UNIVERSITY ABOU BEKR BELKAID - TLEMCEN
FACULTY OF ENGINEERING
Department of Electronics

Visual Navigation using Adaptive Learning

Thesis submitted in fulfilment of the requirements for the award of the degree of Doctor in Engineering
by

Sid Ahmed Berrabah

Address
BP 230 - 13000 Chetouane Tlemcen, Algeria
Tel: (+213) 43 28 56 89
Fax: (+213) 43 28 56 85
Email: sa$_b errabah@mail.univ - tlemcen.dz$

Tlemcen, 2012

Acknowledgments

Tlemcen, 2012
Sid Ahmed Berrabah

Contents

List of Figures

List of Tables

Abbreviations

CMEC	Camera Motion Estimation and Compensation
DFD	Displaced Frame Difference
FC	Fuzzy Controller
FL	Fuzzy Logic
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HCF	Highest Confidence First
ICM	Iterative Conditional Modes
INS	Inertial Navigation System
IR	Infrared Sensors
MAP	Maximum a Posteriori Probability
MRF	Markov Random Field
OM	Object Mask
pdf	probability density function
RL	Reinforcement Learning
SA	Simulated Annealing
SDD	Squared Displacement Difference
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
US	Ultrasound Sensors

Mathematical Notations

Image and Sequence

$t \in \mathbb{R}^+$	time, order of a frame in a video sequence
n	dimension of the color space
f^t	sequence frame at instant t
$\tilde{f}$	compensated frame from camera motion
$d^t(f^1, f^2, ...)$	distance function between images f_1, f_2, ... at instant t
$\mathbf{x}_s$	vector of spacial coordinates of site (pixel) s
Bg	Background
Fg	Foreground

GMM modeling

K	Number of components
Σ_k	$(n \times n)$ covariance matrices of component k
$p(f_s^t)$	statistical background model
$\sigma_{k,i}, i \in 1, 2, 3$	$i^t h$ channel covariance of component k
μ_k	mean of component k
ω_k	mixing weight of component k
$\mathcal{N}(f_s^t, \mu_k, \omega_k)$	normal distribution of component k
$\hat{k}$	a matched component to given data
α, α', $0 \leq \alpha < \alpha' \leq 1$	learning rates for mixing weights
ρ, ρ'	learning rates for component parameters
Th	weight sum threshold

MRF modeling

S	set of sites, pixel indices
N_s	neighbors set for pixel s
L^t	segmentation label field at time t
$l^t = \{l_s^t, s \in S\}$	realizations of segmentation label field at time t
$\mathcal{G}$	random field of observations
$\mathfrak{g}^t = \}_s^t, s \in S\}$	realization of field of observations
$\mathcal{U}^t$	motion field at time t
$\mathbf{u}^t = \{\mathbf{u}_s^t, s \in S\}$	realizations of motion field at time t
$U(x)$	Gibbs energy function
Vc	clique potential
ε_s	displaced frame difference at site s
$\varepsilon'_{s,r}$	squared displacement difference between neighboring site s and r
$\delta(x)$	Kronecker function
λ, β, γ	constant parameters

SLAM modeling

$\mathbf{y}_R^t$	Vehicle state vector
$\mathbf{x}_i^t$	feature state vector
$\mathbf{x}^t$	System state vector ($\mathbf{x}^t = [\mathbf{y}_R^t, \mathbf{x}_i^t]^T$)
$\mathbf{P}^t$	Error covariance matrix
$\mathcal{M} = (\mathbf{x}^t, \mathbf{P}^t)$	Stochastic map
$\mathbf{f}$	transition function (dynamic model)
$\mathbf{z}^t$	observation vector
$\mathbf{h}$	observation model
$\mathbb{F}$	Fundamental matrix
$\mathbf{E}$	Essential matrix
$\mathbf{K}$	Calibration matrix
$\mathcal{P}, \mathcal{P}'$	projection matrices
SVD	Singular value decomposition

Abstract

In this Thesis we present a navigation solution for a mobile robot. The proposed system uses vision input to enable the robot to build a feature based map of its environment, localize efficiently itself without any artificial markers or other modification, detect and track the moving objects, and navigate without colliding with obstacles.

The Simultaneous Localization and Mapping (SLAM) process is tackled as a stochastic problem using Extended Kalman Filter (EKF). Our contribution consists in building a global map of the environment based on several local maps. The SIFT features are used in our implementation and their descriptors are used as a matching constraint. The 3D initialization of the features is based on the visual geometry theory.

To avoid using outlier features, a motion segmentation and estimation (MSE) process is used to detect the moving part of the scene. Subsequently, during map building, the detected features on the moving parts are excluded. The MSE process consists in camera motion estimation and compensation, scene cut or strong camera motion detection, background Gaussian Mixture Model (GMM) model update, and a Maximum a Posteriori Probability Markov Random Field (MAP-MRF) framework to detect the moving objects in the scene and estimate their motion. Two methods for camera motion estimation and compensation are used, one uses the 2D projection of the 3D motion estimated in the SLAM process and the other method uses the dense motion analysis proposed by Dufaux and Konrad.

We considered also the case, where the robot is equipped with other localization sensors such as inertial navigation system (INS), wheel encoders, and a global positioning system (GPS). Two solutions are considered. The first consists in integrating the data from the INS and encoders data in the dynamic model of the vehicle to estimate it's motion in the SLAM process and using the GPS data for geo-localizing the robot and the built map. The second solution consists in using the data from the INS and encoders in a Kalman filter to correct the GPS data, the estimated linear and angular velocities by this filter are used as prediction in the SLAM filter and the output of this one are used to update the dynamics estimation in the integration filter. This solution will increase the accuracy and the robustness of the positioning during the outage of the GPS system and allows a SLAM in less featured environments.

The estimated map is used in a path planning process to generate a list of waypoints allowing the robot to reach the user defined goals. The robot uses then a navigation process to follow the planned path and avoiding the unplanned obstacles or moving objects in the scene. For this procedure, added to vision, infrared and ultrasound sensors are used for obstacles detection.
The navigation procedure is based on two fuzzy logic controllers: a goal seeking controller and an obstacle avoidance controller. The goal seeking controller tries to find the path to the intermediate waypoints, while the obstacle avoidance controller has for mission to avoid obstacles. The 3D position of the moving obstacles is estimated using the epipolar geometry applied to the detected features on the moving objects. A command fusion scheme based on a conditioned activation for each controller arbitrates between the two behaviors. The reinforcement learning algorithm is used to adapt the obstacle avoidance fuzzy controller.

Résumé

Cette thèse introduit une approche pour la navigation de robot mobile en utilisant la vision par mono camera pour la construction d'une carte de l'environnement, la localisation du robot dans la carte construite, et la navigation sécurisée du robot dans son environnement.
La localisation et la cartographie de l'environnement sont traitées simultanément par le processus SLAM sous forme d'un problème stochastique en utilisant le filtre de Kalman étendu. Notre contribution consiste dans la construction d'une carte globale de l'environnement base sur plusieurs cartes locales pour simplifier la complexité du problème. Les points caractéristiques utilisés pour modéliser l'environnement sont de type SIFT et leur profondeur est estimée par la théorie de la géométrie visuelle.
Pour éviter d'utiliser les points caractéristiques associés aux objets mobiles dans le modèle de l'environnement nous utilisons un processus de segmentation et d'estimation de mouvement dans les squences vidéo. Ce processus estime le mouvement de la camera pour l'éliminer, détecte les coupures de scène ou grand mouvement dans la séquence vidéo, met jour le modèle GMM représentant le fond de la scène, et détecte les objets en mouvement et estime leur mouvement par un estimateur du maximum posteriori du Champs aléatoires de Markov. Le mouvement de la camera est estimé par deux approches: la première par la projection du mouvement 3D estimé par le processus du SLAM et la deuxième par la technique de l'analyse du mouvement dans le plan de l'image proposée par Dufaux et Konrad.
Nous avons considéré aussi le cas o le robot est équipé avec d'autres capteurs de localisation tels que les systèmes inertiels INS, les encodeurs des roues, et les systèmes de positionnement global GPS. Deux solutions sont proposées. La première consiste intégrer les données de l'INS et des encodeurs dans le modèle dynamique du véhicule pour estimer son mouvement dans le processus du SLAM et utiliser les données du GPS pour géo-localiser le robot et la carte construite. Tandis que dans la deuxième approche les données de l'INS et des encodeurs sont utilisées dans un filtre de Kalman pour corriger les données du GPS et les vitesses linéaires et angulaires estimées par le filtre sont utilisées comme prédiction au filtre du SLAM. Cette dernière solution augmente la précision et la robustesse du positionnement durant l'absence du signal GPS et permet un SLAM dans des espaces moins texturés.

La carte construite de l'environnement est utilisée pour une planification globale des trajectoires libre. Ces derniers sont définis sous formes d'une liste de points but. Le robot utilise ensuite un processus de navigation pour suivre les chemins planifiés et éviter les obstacles non représenté dans la carte et les objets en mouvement. Pour cette tche de navigation le robot utilise en plus de la vision par camera, des capteurs ultrason et infrarouge. Le processus de navigation est basé sur les contrleurs en logique floue adaptés par l'apprentissage par renforcement.

Introduction

Autonomous navigation of robots has been of research interests since the beginning of robotics. Much progress has been made in the area but building a fully functional autonomous navigation system is still facing technical as well as computational problems. To reach a reasonable degree of autonomy, two basic requirements are needed: sensing and reasoning. Sensing is provided by on board sensory systems that gather information about the robot itself and the surrounding environment. According to the environment state, the reasoning system allows the robot to localize itself in the environment and to seek for free paths to reach its goal.

To accomplish these two tasks, the reasoning system requires a model or a description of the environment, which is not always available. In this case the robot should have the means to build such a model over time as it explores its environments. This is known as mapping problem.

Mapping and localization are interlinked problems: without a precise map, the robot cannot localize itself using the map. On the other hand, without precise knowledge of its pose (spatial position and orientation), the robot cannot build a representation of its environment. In combination, however, the problem is much difficult. In this case, starting from an unknown location in an unknown environment, an autonomous mobile robot should be able to incrementally build a map of the environment using only relative observations of the environment and then use this map to localize itself and navigate. Such is referred to as Simultaneous Localization and Mapping (SLAM) approach [1, 2] or Concurrent Mapping and Localization (CML) [6, 7, 8].

Knowing its position in the scene and based on the constructed map, the robot uses a path planner to compute the more safe trajectory to reach its goal [224, 225]. While navigation, a mobile robot should take into account the possible existence of unmapped obstacles or moving objects in the scene [151].

Several works have been realized in this area of mobile robot navigation [151, 58, 17, 20, 71, 79, 87, 88], but in general, these works are specialized and dedicated to solve a specified part of the problem and only few works have tried to solve the global problem as in [89].

In this dissertation we present a navigation solution for a mobile robot in large sized environments. It enables the robot to build a map of its environment, localize efficiently itself without any artificial markers or other modification, detect and track moving objects, and navigate without colliding with obstacles.

The proposed system uses vision input to detect and track moving objects, build a feature based map of the environment and localize the robot. For the obstacle avoidance, apart from vision based moving object detection, the robot uses ultrasound and infrared sensors. The algorithm consists in three processes as illustrated in figure 1.

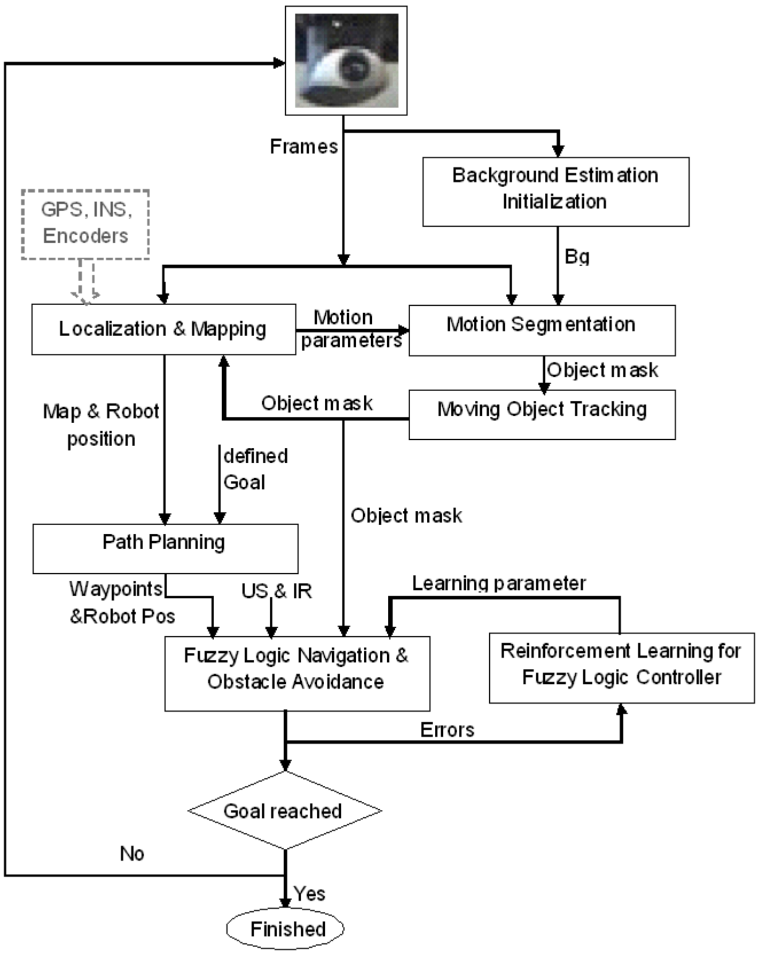


Figure 1: Navigation System for Mobile Robot

Simultaneous Localization and Mapping (SLAM) process

Using this process the robot builds a feature-based map of the environment and uses this map to localize itself. The SLAM problem is tackled as a stochastic problem using Extended Kalman Filter (EKF) with linear Gaussian approximations for the map and vehicle position and uncertainties.

Our contribution consists in building a global map of the environment based on several local maps. Each local map is a feature based 3D representation of the environment. The grouping of the local maps is realized in two manners: by transforming the coordinates of the local map features in a global coordinate frame system or by saving only the robot positions where it started to build the local maps. From the usage point view, SIFT features are used for the 3D reconstruction and their descriptors are used as a matching constraint. For the initialization step we propose an approach based on multiview geometry.

For a robust matching we use a product of three parameters: the Mahalanobis distance between the measurements and their predictions, the Euclidean distance between the descriptor vectors of the features, and the distance of the feature to the induced epipolar line (epipolar constraint). These constraints allow combining the system model, the multiview geometry and the scale-space invariance parameters.

We considered also the case, where the robot is equipped with other localization sensors such as inertial navigation system (INS), wheel encoders, and a global positioning system (GPS). Each of these sensors can be used separately by the robot for its localization but it's subject to a lot of error sources affecting the accuracy of the obtained robot location. In this work we proposed two solutions. The first consists in integrating the data from the INS and encoders data in the dynamic model of the vehicle to estimate it's motion in the SLAM process and using the GPS data for geo-localizing the robot and the built map. The second solution consists in using the data from the INS and encoders in a Kalman filter to correct the GPS data, the estimated linear and angular velocities by this filter are used as prediction in the SLAM filter and the output of this one are used to update the dynamics estimation in the integration filter. This solution will increase the accuracy and the robustness of the positioning during the outage of the GPS system and allows a SLAM in less featured environments.

Motion segmentation and estimation process

To avoid using outlier features in the built map, a motion segmentation and estimation (MSE) process is used to detect the moving part of the scene and estimate their motion. Subsequently, during map building, the detected features on the moving parts are excluded. The MSE process combines a Gaussian Mixture Model (GMM) background subtraction approach and a Maximum a Posteriori Probability Markov Random Field (MAP-MRF) framework. This enables us to exploit the simplicity and capability of the GMM approach to adapt to illumination changes and small motions in the scene and the advantages of spatio-temporal dependencies that moving objects impose on pixels and the interdependence of motion and segmentation fields.

This process compensates the camera motion based on the 3D camera motion estimated by the SLAM process. The compensated frame is used, to update the GMM model of the background. The means and covariances of the Gaussians are learned from color observations in consecutive images.

The MAP-MRF framework is used to optimize the segmentation and estimate the motion filed of the moving objects. A feedback of the MRF optimization results to update the background model enables the acceleration of the learning of the new stationary regions and therefore avoids the fragmentation problem in the detected mask of moving objects.

To be used in the SLAM process, to avoid the outlier features, a tracking function is used to predict the position of the detected moving objects based on their motion.

Path planning and navigation

The obtained feature based map is exploited by a global path planning algorithm to generate a list of waypoint allowing the robot to reach the user defined goals. The robot uses a navigation process to follow the planned path and avoiding the unplanned obstacles or moving objects in the scene. For this procedure, added to vision, infrared and ultrasound sensors are used for obstacles detection. The navigation procedure is based on two fuzzy logic controllers: a goal seeking controller and an obstacle avoidance controller. The goal seeking controller tries to control the robot towards the intermediate waypoints, while the obstacle avoidance controller has for mission to avoid obstacles. The 3D position of the moving obstacles is estimated using the epipolar geometry applied to the detected features on the moving objects. A command fusion scheme based on a conditioned activation for each controller arbitrates between the two behaviors. The reinforcement learning algorithm is used to adapt the obstacle avoidance fuzzy controller.

The remainder of this dissertation is organized as follows. In chapter 1, a state of the art on the SLAM problem is presented and then the proposed algorithm is detailed. Chapter 2 discusses the motion segmentation and estimation problem and the proposed approach. Chapter 3 describes the navigation process. The implementations and the results of the different part of the algorithm are presented in the corresponding chapters. We conclude with a general conclusion.

Chapter 1

Simultaneous Localization And Mapping

1.1 Introduction

The recent research in robotics aims at developing autonomous mobile robots which can navigate independently in often very complex real environments. These robots are equipped with (i) a perception system to collect information about their environments, (ii) a reasoning system to treat the collected information and to decide based on it, and (iii) a locomotive system to reach its goal.

The reasoning system is the central unit in an autonomous robot. According to the environment state, it must allow the robot to localize itself in the environment and to seek for free paths. To accomplish these two tasks, it bases it reasoning on a model or a description of the environment. The environment model is not always available and hence the robot should have the means to build such a model over time. This latest problem is known as mapping problem.

If the robot's pose (spatial position and orientation) is known all along, building a map would be quite simple. In the opposite, if a map of the environment exists already, it will be very easy to determine accurately the robot's pose at any time. Solving the mapping problem in conjunction with the localization problem, however, is much harder. The literature refers to this problem as Simultaneous Localization and Mapping (SLAM) [1, 2, 3], Simultaneous Localization and Map Building (SLAMB) [4, 5], or Concurrent Mapping and Localization (CML) [6, 7, 8].

The SLAM problem is tackled as a stochastic problem and it has been addressed with approaches based on Bayesian filtering. The main problem of those approaches is that the computational complexity growth with the size of the mapped space, which limits their applicability in large-scale areas. In the case of vision based SLAM approaches, other challenges have to be tackled, as the high rate of the input data, the inherent 3D quality of visual data, the lack of direct depth measurement and the difficulty in extracting long-term features to map.

Due to those factors, there have been relatively few successful vision-only SLAM systems which are able to construct persistent and consistent map while closing loops to correct drift.

In this work, we focus on monocular vision-based SLAM where a single camera is moving through a large-scale environments. This case is interesting because it potentially offers a low-cost approach to SLAM in unknown environments. The well known approach in this area, is the MonoSLAM algorithm of Davison et al. [9]. This is a real-time SLAM approach for indoors in room-size domains, which recover the 3D trajectory of a monocular camera, moving rapidly through an unknown scene. Davison's algorithm is not suitable in larger environments.

To be able to use SLAM algorithms in large areas, in this dissertation we propose to build several size limited local maps and combine them into a global map using an 'history memory' which accumulates sensory evidence over time to identify places with a stochastic model of the correlation between map features. In our implementation, the dynamic model of the camera takes into account that the camera is on the top of a mobile robot which moves on a perfect ground-plane at all times and the SIFT feature detector is used instead of Shi and Thomasi algorithm as in [9]. The SIFT features are proved to remain stable to affine distortions, change of viewpoint, noise and change in illumination. Using SIFT features allows also a more reliable feature matching by using the advantage of the scale-space invariance parameters of the SIFT features.

An other problem for SLAM methods is change over time of the environment [10, 11, 12]. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time. Others are faster, such as the change of door status or the location of furniture items, such as chairs. Even faster may be the change of location of other agents in the environment, such as cars or people. Unfortunately, there are almost no mapping algorithms that can learn meaningful maps of dynamic environments. Our study is limited to the case of dynamic environments with moving objects. To deal with this problem, we use the proposed algorithm for motion segmentation in chapter 2 to remove the outliers features which are associated with moving objects.

A successful SLAM algorithm should allow a robot navigating in an environment and returning to the starting point without losing its way. This situation becomes an even more difficult problem if the return route is different from the departure one. This is referred to as *closing the loop*. Without an absolute positioning sensor, the robot cannot be certain about its location relative to the start as it explores unknown areas. Consequently, to recognize when it has returned to the start, the robot should identify some of the perviously seen landmarks. In this sequel, we propose also some solutions to this problem.

The chapter is organized as follows, section 1.3 gives an overview of the localization and mapping problems for mobile robot navigation. Section 1.3.3 introduces the monoSLAM approach [9] and discuss its limitations. Then, section 1.5 describes the proposed approach and section 1.6 discusses the implementation and the experimental results.

At the end of this chapter, we consider the case of a mobile robot for outdoor applications equipped, added to the vision camera, with other sensors for localization as a Global Positioning System (GPS), an Inertial Navigation System (INS), and wheels encoders. Two solutions are proposed in this case, one using the encoders and the INS sensors in the system dynamic model for motion estimation in the SLAM process and the GPS data for geo-referencing the results (the robot position and the map). The second approach uses a separate filter to integrate the GPS, INS, and encoders data to correct the geo-localization. The linear and angular velocities in the dynamic model of the EKF-SLAM algorithm are given by the GPS/INS/Encoders integration filter. In the other hand, the output of the SLAM-EKF is used to update the dynamics estimation in the integration filter and therefore the geo-referenced localization.

1.2 Coordinate Systems - Definition and Transformation

In this section we introduce the different coordinate systems which will be used for robot navigation, localization and mapping.

1.2.1 Earth Centred Earth Fixed (ECEF or ECF) coordinates

The Earth-centered Earth-fixed (ECEF or ECF) or conventional terrestrial coordinate system (X^E,Y^E,Z^E) rotates with the Earth and has its origin at the center of the Earth (figure 1.1). The X^E axis is located in the equatorial plane and points towards the mean Meridian of Greenwich. The Z^E axis is parallel to the Earths mean spin axis. The Y^E axis can be determined by the right-hand rule to be passing through the equator at 90° longitude.

1.2.2 Golobal coordinate system

For the navigation and the SLAM we consider a global coordinate system $G(X^G$,Y^G,$Z^G)$ (figure 1.1) formed by a plane tangent to the Earth's surface, which is fixed to a specific location with known geodetic coordinates (it can be the initial robot position or any other location). The X^G axis points towards the east, the Y^G axis points towards the North and the Z^G axis points vertically upwards.

1.2.3 Transformation from Geodetic coordinates to ECEF coordinates

In geodetic coordinates the Earth's surface is approximated by an ellipsoid and locations near the surface are described in terms of latitude Φ, longitude Γ, and altitude Ψ. Geodetic coordinates can be converted into ECEF coordinates

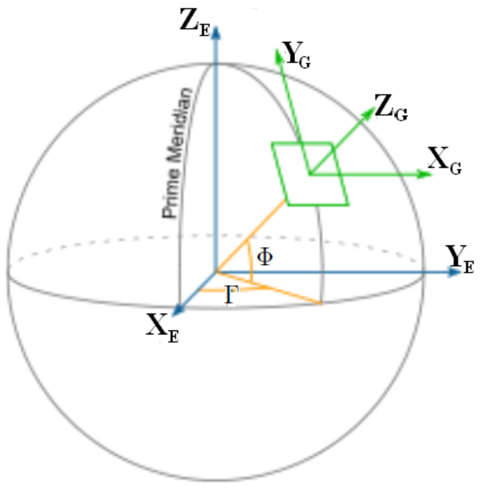


Figure 1.1: The ECEF and the global coordinate systems

(x^E, y^E, z^E) using the following formulas [13]:

$$\begin{aligned} x^E &= (\chi + \Psi)\cos(\Phi)\cos(\Gamma) \\ y^E &= (\chi + \Psi)\cos(\Phi)\sin(\Gamma) \\ z^E &= (\chi(1-e^2) + \Psi)\sin(\Phi) \end{aligned} \tag{1.1}$$

where $\chi = a/\sqrt{(}1 - e^2 sin^2(\Phi))$ is the distance from the surface to the Z^E-axis along the ellipsoid normal. a and e^2 are the semi-major axis and the square of the first numerical eccentricity of the ellipsoid respectively ($a = 6356752.3142m$ and $e^2 = 6.69437999014 \times 10^{-3}$).

1.2.4 Transformation from ECEF coordinates to G coordinates

Let's consider that the coordinate system G is at a location O with geodetic coordinates (Φ_O,Γ_O,Ψ_O) and ECEF coordinates (x_O^E, y_O^E, z_O^E). The location (x^E, y^E, z^E) in the ECEF coordinate system is converted into the G coordinate (x^G, y^G, z^G) using the following equations [13]:

$$\begin{bmatrix} x^G \\ y^G \\ z^G \end{bmatrix} = \begin{bmatrix} -\sin(\Gamma_O) & \cos(\Gamma_O) & 0 \\ -\sin(\Phi_O)\cos(\Gamma_O) & -\sin(\Phi_O)\sin(\Gamma_O) & \cos(\Phi_O) \\ \cos(\Phi_O)\cos(\Gamma_O) & \cos(\Phi_O)\sin(\Gamma_O) & \sin(\Phi_O) \end{bmatrix} \begin{bmatrix} x^E - x_O^E \\ y^E - y_O^E \\ z^E - z_O^E \end{bmatrix} \tag{1.2}$$

1.2.5 Transformation from G coordinates to ECEF coordinates

This is just the inverse of the transformation from ECEF to G systems.

$$\begin{bmatrix} x^E \\ y^E \\ z^E \end{bmatrix} = \begin{bmatrix} -\sin(\Gamma_O) & -\sin(\Phi_O)\cos(\Gamma_O) & \cos(\Phi_O)\cos(\Gamma_O) \\ \cos(\Gamma_O) & -\sin(\Phi_O)\sin(\Gamma_O) & \cos(\Phi_O)\sin(\Gamma_O) \\ 0 & \cos(\Phi_O) & \sin(\Phi_O) \end{bmatrix} \begin{bmatrix} x^G \\ y^G \\ z^G \end{bmatrix} + \begin{bmatrix} x^E_O \\ y^E_O \\ z^E_O \end{bmatrix} \tag{1.3}$$

1.2.6 Robot, INS, Camera and Image coordinate systems

We also define (figure 1.2):

• The robot coordinate system R (X^R,Y^R,Z^R) located at the robot center of gravity, with XY plan parallel to the ground and the X axis points towards the robot's direction. The Z axis points vertically upwards.

• The INS (Inertial Navigation System) coordinate system L with its axes parallel to the robot frame axes and it center at $(x^R_\ell, y^R_\ell, z^R_\ell)$ in the robot frame R.

• The camera coordinate system C with origin at (x^R_c, y^R_c, z^R_c) in the robot frame R and axes parallel to the robot frame axes.

• The image plane coordinate system I with origin at the bottom left corner of the image. The X and Y axes are parallel to the Y and Z axes of the camera, respectively.

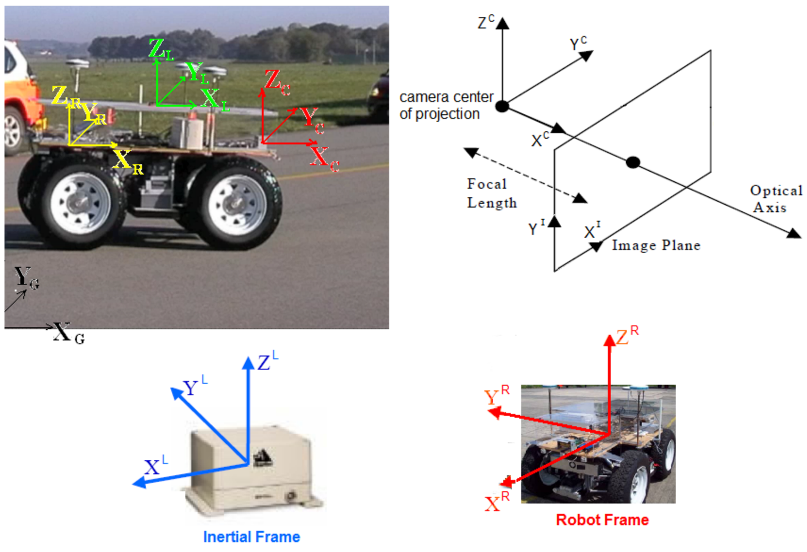


Figure 1.2: The Robot, Camera, and Inertial coordinate systems

The coordinates (x^R, y^R, z^R) of a point int the R frame are transformed to G

coordinates (x^G, y^G, z^G) by:

$$\begin{bmatrix} x^G \\ y^G \\ z^G \end{bmatrix} = \begin{bmatrix} \cos(\gamma_r) & -\sin(\gamma_r) & 0 \\ \sin(\gamma_r) & \cos(\gamma_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix} + \begin{bmatrix} x_r^G \\ y_r^G \\ h \end{bmatrix} \tag{1.4}$$

where the (x_r^G, y_r^G, z_r^G) are the robot coordinates in the G frame systems. In our application z_r^G is always null. h is the hight of the robot coordinate system. γ_r is the robot orientation (yaw).

