,\ldots,k} \left(f(p_i) - \frac{map(h(p_i))}{mask(h(p_i))} \right) \tag{6.22}$$

Finally, a *resamplig step* is performed: a new particle set $S_t = \{S_t^{[1]}, \ldots, S_t^{[M]}\}$ is generated by selecting particles from the temporary population $\hat{S}_t$ with probability proportional to the weight of each one [100]. Some initial particles may be forgotten and some may be duplicated.

6.6 Experiments

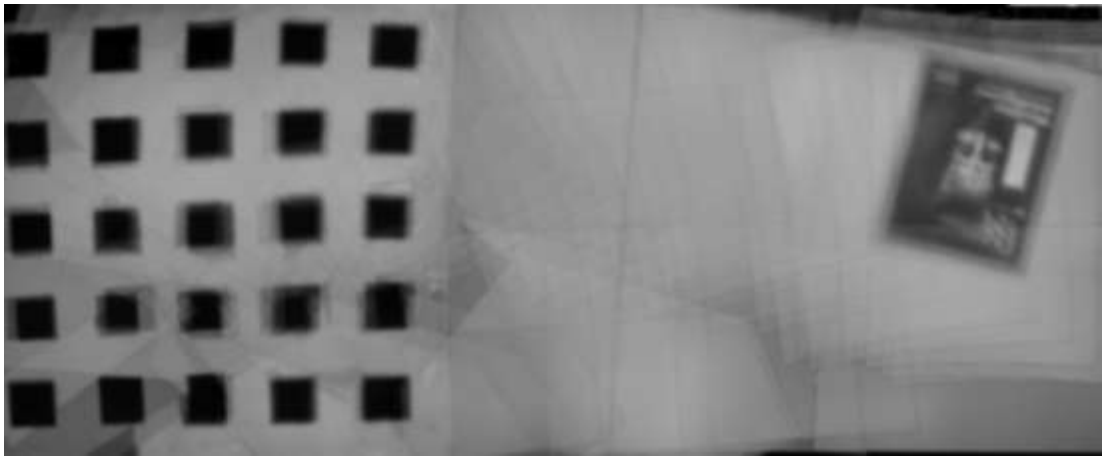


Figure 6.3: An intensity map of the planar surface on which the Kondo KHR-1HV walked.

We tested the proposed Visual SLAM approach in a partially structured envi-

ronment (Fig. 6.1). We use the Kondo KHR-1HV humanoid robot walking slowly on a table. The build intensity map is presented in Fig. 6.3. We can see that both of the planar objects disposed in the plane are well reconstructed.
Due to a non-optimized implementation, the process is not yet able to run in real-time: we are improving the implementation in order to comply with the real-time requirements.

Chapter 7

Conclusions

In this thesis, we faced the Visual SLAM problem with a particular attention to the issues raised by the use of small humanoid robots as experimental platforms. We addressed issues as the low-computational capability of small humanoid robots, the low quality of their sensory information. Moreover we focused on robust and usefull visual environment information like the object planes with respect to less useful point features.
Our goal was to cope with these problems from a vision point-of-view, exploiting new vision algorithms inside the robotics domain and proposing novel techniques, as well.
While most of the contributions in SLAM and Visual SLAM problems still aims to improve the estimation processes involved in these problems, we instead focused on the perception processes involved in Visual SLAM problem, providing novel perceptual techniques well suited to improve and make more stable the current estimation methods.

In this thesis, we presented a new compact image signature based on the Discrete Wavelet Transform that allows memory saving and fast and efficient similarity calculation. We also presented a method to match between perspective images and panoramic images using the proposed signature, where the panoramic images are automatically composed by a humanoid robot with a stitching process from a sequence of perspective images. A loop-closure strategy is hence presented, experiments with real robots shown the effectiveness of our techniques.