The transformation from C and L to R are given by:

$$\begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix} = \begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} + \begin{bmatrix} x_c^R \\ y_c^R \\ z_c^R \end{bmatrix} \tag{1.5}$$

and

$$\begin{bmatrix} x^R \\ y^R \\ z^R \end{bmatrix} = \begin{bmatrix} x^L \\ y^L \\ z^L \end{bmatrix} + \begin{bmatrix} x_\ell^R \\ y_\ell^R \\ z_\ell^R \end{bmatrix} \tag{1.6}$$

In the following all computations will be made in the global coordinate system G and hence we will omit the subscript G for notation clarity.

1.3 Simultaneous Localization and Mapping - An overview

1.3.1 Localization and Mapping

Localization for a mobile robot is the problem of finding out where a robot is, based on sensory data. This localization can be global or local. In the case of local localization, called also pose tracking, the initial estimation of the robot pose (position and orientation) is known. This estimate is updated based on the sensory data during the robot navigation.

Using only sensors that measure relative movements, the error in the pose estimate increases over time as errors are accumulated. Therefore, external sensors are needed to provide information about the absolute pose of the robot. This is achieved by matching the sensors measurements with a model of the environment. If a good initial estimate is given, the correspondence or data association problem is easier. It does not consider the entire space when looking for the correspondence, but rather only a relatively small region around the estimated pose [14].

In the other hand, for global pose estimation [15, 16], being the ability of determining the robot's pose in a map using its sensors, data association is more complicate and depends on the complexity of the environment.

Moreover, if the map does not exist, the robot should have the means to build one. Schematically, the problem of map building consists of the following steps:

(i) Sensing the environment at time t using onboard sensors (e.g., laser scanner, vision, or sonar); (ii) Representing of sensor data ; (iii) Integrating the recently perceived observations at time t with the previously 'learned' structure of the environment estimated at time $t-1$.

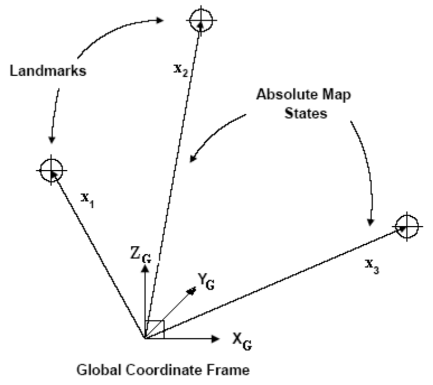
The environment models (maps) used for robot navigation can be grouped into two categories: metric maps or topological map. Metric maps [17, 18, 19] are high resolution geometric representations with an explicit Cartesian reference frame. Building metric maps depends critically on accurate position information. On the other hand topological maps [17, 18, 20, 21] represent the environment as a graph of interconnected places without taking into account their absolute position with respect to a coordinate system. In other words, topological maps can be built and maintained without any estimates for the position of the robot. As a result, they can be used to represent large area maps since all connections between nodes are relative, rather than absolute. Other technics integrate topological and metric representations [17, 18, 22, 23, 24], where the environment is represented globally as a graph of connected regions and each region is represented by a local metric map.

Metric maps can be: (i) feature-based [25] where the environment is represented as a set of geometric primitives such as corners or lines, or (ii) grid-based representations [19, 26, 27, 28] where the environment is represented as a grid of occupied and free spaces. Localization in a metric map can be seen as multiple target tracking problem where the targets are static and the observer in motion [17].

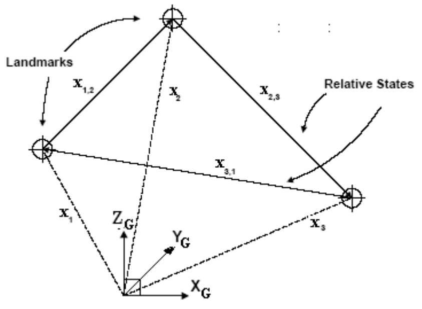
Metric maps can be also classified as absolute or relative. In absolute maps (Figure 1.3(a)), landmarks are represented in a single global coordinate frame. An absolute map $\mathcal{M}_{\text{abs}}$ with N landmarks is denoted in vector form as $\mathcal{M}_{\text{abs}} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^T$. Whereas in relative maps (Figure 1.3(b)), only the relationships between individual landmarks are described. A relative map $\mathcal{M}_{\text{rel}}$ is denoted in vector form as $\mathcal{M}_{\text{rel}} = [..., \mathbf{x}_{i,j}, ...]^T$, where $\mathbf{x}_{i,j}$ is the relative state between landmarks $\mathbf{x}_i$ and $\mathbf{x}_j$. In case of point features the relative states are vectors describing the displacement between point landmarks represented in a common coordinate system:

$$\mathbf{x}_{i,j} = \mathbf{x}_j - \mathbf{x}_i$$

.



(a) A three landmarks absolute map



(b) A three landmark relative map

Figure 1.3: Absolute and relative metric maps

1.3.2 Simultaneous Localization And Mapping

In case of navigation in an unknown environment starting from an unknown location with no a priori knowledge [1, 2, 3, 4, 5, 6, 7, 8], a SLAM system simultaneously computes an estimate of the robot location and the landmarks locations. While continuing its motion, the robot builds a complete map of landmarks and uses these to provide continuous estimates of the vehicle location.

Techniques for solving the SLAM problem have focused in using probabilistic methods taking account of the uncertainty in the measurement. Two main groups of techniques have been considered depending on the way of representing such uncertainty: a) Gaussian filters and b) non-parametric filters, which will be discussed later in this chapter.

1.3.3 Vision based SLAM

SLAM Methods using cameras to observe the environment are referred to as vision based SLAM approaches [29, 30, 14, 31, 32, 33]. In an image, several information associated with objects within the scene, such as width, height, color and texture could be inferred. Using current image processing techniques of edge and corner detection, features can be distinguished and identified in a single image frame. Inherently, 2D images cannot provide the distance information of these features; they do however return a bearing to the feature, relative to the camera. Repeated observations and position estimates of the camera, then allow triangulation of the feature's global position, which can then be used in the SLAM algorithm. This method is known as Bearing-Only SLAM [17, 34, 35, 36, 37].

1.3.4 SLAM Formulation

Consider a mobile robot moving through an unknown static environment. The robot executes controls and collects observations of features in the world. Both the controls and the observations are corrupted by noise. Simultaneous Localization and Mapping (SLAM) process recovers a map of the environment and the path of the robot from a set of noisy controls and observations. As error in the robot's path correlates errors in the map (due to control error, the robot's pose becomes more uncertain as the robot moves and therefore the uncertainty in the estimated positions of newly observed landmarks also increases), the state of the robot and the map must be estimated simultaneously.

The vehicle travels through the environment using its sensors to observe features around it. The state of the system at time t (figure 1.4) can therefore be represented by the augmented state vector, $\mathbf{x}^t$, consisting of the n_r dimensional vector representing $\mathbf{x}_r^t$, and the N observed landmarks, $\mathbf{x}_i^t$, $i = 1, ..., N$, relative to the global frame.

$$\mathbf{x}^t = \begin{bmatrix} \mathbf{x}_r^t \\ \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_N^t \end{bmatrix} \tag{1.7}$$

The vehicle state at time t, $\mathbf{x}_r^t$, may contain any number of properties of the vehicle, including location, velocity and acceleration for example. This information can be generated using odometers attached to the wheels of the robot, inertial navigation units, or simply by observing the control commands executed by the robot. Regardless of origin, any measurement of the robot's motion will be referred to generically as a control. The control at time t will be written ξ^t. The observations $\mathbf{z}^t$ of the robot at time t are the detected landmarks.

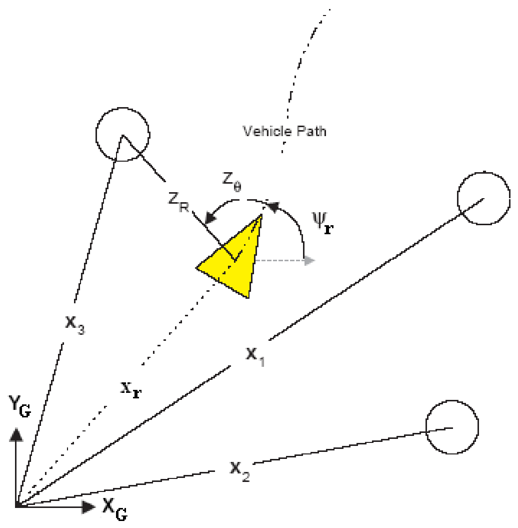


Figure 1.4: System Model in a SLAM algorithm

Using the above defined notation, the primary goal of SLAM is to recover the best estimate of the system state (the robot pose and the landmarks positions), given the set of noisy observations $\mathbf{z}^t$ and controls ξ^t.

In probabilistic terms, this is expressed by the following posterior, referred to in the sequel as the SLAM posterior:

$$p(\mathbf{x}^t \mid \mathbf{z}^t, \xi^t) \tag{1.8}$$

Dynamic Model

The dynamic model or motion model is the relationship between the robot's paste state, $\mathbf{x}_r^{t-1}$, and its current state, $\mathbf{x}_r^t$, given a control input ξ^t

$$\mathbf{x}_r^t = \mathbf{f}\left(\mathbf{x}_r^{t-1}, \xi^t, \nu^t\right) \tag{1.9}$$

Where the transition function $\mathbf{f}$ is a function representing the mobility, kinematics and dynamics of the robot and ν is a random vector describing the unmodelled aspects of the vehicle (process noise such as wheel sleep or odometry error).

The dynamic model can also be represented as a probabilistic function:

$$p(\mathbf{x}_r^t \mid \mathbf{x}_r^{t-1}, \xi^t) \tag{1.10}$$

Observation Model

The observation model describes the physics and the error model of the robot's sensor. The observations are related to the system state according to:

$$\mathbf{z}^t = \mathbf{h}\left(\mathbf{x}^t\right) + \mathbf{w}^t \tag{1.11}$$

Where $\mathbf{z}^t$ is the observation vector at time t and $\mathbf{h}$ is the observation model. The vector $\mathbf{z}_i^t$ is an observation at instant t of the i'th landmark location $\mathbf{x}_i^t$ relative to the robot's location $\mathbf{x}_r^t$. This type of observation will be referred to as a vehicle-landmark observation.

$\mathbf{w}$ is a random vector describing both measurement noise and uncertainties in the measurement model itself (observation noise such as sensor inaccuracy).

In probabilistic notation, the observation model can be represented by:

$$p(\mathbf{z}^t \mid \mathbf{x}^t) \tag{1.12}$$

Given a model for the motion and observation, the SLAM process consists of generating the best estimate for the system states given the information available to the system. This can be accomplished using a recursive, three stage procedure comprising prediction, observation and update of the posterior. This recursive update rule, known as filtering for SLAM, is the basis for the majority of SLAM algorithms.

For a robust estimation of the vehicle and landmarks states, the SLAM problem has been addressed with approaches based on Bayesian filtering, with a prediction and an update steps. A SLAM solution method can be summarized with the algorithm given in figure 1.5. In the prediction step, the system state is predicted from the previous state based on the motion model of the robot

$$p(\mathbf{x}^{t|t-1} \mid \xi^t, \mathbf{x}^{t-1|t-1}) \tag{1.13}$$

The estimated state $\mathbf{x}^{t|t-1}$ is used to predict the measurement $\mathbf{z}^{t|t-1} = \mathbf{h}(\mathbf{x}^{t|t-1})$ which will be matched with current measurement $\mathbf{z}^t$. If a matching is found for a given feature (this means that the feature has been re-observed), the process state is updated based on measurements of the map made at time step t (this is the update step).

$$p(\mathbf{x}^{t|t} \mid \mathbf{z}^t, \xi^t) \tag{1.14}$$

If no matching is found for the current measurements, the algorithm check if these measurements correspond to candidate features and augment the system state with the state of the new features.

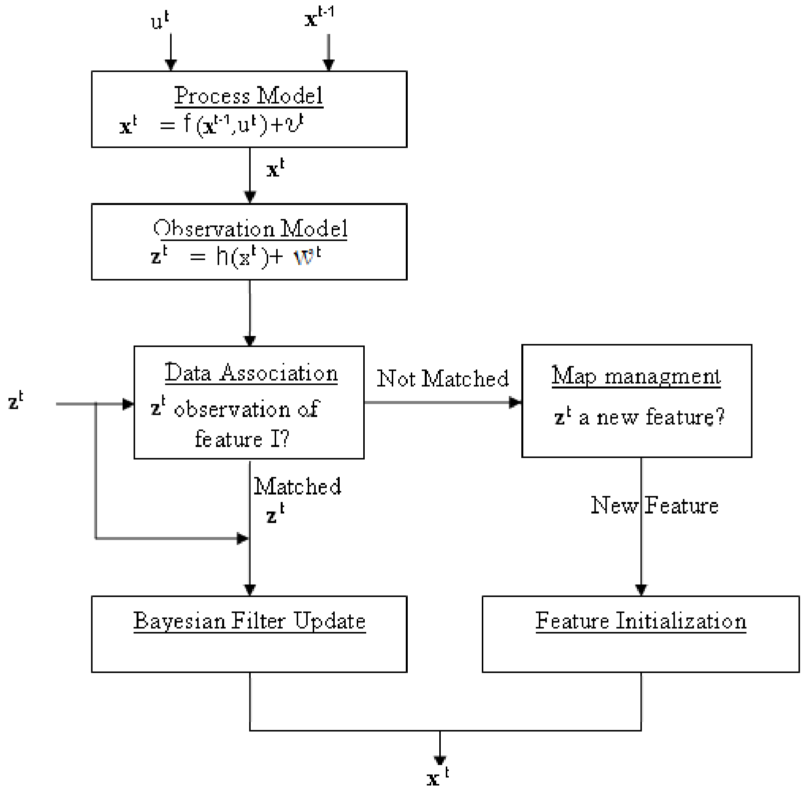


Figure 1.5: The general steps of a SLAM algorithm

SLAM filtering implies that a problem involving N landmark will require the estimation of a mean state vector of size M ($M = N * n_L + n_r$)) and an associated covariance matrix of size M^2 [38], with n_L and n_r are the size of the feature and the robot states respectively. Also each vehicle-observation requires M^2 update time and a storage size of M^2 to maintain the joint covariance matrix. So the overall computational costs grow quadratically with the number of landmarks.

1.3.4.1 Gaussian Filters

Extended Kalman Filter (EKF) is the most well-known Gaussian filter for treating the SLAM problem, where the belief is represented by a Gaussian distribution. The main goal of EKF (see annex C for more details on Kalman Filtering) is the estimation of the current state of a dynamic system by using data provided by the sensor measurements. Extended Kalman Filter is a recursive system, that only uses the information of the previous step and the actual measurements in order to estimate the current state and update the system. Whenever a landmark is observed by the on-board sensors of the robot, the system determines whether it has been already registered and updates the filter. In addition, when a part of the scene is revisited, all the gathered information from past observations is used by the system to reduce uncertainty in the whole mapping. This strategy is known as closing-the-loop.

In EKF-based SLAM approaches [39, 40, 9], the environment is represented by a stochastic map $\mathcal{M} = (\hat{\mathbf{x}}, \mathbf{P})$, where $\mathbf{x}$ is the estimated state vector containing the vehicle state $\mathbf{x}_r$ and the environment features state $\mathbf{x}_i$, $i = 1 \cdots N$, and $\mathbf{P}$ (equation 1.15) is the error covariance matrix, where all the correlations between the elements of the state vector are defined. $\mathcal{M}$ is built incrementally, using the set of measurements $\mathbf{z}^t$ obtained by sensors such as cameras or lasers.

$$\mathbf{P}^t = E[(\mathbf{x}^t - \hat{\mathbf{x}}^t)(\mathbf{x}^t - \hat{\mathbf{x}}^t)^T] = \begin{bmatrix} \mathbf{P}^t_{rr} & \mathbf{P}^t_{r1} & \cdots & \mathbf{P}^t_{rN} \\ \mathbf{P}^t_{1r} & \mathbf{P}^t_{11} & \cdots & \mathbf{P}^t_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^t_{Nr} & \mathbf{P}^t_{N1} & \cdots & \mathbf{P}^t_{NN} \end{bmatrix} \tag{1.15}$$

The sub-matrices, $\mathbf{P}_{rr}$, $\mathbf{P}_{ri}$ and $\mathbf{P}_{ii}$ are, respectively, the robot to robot, robot to feature and feature to feature covariances. The sub-matrices $\mathbf{P}_{ij}$ are the feature to feature cross-correlations. $\mathbf{x}$ and $\mathbf{P}$ will change in dimension as features are added or deleted from the map.

The Extended Kalman Filter performs the two previously described steps: a) the prediction step, which estimates the current state based on the previous states and the control ξ; and b) the update step, which uses the current information provided by robot on-board sensors to refine prediction.

Prediction

At each time step of the filter we obtain the predicted state $\mathbf{x}$ and covariance $\mathbf{P}$ using the state transition function.

$$\mathbf{x}^{t|t-1} = \begin{bmatrix} \mathbf{f}\left(\mathbf{x}_r^{t-1|t-1}, \xi\right) \\ \mathbf{x}_1^{t-1|t-1} \\ \vdots \end{bmatrix} \tag{1.16}$$

$$\mathbf{P}^{t|t-1} = \mathbf{F}\mathbf{P}^{t-1|t-1}\mathbf{F}^T + \mathbf{Q}^{t-1} \tag{1.17}$$

where

$$\mathbf{F} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^{t-1|t-1}} = diag\left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_r}\right|_{\mathbf{x}_r^{t-1|t-1}}, \mathrm{I}\right)$$

is the Jacobian of $\mathbf{f}$ with respect to the state vector $\mathbf{x}$ and $\mathbf{Q}$ is the process noise covariance.

Update

The update to include a new measurement incorporates the innovation ε, which is the difference between the observation and its prediction, and the covariance $\mathbf{S}$. We also inject the measurement noise via covariance $\mathbf{R}$.

$$\mathbf{x}^{t|t} = \mathbf{x}^{t|t-1} + \mathbf{W}^t \varepsilon^t \tag{1.18}$$

$$\mathbf{P}^{t|t} = \mathbf{P}^{t|t-1} - \mathbf{W}^t \mathbf{S}^t {\mathbf{W}^t}^T \tag{1.19}$$

Where

$$\mathbf{W}^t = \mathbf{P}^{t|t-1} \mathbf{H}^T (\mathbf{S}^t)^{-1} \tag{1.20}$$

$$\mathbf{S}^t = \mathbf{H}\mathbf{P}^{t|t-1}\mathbf{H} + \mathbf{R}^t \tag{1.21}$$

$$\varepsilon = \mathbf{z}^t - \mathbf{h}(\mathbf{x}^{t|t-1}) \tag{1.22}$$

$\mathbf{Q}$ and $\mathbf{R}$ are block-diagonal matrices defining the error covariance matrices characterizing the noise in the model and the observations, respectively.

$\mathbf{W}$ is the Kalman Gain, $\mathbf{S}$ is the innovation covariance and $\mathbf{H}$ is the Jacobian of the measurement model $\mathbf{h}$ with respect to the state vector:

$$\mathbf{H} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}^{t|t-1}}$$

A measurement of feature $\mathbf{x}_i$ is not related to the measurement of any other feature so

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}} = [\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_r} \quad 0 \cdots \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i} \quad 0 \quad \cdots] \tag{1.23}$$

where $\mathbf{h}_i$ is the measurement model for the i'th feature and then

$$\mathbf{S}_i = \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_r}\mathbf{P}_{rr}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_r}^T + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i}\mathbf{P}_{ir}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_r}^T + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_r}\mathbf{P}_{ri}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i}^T + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i}\mathbf{P}_{ii}\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_i}^T + \mathbf{R} \tag{1.24}$$

which depends only on the measurements of the feature i and the vehicle state.

The best way to view the update step is as a correction to the prediction, based on the measurement and weighted by $\mathbf{W}$ to favor the prediction or the measurement depending on the relative certainty of each (encoded by $\mathbf{S}$). In the prediction the uncertainty can only ever increase, because of the addition of process noise, whereas the update step allows for a decrease since the measurements are improving the knowledge of the state.

Initialization

In the creation of a new stochastic map at step 0, a base reference must be selected. It is a common practice to build a map relative to a fixed base reference different from the initial vehicle location. This normally requires the assignment of an initial level of uncertainty to the estimated vehicle location. In the theoretical linear case, the vehicle uncertainty should always remain above this initial level. In practice, due to linearizations, when a nonzero initial uncertainty is used, the estimated vehicle uncertainty rapidly drops below its initial value, making the estimation inconsistent after very few EKF update steps [38].

A good alternative is to use, as base reference, the initial vehicle location, and thus we initialize the map with perfect knowledge of the vehicle location.

Given a set of i measurements $\mathbf{Z}^i = [\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^i]$ we wish to initialize a new entity L_i (feature) into a stochastic map modelled as a mean $x_i^{t|t}$ and covariance matrix $P_i^{t|t}$.

Initialization consists of stacking the new landmark position $\mathbf{x}_i$ into the map as

$$\mathbf{x}_+^{t|t} = \begin{bmatrix} \mathbf{x}^{t|t} \\ \mathbf{x}_i^{t|t} \end{bmatrix} \tag{1.25}$$

and defining the *pdf* of this new state conditioned on measurement $\mathbf{z}_i$.

The augmented covariance matrix is specified by:

$$\mathbf{P}_+^{t|t} = \begin{bmatrix} \mathbf{P}^{t|t} & \mathbf{P}_{\mathbf{x}i}^{t|t} \\ \mathbf{P}_{i\mathbf{x}}^{t|t} & \mathbf{P}_i^{t|t} \end{bmatrix} \tag{1.26}$$

Figure 1.6 gives a graphic description to the augmentation of the state vector and the covariance matrix with a new feature measurement.

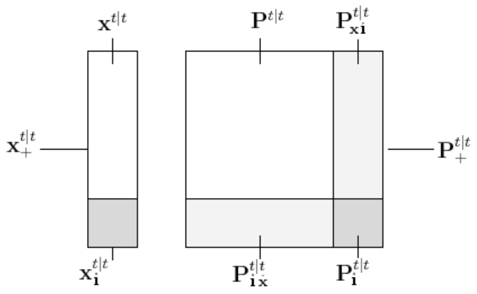

$\mathbf{x}^{t|t}$
$\mathbf{P}^{t|t}$
$\mathbf{P}_{\mathbf{xi}}^{t|t}$
$\mathbf{x}_{+}^{t|t}$
$\mathbf{P}_{+}^{t|t}$
$\mathbf{x}_{\mathbf{i}}^{t|t}$
$\mathbf{P}_{\mathbf{ix}}^{t|t}$
$\mathbf{P}_{\mathbf{i}}^{t|t}$

Figure 1.6: Representation of the EKF-SLAM state vector and covariance matrix augmenting

1.3.4.2 Non-Parametric Filters

Simultaneous Localization and Mapping (SLAM) problem has also been tackled by using non-parametric filters such as the histogram filter or the particle filter (PF). The main difference compared to Gaussian filters is the possibility of dealing with multimodal data distribution, using multiple values (particles) to represent the belief [42].

Particle filter SLAM also known as FastSLAM decouples map of features from pose. FastSLAM exploits the fact that if the robot pose was known all of the features would be uncorrelated and processes all of the feature measurements independently [42].

$$\begin{aligned} p(\mathbf{x}_r^{0:t}, \mathbf{x}_1^t, ..., \mathbf{x}_N^t | \mathbf{z}^t, \xi^t) &= p(\mathbf{x}_r^{0:t} | \mathbf{z}^t, \xi^t) p(\mathbf{x}_1^t, ..., \mathbf{x}_N^t | \mathbf{z}^t, \xi^t) \\ &= p(\mathbf{x}_r^{0:t} | \mathbf{z}^t, \xi^t) \prod_{i=1:N} p(\mathbf{x}_i^t | \mathbf{x}_r^t, \mathbf{z}^t, \xi^t) \end{aligned}$$

where $\mathbf{x}_r^{0:t}$ is the set of robot positions from time 0 to t.

The SLAM problem is then decomposed into a robot path estimation problem and a collection of feature estimation problems that are conditioned on the robot path estimation. The robot path posterior $p(\mathbf{x}_r^{0:t}|\mathbf{z}^t, \xi^t)$ is represented by a set of particles, and the distributions $p(\mathbf{x}_i^t|\mathbf{x}_r^t, \mathbf{z}^t, \xi^t)$ are represented by particle sets, where each set is attached to one particular robot particle.

Robot path estimation FastSLAM maintains a set of K particles $\mathbf{x}_r^{t,[k]}, k = 1 : K$ where the superscript $[k]$ refers to the k-th particle in the set, obtained by sampling from the motion model. Each particle represents a guess of the robot's path.

That is, each estate $\mathbf{x}^t$ of the system can be represented by multiple particles, one for each hypothesis.

$$\mathbf{x}^t = \mathbf{x}_{[m]}^1, \mathbf{x}_{[m]}^2, ..., \mathbf{x}_{[m]}^t \tag{1.27}$$

where each particle $\mathbf{x}^t_{[m]}$ represents m different hypotheses of the estimation of the vehicle pose at a time step t, represented as:

$$\mathbf{x}^t_{[m]} \sim p(\mathbf{x}^t|\mathbf{z}^t, \mathbf{x}^{t-1}_{[m]}) \tag{1.28}$$

where $\mathbf{x}^t$ and $\mathbf{z}^t$ represent the estimated state of the vehicle and the measurements at time step t.

In comparison with EKF-based filters, PF present more robustness to periods of considerably uncertainty and sensor noise, due to its multi-modal data distribution. However, Gaussian filters usually have a polynomic computational cost, whereas the computational cost of a non-parametric filter may be exponential. During last years, several interesting approaches based on particle filters have been presented as an alternative to EKF-based techniques [41, 42], with the aim of solving the SLAM problem. Stachniss [41] proposes the use of a Rao-Blackwellized particle filter for local map representation, combined with some techniques for particle reduction and a "Closing the loop" strategy. The strongest point of this approach is the possibility of dealing with periods of great uncertainty, due to its ability to recover already vanished hypotheses. This represents a considerable improvement with respect to EKF-based approaches, which do not allow to recover hypotheses that have been already vanished in the past even if these hypotheses were correct.

Alternatively, Montemerlo [42] has proposed a new PF-based approach named FastSlam, which combines the use of particles with Kalman filters for map representation. That is, each particle $\mathbf{x}^t_{[m]}$ (composed by all the hypothesis of the robot pose estimation at time state t) has, at the same time, K Kalman filters representing each landmark pose estimation with respect to the vehicle pose.

$$S^t = \mathbf{x}^t_{[m]}, \hat{\mathbf{x}}_{1,[m]}, \mathbf{P}_{1,[m]}, ..., \hat{\mathbf{x}}_{K,[m]}, \mathbf{P}_{K,[m]} \tag{1.29}$$

where $\mathbf{x}^t_{[m]}$ represents all the hypotheses of the robot pose estimation at time t (eq.1.27) and $\hat{\mathbf{x}}_{K,[m]}$ and $\mathbf{P}_{K,[m]}$ represent the estimate state vector (mean) and the covariance matrix of each landmark with respect to each particle. The update process in FastSlam is carried out in the same way as in EKF approaches. In addition, a weight is assigned to each particle depending on its reliability.

This hybrid method has provided reliable solutions to several problems of EKF-based approaches such as the high computational cost that requires to update filters containing considerable amount of data. That is, since the problem is divided into multiple small Kalman Filters containing only Gaussians of two dimensions (for 2D feature location), the computational cost can be reduced to $O(M\log K)$, where M is the number of particles and K the number of landmarks. However, if the complexity of the environment requires the use of 3D data the computational cost increases considerably, forcing the reduction of the number of features at each step, which has a direct effect on the quality of the results.

1.4 SLAM in large-scale areas

The main open problem of the current state of the art SLAM approaches and particularly vision based approaches is mapping large-scale areas [9]. Relevant shortcomings of this problem are, on the one hand, the computational burden, which limits the applicability of the EKF-based SLAM in large-scale real time applications and, on the other hand, the use of linearized solutions which compromises the consistency of the estimation process. Added to this, the challenges posed by vision over laser sensors, which include the very high input data rate, the inherent 3D quality of visual data, the lack of direct depth measurement and the difficulty in extracting long-term features to map [9]. Due to those factors, there have been relatively few successful vision-only SLAM systems which are able to construct persistent and consistent map while closing loops to correct drift.

The computational complexity of the EKF stems from the fact that the covariance matrix $\mathbf{P}^t$ represents every pairwise correlation between the state variables. Incorporating an observation of a single landmark will necessarily have an affect on every other state variable. This makes the EKF computationally infeasible for SLAM in large environment.

Methods like Network Coupled Feature Maps [17], Sequential Map Joining [43], and the Constrained Local Submap Filter (CRSF) [44] have been proposed to solve the problem of SLAM in large spaces by breaking the global map into submaps. This leads to a more sparse description of the correlations between map elements. When the robot moves out of one submap, it either creates a new submap or relocates itself in a previously defined submap. By limiting the size of the local map, this operation is constant time per step. Local maps are joined periodically into a global absolute map, in an $O(N^2)$ step. Each approach reduces the computational requirement of incorporating an observation to constant time. However, these computational gains come at the cost of slowing down the overall rate of convergence.

The Constrained Relative Submap Filter [44] proposes to maintain the local map structure. Each map contains links to other neighboring maps, forming a tree structure (where loops cannot be represented). The method converges by revisiting the local maps and updating the links through correlations. Whereas in the hierarchical SLAM [45], the links between local maps form an adjacency graph. This method allows to reduce the computational time and memory requirements and to obtain accurate metric maps of large environments in real time.

1.4.1 Hierarchical SLAM

The hierarchical SLAM approach [45] consists on the lower (or local) map level, which is composed of a set of local maps that are guaranteed to be statistically independent, and the upper (or global) level, which is an adjacency graph whose arcs are labeled with the relative location between local maps (figure 1.7).

The local level contains the information of the local areas where features and vehicle are represented with respect to a local reference frame. The global level consists on a graph in which the relative locations between local maps are described. The size of local maps can be determined depending on each situation.

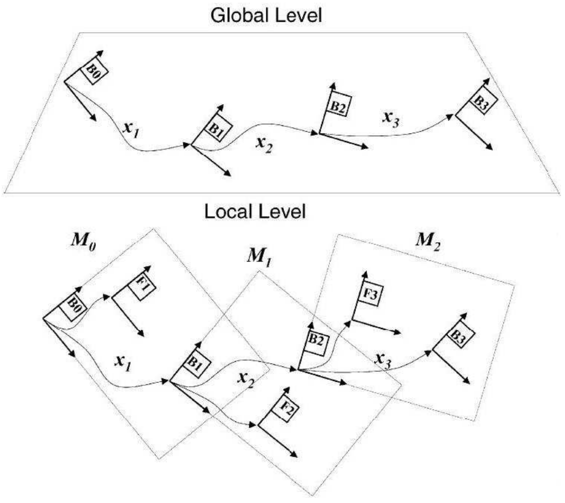


Figure 1.7: Two level hierarchical SLAM model [45]

Each local map $\mathcal{M}_j = \mathcal{M}^{B_j} = (\hat{\mathbf{x}}^{B_j}, \mathbf{P}^{B_j})$ is composed by: i) the state vector $\hat{\mathbf{x}}^B$, containing the pose of the vehicle $\mathbf{x}_r$ and the features $\mathbf{x}_i, i = 1 : N$ with respect to a base reference system B_j, and ii) the covariance matrix $\mathbf{P}^{B_j}$. The base reference B_j may be the initial current vehicle location when the map is initialized, or it may be associated to a set of local features, such as corner, a pair of points, etc. Data association process is carried out using Individual Compatibility Nearest Neighbor (ICCN) to establish correspondences and Joint Compatibility test to ensure the robustness of the matchings. In addition, each local map must contain the relation between its reference frame and the one of its neighbors, in order to estimate the global position of the vehicle when it is required. The decision to close a local map $\mathcal{M}_j$ and start a new local map is made once the number of features in the current local map reaches a maximum, or the uncertainty of the vehicle location with respect to the base reference of the current map reaches a limit. The new local map $\mathcal{M}_{j+1}$ will have the the current vehicle position as base frame, which corresponds to the last vehicle position in the map $\mathcal{M}_j$.

At the upper level, the topology of the environment is represented by a graph in which each node j corresponds to a local map $\mathcal{M}_j$ of the local level. An arc $c(j,k)$ in the graph represents a known topological relation between local maps $\mathcal{M}_j$ and $\mathcal{M}_k$ detected during the mapping process. The properties of those topological relations are the relative transformations between the base reference of both maps,($x_{jk} = \mathbf{x}_{B_k}^{B_j}$). At the global level, these relative transformations are maintained in a relative stochastic map $\mathcal{M}_G = (\hat{\mathbf{x}}_G, \mathbf{P}_G)$.

$$\hat{\mathbf{x}}_G = \begin{bmatrix} \vdots \\ \hat{\mathbf{x}}_{jk} \\ \vdots \end{bmatrix} ; \quad \mathbf{P}_G = \begin{bmatrix} . & 0 & 0 \\ 0 & \mathbf{P}_{jk} & 0 \\ 0 & 0 & . \end{bmatrix} \tag{1.30}$$

where $\hat{\mathbf{x}}_G$ and $\mathbf{P}_G$ represent the state vector and covariance matrix of global stochastic map.

Every time a local map is created or modified, the new values are included in this relative stochastic map. Since local maps are independent, the covariance matrix P is block diagonal by construction (eq.1.30).

While the vehicle is navigating, a set of local maps are built and an estimation of the vehicle pose is computed. Uncertainty grows considerably as the robot is moving within the environment. Therefore, the author proposed to reduce such uncertainty and correct the misalignments by benefiting of the information provided when the vehicle crosses already visited areas. The main idea is to use the information of the robot pose, provided by the relation between the local maps, and the relocation algorithm RS detailed in [67] in order to predict whether an area has already been visited. Once a loop is detected, the next step is to fuse both maps $\mathcal{M}_i^B$ and $\mathcal{M}_j^{B'}$ that are representing the same area. in [45], the author proposes a local map joining technique:

$$\mathcal{M}_{j+k}^B = (\hat{\mathbf{x}}_{j+k}^B, \mathbf{P}_{j+k}^B)$$

were $\hat{\mathbf{x}}_{j+k}^B$ and $\mathbf{P}_{j+k}^B$ represent the state vector and the covariance resultant of the fusion of the maps $\mathcal{M}_j^B$ and $\mathcal{M}_k^{B'}$ in the reference frame B.

Since the relative reference frames of both maps are known, the main goal of the algorithm is to transform one of the maps and its features into the reference system of the other one.

1.4.2 Constrained Local Submap Filter

Williams [44] proposed to reduce the complexity of EKF by using local maps linked among them forming a tree structure, presenting the so-called Constrained Local Submap Filter (CLSF) (figure 1.8). The local map is composed by a set of landmarks located in the surrounding navigation area of the vehicle/sensor. Each single map has its own reference system and all the landmarks are defined with respect to that reference. A relation between the coordinate system of each submap and the global map is computed. Each time a new local map is created, a new covariance matrix is defined containing only the new local landmarks and the pose of the vehicle/camera, initialized in the origin of the submap with zero uncertainty. In this way, the EKF update step of each submap does not require to deal with a huge amount of data and the partial accumulated error is relatively small. The state vector of the submap might contain not only the relative estimation of all the landmarks but also the global position of some of them and the relationship between the local and the global map. therefor the system will detect duplicate landmarks, finding their correspondences in the global map in a data association process.

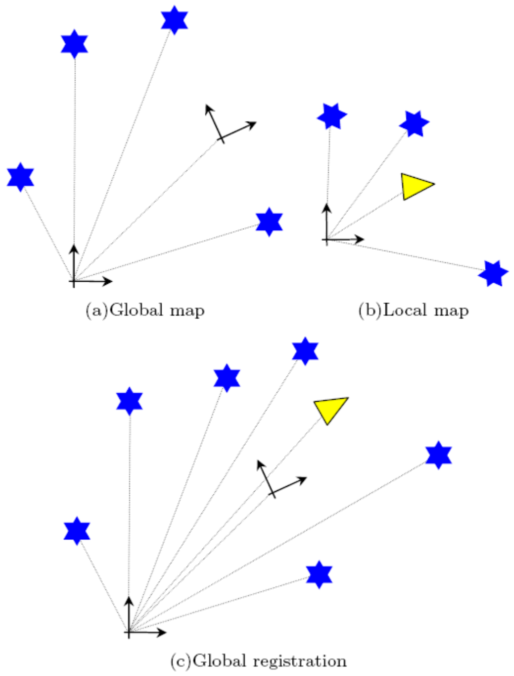


Figure 1.8: The constrained local submap filter. The SLAM frontier is constructed in a local map (b) which periodically registers with a global map (a) to produce an optimal global estimate (c) [44]

At every instant of time, data from local maps are merged with the global map taking into account all the global information and several constraints to ensure the global consistency. The state vector at time t is defined as:

$$\mathbf{x}^t = \begin{bmatrix} \mathbf{x}^{BG} \\ {\mathbf{x}_1^t}^G \\ {\mathbf{x}_1^t}^B \\ {\mathbf{x}_2^t}^B \\ \vdots \\ {\mathbf{x}_n^t}^B \end{bmatrix} \tag{1.31}$$

where $\mathbf{x}^{BG}$ determines the relation between the local B and the global G reference frames and ${\mathbf{x}_i^t}^G$ and ${\mathbf{x}_i^t}^B$ indicate the location of a landmark i with respect to the global frame G and the local frame B, respectively.

During the local update step, only the local covariance matrix is updated:

$$\mathbf{P}^t = \begin{bmatrix} \mathbf{P}^{t^G} & 0 \\ 0 & \mathbf{P}^{t^B} \end{bmatrix} \tag{1.32}$$

were $\mathbf{P}^{t^B}$ represents the covariance matrix of the local area, that is, the correlation of vehicle/sensor and all the landmarks between each other, and $\mathbf{P}^{t^G}$ represents the estimate covariance of the local map and its element respects to the global reference.

When a global update is required, the process needs to determine the global position of each feature. As can be seen in equation (1.31), each local map contains some features that are both related to local and global reference frame. Therefore, using this information and the relation between the landmark's local map and the global frame, the constraint (equation (1.33)) must hold on in order to guarantee the stability of the global system and ensure the correctness of the landmark pose prediction.

$$\mathbf{x}_i^{t^G} - (\mathbf{x}^{BG} \oplus \mathbf{x}_i^{t^B}) = 0 \tag{1.33}$$

the CLSF method presented by Williams provides a reliable solution to reduce the high computational cost of EKF when dealing with a huge amount of data. However, when dealing with huge sensor noise, the method is restricted to small environments due to the lack of closing-the-loop constraints. That is, when the uncertainty of the measurements grows considerably, global constrains might not be enough to assure success in the global estimation process.

1.5 Proposed Vision based SLAM in large-scale areas

We are interested in visual navigation of a mobile robot in large spaces using a mono-camera as sensory input. We propose a procedure to build a global representation of the environment based on several size limited feature based local maps built using an EKF based SLAM approach.

1.5.1 Feature selection

The mobile robot is supposed to travel through a locally planar environment and use its camera to observe features around it. Usually the features used in vision-based SLAM algorithms are salient and distinctive objects detected from images. Typical features might include regions, edges, object contours, corners etc.

In our work, the map features (figure 1.9) are obtained using the SIFT feature detector [46], which maps an image data into scale-invariant coordinates relative to local features (see appendix B section B.1.2). These features are highly distinctive and invariant to image scale and rotation. The work of Mikolajczyk and Schmid [47] proved that SIFT features remain stable to affine distortions, change of viewpoint, noise and change in illumination.

Heuristic map management criteria are used to decide when to initialize new features: essentially, the requirement is to keep a predefined number of features visible from all camera locations. A typical number used is 10; whenever fewer than 10 features are visible new ones are detected and initialized.

Features are not deleted from the map when they leave the field of view, but remain in the map and can be re-observed when the camera moves back and they become visible again. In some cases it is necessary to delete features which are not being reliably matched on a regular basis: some features detected will be frequently occluded or may contain parts of objects at very different depths. These features will lead to failed correlation attempts and can be removed from the map automatically.

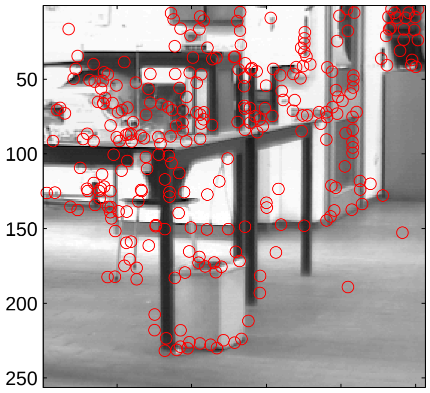


Figure 1.9: Features detected using the SIFT algorithm

1.5.1.1 Feature selection in dynamic environments

One of the problems of SLAM is environment's change over time. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time, or change in illumination. Others are faster, such as the change of door status or the location of furniture items, such as chairs. Even faster may be the change of location of other agents in the environment, such as cars or people. As example, imagine a robot facing a closed door that previously was modelled as open. Such an observation may be explained by two hypotheses, namely that the door status changed, or that the robot is not where it believes to be. Unfortunately, there are almost no mapping algorithms that can learn meaningful maps of dynamic environments.

To deal with the SLAM in dynamic scenes with moving object we propose an algorithm for motion segmentation (chapter 2) to remove the outliers features which are associated with moving objects. In other words, the detected features which correspond to the moving parts in the scene are not considered in the built map. For more feature tracking reliability we use a bounding box around the moving objects (figure 1.10). Moreover, the newly detected features are not added directly to the map but they should be detected and matched in at least n consecutive frames (in our application $n = 5$).

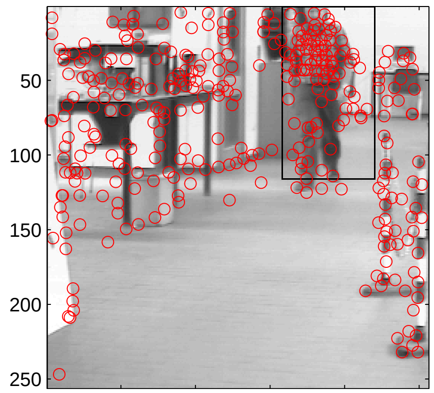


Figure 1.10: Features detected in a scene with moving objects

1.5.2 Local map building

The SLAM system is modeled at each time step t as a multivariate Gaussian with mean $\mathbf{x}^t$ composed of the robot $\mathbf{x}_r$ and features $\mathbf{x}_i$ state vectors in the global coordinate system G (eq. 1.34) and a covariance matrix $\mathbf{P}^t$ of size $(3 + 3N) \times (3 + 3N)$ (eq.1.35). N is the number of features in the map.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \tag{1.34}$$

$$\mathbf{P}^t = \begin{bmatrix} \mathbf{P}^t_{rr} & \mathbf{P}^t_{r1} & \cdots & \mathbf{P}^t_{rN} \\ \mathbf{P}^t_{1r} & \mathbf{P}^t_{11} & \cdots & \mathbf{P}^t_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^t_{Nr} & \mathbf{P}^t_{N1} & \cdots & \mathbf{P}^t_{NN} \end{bmatrix} \tag{1.35}$$

All sub-matrices $\mathbf{P}_{rr}$ (robot to robot covariance), $\mathbf{P}_{ri}$ (robot to feature covariance), $\mathbf{P}_{ii}$ (feature to feature covariance), and $\mathbf{P}_{ij}$ (feature to feature cross-correlations) are of size 3×3.

The robot state vector $\mathbf{x}_r$ is defined by its 2D position vector $(x^t_r, y^t_r)^T$ at time t, (its position in the Z direction being constant $z^t_r = 0$) and the robot's orientation (yaw) relative to the X axis, γ^t_r.

$$\mathbf{x}_r = \begin{bmatrix} x_r \\ y_r \\ \gamma_r \end{bmatrix}$$

The map features L_i, $i = 0, ..., N$ are represented by their 3D position in the global coordinate system:

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

1.5.2.1 Dynamic Model

The dynamic model or motion model is the relationship between the system paste state, $\mathbf{x}^{t-1}$, and its current state, $\mathbf{x}^t$, given a control input ξ^t

$$\mathbf{x}^t = \mathbf{f}\left(\mathbf{x}^{t-1}, \xi^t, \mathbf{v}^t\right) \tag{1.36}$$

Where $\mathbf{f}$ is a function representing the mobility, kinematics and dynamics of the robot (transition function) and $\mathbf{v}$ is a random vector describing the unmodelled aspects of the vehicle (process noise such as wheel sleep or odometry error).

We define a robot control ξ^t at time step t as any "motor actuation" performed by the robot since the last time step $t-1$. A typical situation for ground vehicles is to take the increment of the robot odometry (increments in $x-r$,$y-r$, and the heading γ_r) as the control. One alternative is to set $\xi^t = 0$ with large uncertainty, i.e. the robot is close to its last position up to a degree of uncertainty.

We assume static features. In this case, the only variables of the state vector that actually change over time are those of the robot pose. At each time step, the pose of the robot $\mathbf{x}_r$ changes according to the transition model

$$\mathbf{x}_r^t = \mathbf{f} - r\left(\mathbf{x}_r^{t-1}, \xi^t, \mathbf{v}^t\right) = \begin{bmatrix} x_r^{t-1} + (\nu^{t-1} + \mathbf{V})\cos(\gamma_r^{t-1})\Delta t \\ y_r^{t-1} + (\nu^{t-1} + \mathbf{V})\sin(\gamma_r^{t-1})\Delta t \\ \gamma_r^{t-1} + (\omega^{t-1} + \mathbf{\Omega})\Delta t \end{bmatrix} \tag{1.37}$$

ν and ω are the linear and the angular velocities, respectively. $\mathbf{V}$ and $\mathbf{\Omega}$ are their associated Gaussian distributed perturbations, respectively.

In this part of the chapter we consider that the global coordinate system is the robot's starting position. Therefore the robot pose and the corresponding covariance matrix are initialized to zeros.

Considering a constant velocity model for the smooth robot motion, the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}_r$ is given by:

$$\mathbf{F} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{(\mathbf{x}^{t-1|t-1}, \xi=0)} = diag\left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_r}\right|_{(\mathbf{x}_r^{t-1|t-1}, \xi=0)}, \mathrm{I}\right) \tag{1.38}$$

where

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_r}\right|_{(\mathbf{x}_r^{t-1|t-1}, \xi=0)} = \begin{bmatrix} 1 & 0 & -\sin(\gamma_r^{t-1})(\nu^{t-1} + \mathbf{V})\Delta t \\ 0 & 1 & \cos(\gamma_r^{t-1})(\nu^{t-1} + \mathbf{V})\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{1.39}$$

1.5.2.2 Observation Model

A feature L_i is represented in the state vector by its 3D location in the world coordinate system G:

$$\mathbf{x}_i^t = (x_i^t, y_i^t, z_i^t)$$

Making a measurement of a feature L_i consists of determining its projection in the camera image. Using a perspective projection, the observation model in the image coordinate system I (see section 1.2.6 for frames definition) is obtained as follows:

$$\mathbf{z}_i^t = \mathbf{h}(\mathbf{x}_i^t) = \begin{bmatrix} o_x + f\frac{y_i^{tC}}{x_i^{tC}} \\ o_y + f\frac{z_i^{tC}}{x_i^{tC}} \end{bmatrix} \tag{1.40}$$

where o_x and o_y are the coordinates of the image center and f is the focal length of the camera. $\mathbf{x}_i^{tC} = (x_i^{tC}, y_i^{tC}, z_i^{tC})$ are the coordinates of the feature L_i in the camera coordinate frame C (see section 1.2.6 for frames definition). They are related to $\mathbf{x}_i$ by:

$${}^C\mathbf{x}_i^t = \begin{bmatrix} x_i^{tC} \\ y_i^{tC} \\ z_i^{tC} \end{bmatrix} = \begin{bmatrix} \cos(\gamma_r^t) & \sin(\gamma_r^t) & 0 \\ -\sin(\gamma_r^t) & \cos(\gamma_r^t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^t - x_r^t \\ y_i^t - y_r^t \\ z_i^t - h \end{bmatrix} - \begin{bmatrix} x_c^R \\ y_c^R \\ z_c^R \end{bmatrix} \tag{1.41}$$

h is the high of the robot coordinate system and (x_c^R, y_c^R, z_c^R) are the coordinates of the camera center in the robot frame R.
The measurement model is then:

$$\mathbf{z}_i^t = \mathbf{h}(\mathbf{x}_i^t) = \begin{bmatrix} o_x + f\frac{-(x_i^t - x_r^t)\sin(\gamma_r^t) + (y_i^t - y_r^t)\cos(\gamma_r^t) - y_c^R}{(x_i^t - x_r^t)\cos(\gamma_r^t) + (y_i^t - y_r^t)\sin(\gamma_r^t) - x_c^R} \\ \\ o_y + f\frac{z_i^t - h - z_c^R}{(x_i^t - x_r^t)\cos(\gamma_r^t) + (y_i^t - y_r^t)\sin(\gamma_r^t) - x_c^R} \end{bmatrix} \tag{1.42}$$

1.5.2.2.1 Adding new features

Let $N-1$ denote the number of landmarks in the map at some time step. The system state vector and covariance matrix are then:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{rr} & \mathbf{P}_{r1} & \cdots & \mathbf{P}_{r(N-1)} \\ \mathbf{P}_{1r} & \mathbf{P}_{11} & \cdots & \mathbf{P}_{1(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{(N-1)r} & \mathbf{P}_{(N-1)1} & \cdots & \mathbf{P}_{(N-1)(N-1)} \end{bmatrix}$$

Adding a new feature L_N consists in introducing the feature state in the system state vector and expand the covariance with a new row and column to include its correlation and cross-correlation.

$$\mathbf{x}^+ = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_N \end{bmatrix}$$

$$\mathbf{P}^+ = \begin{bmatrix} & & & & \mathbf{P}_{rr}\frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}^T \\ & & \mathbf{P} & & \mathbf{P}_{1r}\frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}^T \\ & & & & \vdots \\ & & & & \mathbf{P}_{(N-1)r}\frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}^T \\ \frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}\mathbf{P}_{rr} & \frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}\mathbf{P}_{r1} & \cdots & \frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}\mathbf{P}_{r(N-1)} & \frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}\mathbf{P}_{rr}\frac{\partial \mathbf{g}}{\partial \mathbf{x}_r}^T + \frac{\partial \mathbf{g}}{\partial \mathbf{x}_N}\mathbf{R}\frac{\partial \mathbf{g}}{\partial \mathbf{x}_N}^T \end{bmatrix}$$

where $\mathbf{g}$ is the inverse function of $\mathbf{h}$ and $\mathbf{R}$ is the sensor noise covariance matrix.

1.5.2.2.2 Subtracting outliers from the map

Some stationary objects in the scene can start moving. In this case, their corresponding features should be deleted from the built map. Deleting a feature L_i from the map consists in deleting it as element from the system state vector and the corresponding columns and rows in the covariance matrix.

$$\mathbf{x}^t = \begin{bmatrix} \mathbf{x}_r^t \\ \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_{i-1}^t \\ \mathbf{x}_{i+1}^t \\ \vdots \\ \mathbf{x}_n^t \end{bmatrix} \tag{1.43}$$

$$\mathbf{P}^t = \begin{bmatrix} \mathbf{P}_r^t & \cdots \mathbf{P}_{r,i-1}^t & \mathbf{P}_{r,i+1}^t & \cdots & \mathbf{P}_{r,n}^t \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{P}_{n,r}^t & \cdots \mathbf{P}_{i-1,r}^t & \mathbf{P}_{i+1,r}^t & \cdots & \mathbf{P}_n^t \end{bmatrix} \tag{1.44}$$

1.5.3 Features initialization

When a feature is first detected, measurement from a single camera position provides good information on its direction relative to the camera, but its depth is initially unknown.

Since depth information is not provided, EKF can not be directly initialized, leading to a new challenge known as Bearing-Only SLAM. An early approach was proposed by Deans [48], who combined Kalman filter and bundle adjustment in filter initialization, obtaining accurate results at the expense of increasing filter complexity. In [9], Davison uses for initialization an A4 piece of paper as a landmark to recover metric information of the scene. Then, whenever a scene feature is observed a set of depth hypotheses are made along its direction. In subsequent steps, the same feature is seen from different positions reducing the number of hypotheses and leading to an accurate landmark pose estimation. Besides, Lemaire [37] proposed a 3D Bearing-Only SLAM algorithm based on EKF filters, in which each feature is represented by a sum of Gaussians.

In our application, to estimate the 3D position of the detected features, we use an approach based on *epipolar geometry* [49, 50]. This geometry represents the geometric relationship between multiple viewpoints of a rigid body and it depends on the internal parameters and relative positions of the camera (see appendix A for more details). The two and three views geometry have been used in our case.

1.5.3.1 Two-views geometry

The epipolar geometry is illustrated in figure 1.11.

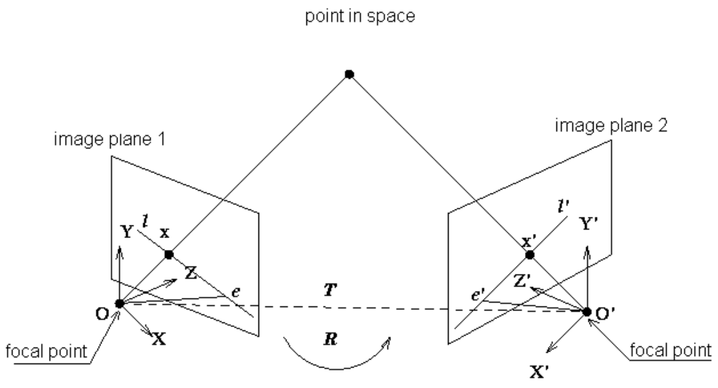


Figure 1.11: Illustration of the epipolar geometry

The fundamental matrix $\mathbb{F}$ (a 3×3 matrix of rank 2) encapsulates this intrinsic geometry. It describes the relationship between matching points: if a point $\tilde{\mathbf{X}}$ is imaged as $\mathbf{x}_i$ in the first view, and $\mathbf{x'}_j$ in the second, then the image points must satisfy the relation $\mathbf{x}_i^T \mathbb{F} \mathbf{x}_j = 0$. The fundamental matrix is independent of scene structure. However, it can be computed from correspondences of imaged scene points alone, without requiring knowledge of the cameras internal parameters or relative pose.

Given a set of n pairs of image correspondences $(\mathbf{x}_j, \mathbf{x'}_j), j = 1..n$, we compute $\mathbf{R}$ and $\mathbf{t}$ such the epipolar error is minimized

$$\min_{\mathbb{F}} \sum_{j=1}^{n} \mathbf{x'}_j \mathbb{F} \mathbf{x}_j$$

For the minimization, we use the Random Sample Consensus (RANSAC) algorithm [51] (see appendix A for more details on using RANSAC for motion parameters estimation).

Essential matrix computing:
Knowing the camera calibration matrix $\mathbf{K}$, we can calculate the essential matrix $\mathbf{E}$ as follows (see appendix A):

$$\mathbf{E} = \mathbf{K}^T \mathbb{F} \mathbf{K} \tag{1.45}$$

The camera calibration matrix $\mathbf{K}$ encodes the transformation from image coordinates to pixel coordinates in the image plane. It depends on the so-called intrinsic parameters:

- focal distance f (in mm),
- principal point (or image centre) coordinates o_x,o_y (in pixel),
- width (s_x) and height (s_y) of the pixel footprint on the camera photosensor (in mm),
- angle θ between the axes (usually $\pi/2$).

The ratio s_y/s_x is the aspect ratio (usually close to 1).

$$\mathbf{K} = \begin{bmatrix} f/s_x & f/s_x \cos\theta & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

compute the camera matrices and motion parameters:
The camera coordinate systems corresponding to tow views are related by a rotation matrix, $\mathbf{R}$, and a translation vector, $\mathbf{t}$:

$$\mathbf{x}_j = \mathbf{R}\mathbf{x}_i + \mathbf{t} \tag{1.46}$$

Taking the vector product with $\mathbf{t}$, followed by the scalar product with $\mathbf{x}_j$, we obtain:

$$\mathbf{x}_j.(\mathbf{t} \wedge \mathbf{R}\mathbf{x}_i) = \mathbf{0} \tag{1.47}$$

This can also be written as

$$\mathbf{x}_j \mathbf{E} \mathbf{x}_i = \mathbf{0} \tag{1.48}$$

where

$$\mathbf{E} = \mathbf{t}_\times \mathbf{R} \tag{1.49}$$

is the essential matrix, and $\mathbf{t}_\times$ denotes skew symmetric cross product matrix for $\mathbf{t}$

$$\mathbf{t}_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

The rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the two camera frames are then calculated by singular value decomposition SVD decomposition of $\mathbf{E}$.

Suppose that the SVD decomposition of $\mathbf{E}$ is $\mathbf{U}\text{diag}(1, 1, 0)\mathbf{V}^T$. The factorization $\mathbf{E} = \mathbf{t}_\times \mathbf{R}$ corresponds to (see appendix A):

$$\mathbf{t}_\times = \mathbf{U}\mathbf{Z}\mathbf{U}^T \qquad \mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T \tag{1.50}$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textit{and} \qquad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The SVD decomposition of $\mathbf{E}$ allows also to compute the camera projection matrices. Supposing that the projection matrix at the first view is the $\mathcal{P} = [I|0]$, the projection matrix for the second view is then given by: $\mathcal{P}' = [\mathbf{U}\mathbf{W}\mathbf{V}^T|t]$.

Triangulation and 3D reconstruction:
Given the camera matrices $\mathcal{P}$ and $\mathcal{P}'$, let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two corresponding points satisfying the epipolar constraint $\mathbf{x}_i^T \mathbb{F} \mathbf{x}_j = 0$. It follows that $\mathbf{x}_i$ lies on the epipolar line $\mathbb{F}\mathbf{x}_j$ and so the two rays back-projected from image points $\mathbf{x}_i$ and $\mathbf{x}_j$ lie in a common epipolar plane. Since they lie in the same plane, they will intersect at some point. This point is the reconstructed 3D scene point M.

Analytically, the reconstructed 3D point $\tilde{X}$ can be found the following equation:

$$\mathbf{e} = z_i \mathbf{x}_i - z_j \mathbf{x}_j \tag{1.51}$$

where $\mathbf{e}$ is the epipole at the first view. z_i and z_j are the depth of the space point $\tilde{X}$ with respect to the first and second views, respectively.