The contributions proposed in this Chapter resulted in two publications in [86, 85].
Another contribution presented in this thesis is a 3D visual odometry framework based on monocular images robust even to non-uniform motion blur. The system is based on a new features detection and description scheme in which a possible motion blur effect that affects the images is estimated and minimized, allowing an accurate and robust feature tracking process. The matched features are used to estimate the motion of the robot in 3D, without the usual planar motion assumption that constraint the movement in 2D, using the Five Point algorithm, an efficient and stable epipolar geometry based technique.
The presented approaches are tested on small humanoid robots and on a wheeled robot, as well.
The contributions proposed in this Chapter have been published in [84, 88, 87].
Finally, we propose a Visual SLAM strategy that aims to reconstruct an intensity map of the floor plane on which a humanoid robot walks. The method is based on the extraction of the homographies that relate the pixels in the camera image plane and the points that lie in the floor plane, using an efficient tracking method. The map estimation is performed using a probabilistic techniques called Rao-Blackwellized Particle filter.
Currently, we are improving this approach and we are writing a new paper to be submitted to a international conference.

7.1 Future Works

In order to extend the topics presented in this thesis, we are working on a complete framework that provide an accurate 3D visual reconstruction of indoor unknown environments explored by a humanoid robot.
Our idea is to improve and extend the Visual SLAM strategy based on planar surfaces introduced in Chapter 6, allowing the tracking of multiple planes that are mapped and updated inside a 3D map of intensities. The 3D map can be considered a *3D image* of the environment, where its *voxel* (3D pixels) represent the intensities of the mapped planes.
The idea in this framework is to take advantage of the proposed visual odometry

strategy presented in Chapter 5 for the robot ego-motion estimation, and to match inside a 3D visual maps the tracked planar features. A topological loop-closure strategy like the one presented in Chapter 4 should be exploited in order to correctly detect previously visited locations.

Bibliography

[1] H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localisation using omnidirectional images. In Anil K. Jain, Svetha Venkatesh, and Brian C. Lovell, editors, *Proc. of the 14th International Conference on Pattern Recognition*, volume vol. I, pages pp. 1799–1803, 1998.

[2] Henrik Andreasson, Tom Duckett, and Achim Lilienthal. Mini-slam: Minimalistic visual slam in large-scale environments based on a new interpretation of image similarity. In *Proc. of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[3] A. Angeli, S. Doncieux, J.A. Meyer, and D. Filliat. Incremental vision-based topological slam. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031–1036, 22-26 Sept. 2008.

[4] T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. Ph.D. dissertation, Univ. Sydney, Australian Ctr. Field Robotics, 2002.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proc. of the ninth European Conference on Computer Vision (ECCV)*, 2006.

[6] P. R. Beaudet. Rotationally invariant image operators. *International Joint Conference on Pattern Recognition*, pages 579–583, 1978.

[7] S. Behnke, M. Schreiber, J. Stückler, R. Renner, and H. Strasdat. See, walk, and kick: Humanoid robots start to play soccer. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 497–503, Dec. 2006.

[8] J. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(4):509–522, 2005.

[10] S. Benhimane and E. Malis. Homography-based 2d visual tracking and servoing. *International Journal of Robotic Research*, 2007.

[11] Maren Bennewitz, Cyrill Stachniss, Wolfram Burgard, and Sven Behnke. Metric localization with sift features using a single camera. In *Proceedings of European Robotics Symposium (EUROS), Palermo / Italy*, 2006.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. 2007.

[14] J.Y. Bouguet. Camera calibration toolbox for matlab.

[15] Gary Rost Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[16] M. Brown and D. Lowe. Invariant features from interest point groups. In *BMVC02*, 2002.

[17] J. Canny. A computational approach to edge detection. In *Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 679–697. IEEE, 1986.

[18] R. Cassinis, D. Duina, S. Inelli, and A. Rizzi. Unsupervised matching of visual landmarks for robotic homing using Fourier-Mellin transform. *Robotics and Autonomous Systems*, 40(2-3), August 2002.

[19] K. Choset, H.and Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125 – 137, Apr 2001.

[20] J. Civera, A.J. Davison, and J Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24:932–945, 2008.

[21] JPEG Committee. Jpeg home page. http://www.jpeg.org.

[22] A.I. Comport, E. Malis, and P. Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *IEEE International Conference on Robotics and Automation*, pages 40–45, 2007.