The three points $\mathbf{x}_i$, $\mathbf{e}$, and $\mathbf{x}_j$ are known and are collinear, so we can solve for Z:

$$z = \frac{(\mathbf{e} \times \mathbf{x}_j).(\mathbf{x}_i \times \mathbf{x}_j)}{\|\mathbf{x}_i \times \mathbf{x}_j\|^2} \tag{1.52}$$

1.5.3.2 Three-views geometry

The trifocal tensor approach is an extension to the case of three views of the two-view geometry description (see appendix A). The $3 \times 3 \times 3$ trifocal tensor $\mathfrak{T}$ describes the relationship between three images of the same static scene. It encapsulates the projective geometry between the different viewpoints and is independent from the structure of the scene. Knowing the projection matrices of the three cameras, $\mathcal{P}_{i=1,2,3}$, the entries of the trifocal tensor are given by

$$\mathfrak{T}_{i,j,k} = (-1)^{i+1}.det \begin{pmatrix} a^i \\ b^j \\ c^k \end{pmatrix} \tag{1.53}$$

where a^i denotes matrix $\mathcal{P}_1$ without row i and b^j and c^k represent the j-th row of $\mathcal{P}_2$ and the k-th row of $\mathcal{P}_3$ respectively [49, 50].

1.5.4 Features Matching

At step t, the onboard camera obtains a set of measurements $\mathbf{z}_i^t$ $(i = 1, ..., m)$ of m environment features. Feature matching corresponds to data association, also known as the correspondence problem, which consists in determining the origin of each measurement, in terms of the map features L_j , $j = 1, ..., N$. The result is a hypothesis

$$\mathcal{H}^t = [\mathfrak{h}_1^t, ..., \mathfrak{h}_m^t] \tag{1.54}$$

associating each measurement $\mathbf{z}_i^t$ with its corresponding map feature. $\mathfrak{h}_i^t = 0$ indicates that $\mathbf{z}_i^t$ does not come from any feature in the map. For data association a measure of the discrepancy between a predicted measurement that each feature would generate and an actual sensor measurement is measured by the innovation ε given by (1.22).

The measurement $\mathbf{z}_i^t$ can be considered corresponding to the feature j if the Mahalanobis distance ${D_{ij}^2}^t$ [53] satisfies:

$${D_{ij}^2}^t = {\varepsilon^T}^t {\mathbf{S}^{-1}}^t \varepsilon^t < th \tag{1.55}$$

Where the covariance $\mathbf{S}^t$ and the innovation ε^t are given by equations (1.21) and (1.22), respectively.

In order to establish the consistency of a hypothesis $\mathcal{H}^t$, measurements can be jointly predicted using the function $\mathbf{h}_{\mathcal{H}^t}$

$$\mathbf{h}_{\mathcal{H}^t}(\mathbf{x}^{t|t-1}) = \begin{bmatrix} \mathbf{h}_{\mathfrak{h}_1^t}(\mathbf{x}^{t|t-1}) \\ \vdots \\ \mathbf{h}_{\mathfrak{h}_m^t}(\mathbf{x}^{t|t-1}) \end{bmatrix} \tag{1.56}$$

which can also be linearized around the current estimate to yield:

$$\mathbf{h}_{\mathcal{H}^t}(\mathbf{x}^{t|t-1}) \simeq \mathbf{h}_{\mathcal{H}^t}(\hat{\mathbf{x}}^{t|t-1}) + \mathbf{H}_{\mathcal{H}^t}(\mathbf{x}^{t|t-1} - \hat{\mathbf{x}}^{t|t-1})$$

$$\mathbf{H}_{\mathcal{H}^t} = \begin{bmatrix} \mathbf{H}_{\mathfrak{h}_1^t} \\ \vdots \\ \mathbf{H}_{\mathfrak{h}_m^t} \end{bmatrix} \tag{1.57}$$

The joint innovation and its covariance are:

$$\varepsilon_{\mathcal{H}^t}^t = \mathbf{z}^t - \mathbf{h}_{\mathcal{H}^t}(\hat{\mathbf{x}}^{t|t-1})$$

$$\mathbf{S}_{\mathcal{H}^t} = \mathbf{H}_{\mathcal{H}^t}\mathbf{P}^t\mathbf{H}_{\mathcal{H}^t}^T + \mathbf{R}_{\mathcal{H}^t} \tag{1.58}$$

Measurements $\mathbf{z}^t$ can be considered compatible with their corresponding features according to $\mathcal{H}^t$ if the Mahalanobis distance satisfies:

$$D_{\mathcal{H}^t}^2 = \varepsilon_{\mathcal{H}^t}^T \mathbf{S}_{\mathcal{H}^t}^{-1} \varepsilon_{\mathcal{H}^t} < th \tag{1.59}$$

In our application, as we are using SIFT features, the matching between feature is checked using a product of the Mahalanobis distance between measurements and their predictions and the Euclidean distance between the descriptor vectors of the features. This will allow using the advantage of looking for feature matching based on the prediction of their position based on the system model and the advantage of the scale-space invariance parameters.

$$D^2 = D_{\mathcal{H}^t}^2 + D_{desc}^2 < th \tag{1.60}$$

where $D_{desc}^2 = \| desc_1 - desc_2 \|$ is the Euclidean distance between the descriptor vectors of the features.

Additionally, corresponding features should satisfy the epipolar constraint (see appendix A), hence an image point $\mathbf{x}_i^t$ that corresponds to $\mathbf{x}_i^{t-1}$ is located on or near the epipolar line that is induced by $\mathbf{x}_i^{t-1}$. The distance of the image point $\mathbf{x}_i^t$ from that epipolar line is computed as follows:

$$D_{epi}^2 = \frac{(\mathbf{x}_i^{t^T} \mathbb{F} \mathbf{x}_i^{t-1})^2}{(\mathbb{F}\mathbf{x}_i^{t-1}\big|_1)^2 + (\mathbb{F}\mathbf{x}_i^{t-1}\big|_1)^2} \tag{1.61}$$

where $\mathbb{F}\mathbf{x}_i^{t-1}\big|_j$ is the j component of the vector $\mathbb{F}\mathbf{x}_i^{t-1}$. $\mathbb{F}$ is the fundamental matrix which is computed based on the estimations from the Extended Kalman Filter. Knowing the camera position at instants t, the estimated camera position at instant $t-1$, and the camera calibration matrix $\mathbf{K}$ (see appendix A for the definition of $\mathbf{K}$), the fundamental matrix $\mathbb{F}$ is computed as follows:

$$\mathbb{F} = \mathbf{K}^{-T}\mathbf{R}[\mathbf{t}]_\times \mathbf{K}^{-1} \tag{1.62}$$

where $\mathbf{R}$ is the rotation matrix and $[\mathbf{t}]_\times$ is the skew matrix corresponding to the translation vector $\mathbf{t}$. The notation $\mathbf{K}^{-T}$ denotes the transpose of the inverse $\mathbf{K}$.

Therefore, our cost function for features matching is the sum of D^2 and D^2_{epi}

$$D^2_{match} = D^2 + D^2_{epi} \tag{1.63}$$

Measurements for which correspondences in the map cannot be found by data association can be directly added to the current stochastic state vector as new features.

Feature close to an obstacle
If a well initialized feature is detected too close to a moving object (less then the image patches half size), this feature is considered as occluded and no matching is performed. These features are not used as well for motion estimation.

1.5.5 Local and Global Mapping

In our study, we are interested in robot navigation in large spaces. For that, we propose a procedure to build a global representation of the environment based on several size limited local maps built using the previously described approach. Two methods for local map joining are proposed, the first method consists in transforming each local map into a global frame before to start building a new local map. While in the second method, the global map consists only in a set of robot positions where new local maps started (i.e. the base references of the local maps). In both methods, the base frame for the global map is the robot position at instant t_0.

Each local map is built as follows: at a given instant t_k, a new map is initialized using the current vehicle location, $\mathbf{x}_R^{t_k}$, as base reference $B_k = \mathbf{x}_r^{t_k}$, $k = 1, 2, ...$ being the local map order. Then, the vehicle performs a limited motion acquiring sensor information about the L_i neighboring environment features. An EKF-based technique is used to model the local maps.

The 'k'th local map is defined by:

$$\mathfrak{M}_k = (\mathbf{x}_k, \mathbf{P}_k)$$

where $\mathbf{x}_k$ is the state vector in the base reference B_k of the L_k detected features and $\mathbf{P}_k$ is their covariance matrix estimated in B_k.

1.5.5.1 Key-Instants

The decision to start building a new local map at an instant t_k is based on two criteria: the number of features in the current local map and the scene cut detection result (see section 2.3.2 for the definition of scene cut detection). The instant t_k is called a key-instant.

In our application we defined two thresholds for the number of features in the local maps: a lower Th^- and a higher Th^+ thresholds. A key-instant is selected if the number of features n_l^k in the current local map k is larger then the lower threshold and a scene cut has been detected or the number of features has reached the higher threshold. This allows keeping reasonable dimensions of the local maps and avoids building too small maps.

```
IF ( $n_l^k > Th^-$ AND scene-cut ) OR ($n_l^k > Th^+$)
    start a new local map $\mathfrak{M}_{k+1}$
ELSE
    continue with the local map $\mathfrak{M}_k$
END IF
```

1.5.5.2 Key-Frames

At the key-instants t_k some frames are selected to be used for features initialization in the new local map. These frames are called key-frames. The motion between two frames must be sufficiently large to accurately compute the 3D positions of matched points. For that we select frames relatively far from each other.

1.5.5.3 First Method for Global Map Building

In this method the first local map is used as global map. Each finalized local map is transferred to the global map before starting a new one, by computing the state vectors and the covariance matrix.

The goal of map joining is to obtain one full stochastic map:

$$\mathfrak{M} = (\mathbf{x}^0_{(0\oplus 1\oplus 2\oplus ...)}, \mathbf{P}^0_{(0\oplus 1\oplus 2\oplus ...)}) \tag{1.64}$$

where $\mathbf{x}^0_{(0\oplus 1\oplus 2\oplus ...)}$ is a concatenation in the frame B_0 of all sets of features from local maps $\mathfrak{M}_0$, $\mathfrak{M}_1$, $\mathfrak{M}_2$, ...:

$$\mathbf{x}^0_{(0\oplus 1\oplus 2\oplus ...)} = (\mathbf{x}^0_0, \mathbf{x}^0_1, \mathbf{x}^0_2, ...)^T \tag{1.65}$$

The location $\mathbf{x}^1_i$ of feature i from the local map $\mathfrak{M}_1$ is given in the frame B_0 as follows:

$$\begin{pmatrix} \mathbf{x}^0_i \\ 1 \end{pmatrix} = \mathcal{T}_{1\to 0} \cdot \begin{pmatrix} \mathbf{x}^1_i \\ 1 \end{pmatrix} \tag{1.66}$$

$\mathcal{T}_{1\to 0} = (\mathcal{R}|\mathbf{t})$ is the transformation matrix corresponding to rotation $\mathcal{R}$ and $\mathbf{t}$ from frame $B_1 = \mathbf{x}^{t_1}_x = (x^{t_1}_r, y^{t_1}_r, \gamma^{t_1}_r)^T$ to frame $B_0 = \mathbf{x}^0_r = (0, 0, 0)^T$:

$$\mathcal{T}_{1\to 0} = \begin{pmatrix} \cos(\gamma_r^{t_1}) & 0 & -\sin(\gamma_r^{t_1}) & y_1^{t_1} \\ 0 & 1 & 0 & 0 \\ \sin(\gamma_r^{t_1}) & 0 & \cos(\gamma_r^{t_1}) & y_2^{t_1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.67}$$

As each camera position $\mathbf{x}_r^{t_k}$ corresponding to the frame reference B_k is given in the previous frame reference B_{k-1}, the coordinates of a feature point i vector from frame B_k to B_0 is obtained by successive transformations:

$$\begin{pmatrix} \bar{\mathbf{x}}_i^0 \\ 1 \end{pmatrix} = \mathcal{T}_{k\to(k-1)}\mathcal{T}_{(k-1)\to(k-2)}...\mathcal{T}_{1\to 0}.\begin{pmatrix} \mathbf{x}_i^k \\ 1 \end{pmatrix} \tag{1.68}$$

Where the transformation matrix from B_k to B_{k-1}, $\mathcal{T}_{k\to k-1} = (\mathcal{R}|\mathbf{t})$, is given by:

$$\mathcal{T}_{k\to(k-1)} = \begin{pmatrix} \cos(\gamma_r^{t_k}) & 0 & -\sin(\gamma_r^{t_k}) & x_r^{t_k} \\ 0 & 1 & 0 & 0 \\ \sin(\gamma_r^{t_k}) & 0 & \cos(\gamma_r^{t_k}) & y_r^{t_k} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.69}$$

The covariance $\mathbf{P}^0_{(0\oplus 1\oplus 2\oplus ...)}$ of the joint map is obtained from the linearization of the state transition function $\mathbf{f}$. As the local maps are independents, the Jacobian (from linearization) is then applied separately to the local map covariance:

$$\mathbf{P}^0_{(0\oplus 1\oplus 2\oplus ...)} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_0 P_0 \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_0^T + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_1 P_1 \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_1^T + ... + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_i P_i \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}_r}\right|_i^T + ... \tag{1.70}$$

where $\left.\frac{\partial\mathbf{f}}{\partial\mathbf{y}_r}\right|_k$ is the Jacobian of the state transition function $\mathbf{f}$ with respect to $\mathbf{y}_r$ in the reference frame k.

1.5.5.4 Second Method for Global Map building

In this method, the global map is limited to the set of the coordinates of the origins of the local maps:

$$\mathfrak{M}_G^B = (\bar{\mathbf{x}}_r^0, \bar{\mathbf{x}}_r^1, \bar{\mathbf{x}}_r^2, ...) \tag{1.71}$$

where $\bar{\mathbf{x}}_r^k$ are the robot coordinates in B_0, where it decides to build the local map $\mathfrak{M}_k$ at instant t_k.

$$\begin{pmatrix} \bar{\mathbf{x}}_r^k \\ 1 \end{pmatrix} = \mathcal{T}_{k\to 0}.\begin{pmatrix} \mathbf{x}_r^{t_k} \\ 1 \end{pmatrix} \tag{1.72}$$

$t_0 = 0$ and $\bar{\mathbf{x}}_r^0 = \mathbf{x}_r^{t_0} = (0, 0, 0)$.

The transformation matrix $\mathcal{T}_{k\to 0}$ is obtained by successive transformations:

$$\mathcal{T}_{k\to 0} = \mathcal{T}_{1\to 0}.\mathcal{T}_{2\to 1}...\mathcal{T}_{(k-1)\to(k-2)} \tag{1.73}$$

where $\mathcal{T}_{i \to i-1} = (\mathcal{R}|\mathbf{t})$ is the transformation matrix corresponding to rotation $\mathcal{R}$ and translation $\mathbf{t}$ of frame B_i regarding to frame B_{i-1}:

$$\mathcal{T}_{i \to i-1} = \begin{pmatrix} \cos(\gamma_r^{t_i}) & 0 & -\sin(\gamma_r^{t_i}) & x_r^{t_i} \\ 0 & 1 & 0 & 0 \\ \sin(\gamma_r^{t_i}) & 0 & \cos(\gamma_r^{t_i}) & y_r^{t_i} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.74}$$

In this case, for feature matching at instant t, the robot uses the local map with the closest base frame to its current location:

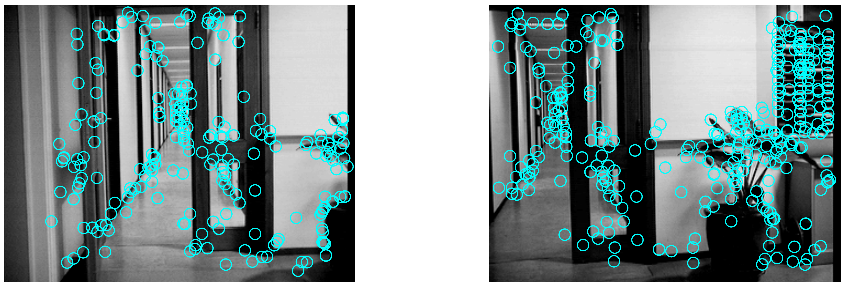
$$\arg\min_i (\| \bar{\mathbf{x}}_i^k - \bar{\mathbf{x}}_r^t \|) \tag{1.75}$$

where $\bar{\mathbf{x}}_r^t$ is the robot position at instant t in B_0.

1.6 Experimental Results & Evaluation

1.6.1 Feature selection and matching

Figure 1.13 shows a comparison between matching of features detected in two frames (figure1.12), using Mahalanobis distance between measurements and their predictions with consistency hypothesis (1.13.c), by using Euclidean distance between features descriptors as in [46] (1.13.d), and using the proposed method with a combination of the Mahalanobis distance with consistency hypothesis and Euclidean distance between features descriptors (1.13.e). We can see that the proposed matching is more accurate.



a) SIFT features on the first frame

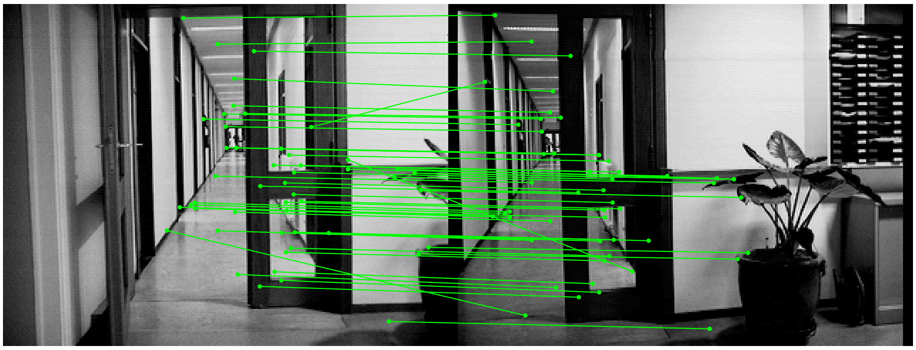
b) SIFT features on the second frame

Figure 1.12: SIFT feature on two consecutive frames

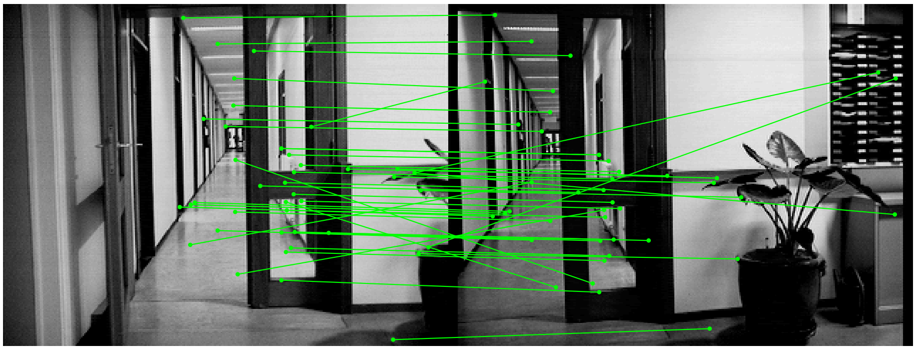
Figures 1.15 and 1.16 represent a comparison of the number of matches between the three methods for feature matching: The method used in [9] based on the Shi and Thomasi feature detector, the SIFT features matched using the Mahalanobis distance and SIFT features matched using the proposed approach.

Two data set have been used for this evaluation with changes in scaling (the sequence with corridor used in the previous tests) and change in oriantation (the building sequence 1.14). For each data set, we matched the feature in image 1 to the features in image n (where n = 2, 3, ..., 6).

From those experimental result, we can observe that our feature matching outperforms the Shi and Thomasi feature matching used in [9] and the SIFT features matched by only Mahalanobis distance.


c) Features matching based on Mahalanobis distance with consistency hypothesis

-


d) Features matching based on Euclidean distance between features descriptors (Lowe method)

-

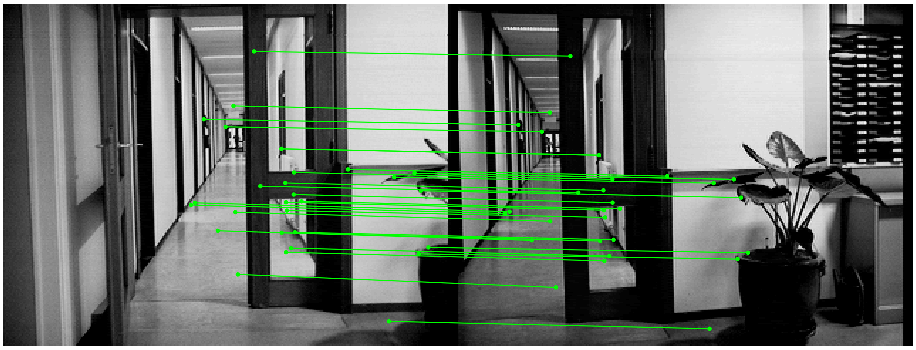

e) Features matching based on Eq.(1.63)

Figure 1.13: SIFT feature matching

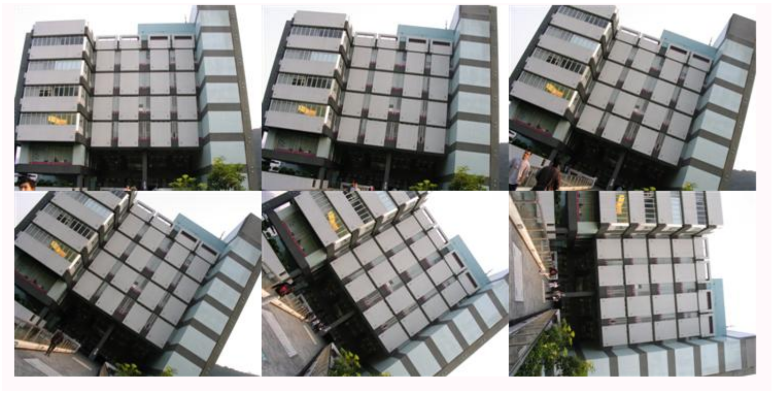


Figure 1.14: Building sequence

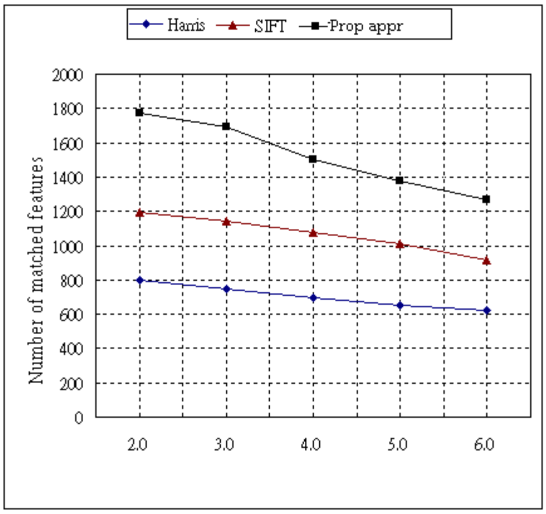


Figure 1.15: Feature matching in case of change in scaling

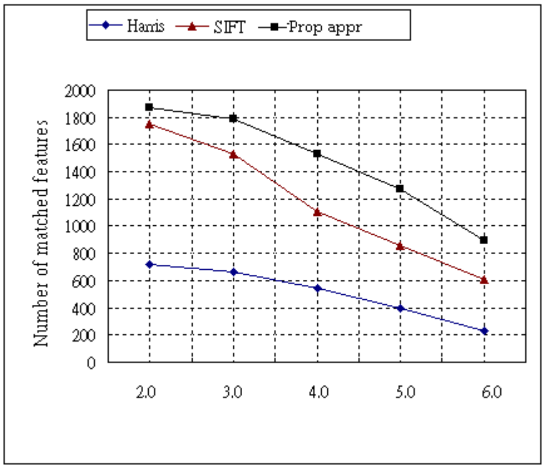


Figure 1.16: Feature matching in case of change in orientation

Figure 1.17 shows an example with a feature detected on a moving object. As the feature has not been detected during 10 consecutive frames before being occluded by the moving objects, this feature will not be added to the system state. The figure shows also an example with well initialized features which will be occluded by the moving object. These features stay in the system state.

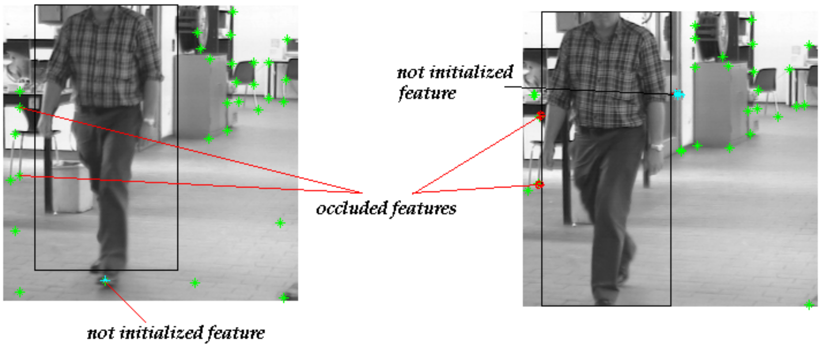


Figure 1.17: Features initialized on a moving object or occluded by a moving object

1.6.2 Mapping and localization

Figure 1.18 shows an example of SLAM using the proposed algorithm for global map building. Black squares describes the positions where the feature matching has passed from a local frame to an other. The ellipses around the features on the original frame represents the estimated covariance. The ellipses are drown in cyan color around non matched feature and in red for matched features.

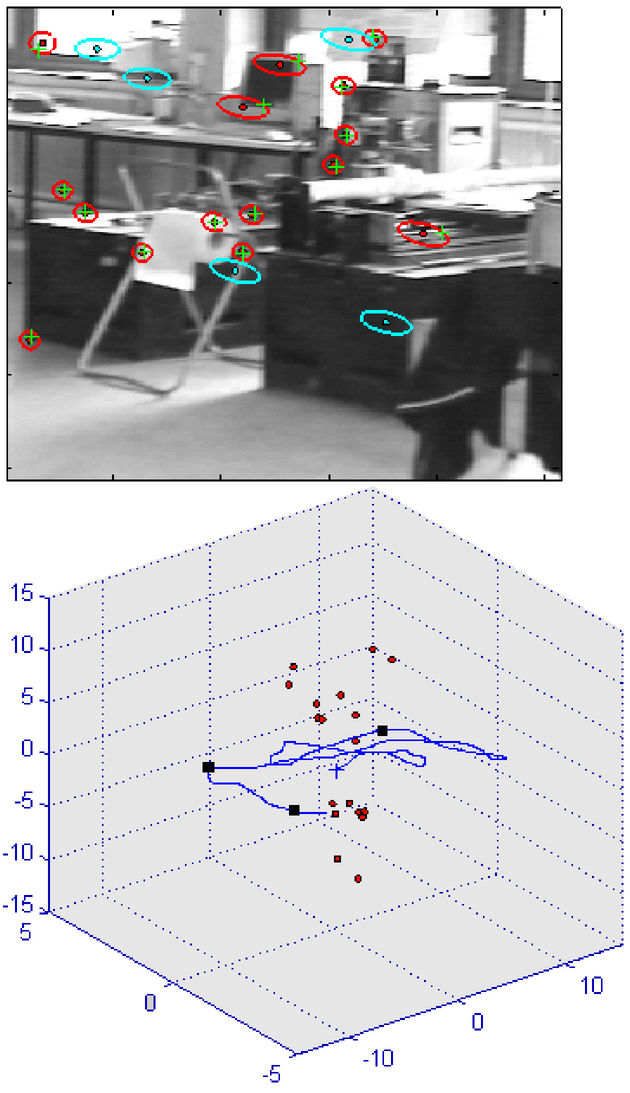


Figure 1.18: Example of Mapping and localization in a real scene

Figure 1.19 represents for the same image sequence an estimation of the camera position error and its corresponding 2σ variance bounds. Position errors are plotted as x and y distances to camera location (calculated using the vehicle odometry) in meters.

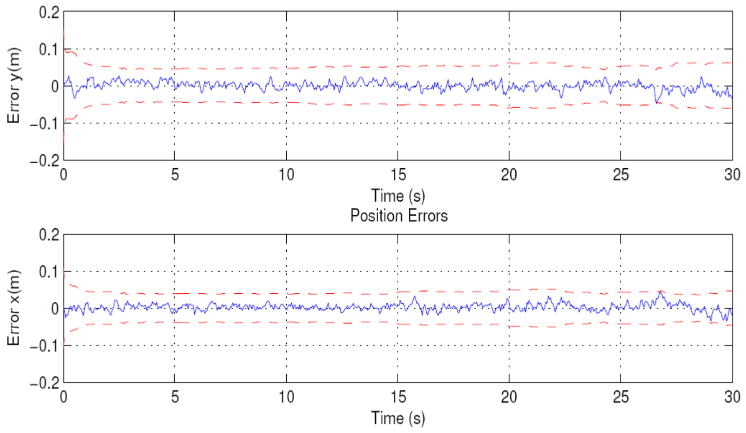


Figure 1.19: Estimation errors for camera position and its corresponding 2σ variance bounds. Position errors are plotted as x and y distances to camera location (calculated using the vehicle odometry) in meters.

Figure 1.20 shows the result of the algorithm in a scene with moving objects.