[23] Franklin Crow. Summed-area tables for texture mapping. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984.

[24] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[25] Andrew J. Davison, Ian D. Reid, , Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1–16, 2007.

[26] F. Dellaert. Square root sam. In *In Proc. of Robotics: Science and Systems (RSS)*, page 177184, 2005.

[27] M. N. Do and M. Vetterli. Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Tr. Im. Proc.*, 11(2):146–158, February 2002.

[28] G. Dudek and D. Jugessur. Robust place recognition using local appearance based methods. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 2, 2000.

[29] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *Robotics and Automation Magazine, IEEE*, 13:99–110, June 2006.

[30] E. Eade and T. Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476, June 2006.

[31] E. D. Eade and T. W. Drummond. Edge landmarks in monocular slam. In *British Machine Vision Conference (BMVC)*, volume 1, pages 469–476, 2006.

[32] R. Elinas, P. Sim and J. Little. slam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proc. of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[33] D. Filliat. A bag of words method for interactive visual qualitative localization and mapping. 2007.

[34] D. Filliat. A visual bag of words method for interactive qualitative localization and mapping. 2007.

[35] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[36] J. Folkesson and H. I. Christensen. Graphical slam - a self-correcting map. In *Proceedings of the IEEE International Conference on Robotics and Automation*, page 791798, 2004.

[37] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[38] Emanuele Frontoni and Primo Zingaretti. An efficient similarity metric for omnidirectional vision sensors. *Robotics and Autonomous Systems*, 54(9):750–757, 2006.

[39] Jos Gaspar, Niall Winters, and Jos Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE*

Transaction on Robotics and Automation, Vol 16(number 6), December 2000.

[40] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, Feb. 2007.

[41] H.-M. Gross, A. Koenig, Ch. Schroeter, and H.-J. Boehme. Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2003)*, pages pp. 1505–1511, October 2003, Las Vegas USA.

[42] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[43] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[44] R. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[45] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[46] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH 95 (Los Angeles, California, August 6–11, 1995)*, New York, 1995.

[47] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Proc. of the 15th Int.Conference on Pattern Recognition (ICPR00)*, volume 4, pages pp. 136–139. IEEE Computer Society, September 2000.

[48] Il-Kyun Jung and Simon Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 946, Washington, DC, USA, 2003. IEEE Computer Society.

[49] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proc. of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[50] J. Koenderink and A. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55.

[51] B.J.A. Krse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appareance-based robot localization. *Image and Vision Computing*, vol. 19(6):pp. 381–391, April 2001.

[52] S. Lacroix, T. Lemaire, and C. Berger. More vision for SLAM. In *IEEE International Conference on Robotics and Automation, Roma (Italy), workshop on Unifying Perspectives in Computational and Robot Vision*, April 2007.

[53] R.L. Lagendijk, J. Biemond, and D.E Boekee. Identification and restoration of noisy blurred images using the expectation-maximization algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7):1180 – 1191, 1990.

[54] J.J. Leonard and H.F. Durrant-whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Workshop on Intelligence for Mechanical Systems, IROS '91*, pages 1442–1447, 1991.

[55] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270, 1994.

[56] B. Lisien, Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 448453, 2003.

[57] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(10):133–135, 1981.

[58] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[59] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333 – 349, October 1997.

[60] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79.

[61] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24 (3):169–186, 2007.

[62] C. Mayntx, T. Aach, and D. Kunz. Blur identification using a spectral inertia tensor and spectralzeros. In *Proc. of 1999 International Conference on Image Processing (ICIP)*, 1999.

[63] E. Menegatti, A. Pretto, A. Scarpa, and E. Pagello. Omnidirectional vision scan matching for robot localization in dynamic environments. *IEEE Transactions on Robotics*, 22:523–535, June 2006.

[64] Emanuele Menegatti, Takeshi Maeda, and Hiroshi Ishiguro. Image-based memory for robot navigation using properties of the omnidirectional images. *Robotics and Autonomous Systems, Elsevier*, 47(4):pp. 251–267, July 2004.

[65] Emanuele Menegatti, Alberto Pretto, and Enrico Pagello. A new omnidirectional vision sensor for Monte-Carlo localisation. In *RoboCup Symposium 2004*, 2004.

[66] Emanuele Menegatti, Alberto Pretto, and Enrico Pagello. Testing omnidirectional vision-based Monte-Carlo localization under occlusion. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS04)*, pages pp. 2487–2494, September 2004.

[67] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

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

[69] M.E. Moghaddam and M Jamzad. Motion blur identification in noisy images using fuzzy sets. In *Proc. of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, 2005.

[70] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[71] Michael Montemerlo and Sebastian Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, January 2007.

[72] Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. Walrus: a similarity retrieval algorithm for image databases. *SIGMOD Rec.*, 28(2):395–406, 1999.

[73] S. K. Nayar, H. Murase, and SA Nene. Learning, positioning, and tracking visual appearance. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3237–3244, 1994.

[74] J. Neira, A. J. Davison, and J. J. Leonard. Guest Editorial Special Issue on Visual SLAM. *Robotics, IEEE Transactions on*, 24:929 – 931, Oct 2008.

[75] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Automat.*, 17(6):890897, 2001.

[76] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. 2006.

[77] M. E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. 2006.

[78] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Proc. of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[79] David Nistér. An efficient solution to the five-point relative pose problem. 2003.

[80] David Nistér. Preemptive ransac for live structure and motion estimation. 2003.

[81] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[82] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

[83] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Numerical Recipes in C, 1988.

[84] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA09)*, 2009.

[85] A. Pretto, E. Menegatti, Y. Jitsukawa, R. Ueda, and T. Arai. Toward image-based localization for aibo using wavelet transform. pages 831–838, 2007.

[86] A. Pretto, E. Menegatti, Y. Jitsukawa, R. Ueda, and T. Arai. Image similarity for image-based localization of robots with low-computational resources. *Robotics and Autonomous Systems, Elsevier*, (in press).

[87] A. Pretto, E. Menegatti, and E. Pagello. Reliable features matching for humanoid robots. *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, Dec. 2007.

[88] A. Pretto, E. Menegatti, M. Takahashi, T. Suzuki, and E. Pagello. Visual odometry for an omnidirectional-drive robot. In *6th International Symposium on Mechatronics and its Applications (ISMA09)*, 2009.

[89] A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics*, 22:92–107, 2006.

[90] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume 1. Academic, 1982.

[91] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on Robotics*, 28 (2), October 2008.

[92] S. Se, D. Lowe, and J. Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, 21(8):735, 2002.

[93] J. Shi and C. Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[94] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[95] G. Silveira, E. Malis, and P. Rives. An efficient direct method for improving visual SLAM. In *Proc. of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4090–4095, Italy, 2007.

[96] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. Oct. 2003.

[97] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *International Journal of Robotics Research*, 5:56 – 68, 1986.

[98] Olivier Stasse, Andrew J. Davison, Ramzi Sellaouti, and Kazuhito Yokoi. Real-time 3d slam for a humanoid robot considering pattern generator information. In *Proc. of the 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[99] Adrian Stern, Inna Kruchakov, Eitan Yoavi, and Norman S. Kopeika. Recognition of motion-blurred images by use of the method of moments. *Applied Optics*, 41(11):2164–2171, 2002.

[100] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[101] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *International Journal of Robotics Research*, 23:693 – 716, Aug 2004.

[102] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[103] Tinne Tuytelaars. Local invariant features: What? how? why? when? In *9th European Conference on Computer Vision (ECCV 2006)*, Graz, Austria, May 2006.

[104] M. Vetterli and J Kovacevic. *Wavelets and Subband Coding*. Signal Processing Series. 1995.

[105] F. W. Warner. *Foundations of differential manifolds and Lie groups*. Springer Verlag, 1987.

[106] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, 21(2):208–216, 2005.

[107] Y. Yitzhaky, I. Mor, A. Lantzman, and N. S. Kopeika. Direct method for restoration of motion-blurred images. *Journal of the Optical Society of America A*, 15:1512–1519, June 1998.

[108] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):13301334, 2000.