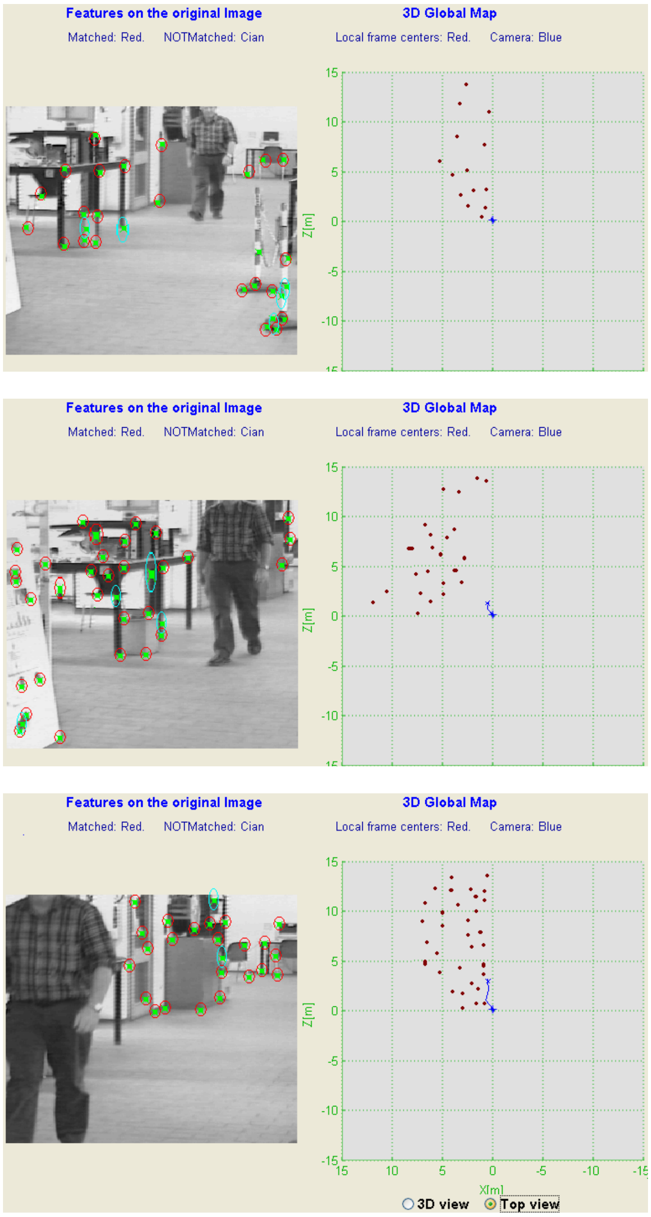
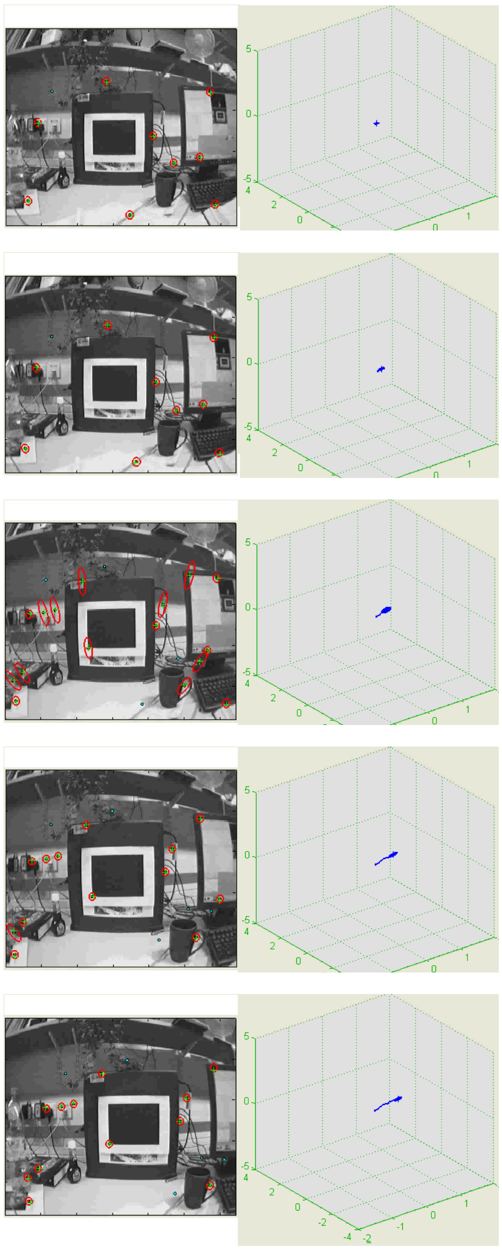
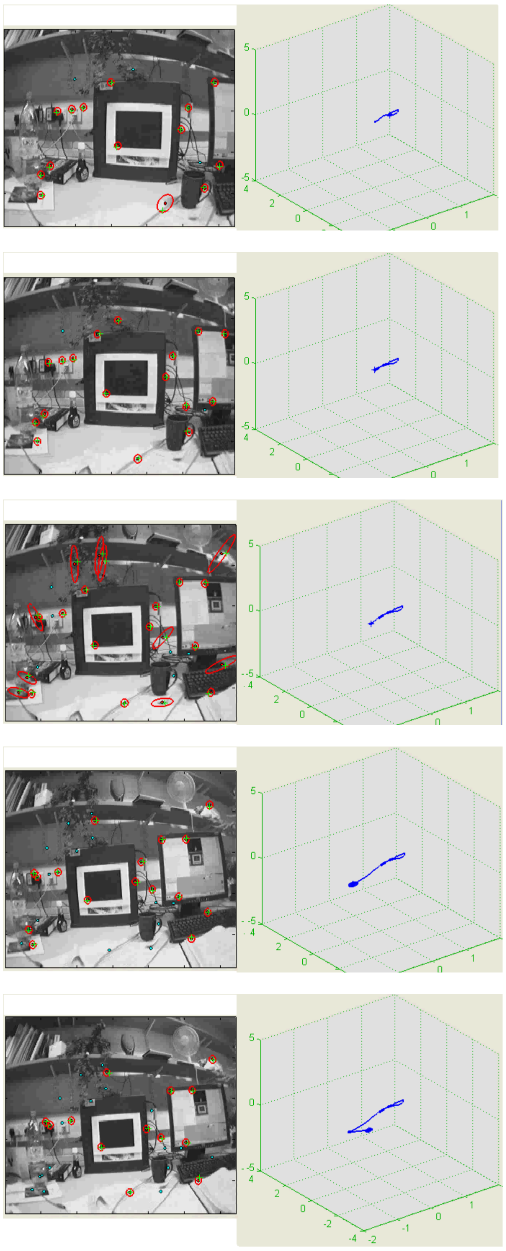

Features on the original Image
Matched: Red. NOTMatched: Cian
3D Global Map
Local frame centers: Red. Camera: Blue
Z[m]
X[m]
3D view
Top view

Figure 1.20: Results in a scene with moving objects

Figure 1.21 represents the evolution in time of the uncertainty around the camera position. The uncertainty continues growing until a previous position has been identified (closing loop) by feature matching.


5
0
-5
4
2
0
-2
-4
-2
-1
0
1



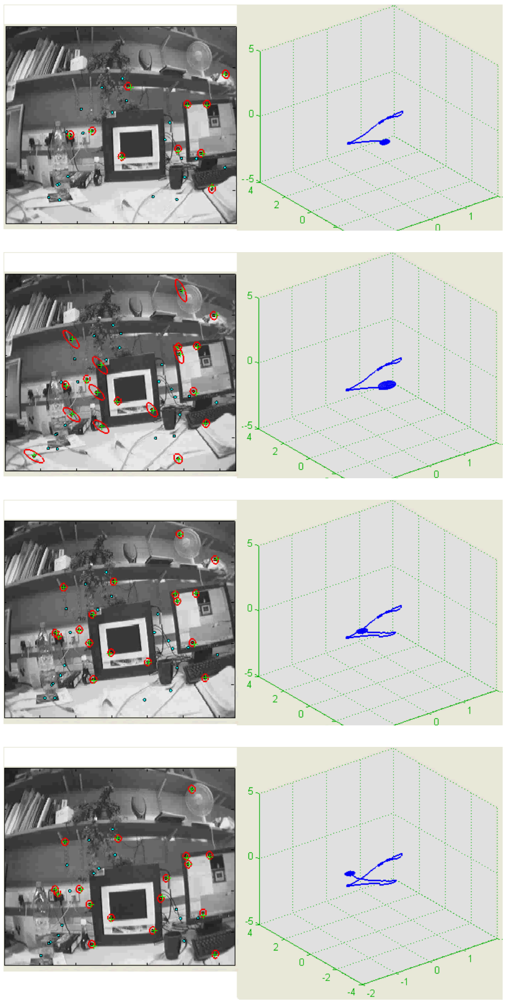


Figure 1.21: Frame-by-frame camera position uncertainty evolution

Figure 1.22, shows an example for the detection of loops using the SLAM process. In this example, the robot cruises two times across a defined path (real path drawn in red in the figure). At the first round the position error accedes 5m and the uncertainty around the robot position reaches 6.2m (cyan ellipses in the figure). After loop detection, the uncertainty is reduced. During the second round the position error is limited to a maximum of 1m and the uncertainty to a maximum of 2 meters (magenta ellipses in the figure) before the detection of the close the loop.

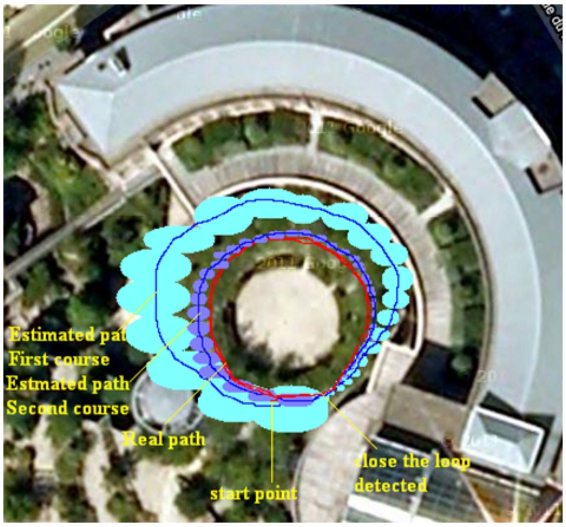


Figure 1.22: Closing the loop

1.7 Integrating other sensors data in the SLAM process

In this section we suppose that, added to the monocular camera, the robot is equipped with an inertial navigation system (INS), wheel encoders, and a global positioning system (GPS) for outdoor navigation. Each of these sensors can be used separately by the robot for its localization but they are subject to a lot of error sources affecting the accuracy of the obtained robot positioning.

Two schemas of integration can be imagined, in the first case, the data from the INS and encoder sensors are used in the SLAM algorithm for angular and linear velocity estimation, respectively, and the data from the GPS, when available, is used for a geo-referencing the localization of the robot and therefor the map features. While in the seconde case, the estimated camera motion from camera/INS/encoders is used to improve the GPS positioning. In both cases, the robot can continue tracking its GPS position even if the GPS signal is lost.

The following paragraphs give an overview of the functioning principles of the different sensors

1.7.1 GPS data correction using INS/Encoders data

Positioning using global positioning systems (GPS) is determined, at any time, by measuring the time delay in a radio signal broadcast from several satellites, and using this and the speed of propagation to calculate the distance to the satellites. Position on earth is then calculated by triangulation of intersecting radio signals at the GPS receiver. Using the GPS for positioning is subject to several sources of errors, as listed below.

Clock inaccuracies and rounding errors: Despite the synchronization of the receiver clock with the satellite time during the position determination, the remaining inaccuracy of the time still leads to an error of about 2 m in the position determination. Rounding and calculation errors of the receiver translate into approximately 1 m of error.

Multipath effect: The multipath effect is caused by reflection of satellite signals (radio waves) on objects. This effect mainly appears in the neighborhood of large buildings or other elevations. The reflected signal takes more time to reach the receiver than the direct signal. The resulting error typically lies in the range of a few meters.

Satellite orbits: Although the satellites are positioned in very precise orbits, slight shifts of the orbits are possible due to gravitation forces. The sun and the moon have a weak influence on the orbits. The orbit data are controlled and corrected regularly and are sent to the receivers in the package of ephemeris data.

Other spatial and geometry factors affect the accuracy of the GPS measurements. A typical civilian GPS receiver provides 6 to 12 meters accuracy, depending on the number of satellites available. This accuracy can be reduced to 1 m by using a differential GPS (DGPS) which employs a second receiver at a fixed location to compute corrections to the GPS satellite measurements.

The GPS measurements are called pseudo-ranges (instead of ranges) since the estimated times of transmission are corrupted by different biases. The positioning equations for n_s satellites in sight at time instant t can be defined as:

$$r_i^t = \sqrt{\left(X_i^t - x^t\right)^2 + \left(Y_i^t - y^t\right)^2 + (Z_i^t)^2} + b^t + w_i^t \tag{1.76}$$

for $i = 1, ..., n_s$, where r_i^t is the pseudo-range between the user and the ith satellite, $[X_i^t, Y_i^t, Z_i^t]^T$ is the position of the ith satellite, b^t is a bias term resulting from the clock offset, w_i^t is the measurement error and $[x^t, y^t]$ is the vehicle position to be estimated (the vehicle altitude is $z^t = 0$ in our application). These equations will be used as measurement equations in the proposed navigation solutions.

1.7.1.1 Wheel encoders

This section describes the main elements of differential odometry. The idea is to integrate information regarding distance and yaw rate using the measurements given by the vehicle odometers. Fig. 1.23 shows encoders located on the front or rear wheels. The first index of the different variables refers to the front f or rear r axes whereas the second index corresponds to the left l and right r sides of the car. Consequently, the wheel radii (resp. angular velocities) are denoted as R_{rl} ,R_{rr}, R_{fl} and R_{fr} (resp. w_{rl}, w_{rr}, w_{fl}, and w_{fr}). The other notations used in Fig. 1.23 are L for the length between wheels and γ_r for the vehicle yaw rate (change of angle of direction).

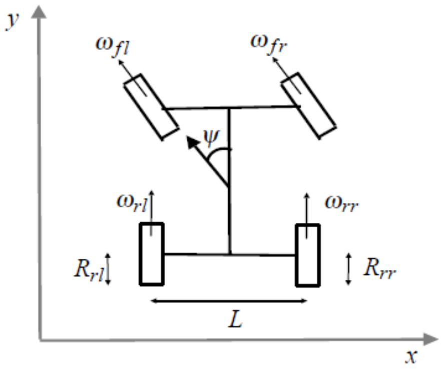


Figure 1.23: Illustration of the wheel encoders

The mean speed of the vehicle at time t can be computed as [207]:

$$V(w_{rr}^t, w_{rl}^t, R_{rr}, R_{rl}) = \frac{w_{rr}^t R_{rr} + w_{rl}^t R_{rl}}{2} \tag{1.77}$$

The yaw rate of the vehicle can be calculated as a function of the angular velocities of each wheel. By neglecting side slip effects and modeling the vehicle as a rigid body, the vehicle yaw rate at time t expresses as

$$\dot{\gamma}_r^t = h(w_{rr}^t, w_{rl}^t, R_{rr}, R_{rl}) = \frac{w_{rr}^t R_{rr} + w_{rl}^t R_{rl}}{L} \tag{1.78}$$

1.7.1.2 Inertial navigation System INS

The inertial navigation system (INS) is a self-contained navigation technique in which measurements provided by accelerometers and gyroscopes are used to track the position and orientation of an object relative to a known starting point, orientation and velocity. INS typically contain three orthogonal rate-gyroscopes and three orthogonal accelerometers, measuring angular velocity and linear acceleration respectively. By processing signals from these devices it is possible to track the position and orientation of a robot on which the INS device is mounted.

Inertial navigation systems usually can only provide an accurate solution for a short period of time. As the acceleration is integrated twice to obtain the position, any error in the acceleration measurement will also be integrated and causes a bias on the estimated velocity and a continuous drift on the position estimate by the INS. Additionally, the INS software must use an estimate of the angular position of the accelerometers when conducting this integration. Typically, the angular position is tracked through an integration of the angular rate from the gyro sensors. These also produce unknown biases that affect the integration to get the position of the unit.

The accelerometers deliver a non gravitational acceleration (also referred to as specific force f_p) and the gyrometers measure the rotation rate of the sensor cluster Ω_{ip} in order to keep track of the vehicle orientation.

The differential equations relating the measured quantities to the dynamics are defined as follows:

$$\dot{v}_{en} = R_{p2n} f_p + g_n - (\Omega_{en} + 2\Omega_{ie}) v_e - \Omega_{ie}^2 p_n \tag{1.79}$$

R_{a2b}: rotation matrix from frame a to frame b,
p_b: location of the vehicle in the frame b,
Ω_{ab}: rotation rate from frame a to frame b,
$v_a = [\dot{y}^t, \dot{x}^t]$: velocity relative to frame a

The subscripts refer to the different coordinate frames, i.e., i: inertial frame, e: earth centered earth fixed frame, n: local geographic frame, p: platform frame.

The velocity $\dot{x}_R$ of the vehicle in the G frame is given by:

$$\dot{x}_R = \begin{pmatrix} \dot{\lambda} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \frac{1}{R_\lambda} & 0 \\ 0 & \frac{1}{R_\phi cos\lambda} \end{pmatrix} \tag{1.80}$$

where g_n is the gravitational acceleration in the local geographic frame, λ and ϕ are the latitude and longitude of the mobile, R_λ is the earth radius of curvature in a meridian at a given latitude and R_ϕ is the transverse radius (we consider the WGS84 model for which the earth is an ellipsoid). These equations are integrated to obtain the vehicle position and velocity. This integration will entail a drift in stand-alone INS estimates due to a bias affecting the INS measurements (denoted as ba for the accelerometer bias and as bg for the gyroscope bias).

1.7.1.3 Integration GPS/INS/Encoders

Kalman filtering provides a powerful tool to create synergism between two navigation sensors - GPS receivers and INS - since it is able to take advantage of both systems' characteristics to provide a common, integrated navigation implementation with a performance superior to either of the sensor subsystems

The wheel encoders measures the Y-direction velocity in the vehicle frame, while two non-holonomic constraints are applied to the X and Z directions of the vehicle frame. The non-holonomic constraints imply that the vehicle does not move in the up or transverse directions, which holds in most cases. The Wheel encoders provides the absolute velocity information to update the centralized Kalman filter. During GPS outages, the non-holonomic constraints as well as the absolute velocity information can constrain the velocity and consequently the position drift of the free inertial system.

The error states estimated by the GPS/INS Extended Kalman filter include position errors, velocity errors, misalignment angles, as well as accelerometer and gyro biases. All of these error states are 3×1 vectors. Due to the centralized processing approach, the double differenced ambiguities ($\Delta\nabla N$) are also contained in the error states. The dynamic model for the GPS/INS Extended Kalman filter is given in equation 1.81 []

$$\begin{bmatrix} \delta\dot{r}^e \\ \delta\dot{v}^e \\ \dot{\epsilon}^e \\ \delta\dot{b}^b \\ \delta\dot{d}^b \\ \Delta\nabla N \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 \\ N^e & -2\Omega_{ie}^e & -F^e & R_b^e & 0 & 0 \\ 0 & 0 & -\Omega_{ie}^e & 0 & R_b^e & 0 \\ 0 & 0 & 0 & -diag(\alpha_i) & 0 & 0 \\ 0 & 0 & 0 & 0 & -diag(\beta_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} \delta r^e \\ \delta v^e \\ \epsilon^e \\ \delta b^b \\ \delta d^b \\ \Delta\nabla N \end{bmatrix}$$
$$+ \begin{pmatrix} 0 & 0 & 0 & 0 \\ R_b^e & 0 & 0 & 0 \\ 0 & R_b^e & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{pmatrix} . \begin{bmatrix} w_f \\ w_w \\ w_b \\ w_d \end{bmatrix} = F_{GPS/INS}.\delta x + G.w \tag{1.81}$$

where

δr^e	is the position error vector
δv^e	is the velocity error vector
ϵ^e	is the misalignment angle error vector
w_f	is the accelerometer noise
w_w	is the gyro noise
δb^b	is the vector of the accelerometer bias errors
δd^b	is the vector of the gyro bias errors
$diag(\alpha_i)$	is diagonal matrix of time constants for the accelerometer bias models
$diag(\beta_i)$	is diagonal matrix of time constants for the gyro bias models
w_b	is the driving noise for the accelerometer biases
w_d	is the driving noise for the gyro biases
R_b^e	is the direction cosine matrix between b frame and e frame
δx	is the error states vector, and
$F_{GPS/INS}$	is the dynamic matrix for GPS/INS integration strategy

To take into account that in practice the tire radius may change based on the load and driving conditions and that the INS body frame does not always coincide with the vehicle frame, the wheel encoders scale factor and the tilt angles between the b and v frames are modeled as random constants.

$$\begin{bmatrix} \delta\dot{r}^e \\ \delta\dot{v}^e \\ \dot{\epsilon}^e \\ \delta\dot{b}^b \\ \delta\dot{d}^b \\ \Delta\nabla N \\ \delta\dot{S} \\ \dot{\epsilon}_{b-v} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ N^e & -2\Omega_{ie}^e & -F^e & R_b^e & 0 & 0 & 0 & 0 \\ 0 & 0 & -\Omega_{ie}^e & 0 & R_b^e & 0 & 0 & 0 \\ 0 & 0 & 0 & -diag(\alpha_i) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -diag(\beta_i) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} \delta r^e \\ \delta v^e \\ \epsilon^e \\ \delta b^b \\ \delta d^b \\ \Delta\nabla N \\ \delta S \\ \epsilon_{b-v} \end{bmatrix}$$

$$+ \begin{pmatrix} 0 & 0 & 0 & 0 \\ R_b^e & 0 & 0 & 0 \\ 0 & R_b^e & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} . \begin{bmatrix} w_f \\ w_w \\ w_b \\ w_d \end{bmatrix} = F_{GPS/INS/WE}.\delta x + G.w \tag{1.82}$$

where $F_{GPS/INS/WE}$ is the dynamic matrix for GPS/INS/Wheel encoders integration strategy, δS is the Wheel encoders scale factor error state, and $\epsilon_{b-v} = [\delta\alpha \quad \delta\beta \quad \delta\gamma]^T$ is the error vector of the tilt angles between the body frame and the vehicle frame corresponding to the X, Y and Z axes respectively.

Since the wheel speed is measured in the vehicle frame, and the velocities in GPS/INS system are parameterized in ECEF frame, the Wheel encoders update can be either carried out in the e frame by transforming the Wheel encoders measurement into the e frame or carried out in the v frame by transforming the GPS/INS integrated velocities into the v frame. In our application, the Wheel encoders update is carried out in v frame, and the GPS/INS integrated velocities are transformed from e frame into v frame. The measurement equation is expressed in equation 1.83 with two non-holonomic constraints being applied into the X and Z axes of the vehicle frame.

$$\begin{bmatrix} 0 \\ S.v_{WE} \\ 0 \end{bmatrix} = R_b^v.(R_b^e)^T.v^e \tag{1.83}$$

where v_{WE} is the Wheel encoder measurement, S is the Wheel encoder scale factor, and R_b^v is the direction cosine matrix between the b and v frames calculated by the following:

$$R_b^v = R_3(\gamma).R_1(\alpha).R_2(\beta) \tag{1.84}$$

where α, β, γ are the tilt angles between the b and v frames with respect to the X, Y and Z axes, respectively. The measurement model in the extended Kalman filter is generally expressed by equation 1.85:

$$Z = H.\delta x + w_m \tag{1.85}$$

where H is the design matrix, w_m is the measurement noise and Z is the measurement residual.

By linearizing equation 1.82, the measurement residual is expressed by:

$$Z = \begin{bmatrix} 0 \\ S.v_{WE} \\ 0 \end{bmatrix} - R_b^v.(R_b^e)^T.v^e = \begin{bmatrix} 0 \\ S.v_{WE} \\ 0 \end{bmatrix} - v^R \tag{1.86}$$

where v^R is the integrated velocity expressed in the v frame.

The design matrix is expressed by a matrix in equation **??**

$$H = \begin{bmatrix} O_{3\times3} & R_b^v.(R_b^e)^T & R_b^v.(R_b^e)^T.V^E & O_{3\times3} & \cdots \\ O_{3\times3} & O_{AR\times AR} & -v_{WE} & V^V & \cdots \end{bmatrix} \tag{1.87}$$

where V^E is the skew symmetric matrix of the integrated velocity in ECEF frame v^e , V^V is the skew symmetric matrix of the integrated velocity expressed in vehicle frame v^v , 0 is a zero matrix with the subscripted dimensions and AR is the number of float ambiguities. AR is equal to zero when all the ambiguities are fixed.

1.7.2 GPS/INS/Encoders integration in the VSLAM process

In this section we propose to use the integrated sensors technique GPS/INS/Encoders on the VSLAM process for geo-localizing the obtained robot position and feature map from the VSLAM. This will increase the robustness of the positioning during the outage periods and in less featured environments.

In this case the the camera linear and angular velocities in the VSLAM process are updated based on the data from the Wheel encoders and INS sensors, respectively. Tow Extended Kalman Filter works in parallel mode. The estimated position of the VSLAM EKF is used to update the GPS/INS/Encoders integration EKF and the estimated velocities (linear and angular) from the GPS/INS/Encoders integration EKF is used to update the camera motion in the VSLAM EKF.

The proposed framework has been analyzed with different simulated GPS outages. Figure 1.24 illustrates an example where the GPS signal was lost during $30 seconds$. The black curve shows that the GPS localization error is $3.6 meters$. The dashed blue curve shows that even if the integration of GPS, INS, and wheels encoders data reduces the error on the robot position to less than 1m, it is not reliable during the GPS outage where the error grows continuously. While the proposed framework combining GPS/INS/Encoders localization, and visual SLAM localization, remains stable even during the GPS outage (red curve).

Figure 1.25 shows an example of the robot localization in a real environment. The blue curve in figure 1.25.a represents the obtained robot path using the proposed algorithm, and the red curve is the GPS data. The initial GPS position is the mean of the GPS measurements during few minutes of initialization. Figure 1.25.b represents the built local maps. Each local map is represented by a different color. In this experiment the maximum number of features in the local maps is fixed to 60 features.

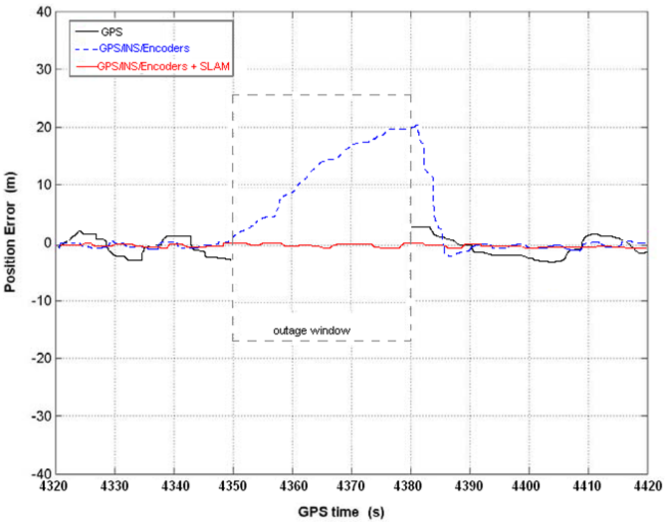
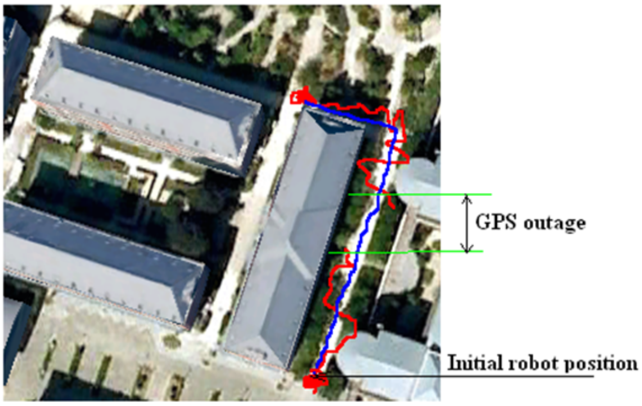
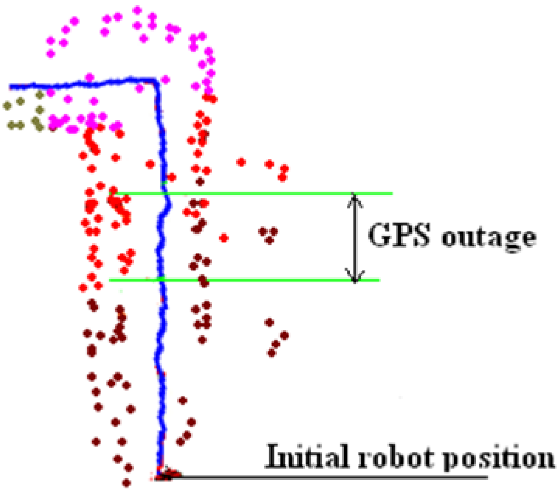


Figure 1.24: Robot Localization in case of GPS outage



a) Robot path superimposed to a geo referenced map



b) The built local maps

Figure 1.25: Robot localization in a real environment

1.8 Conclusion

In this chapter we introduced a feature based method for simultaneous localization and mapping in large environments for a mobile robot using a mono-camera as sensory input. The proposed approach builds several size limited local maps and combine them into a global map. The extended kalman filtering is used to build the local maps where the 3D position of the features is initialized by the epipolar geometry principle.

For a robust matching, the algorithm uses a product of three parameters: the Mahalanobis distance between measurements and their predictions, the Euclidean distance between the descriptor vectors of the features, and the distance of the feature to the induced epipolar line (epipolar constraint). This allows using the advantage of looking for feature matching based on the prediction of their position based on the system model and the advantage of the space-scale invariance parameters.

The chapter introduces also the use of other sensors (GPS, INS and encoders) data in the visual SLAM process. The proposed framework combines two Extended Kalman Filters (EKF), the first one, referred to as the integration filter, is dedicated to the improvement of the GPS localization based on data from an Inertial Navigation System (INS) and wheels' encoders. The second EKF implements the V-SLAM process. The linear and angular velocities in the dynamic model of the V-SLAM EKF filter are given by the GPS/INS/Encoders integration filter. In the other hand, the output of the V-SLAM EKF filter is used to update the dynamics estimation in the integration filter and therefore the geo-referenced localization. Tests in real environment demonstrated the increase of the accuracy and the robustness of the positioning during GPS outage and allows a SLAM in less featured environments.

Chapter 2

Motion Estimation and Segmentation

2.1 Introduction

The apparent motion field existing in an image sequence is a rich source of information about the composition of the viewed scene. In dynamic scene understanding, the recovering and interpretation of this motion field is of importance for a range of applications, such as moving object detection and tracking, image indexing, and compression.

Moving object detection or motion segmentation can be considered as a classification problem. Given a set of data points (the intensity values in each frame of the sequence) we want to find a clustering of these data points that best correspond to some properties of the scene. A segmentation is considered good if the computed clusters are homogeneous with respect to the feature vector, with the additional constraint that no two clusters can be similar (otherwise their pixels might be grouped together into the same class).

This work introduces a new method for simultaneously detecting the moving objects (foregrounds) and estimating their motion in image sequences taken with a moving observer. The algorithm combines a Gaussian Mixture Model (*GMM*) background subtraction approach and a Maximum a Posteriori Probability (*MAP*)- MRF framework. This enables us to exploit the simplicity and capability of the *GMM* approach to adapt to illumination changes and small motions in the scene, the advantages of spatio-temporal dependencies that moving objects impose on pixels, and the interdependence of motion and segmentation fields.

The algorithm starts with the estimation and the compensation of the camera motion between the previous and the current frames using method based on dense motion analysis. To deal with the problem of outliers in the estimation of the camera motion, we propose to use an initialization phase. This means that before starting moving the robot builds a model of the scene by detecting the static and the moving parts. Subsequently, during navigation, the initially detected moving parts are not considered for the camera motion estimation.

In a second step, an apparent scene cut or strong camera pan is detected by evaluating whether or not the squared difference between the current frame and the compensated frame from the camera motion, exceeds a given threshold. The evaluation is performed only within the background regions of the previous frame. In case of a scene cut between two consecutive frames, the segmentation algorithm is reset, i.e. all parameters are set to their initial values, as the camera motion in this case cannot be correctly compensated.

In a third step, the new compensated frame is then used to update the GMM model of the background, as in the case of a static camera. In this model, the pixel intensity is modeled by a mixture of Gaussian distributions to model variations in the background. The GMM parameters are learned from color observations in consecutive images.

In a fourth step, the background model, the current frame, the compensated frame from camera motion, and the previous segmentation (object mask) are used in a *maximum a posteriori probability* (*MAP*) framework to detect moving objects in the current frame and estimate their motion field.

In the last step, in order to avoid the common problem of fragmentation in background subtraction methods and accelerate the learning of the background model (new static regions), the background model is re-updated based on the results of the MAP-MRF optimization.

The first part of this chapter gives a literature overview of the most important methods for motion segmentation, and in the second part, the proposed approach is described in more details.

2.2 Motion Segmentation Methods - An overview

In the literature we can find different classification of the methods dealing with moving objects Segmentation. The classification could be based on the information used for segmentation (motion-based methods, spatiotemporal methods, 2D methods, 3D methods, change detection methods, optical flow based methods, joint estimation/segmentation methods), or it could be based on the sequences to be segmented (methods for sequences taken with a static camera, methods for sequences taken with a moving camera, methods for sequences with a static background, methods for sequences with a dynamic background).

In this chapter, we will describe the well known methods for motion segmentation based on the used information. Moreover, we describe how these methods can be used in the case of image sequences taken with a moving camera.

2.2.1 Motion segmentation based on change detection

Change detection is probably the oldest and the most popular technique in motion segmentation. It attempts to detect moving regions in a sequence frame by background subtraction or by differencing between two or three successive frames. The differentiation is made in a pixel-by-pixel fashion. The decision, whether a pixel $s \in S$ (S being the set of pixels indices) at spatial position $\mathbf{x} = (x, y)$ belongs to a changed or to an unchanged image part, is based on the evaluation of the frame difference ($d(f^t, f')$):

$$d(f_s^t, f_s') = \parallel f_s^t - f_s' \parallel \tag{2.1}$$

where f^t is the current frame and f' is the previous frame ($f' = f^{t-1}$) or a model of the background.

In order to distinguish between relevant changes due to object's motion or brightness changes, as well as irrelevant temporal changes due to noise, the frame difference is compared to a threshold T_{ch}. In the simplest case this threshold is tuned manually. A pixel $s \in S$ is considered as part of a changed region (foreground), if the frame difference exceeds the threshold.

Change detection through simple thresholding can lead to significant errors and inaccuracy in general situations. For this reason, change detection is usually inserted into a hierarchical or a relaxation algorithm [120, 128]. In this case, the initial change detection is refined by a motion compensated prediction/update or a threshold evaluation-update process (see Figure 2.1 for example [121]).

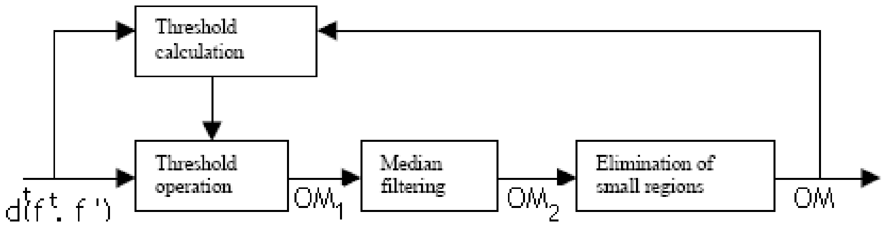


Figure 2.1: Change detector block diagram. ($d(f^t, f')$: frame difference, OM, OM_i: object masks)

In the first step, the threshold operation is performed using an initial value for the threshold. Then, a two dimensional median filter is used in order to smooth the boundaries of the changed regions. The last step is to eliminate small isolated regions of the change detection mask. After these three steps, the initial threshold is re-evaluated, it is adapted to the standard deviation of the noise based on the available unchanged regions. The process repeats with the new threshold until the system is stable. The result is the objects Mask (OM) (change detection mask).

The hard problem for motion segmentation methods based on change detection is the occlusion problem where two or more regions overlap in the images.

2.2.1.1 Temporal differencing

The approach of temporal differencing [120, 121, 127, 128, 200] makes use of pixel-wise difference between two or three consecutive frames in an image sequence to extract moving regions. Temporal differencing methods are adaptive to dynamic environments, but they detected only the boundaries of moving objects and not the objects themselves unless they are highly textured [130].

To void some false detections, an improved version based on three-frame difference (equation 2.2) instead of two-frame difference (equation 2.1) has been proposed in [200]:

$$d(f_s^t, f_s^{t-1}, f_s^{t+1}) = |f_s^t - f_s^{t-1}|.|f_s^t - f_s^{t+1}| \tag{2.2}$$

Where f^{t-1}, f^t, f^{t+1} are three consecutive frames in an image sequence with f^t the reference frame (current frame). In this case, there is difference only if there are differences among all the three frames, and foregrounds are detected also by thresholding the frame difference.

2.2.1.2 Background subtraction

Background subtraction is one of the most successful method for obtaining silhouettes of moving objects by comparing the sequence frames to a background model. The numerous approaches to this method differ in the type of the background model, the procedure used to update the background model, and in the subtraction and decision procedure. In the following we present the well known approaches for background subtraction and background modeling.

2.2.1.2.1 subtraction and decision

In this paragraph, we focus on the ways in which backgrounds are subtracted from image frames containing moving objects to produce the segmentation, l^t, at time t. All of the methods involve measuring some distance $d(f_s^t, Bg_s)$ between an incoming image sites (pixels), s, and the background model, and then thresholding the differences:

$$l_s^t = \begin{cases} 1, & \text{if } d(f_s^t, Bg_s) \geq Th(.); \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

where $Th(.)$ is a threshold function that is typically constant.
The simplest distance metric is the Euclidean distance [91, 136]

$$d(f_s^t, Bg_s) = \| f_s^t - \mu_s^t \| \tag{2.4}$$

with μ_s^t is the mean value of the background, Bg, at site s.

A generalized version of the Euclidean distance takes variance into consideration [99]

$$d(f_s^t, Bg_s) = \left\| \frac{f_s^t - \mu_s^t}{\Sigma_s} \right\| \tag{2.5}$$

Rather than finding some absolute distance away from a mean, this standardizes the distances across pixels with different variances from the background model

(as described in the next section).
A more general version of the standardized Euclidean distance is the Mahalanobis distance [115]

$$d(f_s^t, Bg_s) = \sqrt{(f_s^t - \mu_s^t)^T \Sigma_s^{-1} (f_s^t - \mu_s^t)} \tag{2.6}$$

with Σ_s is $n \times n$ variance matrix at site s in the background model (n is the dimension of the color space). This metric is more accurate, as it does not assume that the data is clustered spherically about the mean.

2.2.1.2.2 Background modeling

Background modeling is the most important part of any background subtraction algorithm. The goal is to construct and maintain a representation of the scene. The simplest background model is the temporally averaged image, a background approximation that is similar to the current static scene.

A good model for the background to be used for motion segmentation based on background subtraction must handel some of the inconsistencies due to:

- Moving shadows:
  They differ from the background image and are therefore identified as parts of the moving objects.

- Stationary objects:
  The background image can become corrupted by new stationary objects. For example a parked car in a street should be classified as a part of the background, otherwise it will continue to be detected as moving object.

- Non-stationary background:
  The background can contain moving parts, for example, in an outside scene, the wind can make tree branches move. This kind of background motion causes the color pixel intensity values to vary significantly with time. For example, one pixel can be part of a leaf at one frame and a part of the sky on the seconde.

- Camouflage:
  If some of the foreground pixels have same colors as the background, this may produce holes in the computed foreground object.

- Illumination change:
  For example, shadows from slowly moving clouds or other variable lighting conditions cause inclusion of background elements in the computed foreground. Also sudden changes in illumination alter the appearance of the background.

In order to reduce the influence of these problems on motion segmentation, several methods have been proposed to build an adaptive model for the background:

Accumulative Difference

The algorithm add a memory to the motion detection process to forme accumulative difference images [100]. This memory contains a counter for each pixel accounting the number of its consecutive classification as background. If this number exceeds a given threshold, the pixel is finally identified as background pixel.

Kalman Filter for background modeling

The Kalam filter has been used in several work to model the background [91, 136, 198, 199]. In this model each pixel is modeled by a Kalman filter to predict its next value. The background is updated according to the equation:

$$Bg_s^{t+1} = Bg_s^t + (\theta_1(1 - l_s^t) + \theta_2 l_s^t) d(f_s^t, Bg_s^t) \tag{2.7}$$

where Bg^{t+1} is the next predicted background, l^t is the current binary segmentation (foreground=1, background=0), and $d(f_s^t, Bg_s)$ is the difference between the current frame, f^t, and the currently predicted background, Bg^t:

$$d(f_s^t, Bg_s^t) = \|f_s^t - Bg_s^t\| \tag{2.8}$$

The learning rates, θ_1 and θ_2, are chosen (or learned) based on how quickly the background of the scene is changing. Typical values for θ_1 and θ_2 are 0.1 and 0.01, respectively [136].

By using equation (2.7), the background model is updated quickly when a pixel is labeled 'background' than when it is labeled 'foreground'. A value of $\theta_2 = 0$ implies that pixels labeled 'foreground' have no effect on the background model. While at first this may seem appropriate, it also implies that moving objects that come to rest will never be incorporated into the background model. But with $\theta_2 \gg 0$ the ghost foreground objects will have an adverse effect on background model.

A limitation of this model is that it considers a constant noise over all pixels which cannot model the local variations in pixel intensity caused by lighting effects.

Statistical modeling of the background

The principle in statistical background modeling for motion segmentation is deduced from a Bayesian decision for pixel classification as background (Bg) or foreground (Fg).

Given the current frame value, f_s^t, at site $s \in S$, the Bayesian decision R:

$$R_s = \frac{p(Bg_s \mid f_s^t)}{p(Fg_s \mid f_s^t)} = \frac{p(f_s^t \mid Bg_s)p(Bg_s)}{p(f_s^t \mid Fg_s)p(Fg_s)} \tag{2.9}$$

In generale case we don't know anything about the foreground objects that can be seen nor when and how often they will be present. Therefore, we set $p(Fg_s) = p(Bg_s)$ and assume uniform distribution for the foreground object appearance $p(f_s^t \mid Fg_s) = c_{Fg}$. Then we decide that the pixel belongs to the background if:

$$p(f_s^t \mid Bg_s) > c_{th}(= R.c_{Fg}) \tag{2.10}$$

where c_{th} is a threshold value. We will refer to $p(f_s^t \mid Bg_s)$ as the background model and is noted in the remainder of this document as $p(f_s^t)$.

The problem is then how to efficiently estimate the density function and to adapt it to possible changes as the change in illumination or a new stationary objects.

Several models have been proposed to solve the problem of estimating and adapting the background density function. In the following, two of the well known methods are presented: a parametric Gaussian Mixture model (GMM) and a non-parametric k Nearest Neighbors (k-NN) model.

- **Parametric Gaussian Mixture Model**

The basis of this model has been proposed by Stauffer and Grimson [99]. In this model [92, 94, 99, 100, 115, 147] each pixel is modelled as a mixture of Gaussians (equation 2.11) and an algorithm is used to figure out which pixel belongs to which normal distribution. The number of normal distributions (Gaussians), K, is normally set as a parameter from the beginning, however some algorithms also estimate this number [115, 201].

$$p(f_s^t) = \sum_{k=1}^{K} \omega_{s,k}^t \mathcal{N}(f_s^t, \mu_{s,k}^t, \Sigma_{s,k}^t) \tag{2.11}$$

where $\omega_{s,k}^t$'s (with $\sum_{k=1}^{K} \omega_{s,k}^t = 1$) denote the mixing weights of Gaussian distributions and define how the pervious pixel evidence (past frames) correlate with the associated distributions.
$\mathcal{N}(f_s^t, \mu_{s,k}^t, \Sigma_{s,k}^t)$ is the normal distribution of the component k and is given by:

$$\mathcal{N}(f_s^t, \mu_{s,k}^t, \Sigma_{s,k}^t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_{s,k}^t|}} e^{\left(-\frac{1}{2}(f_s^t - \mu_{s,k}^t)^T {\Sigma_{s,k}^t}^{-1} (f_s^t - \mu_{s,k}^t)\right)} \tag{2.12}$$

with $\mu_{s,1}^t, ..., \mu_{s,K}^t$ are the means and $\Sigma_{s,1}^t, ..., \Sigma_{s,K}^t$ are the covariances matrices that describe the Gaussian components at pixel s. n is the dimension of the color space (number of channels RGB, YUV,...). If the channels are considered independent, then $\Sigma_{s,k}^t = {\sigma_{s,k}^t}^2 I$ with I is the identity matrix.

Model update procedure

Each incoming pixel value f_s^t is checked against all the gaussians at that location. Component k is declared as the matched component if the observation is within $B\sigma_{s,k}^t$ from its mean $\mu_{s,k}^t$.
For Stauffer [99], the comparison is done by comparing the gaussian probability value. A match with component $\hat{k}$ is found if:

$$p(f_s^t) > p(\mu_{s,\hat{k}}^t + B\sigma_{s,\hat{k}}^t) \tag{2.13}$$

Instead of doing the comparison by an expensive evaluation of the density function to verify the ownerships of the measurement f_s^t to the components, Gordon et Al.

[94] use the Euclidian distance

$$d_s^t = \left(f_s^t - \mu_{s,\hat{k}}^t\right)^2 \tag{2.14}$$

to do the comparison. A match with the component $\hat{k}$ is found if:

$$d_s^t < B\sigma_{s,\hat{k}}^t \tag{2.15}$$

In [115], it is demonstrated that the parameter B can be approximated by $B = \sqrt{-2ln(\sqrt{2\pi}\sigma c_{thr})}$.

When a matching is found using 2.13 or 2.15, the parameters of the matched component $\hat{k}$ are updated as follows:

$$\begin{aligned} \omega_{s,\hat{k}}^{t+1} &= (1-\alpha)\omega_{s,\hat{k}}^t + \alpha \\ \mu_{s,\hat{k}}^{t+1} &= (1-\rho)\mu_{s,\hat{k}}^t + \rho f_s^t \\ (\sigma_{s,\hat{k}}^{t+1})^2 &= (1-\rho)(\sigma_{s,\hat{k}}^t)^2 + \rho\left(f_s^t - \mu_{s,\hat{k}}^t\right)^2 \end{aligned} \tag{2.16}$$

Where α is a user-defined learning rate with $0 \leq \alpha \leq 1$. This parameter incorporates exponential forgetting which enables to control how long past pixel values influences the model of the background. The higher the value of α, the faster the past background model is forgotten and vice versa. In [160], a time-varying gain is used $\alpha_t = 1/t$ with a lower bound α_{min}.
The learning rate for the parameters ρ is computed in [99] by $\alpha N(f_s^t, \mu_{s,\hat{k}}^t, \sigma_{s,\hat{k}}^t)$ and in [160] it is approximated as follows by $\rho \approx \alpha/\omega_{s,\hat{k}}^t$.

If no matched component can be found, the component with the least weight is replaced by a new component with mean equal to the new data, f_s^t, a large initial variance, σ_0, and a small weight, ω_0. In [161], ω_0 is equal to α and the component with the least weight is discarded only if the maximum number of components is reached. The rest of the components maintain the same means and variances, but lower their weights to achieve exponential decay:

$$\omega_{s,k}^{t+1} = (1-\alpha)\omega_{s,k}^t \tag{2.17}$$

Finally, all the weights are re-normalized to sum up to one.

Pixels Classification

To determine whether the pixel s is a background or a foreground pixel, given its value f_s^t, we first order all components by their values of $(\omega_{s,k}^t/\sigma_{s,k}^t)$. Higher-rank components thus have low variances and high probabilities, which are typical characteristics of background. If $i_1, i_2, ..., i_K$ are the component order after sorting, the first M components that satisfy the following criterion are declared to be the background components:

$$\sum_{k=i_1}^{i_M} \omega_{s,k}^t \geq Th \tag{2.18}$$

where Th is the weight threshold. It represents a measure of the minimum portion of the data that should be accounted for by the background. This takes the "best" distributions until a certain portion, Th, of the recent data has been accounted for. If a small value for Th is chosen, the background model is usually unimodal. If this is the case, using only the most probable distribution will save processing.

If Th is higher, a multimodal distribution caused by a repetitive background motion (e.g., leaves on a tree) could result in more than one color being included in the background model. This results in a transparency effect which allows the background to accept two or more separate colors [99].

s is declared as a background pixel if f_s^t is within B times the standard deviation from the mean of any one of the background components.

From this model we see that if a new object comes into the scene and remains static for long time, its weight will be constantly increasing and becomes larger than the threshold and therefore it can be considered to be a part of the background. The object should be static for approximately $log(Th)/log(1-\alpha)$ frames [161]. For example for $Th = 0.9$ (for a multimodel learning) and $\alpha = 0.001$ (to keep trace of the history) we get 105 frames.

Number of components

One of the improvement proposed for this method for background modeling using a Mixture of Gaussians is the work of Zivkovic [201], in which not only the parameters but also the number of components of the mixture is constantly adapted for each pixel. In this model, at each time a component is generated or deleted to update the number of components and choose compact models for the data. The generation and delation rules are inspired by the works in [202] and [203], respectively.

The generation rule simply adds a new component whenever there is a new sample that is not well described by the current distribution, i.e. no matching is found as defined above. When a new component k is generated for the pixel s, it is initialized by:

$$w_{s,k}^t = \alpha, \mu_{s,k}^t = f_s^t, \sigma_{s,k}^t = \sigma_0$$

where σ_0, in [201], is computed by:

$$\sigma_0 = 0.2\frac{med}{0.68\sqrt{2}}$$

with med is the median of differences $\| f^t - f^{t-1} \|$ for all pixels between two successive frames and for a number of pairs of images. In practice, in [201] the maximal number of components is fixe.

The deletion rule is added to discard the obsolete components. A component k is deleted if its weight becomes negative $w_{s,k} < 0$. This also ensures that the weights stay positive.

- **Non-parametric k Nearest Neighbors (k-NN) model**

These methods [114, 159], initially founded by El Gammel & Al. [114], show some trade off in using mixture model with adaptive scheme: if the background model adapts too slowly to changes in the scene, then we will construct a very wide and inaccurate model that will have low detection sensitivity. On the other hand, if the model adapts too quickly, this will lead to two problems: the model may adapt to the targets themselves, as their speed cannot be neglected with respect to the background variations, and it leads to inaccurate estimation of the model parameters.

To overcome these problems, a model which keeps a sample for each pixel of the scene and estimates the probability that a newly observed pixel value is from the background has been proposed is [114]. The model estimates these probabilities independently for each new frame as follows.

Let $f_s^1, f_s^2, ..., f_s^N$ be a recent sample of intensity values for a pixel at site s. Using this sample, the probability density function that this pixel will have intensity value f_s^t at time t can be non-parametrically estimated using the kernel estimator $\mathcal{K}$ as

$$p(f_s^t) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{K}(f_s^t - f_s^i) \tag{2.19}$$

If a Normal function $\mathcal{N}(0, \Sigma)$ is used for the kernel estimator function $\mathcal{K}$, where Σ represents the kernel function bandwidth, then the density can be estimated as

$$p(f_s^t) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^n \mid \Sigma \mid}} e^{-\frac{1}{2}\left(f_s^t - f_s^i\right)^T \Sigma^{-1}\left(f_s^t - f_s^i\right)} \tag{2.20}$$

with n is the dimension of the color space.

If we assume independence between the different color channels with a different kernel bandwidths σ_j^2 for the j^{th} color channel, then

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}$$

and the density estimation is reduced to

$$p(f_s^t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{\left(f_{s,j}^t - f_{s,j}^i\right)^2}{2\sigma_j^2}} \tag{2.21}$$

where $f_{s,j}^t$ is the jth color channel for pixel s at time t.

Using this probability estimate, the pixel s is considered a foreground pixel if $p(f_s^t) < Th$ where the threshold Th is a global threshold over all the image that can be adjusted to achieve a desired percentage of false positives.

Density estimation using a normal kernel function is a generalization of the Gaussian mixture model, where each single sample of the N samples is considered to be a Gaussian distribution $\mathcal{N}(0, \Sigma)$ by itself. This allows the estimation of the density function more accurately and depending only on recent information from the sequence. This also enables the model to quickly "forget" about the past and concentrate more on recent observation. At the same time, it avoids the inevitable errors in parameter estimation, which typically require large amounts of data to be both accurate and unbiased.

The N samples used to model the background needs to be updated continuously to adapt to changes in the scene. The update is performed in a first-in first-out manner. That is, the oldest sample/pair is discarded and a new sample/pair is added to the model. The new sample is chosen randomly from each time interval of length N frames.

Given a new pixel sample, there are two alternative mechanisms to update the background model:

1. Selective Update: add the new sample to the model only if it is classified as a background sample.

2. Blind Update: just add the new sample to the model.

There are tradeoffs to these two approaches. The first enhance detection of the targets, since target pixels are not added to the model. This involves an update decision: we have to decide if each pixel value belongs to the background or not. The simplest way to do this is to use the detection result as an update decision. The problem with this approach is that any incorrect detection decision will result in persistent incorrect detection later, which is a deadlock situations. So, for example, if a tree branch might be displaced and stayed fixed in the new location for a long time, it would be continually detected.

The second approach does not suffer from this deadlock situation since it does not involves any update decisions; it allows intensity values that do not belong to the background to be added to the model. This leads to bad detection of the targets (more false negatives) as they erroneously become part of the model. This effect is reduced by increasing the time window over which samples are taken, as a smaller proportion of target pixels will be included in the sample. But as we increase the time window more false positives will occur because the adaptation to changes is slower and rare events are not as well represented in the sample.

2.2.2 Motion segmentation based on motion field

These methods [102, 107, 148] perform segmentation based on the estimated displacement or optical flow of image pixels.

The optical flow describes how the image is changing with time. The estimation of the optical flow is itself a very active research topic [103, 112]. Ideally, the projection into the two dimensional image plane of the three dimensional velocity field seen by the camera should be computed. However, this is not just difficult to achieve in practice, it is usually impossible to achieve perfectly even in theory [103].

Once the dense optical flow field is obtained, motion segmentation algorithms try to extract regions with homogeneous motions. The principles of these approaches can be summarized as follows: Suppose we have M sets of parameters vectors, where each set defines a correspondence or a flow vector at each pixel. Flow vectors defined by the mapping parameters are called model-based or synthesized flow vectors. Thus, we have M synthesized flow vectors at each pixel.

The segmentation procedure then assigns the label of the synthesized vector which is closest to the estimated flow vectors at each site. However, there is a problem with this simple scheme: both the number of classes, M, and the mapping parameters for each class are not known a priori. Assuming a particular value for M, the mapping parameters for each class could be computed in the least squares sense provided that the estimated optical flow vectors associated with the perspective classes are known. That is, we need to know the mapping parameters to find the segmentation labels, and the segmentation labels are needed to find the mapping parameters. This suggests an iterative procedure for clustering where both the segmentation labels and the class means are unknown as in the K-means approach proposed by Wand and Adelson [104], and the MAP technique of Murray and Buxton [107] (see section 2.2.3).

Wand and Adelson [104] proposed a layered video representation in which the image sequence is decomposed into layers by means of an optical-flow-based image segmentation, and ordered in depth along with associated maps defining their motions, opacities, and intensities. In the layered representation, the segmentation labels denote the layer in which a particular pixel resides. The segmentation method is based on the affine motion model and clustering in a six-dimensional parameter space. The image is initially divided into small blocks. Given an optical flow field, a set of affine parameters are estimated for each block. To determine the reliability of the parameters estimates, the sum of squares distances between the synthesized, $\tilde{\mathbf{u}}$, and estimated, $\mathbf{u}$, flow vectors is computed as:

$$\epsilon = \sum_{s \in \mathcal{B}} \| \mathbf{u}_s - \tilde{\mathbf{u}}_s \|^2$$

where $\mathcal{B}$ refers to a block of pixels. Obviously, if the flow within the block complies with an affine model, the residual will be small. On the other hand, if the block falls on the boundary between two distinc motions, the residual will be large. The motion parameters for blocks with acceptably small residuals are selected as candidate layer models. to determine the appropriate number M of layers, the motion parameters of the candidate layers are clustered in the six-dimensional parameter space. The initial set of affine model parameters are set equal to the mean of the M clusters. Then the segmentation label of each pixel site is selected as the index of the parameter set that yields the closest optical flow vector at that site. After all sites are labeled, the affine parameters of each layer are recalculated based on the new segmentation labels. This procedure is repeated until the segmentation labels no longer change or a fixed number of iterations is reached.

A limitation of this segmentation method is that it lacks constraints to enforce spatial and temporal continuity of the segmentation label.

2.2.3 Motion segmentation based on Markov random fields

Markov random fields (*MRF*) define an efficient and powerful framework for specifying nonlinear interactions between features of the same nature or of a different one. They help to combine and organize spatial and temporal information by introducing strong generic knowledge about the features to be estimated. MRF models have been successfully introduced in many fundamental issues of image analysis and computer vision such as image restoration [182], edge detection [183], image segmentation [184, 185], surface reconstruction [186], motion analysis [162, 163, 166, 167, 187], or scene interpretation [188].

We will start with a short introduction to Markov Random Fileds and Gibbs distribution then we describe some of the well known methods using MRF for motion segmentation.

2.2.3.1 MRF labeling - Theoretical background

Contextual constraints are ultimately necessary in the interpretation of visual information. A scene is understood through the spatial and visual context of the objects in it; the objects are recognized in the context of object features at a lower level representation; the object features are identified based on the context primitives at an even lower level; and the primitives are extracted in the context of image pixels at the lowest level of abstraction. The use of contextual constraints is indispensable for a complex vision system.

Markov random field theory provides a convenient and consistent way of modeling context dependent entities. This is achieved through characterizing mutual influences among such entities using MRF probabilities. The theory tells us how to model the *a priori* probability of contextual dependent patterns. A particular MRF model favors its own class of patterns by associating them with larger probabilities than other pattern classes. In the following, we will briefly review the concept of MRF defined on graphs [190].

Let $\mathbf{G}$=$\{\mathbf{S}$,$\mathbf{A}\}$ be a graph, where $\mathbf{S}$=$\{s_1, s_2, ..., s_m\}$ is the set of nodes and $\mathbf{A}$ is the set of arcs connecting them. We define a *neighbourhood system* on $\mathbf{G}$, denoted by:

$$\mathbf{N} = \{\mathbf{N}(s_1), \mathbf{N}(s_2), ..., \mathbf{N}(s_m)\}$$

where $\mathbf{N}(s_i), i = 1, 2, ..., m$ is the set of all nodes in $\mathbf{S}$ that are neighbors of S_i, such that:
i) $s_i \notin \mathbf{N}(s_i)$, and
ii) if $s_j \in \mathbf{N}(s_i)$ then $s_i \in \mathbf{N}(s_j)$
Let $\mathbf{L}$=$\{L_1, L_2, ..., L_m\}$ be a family of random variables defined on $\mathbf{S}$, in which each random variable L_i takes a value l_i in a given set Λ (the random variables L_i's can be numerical as well as symbolic, e.g. interpretation labels). The family $\mathbf{L}$ is called a random field. $\mathbf{L}$ is a MRF on $\mathbf{G}$, with respect to the neighbourhood system $\mathbf{N}$ if and only if

1. $p(\mathbf{L} = l) > 0$, for all realizations l of $\mathbf{L}$;
2. $p(l_i|l_j, \forall s_j \neq s_i) = p(l_i|l_j, s_j \in \mathbf{N}(s_i))$

where $p(\mathbf{L} = l) = p(L_1 = l_1, L_2 = l_2, ..., L_m = l_m)$ (abbreviated by $p(l)$) and $p(l_i|l_j)$ are the joint and conditional probability functions, respectively. Intuitively, the MRF is a random field with the property that the statistics at a particular node depend mainly on that of its neighbors.

An important feature of the MRF model defined above is that its joint p.d.f has a general functional form, known as *Gibbs distribution*, which is defined based on the concept of *cliques*. A clique c, associated with the graph $\mathbf{G}$, is a subset of $\mathbf{S}$ such that it contains either a single node, or several nodes that are all neighbors of each other. If we denote the collection of all the cliques of $\mathbf{G}$, with respect to the neighbourhood system $\mathbf{N}$, as $\mathbf{C(G,N)}$, the general form of a realization of $p(l)$ can be expressed by the following Gibbs distribution:

$$p(l) = \frac{1}{Z} e^{-U(l)} \tag{2.22}$$

where $U(l) = \sum_{c \in \mathbf{C}} V_c(l)$ is called the *Gibbs energy function* and $V_c(l)$ the *clique potential functions* defined on the corresponding cliques $c \in \mathbf{C}(\mathbf{G}, \mathbf{N})$. Finally, $Z = \sum_{l \in \mathbf{L}} e^{-U(l)}$ is a normalizing constant called the *partition function.*

In case of a *labeling* problem, when we have both prior information together with knowledge about the distribution of our data, the most optimal labeling of the graph $\mathbf{G}$ can be obtained based on the maximum *a posteriori* probability (MAP)-MRF framework. According to the Bayes rule, the posterior probability of our system can be computed by using the following formulation:

$$p(l|\mathfrak{g}) = \frac{p(\mathfrak{g}|l)p(l)}{p(\mathfrak{g})} \tag{2.23}$$

where $p(l)$ is the prior probability of labelings l, $p(\mathfrak{g}|l)$ is the conditional probability distribution function (p.d.f.) of the observations $\mathfrak{g}$, also called the *likelihood function* of l for $\mathfrak{g}$ fixed, and $p(\mathfrak{g})$ is the density of $\mathfrak{g}$ which is a constant when $\mathfrak{g}$ is given.

By associating an energy function to $p(\mathfrak{g}|l)$ and $p(l)$ (denoted by $U(\mathfrak{g}|l)$ and $U(l)$ respectively), we can express the posterior probability as:

$$p(l|\mathfrak{g}) \propto e^{-U(l|\mathfrak{g})} \tag{2.24}$$

where

$$U(l|\mathfrak{g}) = U(\mathfrak{g}|l) + U(l) \tag{2.25}$$

The most optimal labeling, given the observation field $\mathfrak{g}$, can be found by minimizing the energy function $U(l|\mathfrak{g})$.

Due to its unique property of combining both global and local information, the MRF model-based approach, applied to image processing, provides potential advantages in knowledge representation, learning and optimization. This is the reason why we use this approach.

Several methods have used Markov Random Filed for motion segmentation. These methods can be grouped into two categories: pixel based methods and region based methods. The work of Wang et al. [176] will be discussed separately as it includes in the same framework pixel based and region based processing.

2.2.3.2 Pixel based MRF methods for motion segmentation

In the algorithms described in [162, 163, 164, 165, 166], the Markov Random Field has been introduced to model the spatial and/or the temporal dependency between pixels during the motion segmentation process. The MRF modeling in these works is summarized in following paragraphs.

2.2.3.2.1 Murray's method

In [107] Murray and Buxton search for the maximum of the a *posteriori* probability of the motion segmentation label fields, L, given the optical flow data. It is a measure of how well the current segmentation l^t explains the observed optical flow data $\mathbf{u}^t$ and how well it conforms to the prior expectations.

Using the Bayes theorem, the a *posteriori* probability is expressed as:

$$p(L^t = l^t|\mathcal{U} = \mathbf{u}^t) = p(l^t|\mathbf{u}^t) = \frac{p(\mathbf{u}^t|l^t)p(l^t)}{p(\mathbf{u}^t)} \tag{2.26}$$

where L^t and $\mathcal{U}^t$ are the segmentation labels and the motion fields and $u^t = \{u_s^t, s \in S\}$ and $l^t = \{l_s^t, s \in S\}$ are their realizations, respectively, with S being the set of sites (pixels) indices. $p(\mathbf{u}^t|l^t)$ is the conditional pdf of the optical flow, $\mathbf{u}^t$, given the segmentation, l^t, and $p(l^t)$ is the a *priori* pdf of the segmentation. $p(\mathbf{u}^t)$ is constant with respect to the segmentation labels, and hence can be ignored during MAP optimization.

The conditional probability $p(\mathbf{u}^t|l^t)$ is a measure of how well the synthesized optical flow, $\tilde{\mathbf{u}}$, modeled by a quadratic parameter model that depends on the segmentation label, fits the estimated optical flow, $\mathbf{u}$. Assuming that the mismatch between the observed flow and the synthesized flow is modeled by a white Gaussian noise with zero mean and variance σ^2, the conditional pdf $p(\mathbf{u}^t|l^t)$ is expressed as:

$$p(\mathbf{u}^t|l^t) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\sum_{i=1}^{m}(\epsilon_{s_i}^t)^2/2\sigma^2\right) \tag{2.27}$$

where m is the number of flow vectors available at site s_i, and

$$(\epsilon_{s_i}^t)^2 = \| \mathbf{u}_{s_i}^t - \tilde{\mathbf{u}}_{s_i}^t \|^2$$

The prior pdf $p(l^t)$ is modeled by a Gibbs distribution which introduces local constraints on the segmentation:

$$p(l^t) = \frac{1}{Q} \sum_{\zeta \in \Omega} \exp\left(-U(l)\right) \delta(l^t - \zeta) \tag{2.28}$$

where Ω denotes the discrete sample space of l, Q is the partition function

$$Q = \sum_{\zeta \in \Omega} \exp\left(-U(l)\right)$$

and $U(l)$ is a potential function expressed as a sum of local cliques $V_c(l_s^t, l_r^t)$:

$$U(l) = \sum V_c(l_s^t, l_r^t)$$

$$V_c(l_s^t, l_r^t) = \begin{cases} -\beta, & l_s^t = l_r^t; \\ +\beta, & \text{otherwise}; \end{cases}$$

with β is positive number.

Because the model parameters corresponding to each label are not known a *priori*, the MAP optimization alternates between estimation of the model parameters and assignment of the segmentation labels and it is performed by the simulated annealing algorithm [205].

Note that this method is limited by the accuracy of the available optical flow estimates.

2.2.3.2.2 Migdal's method

Based on the Gaussian Mixture Model of Stauffer and Grimson [99], Migdal et al. [162] used three Markov random fields to improve the pixels classification as background (Bg) and foreground (Fg) by modeling the spatial and temporal dependencies between adjacent pixels. The first MRF M_1 is used to encode spatial relationship and the second MRF M_2 is used for temporal segmentation continuity by looking at past segmentation. The third MRF M_3 exploits a full temporal constraint using a batch computation model.

The segmentation at each time t, l^t, is obtained by estimating the maximum a posteriori (MAP):

$$p(l^t \mid f^t) = \frac{p(f^t \mid l^t)p(l^t)}{p(f^t)} \tag{2.29}$$

As the optimization is done over segmentation variate:

$$p(l^t \mid f^t) \propto p(f^t \mid l^t).p(l^t)$$

The prior, $p(l^t)$, and the likelihood function, $p(\mathfrak{g}^t \mid l^t)$, are modeled by Gibbs distributions and therefor the maximization of the MAP is equivalent to the minimization of the sum of their corresponding energy functions .

$$U = U(f^t \mid l^t) + U(l^t)$$

Likelihood term

$$U(f^t \mid l^t) = \sum_s V(f_s^t \mid l_s^t)$$

$$V(f_s^t \mid l_s^t) = \begin{cases} d(f_s^t), & l_s^t = 0; \\ \ln 2^{24}, & l_s^t = 1; \end{cases}$$

with $d(f_s^t) = \frac{\|\mathfrak{g}_s^t - \mu\|}{2\sigma^2}$ is the Mahalanobis distance between the current observation f_s^t and the matched Gaussian model and μ and σ are, respectively, the mean and the variance of this model.

As no model is used for the foreground, the probability of seeing a pixel s with color f_s, given that the segmentation of that pixel has been determined to be foreground is assumed to be a uniform distribution in the RGB space and is therefore equal to $\frac{1}{2^{24}} = e^{-\ln 2^{24}}$.

Prior term

The energy function of the prior is composed of two terms, a spatial, U_S, and a temporal terms, U_T.

- Spatial term

$$U_S(l^t) = \sum_{s,r} V_S(l_s^t, l_r^t)$$

$$V_S(l_s^t, l_r^t) = \begin{cases} \psi_1, & l_s^t = l_r^t = 1; \\ \psi_2, & l_s^t = l_r^t = 0; \\ \psi_3, & l_s^t \neq l_r^t; \end{cases}$$

where V_S is a two-clique potential function and ψ_1, ψ_2 and ψ_3 are constants with $(\psi_1 < \psi_2 < \psi_3)$.

- Temporal term

$$U_T(l^t) = \sum_{s} V_T(l_s^t, l_s^{t-1})$$

$$V_T(l_s^t, l_s^{t-1}) = \begin{cases} \theta_1, & l_s^t = l_s^{t-1} = Bg; \\ \theta_2, & l_s^t = l_s^{t-1} = Fg; \\ \theta_3, & l_s^t \neq l_s^{t-1}; \end{cases}$$

where V_T is a two-clique potential function and θ_1, θ_2 and θ_3 are constants with $(\theta_1 < \theta_2 < \theta_3)$, i.e. more emphasis is done for foreground classification.

2.2.3.2.3 Zhou's method

In this work [163] also a MRF is used to model the spatial-temporal connectivity between pixels augmented with connectivity of pixels in different resolutions for classification foreground/background. The GMM is used also in this case to maintain the background model.

Image pyramids are built to model the phenomena of different temporal-spatial resolution scales. Each node in the graph represents the state of a pixel at certain resolution scale. The spatial-temporal dependencies are modeled using the links between the nodes. The classification foreground/background exploits the inference within and between different spatial-temporal resolution scales using a Markov random field representation.

Spatial term

The potential function representing the connection between two adjacent pixels in the same resolution scale is defined mathematically as follows:

$$V_S(l_s^t, l_r^t) = \begin{cases} c_1, & l_s^t \neq l_r^t; \\ 0, & l_s^t = l_r^t; \end{cases}$$

s and r are adjacent pixels in the same resolution scale.

V_S has a positive value when there exists discontinuity in foreground label image and is 0 otherwise. So, this connection is used to penalize the discontinuity in foreground blob extraction.

Scaling-Spatial term

The connectivity between pixels at multiple resolution scales is represented by the following potential function:

$$V_{SS}(l_s^t, l_r^t) = \begin{cases} c_2, & l_s^t \neq l_r^t; \\ 0, & l_s^t = l_r^t; \end{cases}$$

l_s^t, l_r^t are corresponding pixels segmentation at adjacent resolution scales in the pyramid.

V_{SS} has a positive value when the corresponding labeling at two adjacent resolution levels do not agree.

Temporal term

The temporal cotinuity of the segmentation process is modeled with the following function

$$V_T(l_s^t, l_s^{t-1}) = \begin{cases} c_2, & l_s^t \neq l_s^{t-1}; \\ 0, & l_s^t = l_s^{t-1}; \end{cases}$$

The authors introduced another musure to represent the probability of new object appearing and disappearing.
The optimization is based on Gibbs sampling with the simulated annealing algorithm.

2.2.3.3 Region Based MRF Methods for motion segmentation

These methods [167, 168, 169, 170, 171] start with segmenting the current frame and estimating the motion vector on each region and then apply MRF process to merge and classify regions as moving or not moving. In the following paragraphs we will describe some of the well known methods using MRF at region level for motion segmentation. For these methods, a site corresponds to region.

2.2.3.3.1 Cucchiara and Gelgon Methods

In the methods proposed by Cucchiara et al. [170] and Gelgon and Bouthemy [171], each frame of the image sequence is segmented into regions using a region growing algorithm based on color [174] and a Region Adjacency Graph (RAG) is created. The motion vector in the regions is estimated through partitioned region matching. A measure of motion reliability is also computed and a MRF framework is used to merge regions according to the motion parameters (velocity and reliability). The largest region is assumed to be the background and its motion is considered as the motion of the camera, while other regions are classified as moving objects. [170] and [171] differ in the method used for region motion estimation and the algorithm used MRF optimization.

The joint probability distribution of the motion segmentation field and the observation field is modeled by a Gibbs distribution with the following energy functions:

$$U(l^t, \mathfrak{g}^t) = U_1(l^t, \mathfrak{g}^t) + U_2(l^t) + U_3(l^t) \tag{2.30}$$

The observation field, $\mathfrak{g}^t$, is composed of the region motion vector and the reliability of the motion estimation:

$$\mathfrak{g}_s^t = \{\mathbf{u}_s^t, \varphi_s^t\}, \qquad s \in S$$

with S is the set of region indices.

Motion term

This term penalizes the segmentation if the same label has been affected to neighbors regions with different motion vectors.

$$U_1(l^t, \mathfrak{g}^t) = \sum_{s,r \in S} V_1(l_s^t, \mathfrak{g}_s^t, l_r^t, \mathfrak{g}_r^t)$$

$$V_1(l_s^t, \mathfrak{g}_s^t, l_r^t, \mathfrak{g}_r^t) = \begin{cases} c_1 \parallel \mathbf{u}_s^t - \mathbf{u}_r^t \parallel \sqrt{\rho_s^t.\rho_r^t}, & l_s^t = l_r^t \\ 0, & l_s^t \neq l_r^t \end{cases}$$

Geometrical regularization term

This term enhances the fusion of two adjacent regions with a long shared border and a small distance between centroids.

$$U_2(l^t) = \sum_{s,r \in S} V_2(l_s^t, l_r^t)$$

$$V_2(l_s^t, l_r^t) = \begin{cases} -c_2 \frac{\zeta_{s,r}}{\zeta_{s,r} + \sqrt{(\mathbf{x}_s - \mathbf{x}_r)^2}}, & l_s^t = l_r^t \\ 0, & l_s^t \neq l_r^t \end{cases}$$

with $\zeta_{s,r}$ is the length of the shared border between sites s and r, and $\mathbf{x}_s$ and $\mathbf{x}_r$ are the coordinates of their centers.

labeling optimization term

$$U_3(l) = c_3. \sum_{s,r \in S} V_3(l_s, l_r)$$

$$V_3(l_s, l_r) = \begin{cases} 0, & l_s^t = l_r^t; \\ 1, & l_s^t \neq l_r^t. \end{cases}$$

c_1, c_2, c_2 are constants defined empirically.

Optimization

In the work of Cucchiara the optimization of the energy function is performed with a multi-scale iterative algorithm. Whereas, in the work of Gelgon and Bouthemy the optimization is done using the Highest Confidence First (HCF) algorithm [175].

2.2.3.3.2 Patras' Method

In [167], Patras applies the watershed segmentation algorithm, proposed by [172], to obtain an oversegmentation on the intensity image of the current frame. After building the Region Adjacency Graph RAG from the intensity segments, with the assumption that each watershed segment has a correspondence in at least two consecutive frames (a correspondence either in the next or in the previous frame), the author apply the Markov Random Field to detect the independently moving objects. The number of these regions is considered as known and provided as user-specified parameter. The motion field induced by the regions is approximated by 6-parameter affine models and estimated jointly with the MRF classification procedure.

The energy function for the Gibbsian model of the joint probability function distribution of the motion segmentation field and the observation is given by:

$$U(l^t, \mathfrak{g}^t) = U_1(l^t, \mathfrak{g}^t) + U_2(l^t, \mathfrak{g}^t) + U_3(l^t) \qquad (2.31)$$

In this case the observation is composed of the previous estimated segmentation, l^{t-1}, the 6-parameter affine motion model at each site (region), θ_s, the previous, f^{t-1}, the current, f^t, and the next, f^{t+1}, frames in the sequence:

$$\mathfrak{g}_s^t = \{l_s^{t-1}, \theta_s^t, f^t, f^{t-1}, f^{t+1}\} \qquad s \in S$$

with S is the set of region indices.

Motion term

This term therm express how well the current motion model and label field conform with the image intensities.

$$U_1(l^t, \mathfrak{g}^t) = \sum_{s \in S} V_1(l^t_s, \mathfrak{g}^t_s)$$

$$V_1(l^t_s, \mathfrak{g}^t_s) = \min\left(\sum_{i \in R_s}\left(d_i\left(f^t, \tilde{f}^{t-1}\right)\right)^2, \sum_{i \in R_s}\left(d_i\left(f^t, \tilde{f}^{t+1}\right)\right)^2\right)$$

$d_i(f^t, \tilde{f}^{t+1})$ and $d_i(f^t, \tilde{f}^{t-1})$ are, respectively, the forward and backward motion compensated intensity differences at pixels i of region s:

$$d_i(f^t, \tilde{f}^{t+1}) = f^t(\mathbf{x}_s) - f^{t+1}(\mathbf{x}_s + \mathbf{u}_s(\theta_s))$$
$$d_i(f^t, \tilde{f}^{t-1}) = f^t(\mathbf{x}_s) - f^{t-1}(\mathbf{x}_s - \mathbf{u}_s(\theta_s))$$

where $\mathbf{x}_s$ denotes the spatial coordinates of a pixel at site (region) s and $\mathbf{u}_s(\theta_s)$ is the optical flow at site s based on the affine parameters θ_s.

Temporal term

$$U_2(l^t, \mathfrak{g}^t) = \sum_{s \in S} V_2(l^t_s, \mathfrak{g}^t_s)$$
$$V_2(l^t_s, \mathfrak{g}^t_s) = c_1 Q_s$$

where Q_s is the number of pixels of region R_s whose motion-based projections in the previous frame have a label different than l^t_s. The smaller the Q_s, the higher the probability that R_s has the label l^t_s.
c_1 is a constant that controls the temporal consistency of the label field.

Spatial term

$$U_3(l^t) = \sum_{s,r \in S} V_3(l^t_s, l^t_r)$$

$$V_3(l^t_s, l^t_r) = \begin{cases} -c_2.\zeta_{s,r}, & l^t_s = l^t_r; \\ c_2.\zeta_{s,r}, & l^t_s = l^t_r. \end{cases}$$

with $\zeta_{s,r}$ is the length of the shared border between neighbor sites s and r, and c_2 is a constant defined empirically.

Optimization with respect to the spatial energy term $U_3(l)$ tends to minimize the total border length between neighboring objects.

The optimization process of the total energy function iterates between the following phases:

$$\begin{array}{ll} \text{Labeling phase:} & l^{t,k+1} = \arg\min_{l} U(l^t, \theta^{t,k}, l^{t-1}, f^t, f^{t-1}, f^{t+1}) \\ \text{Motion estimation phase:} & \theta^{t,k+1} = \arg\min_{\theta} U(l^{t,k}, \theta^t, l^{t-1}, f^t, f^{t-1}, f^{t+1}) \end{array}$$

where k denotes the iteration index.

In the labeling phase, the minimization of the energy function is done with respect to l^t, keeping the motion parameters θ^t "frozen", using the Iterative Conditional Modes (ICM). In the motion estimation phase, the minimization of the energy function is done with respect to the motion parameters θ^t, keeping the label field l^t "frozen".

2.2.3.3.3 Tsaig Method

As in the previous method, the algorithm proposed by Tsaig and Averbuch [168, 169] start with color-based watershed segmentation of the current frame and spatio-temporal merging (post-processing step) of the small regions. The motion field for the obtained regions is estimated using a robust gradient-based parametric technique [173]. Then it uses MRF process for graph labeling over the Region Adjacency Graph. In this case only two regions are considered, foreground and background.

The energy function for the Gibbsian model of the joint probability function distribution of the motion segmentation field and the observation is given by:

$$U(l^t, \mathfrak{g}^t) = U_1(l^t, \mathfrak{g}^t) + U_2(l^t, \mathfrak{g}^t) + U_3(l^t, \mathfrak{g}^t) \tag{2.32}$$

The measurement correspond to the Motion Vector of the region, $\mathbf{u}_s$, the sum of the mean values of the color components within the region, A_s, and a memory of the number of times the region was classified as foreground in the past:

$$\mathfrak{g}_s^t = \{\mathbf{u}_s^t, A_s^t, M_s^t\}, \qquad s \in S$$

with S is the set of region indices.

Motion term

this term states that moving regions should be classified as foreground, whereas static regions should be classified as background. The magnitude of the motion vector is not taken into consideration.

$$U_1(l^t, \mathfrak{g}^t) = \sum_{s \in S} V_1(l_s^t, \mathfrak{g}_s^t)$$

$$V_1(l_s^t, \mathfrak{g}_s^t) = \begin{cases} -c_1 Q_s^t, & (l_s^t = Fg, \mathbf{u}_s^t \neq 0) or (l_s^t = Bg, \mathbf{u}_s^t = 0); \\ c_1 Q_s^t, & (l_s^t = Fg, \mathbf{u}_s^t = 0) or (l_s^t = Bg, \mathbf{u}_s^t \neq 0). \end{cases}$$

With Q_s^t is the number of pixels in region s at time t.

Temporal continuity term

This term is used to maintain the coherency of the segmentation through time.

$$U_2(l^t, \mathfrak{g}^t) = \sum_{s \in S} V_2(l_s^t, \mathfrak{g}_s^t)$$

$$V_2(l_s^t, \mathfrak{g}_s^t) = \begin{cases} -c_2 M_s^t Q_s^t, & l_s^t = Fg; \\ c_2 M_s^t Q_s^t, & l_s^t = Bg. \end{cases}$$

Spatial continuity term

This term expresses the relationships between pairs of regions. A similarity measure is used in order to incorporate the spatial properties of the regions into the optimization process. Specifically, two regions with similar spatial properties are likely to belong to the same moving object, thus more weight is given to an equal labeling for two regions.

$$U_3(l^t, \mathfrak{g}^t) = \sum_{s,r \in S} V_3(l_s^t, \mathfrak{g}_s^t, l_r^t, \mathfrak{g}_r^t)$$

$$V_3(l_s^t, \mathfrak{g}_s^t, l_r^t, \mathfrak{g}_r^t) = \begin{cases} -\gamma_1 \hbar(A_s^t - A_r^t)\zeta_{s,r}, & l_s^t = l_r^t = Fg; \\ \gamma_2 \hbar(A_s^t - A_r^t)\zeta_{s,r}, & l_s^t = l_r^t = Bg; \\ \gamma_3 \hbar(A_s^t - A_r^t)\zeta_{s,r}, & l_s^t \neq l_r^t; \end{cases}$$

With $\hbar(d)$ is the similarity function shown in figure 2.2
where the constants T_l, T_h, d_l and d_h determine the effect of neighboring regions

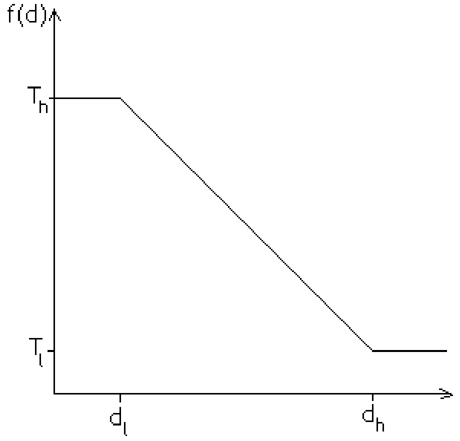


Figure 2.2: The similarity function $\hbar(d)$

on one another, and $\zeta_{s,r}$ is the length of the shared border between neighbor sites s and r.

The constants $c_1, c_2, \gamma_1, \gamma_2$, and γ_3 determine the relative contributions of the three terms to the energy functions and they are defined from experience.

Optimization
The minimization of the energy function is performed using HCF (iterative deterministic relaxation scheme)[87].

2.2.3.4 Joint intensity and spatio-temporal segmentation

Although the above algorithms can identify successfully multiple moving objects in the scene, the boundaries are inaccurate. The boundary information neglected by the initial intensity segmentation field could no longer be recovered by the motion vector field, and the temporal information could not act on the spatial information.

To deal with this problem, wang et al. [176] proposed an algorithm where the intensity segmentation, motion estimation and the spatio-temporal segmentation are performed jointly in the same framework.

The method tends to maximize a posterior MAP estimation of the displacement vector field U, the intensity segmentation field Z, and the spatio-temporal segmentation field L. The realization at time t of U, Z, and L are denoted by $\mathbf{u}^t, z^t$, and l^t, respectively.

$$(\hat{\mathbf{u}}^t, \hat{z}^t, \hat{l}^t) = \arg \max_{(\mathbf{u}_t, z^t, l^t)} p(\mathbf{u}^t, z^t, l^t \mid f^t, f^{t-1}, f^{t+1}) \tag{2.33}$$

where $p(\mathbf{u}^t, z^t, l^t \mid f^t, f^{t-1}, f^{t+1})$ is the posterior probability density function (pdf) given the three video frames.

$$p(\mathbf{u}^t, z^t, l^t \mid f^t, f^{t-1}, f^{t+1}) = \frac{p(\mathbf{u}^t, z^t, l^t, f^t, f^{t-1}, f^{t+1})}{p(f^t, f^{t-1}, f^{t+1})} \tag{2.34}$$

The nominator is constant with respect to the unknowns and

$$p(\mathbf{u}^t, z^t, l^t, f^t, f^{t-1}, f^{t+1}) = p(f^{t-1}, f^{t+1} \mid f^t, \mathbf{u}^t) p(f^t \mid z^t) p(z^t) p(\mathbf{u}^t \mid l^t) p(l^t \mid z^t) \tag{2.35}$$

Motion term
The conditional probability density function $p(f^{t-1}, f^{t+1} \mid f^t, u^t)$ qualifies how well the motion estimation fits the given frames. This likelihood is modeled by a Gibbs distribution with the energy function:

$$U_1(l \mid \mathbf{u}^t) = \sum_{s \in S} \Big((d(f_s^t, \tilde{f}_s^{t-1}))^2 - 2\beta d(f_s^t, \tilde{f}_s^{t-1}) d(f_s^t, \tilde{f}_s^{t+1}) + (d(f_s^t, \tilde{f}_s^{t+1}))^2 \Big)$$

with S is the set of pixels indices.
$d(f_s^t, \tilde{f}_s^{t-1}) = f^t(\mathbf{x}_s) - f^{t-1}(\mathbf{x}_s - \mathbf{u}_s^t)$ and $d(f_s^t, \tilde{f}_s^{t+1}) = f^t(\mathbf{x}_s) - f^{t+1}(\mathbf{x}_s + \mathbf{u}_s^t)$ are the backward and forward displaced frame difference at site s, respectively.
β is a correlation coefficient of $d(f_s^t, \tilde{f}_s^{t-1})$ and $d(f_s^t, \tilde{f}_s^{t+1})$, and $\mathbf{x}_s$ is the spatial coordinates of the site s.

Spatial term

The term $p(f^t \mid z^t)$ shows how well the intensity segmentation fits the scene. Assuming Gaussian distribution for each segmented region in the frame, the conditional probability density is expressed as Gibbs distribution with the energy function:

$$U_2(f^t \mid z^t) = c_1 \sum_{s \in S} (f_s^t - \mu_{z_s^t})^2$$

where $z_s^t = j$ designates the assignment of site s to region j, μ_j is the mean of the intensity within region j.

The intensity segmentation is also constrained with the a priori probability $p(z^t)$ which is modeled by a Markov Random Filed with the Gibbs energy function:

$$U_3(z^t) = c_2 \sum_{s \in S} \sum_{r \in N_s} \frac{1}{\| \mathbf{x}_s - \mathbf{x}_r \|^2} (1 - \delta(z_s^t - z_r^t))$$

where $\delta(.)$ is the Kronecker delta function, $\| \,.\, \|$ denotes the Euclidian distance, $\mathbf{x}_s$ is the spatial position of the site s, and N_s is the set on neighbors of s.
Thus two neighboring pixels are more likely to belong to the same class than to different classes. The constraint becomes strong with the decrease of the distance between the neighboring sites.

Temporal term

The term $p(\mathbf{u}^t \mid l^t)$ is the conditional pdf of the displacement field given the video segmentation field. As Gibbs distribution, the energy function for $p(\mathbf{u}^t \mid l^t)$ is:

$$U_4(\mathbf{u}^t \mid l^t) = c_3 \sum_{s \in S} \sum_{r \in N_s} \frac{1}{\| \mathbf{x}_s - \mathbf{x}_r \|^2} \delta(l_s^t - l_r^t) \| \mathbf{u}_s^t - \mathbf{u}_r^t \|^2$$

The smoothness constraint of the motion vectors is imposed only when the two pixels have the same spatio-temporal segmentation label.

Spatio-Temporal term

The last term $p(l^t \mid z^t)$ represents the probability density of the spatio-temporal segmentation field when the intensity segmentation field is given. The energy function for a representation by a Gibbs distribution of $p(l^t \mid z^t)$ is the following:

$$U_5(l^t \mid z^t) = c_4 \sum_{s \in S} \sum_{r \in N_s} \frac{1}{\| \mathbf{x}_s - \mathbf{x}_r \|^2} \left(1 - \delta(l_s^t - l_r^t)\right) \left(1 + \beta g(D_s^t, D_r^t)\right)$$

$$g(D_s^t, D_r^t) = \begin{cases} 1, & D_s^t < D_r^t; \\ 0, & \text{otherwise} \end{cases}$$

where N_s is the set on neighbors of s and D_s^t is the distance between the pixel (site) s and the nearest boundary pixel in z^t.

The first term in the potential function encourages the spatial connectivity, while the second term gives a penalty on the pixel closer to the boundary of the intensity segmentation field if the two pixels are not of the same video segmentation class. The parameter β controls the strength of the constraint imposed by the intensity segmentation field.

The contributions of the motion, spatial, temporal and saptio-temporal terms are controlled with parameters c_1, c_2, c_3, and c_4 defined empirically.

Optimization

the minimization is done using the iterated conditional modes (ICM) algorithm in two steps. Firstly, $\mathbf{u}^t$ and z^t are updated given the estimate of the spatio-temporal segmentation field l^t as $\mathbf{u}^t$ and z^t are conditionally independent and secondly, the spatio-temporal segmentation l^t is updated assuming the motion field $\mathbf{u}^t$ and the intensity segmentation field z^t are given.

2.2.4 Motion segmentation with a moving observer

Motion segmentation in the case of a moving camera is much more complicate because we need to separate the motion due to the camera (egomotion) from the motion due to the objects. As the observer moves, the images it collects representing the world change, even if everything in the field of view is static, i.e., nothing else is moving. From this sequence of images the observer can infer information relating to its own motion, as well as the depth structure of the environment in witch it is moving and this can be used after by the observer for path planning and obstacle avoidance. The non static objects in the environment of the observer are referred to as 'moving independently'.

Motion segmentation methods in the case of a moving camera can be classified in two groups: layered motion approaches and hierarchical approaches.

2.2.4.1 Layered Motion Methods

These methods for motion segmentation [104, 109, 113] are much suitable for video coding. They try to group scattered regions with same motion into a single model, called layer. More gain can be obtained by representing a motion layer with a single set of motion parameters than by representing each region with an individual set of parameters. By grouping the motions in the scene into different motion layers, more efficient coding can be achieved through a hierarchical manner. Besides, segmentation for coding purpose needs a more accurate motion boundary to avoid the prediction artifact. For this purpose, the algorithms focus on finding global motion homogeneity instead of local motion homogeneity. The idea is to estimate the dominant motion over the entire image, warp the images according to the dominant motion, extract the regions that do not agree with the dominant motion, and for each region estimate the dominant motion again over that region and extract the moving objects in the region by warping and differencing.

2.2.4.2 Hierarchical methods

In these approaches the dominant motion over the entire image is recovered. Assuming that most of the image is occupied by a static scene, the motion of most pixels in the image is due to the egomotion, this is the dominant motion. The idea is to align the scene by compensating out the global motion, then apply motion segmentation methods (described in sections 2.2.1 and 2.2.3) as in the case of a static camera [96, 120, 126, 136, 142].

In [96], the pixels in each image are shifted to compensate for camera rotation. The optical flow is computed with a gradient-based technique and the optical flow vectors with small magnitudes are discarded. Then the vectors with similar locations, magnitudes, and directions are clustered together using a spatial consistency test.

In [142], The apparent motion induced by the camera motion is represented by a 2D parametric motion model, and compensated for using the values of the motion model parameters estimated by a multi-resolution robust statistical technique. Then, regions whose motion cannot be described by this global model estimated over the entire image, are extracted. The detection of these non conforming regions is achieved through a statistical regularization approach based on multiscale Markov random field (MRF) models.

In the following, we will present the well known methods for camera motion estimation and compensation.

2.2.4.3 Methods for camera motion estimation and compensation

Motion compensation in an image sequence is the warping of the global image changes due to the camera motion. Before a sequence of images can be motion compensated efficiently, the motion of the camera have to be estimated.

The camera motion is generally represented by a parametric motion model where the spacial coordinates, $\mathbf{x'}_s = (x'_s, y'_s)^T$, of a pixel (site) s ($s \in S$) in the current frame, f^t, are function of their corresponding location, $\mathbf{x}_s = (x_s, y_s)^T$, in the pervious frame, f^{t-1}. Table 2.1 gives a summary of the mostly used motion models.

Parameters $\mathbf{a} = (a_1, ..., a_n), n \in \{2, 3, 6, 8, 17\}$ in the motion models are obtained by minimizing a disparity measure between a region R in the current frame and the mapped region in the previous frame.

$$\min_{\theta} \sum_{s \in R} \parallel f^t(\mathbf{x}_s) - f^t(\Theta(\mathbf{x}_s)) \parallel \tag{2.36}$$

Model / Number of Parameters	formula	Physical meaning
Translation / 2	$x'_s = x_s + a_1$ $y'_s = y_s + a_2$	2D translation
Euclidean / 3	$x'_s = x_s \cos\beta - y_s \sin\beta + a_1$ $y'_s = x_s \sin\beta - y_s \cos\beta + a_2$	2D translation + rotation
Affine / 6	$x'_s = a_1 x_s + a_2 y_s + a_3$ $y'_s = a_4 x_s + a_5 y_s + a_6$	above + skew
Pseudo Projective/ 8	$x'_s = a_1 x_s + a_2 y_s + a_3 + a_7 x_s y_s + a_8 x_s^2$ $y'_s = a_4 x_s + a_5 y_s + a_6 + a_8 x_s y_s + a_7 x_s^2$	(none)
Homography / 8	$x'_s = (a_1 x_s + a_2 y_s + a_3)/(a_7 x_s + a_8 y_s + 1)$ $y'_s = (a_4 x_s + a_5 y_s + a_6)/(a_7 x_s + a_8 y_s + 1)$	motion of a 3D planar surface
Q-warping/ 17	$x'_s = \frac{a_1 x_s + a_2 y_s + a_3 + a_4 x_s y_s + a_5 x_s^2 + a_6 y_s^2 + a_7 x_s^2 y_s + a_8 x_s y_s^2 + a_9 x_s^3}{a_{16} x_s + a_{17} y_s + 1}$ $y'_s = \frac{a_{10} x_s + a_{11} y_s + a_{12} + a_{13} x_s y_s + a_{14} x_s^2 + a_{15} y_s^2 + a_9 x_s^2 y_s + a_7 x_s y_s^2 + a_8 x_s^3}{a_{16} x_s + a_{17} y_s + 1}$	motion of a 3D quadratic surface

Table 2.1: motion models

where $\| \, . \, \|$ is the distance measure and $\Theta(\mathbf{x}_s)$ is the transformation applied to the coordinates of site s and is given by one of the models in table 2.1.

The widely used methods to solve this minimization problem are robust regression methods like random sample consensus (RANSAC) [51], the least median of squares (LMedS) regression [189] and the least trimmed squares (LTS) regression [101].

RANSAC selects P number of samples out of M measurements where P is the number of parameters of the model to estimate. Given a priori knowledge of the maximum measurement error, a number of points are selected from a the region R in the previous frame. If at least a minimum number of points are within the expected measurement error (inlier points), the estimate is accepted. Finally, a least squares fit to the selected points is computed and this is the estimate produced by RANSAC. RANSAC is guaranteed to succeed if the minimum number of accepted points (inlier points) is greater than $(M + P + 1)/2$. LMS proceeds identically except it does not use an expected error to classify selected points. Instead all possible P sets are evaluated and the estimate with least median of squared residual is selected. All points with a squared residual smaller than or equal to the least median of squared residual are considered good points. The final least squares step and the number of inlier points necessary for LMS to succeed are identical to RANSAC.

Finally, LTS is an improvement over LMS. In LMS the evaluation of a trial set P depends on the errors in the data points P, i.e., the median of squared residuals of a data set may have been considerably lower if the initial estimate would have been improved by a least squares step. Therefore, LTS improves each subset of "good" points by improving the fit with repeated least squares [101]. The least squares iteration has converged when the set of good points has become stable. LTS finds the good points set with the minimum least squares residual. The LTS process is robust even if $(M - P - 1)/2$ measurements is not respected.

Methods for camera motion estimation can be grouped in two categories: methods based on tracking particular features and methods based on dense motion analysis.

2.2.4.3.1 Camera motion estimation based on features tracking

As said above, if two successive frames in a video sequence taken from a moving camera can be related, it will be possible to extract information regarding the depth of objects in the environment and the speed of the camera. But comparing every pixel in the two images is computationally prohibitive. Intuitively, one can imagine relating two images by matching only locations in the image that are in some way interesting. Such locations are referred to as interest points or features and are located using an features detector. Finding a relationship between images is then performed using only these points. This reduces the required computation time.

Many different feature detectors have been proposed with a wide range of definitions for what points in an image are interesting. Some detectors find points of high local symmetry, others find areas of highly varying texture, while others locate corner points. Some of the well known methods for feature points detection are describe in appendix B.

The camera motion is estimated by tracking the detected feature points in multiple views. The mostly used algorithm for feature tracking is the one using Lukas-Kanade optical flow based algorithm [196]. The tracker takes as input the coordinates of the feature points to track in the first image and at each iteration, it takes a new image and returns the location of these feature points in that image using information from the previous image.

The Lukas-Kanade algorithm is an optical flow based technique which relies on the assumption that brightness of every point of a moving or a static objects does not change in time.

Let's consider a point at spatial position $\mathbf{x} = (x, y)$ undergoes a small displacement in time dt, such that its new position is given by $\mathbf{x}' = (x + dx, y + dy)$. Using Taylor series for brightness gives the following:

$$f(x+dx, y+dy, t+dt) = f(x, y, t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt + ... \tag{2.37}$$

where "..." are higher order terms. $\frac{\partial f}{\partial x} = f_x$, $\frac{\partial f}{\partial y} = f_y$ and $\frac{\partial f}{\partial t} = f_t$ are the derivatives of the image at (x, y, t) respectively.
$f(x, y, t)$ is the frame brightness at spatial position $\mathbf{x} = (x, y)$ and time t.
According to the brightness assumption, $f(x + dx, y + dy, t + dt) = f(x, y, t)$:

$$\frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt + ... = 0 \tag{2.38}$$

Neglecting the higher order terms and dividing the equation by dt we obtain the optical flow equation:

$$f_x\frac{dx}{dt} + f_y\frac{dy}{dt} + f_t = 0 \tag{2.39}$$

where dx/dt and dy/dt are the optical flow components in the x and y directions respectively:

$$\mathbf{u} = (u_x, u_y) = (\frac{dx}{dt}, \frac{dy}{dt})$$

The equation (2.39) alone does not suffice for the computation of the flow components and additional constraints are required. Several approaches have been proposed to obtain additional constraints for the solution of the optical flow constraint equation. The solution as given by Lucas and Kanade is a non-iterative method which assumes a locally constant flow.

Assuming that the flow $\mathbf{u} = (u_x, u_y)$ is constant in a small window of size $m \times m$ with $m > 1$, which is centered at pixel x, y and numbering the pixels within as $1...n$, $n = m3$, a set of equations can be found:

$$\begin{aligned} f_{x_1}u_x + f_{y_1}u_y &= -f_{t_1} \\ f_{x_2}u_x + f_{y_2}u_y &= -f_{t_2} \\ &\vdots \\ f_{x_n}u_x + f_{y_n}u_y &= -f_{t_n} \end{aligned} \tag{2.40}$$

which can be written as

$$\begin{bmatrix} f_{x_1} & f_{y_1} \\ f_{x_2} & f_{y_2} \\ \vdots & \vdots \\ f_{x_n} & f_{y_n} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} -f_{t_1} \\ -f_{t_2} \\ \vdots \\ -f_{t_n} \end{bmatrix}$$

or

$$\mathbf{A}\vec{\mathbf{u}} = -\mathbf{b} \tag{2.41}$$

With this there are more than two equations for the two unknowns and thus the system is over-determined. To solve the over-determined system of equations, the least squares method is used in the LucasKanade optical flow estimation:

$$\begin{aligned} \mathbf{A}^T\mathbf{A}\vec{\mathbf{u}} &= \mathbf{A}^T(-\mathbf{b}) \\ \vec{\mathbf{u}} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(-\mathbf{b}) \end{aligned}$$

or

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i}f_{y_i} \\ \sum_i f_{x_i}f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix} \begin{bmatrix} -\sum_i f_{x_i}f_{t_i} \\ -\sum_i f_{x_i}f_{t_i} \end{bmatrix}$$

A modified version of Lucas & Kanade approach which has been used in our implementations is given in appendix B.2.

2.2.4.3.1.1 Camera motion estimation

Once the correspondence (f^{t-1}, f^t) is known, the ego-motion of the camera or the mapping between the images (f^{t-1}, f^t), can be estimated using a transformation model and an optimization method.

The mapping between a reference image f^{t-1} and a target image f^t can be expressed as:

$$f^t(\mathbf{x}_s) = g(f^{t-1}(T(\mathbf{x}_s))) \tag{2.42}$$

where T is a coordinate transformation and g is an intensity transformation.

If we assume that no intensity changes are present, i.e. $g(\mathbf{x}_s) = \mathbf{x}_s$, the mapping equation can be simplified to:

$$f^t(\mathbf{x}_s) = f^{t-1}(T(\mathbf{x}_s)) \tag{2.43}$$

This equation gives us a mapping between the points in the two images, namely:

$$\mathbf{x}_r = T(\mathbf{x}_s) \tag{2.44}$$

where $\mathbf{x}_s$ is the spatial position of the point s in the image f^{t-1} and $\mathbf{x}_r$ is the spatial position of its corresponding point in f^t.

The model parameters are estimated ,then, by minimizing using following squared error:

$$\epsilon = \frac{1}{2} \sum_{i=1}^{N} (\mathbf{x}_r - T(\mathbf{x}_s))^2 \tag{2.45}$$

where N is the number of features (corners).

The most popular approach for this robust estimation problem is the RANSAC (RANdom SAmple Consensus) algorithm. The idea is to repeatedly guess a set of model parameters using small subsets of correspondences that are drawn randomly from the input set of correspondences $\mathcal{C}$. With a large number of draws, there is a high probability to draw a subset of correspondences that are part of the good motion model. After each subset draw, the motion parameters for this subset are determined and the number of correspondences in $\mathcal{C}$ that are consistent with these parameters is counted. The set of model parameters with the largest support is considered to be the camera motion model.

- **Remove outliers features**

To delete the outliers features, i.e. the features associated with moving objects, two methods have been proposed. In [151] the model parameter estimation is performed in two steps:

- Compute the initial estimate of the camera motion model T_0 using the full feature set.
- discard the features with squared error exceeding a given error and recompute the final estimate of the camera motion using only remaining feature set.

In this method, it is assumed for outliers detection that the portion of moving objects in the images is relatively small compared to the background.

In [120], the considered features are only those within background regions of the previous frame. In case of the first frame the background regions are not known and therefor, pixels which distance from the left or right border is less than 10 pixels are used as observations points, assuming that there is no moving object near the left and right image border.

2.2.4.3.2 Camera motion estimation based on dense motion

The well known of these methods is the one proposed by Dufaux and Konrad [173]. The algorithm is hierarchical and consists of three stages. In the first stage, a low-pass image pyramid is built. Then, an initial translation is estimated with full pixel precision at the top of the pyramid using a modified n-step search matching. In the third stage, a gradient descent is executed at each level of the pyramid starting from the initial translation at the coarsest level.

The technique is designed to minimize the sum of the squared differences (SSD) between the current frame f and the motion compensated previous frame f'

$$E = \sum_{s=1}^{N} \epsilon_s^2, \qquad \text{with} \quad \epsilon_s = (f'_{s'} - f_s) \tag{2.46}$$

where f_s and $f'_{s'}$ denote the intensity of image f at pixel s and the intensity of f' at the corresponding pixel s' ($s' = s + d$, where d is the displacement).

Summation is carried out over N pairs of pixels (s) and (s') within image boundaries.

A perspective (eight-parameter) model is used for the camera motion

$$x'_{s'} = \frac{a_0 + a_2 x_s + a_3 y_s}{a_6 x_s + a_7 y_s + 1}$$

$$y'_{s'} = \frac{a_1 + a_3 x_s + a_5 y_s}{a_6 x_s + a_7 y_s + 1}$$

where (x_s, y_s) and $(x'_{s'}, y'_{s'})$ are spatial coordinates of pixels s and s' in images f and f', respectively. The motion parameters $(a_0, ..., a_7)$ are computed using a gradient descent method.

This model is suitable when the scene can be approximated by a planar surface, or when the scene is static and the camera motion is a pure rotation around its optical center.

In order to improve convergence and reduce computational complexity, a low-pass image pyramid is used; the gradient descent is applied at the top of the pyramid and then iterated at each level until convergence is achieved. To assure convergence in the presence of large displacements, the gradient descent should start within a convergence "basin" of the global minimum of E. To achieve this, an initial, coarse estimate of the translation component of the displacement at the top level of the pyramid is computed by applying an n-step search matching algorithm proposed by Koga et al. [177].

The motion parameters $\mathbf{a} = (a_0, ..., a_7)$ are estimated using an iterative procedure:

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \mathbf{H}^{-1} b \tag{2.47}$$

where
$\mathbf{a}^t$ and $\mathbf{a}^{t+1}$ are parameters set at iteration t and $t+1$, respectively;
$\mathbf{H}$ is $n \times n$ matrix equal to one-half times the Hessian matrix of E;
b is a n-element vector equal to minus one-half times the gradient of E;
n is the number of parameters of the model (8 in this case).

The coefficients of the matrix H and the vector b are given by:

$$H_{kl} = \frac{1}{2}\sum_{s=1}^{N} \frac{\partial^2 \epsilon_s^2}{\partial a_k \partial a_l} \cong \sum_{s=1}^{N} \frac{\partial \epsilon_s}{\partial a_k}\frac{\partial \epsilon_s}{\partial a_l}$$

and

$$b_k = -\frac{1}{2}\sum_{i=1}^{N} \frac{\partial \epsilon_s^2}{\partial a_k} = -\sum_{s=1}^{N} \epsilon_s \frac{\partial e_i}{\partial a_k}$$

The gradient descent starts at the top level of the pyramid, then follows at the subsequent levels in a top-down approach. At each level, the gradient descent is iterated until a suitable convergence criterion. N_{max} iterations are carried out, but the process may stop earlier if the update term is smaller than a preset threshold. The projection of the motion parameters from one level onto the next one consists merely of multiplying a_0 and a_1 by 2, and dividing a_6 and a_7 by two. Finally, the procedure stops at the base level of the pyramid where the final motion parameters are obtained.

2.2.4.3.3 Image warping After camera motion estimation, the frames are warped or in an other word aligned to compensate out the global motion.

The general meaning of the term "image warping" refers to the process of geometrically transforming two-dimensional images. This transformation can be a simple scaling or rotation and it can be a complex and irregular mapping.

An image warp is defined by a mapping from the coordinate space, (x, y), of a source image, f, to the coordinate space, (x', y'), of a destination image, f'. If the destination coordinates are specified as a function of the source coordinates, the mapping is called a forward warping.

$$(x', y') = \mathcal{T}(x, y) \tag{2.48}$$

$\mathcal{T}$ is the mapping function.

If the source coordinates are specified as a function of the destination coordinates, the mapping is called an inverse warping.

$$(x, y) = \mathcal{T}(x', y') \tag{2.49}$$

A warp described by a forward mapping is performed by scanning the source image pixel by pixel, calculating the corresponding location in the destination image by evaluating the mapping function, and painting that location in the destination image with the color of the source pixel. On the other hand, a warp described by an inverse mapping is performed by scanning the destination image pixel by pixel, calculating the corresponding location in the source image by evaluating the mapping function, and painting the destination pixel with the color of the calculated source location.

The spatial transformation may map the area corresponding to a pixel onto an area that is not centered on integer pixel coordinates and may be of a different size or shape than the mapped pixel. In forward mapping this means that the color of the single source pixel has to be distributed over several destination pixels; in inverse mapping the destination pixel has to be interpolated from several source pixels. If this is done improperly, e.g. by simply rounding the mapped coordinates to the nearest pixel, the result will be nonuniform intensity and/or holes in the destination image in the forward mapping case, and some aliasing artifacts in the inverse mapping case.

In the case of image warping for camera motion compensation, an inverse mapping is applied to the image f^t to move it backward and delete the transformation due to the camera motion.

$$(x, y) = \mathcal{T}(x', y') = T^{-1}(x', y') \tag{2.50}$$

where T^{-1} is the inverse of the camera motion model.

In general, a pixel (x', y') in the warped image do not correspond to a single pixel (x, y) in the original image; that is, it doesn't have integer coordinates. Therefore, the intensity (color) f' value assigned to the pixel (x', y') must be computed as an interpolated combination of the pixel intensity (color) values closest to (x, y) in the source image. The most used method for interpolation are:

• Nearest-neighbor interpolation: also known as zero-order interpolation, which simply assigns to point (x', y') in the destination image the value of the nearest pixel (x, y) in the source image. It is the fastest interpolation method, though it can produce image artifacts called jaggies or aliasing error.

• Bilinear interpolation: also known as first-order interpolation, which assigns to Point (x', y') in the destination image a value that is a bilinear function of the four nearest pixels (x, y) in the source image.

Bilinear interpolation results in an improvement in image quality over nearest-neighbor interpolation, but may still result in less-than-desirable smoothing effects.

• Bicubic interpolation, which assigns to point (x', y') in the destination image a value that is a bicubic function of the 16 nearest pixels (x, y) (4 x 4 neighborhood) in the source image. This interpolation method reduces resampling artifacts even further then in the bilinear interpolation. It preserves the fine detail present in the source image at the expense of the additional time it takes to perform the interpolation.

2.2.5 Conclusion

In this section we summarized the problem of motion segmentation and the well known methods solving it. We grouped these methods into three groups: motion segmentation based on change detection, motion segmentation based on motion field analysis and motion segmentation based on Markov random fields. We also presented how these methods for motion segmentation can be used in the case of a moving observer by estimating and compensating the camera motion.

In the following section, we will present the method we propose to solve the problem of motion segmentation. The method estimates simultaneously the motion field and segment the moving objects in the scene.

2.3 Proposed method for motion estimation and segmentation

This work introduces a new algorithm for simultaneous motion estimation and segmentation of image sequences taken with a moving camera. The algorithm exploits both the advantages of the GMM background subtraction approach with its simplicity and capability to adapt to illumination changes and small motions in the scene, and the advantages of the MRF approach with its power to model non-linear interactions between features of different nature to take advantage of space and time dependencies that moving objects impose on pixels.

The use of the background model and the motion information competitively in the MAP-MRF decision framework produces more accurate and visually attractive silhouettes that are less affected by noise than traditional pixel-wise methods.

Figure 2.18 illustrates the different steps of the proposes approach. At time t, the algorithm starts with the estimation and the compensation of the camera motion between the previous frame, f^{t-1}, and the current frame, f^t using the method proposed by Dufaux and Konrad [173].

In the second step, an apparent scene cut or strong camera pan is detected by evaluating whether or not the squared difference between the current frame, f^t, and the compensated previous frame from camera motion, $\tilde{f}^{t-1}$, exceeds a given threshold, Th_{SC}. The evaluation is performed only within the background regions of the previous frame. In case of a scene cut between two consecutive frames, the segmentation algorithm is reset, i.e. all parameters are set to their initial values, as the camera motion in this case cannot be correctly compensated.

The new compensated frame is used, in the third step, to update the GMM model of the background, Bg, as in the case of a static camera. The model used for the background is the one introduced by Stauffer and Grimson [99].

In a fourth step, the background model, Bg, the current frame, f^t, the compensated frame from camera motion, $\tilde{f}^{t-1}$, and the previous segmentation (object mask), OM, are used in a *maximum a posteriori probability* (*MAP*) framework to detect moving objects in the current frame and estimate their motion field.

In a last step, in order to avoid the common problem of fragmentation in background subtraction methods and accelerate the learning of the background model (new static regions), the background model is re-updated based on the results of the MAP-MRF optimization.

The different steps of the algorithm are described in more details in the following sections.

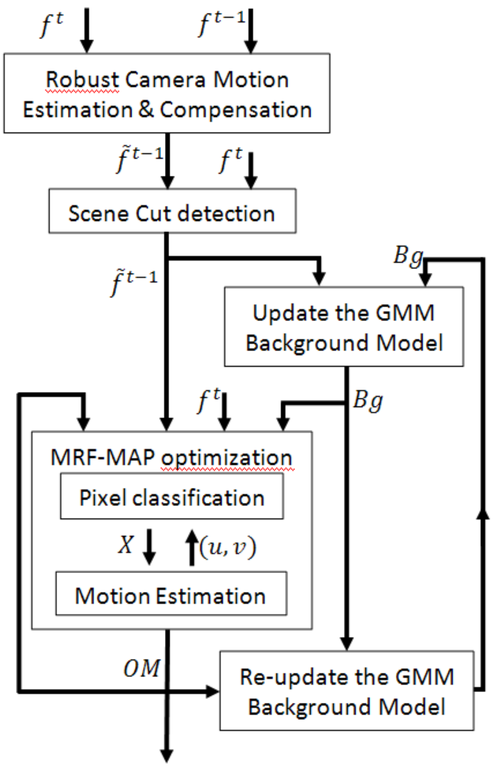


Figure 2.3: Algorithm for motion segmentation with a moving observer

2.3.1 Camera motion estimation and compensation

We implemented two methods for camera motion estimation and compensation (CMEC): a method based on feature tracking and a method based on dense motion analysis (see section 2.2.4.3).

To deal with the problem of outliers in the estimation of the camera motion, we propose to use an initialization phase. This means that before to start moving the robot (the camera) takes some time to build a model for the scene and detect the static and the moving parts. During robot motion, the detected moving parts are not considered for the camera motion estimation.

2.3.1.1 Feature based approach

For the method based on feature tracking, we used the Harris corner detector [191, 194]. Farin et al. [206] have shown that the Harris detector has the best performance for accuracy and repeatability of detection compared to other methods for corner detection.

In our implementation a perspective motion model $\mathbf{x}_r = T(\mathbf{x}_s)$ with eight parameters is used for the camera motion.

$$\mathbf{x}_r = T(\mathbf{x}_s) = \begin{bmatrix} \frac{a_0+a_2x_s+a_3y_s}{a_6x_s+a_7y_s+1} \\ \\ \frac{a_1+a_3x_s+a_5y_s}{a_6x_s+a_7y_s+1} \end{bmatrix} \tag{2.51}$$

The RANSAC algorithm for parameters estimation can be described with the following steps:

1. Draw a subset $\mathcal{S}$ of size $|\mathcal{S}|$ of correspondences. Four correspondences are required to solve for the eight free parameters of the perspective motion model.

2. Compute the parameters $\mathbf{a} = (a_1, ..., a_8)$ of the motion model from the correspondences in $\mathcal{S}$.

3. Determine the set of inliers $\mathcal{I}$, which is the set of correspondences $\mathcal{I} = \{\mathbf{x}_s \leftrightarrow \hat{\mathbf{x}}_r\}$ that comply with the motion model.

4. Repeat steps 1-3 several times (N) and choose the set of inliers for which $|\mathcal{I}|$ is largest.

5. Apply least-squares approximation of the motion parameters with the set of inliers to minimize the squared error (equation 2.45):

$$\epsilon = \frac{1}{2}\sum_i (\mathbf{x}_r - T(\mathbf{x}_s))^2$$

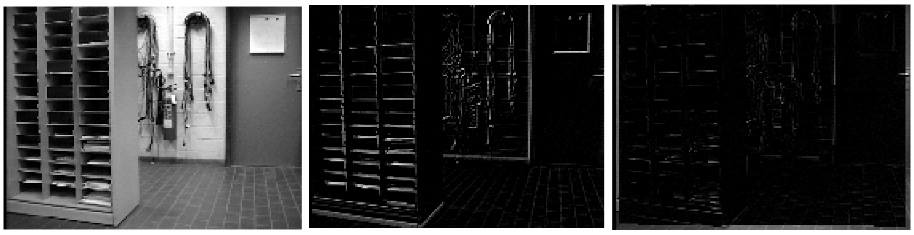
To avoid the requirement of a non-linear optimization, we multiply the Euclidean distance by the denominator of the transform model:

$$\epsilon = \frac{1}{2}\sum_i \left(\left(x_r - \frac{a_0+a_2x_s+a_3y_s}{a_6x_s+a_7y_s+1}\right)^2 + \left(y_r - \frac{a_1+a_3x_s+a_5y_s}{a_6x_s+a_7y_s+1}\right)^2\right)(a_6x_s+a_7y_s+1)^2$$

$$\begin{aligned}\epsilon = \frac{1}{2}\sum_i &[(a_0+a_2x_s+a_3y_s - x_r(a_6x_s+a_7y_s+1))^2 \\ &+ (a_1+a_3x_s+a_5y_s - y_r(a_6x_s+a_7y_s+1))^2]\end{aligned}$$

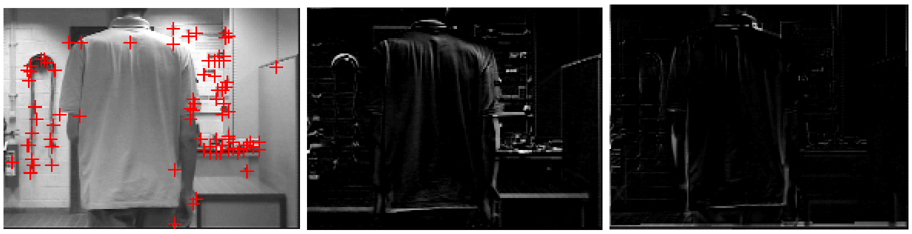
After imposing the necessary condition $\partial\epsilon/\partial a_i = 0$ for an error minimum, this results in a linear equation system from which we can obtain a_i.

Figures 2.4 and 2.5 show examples with no moving objects (static scene) and with moving objects, respectively.



frame f^t frame difference $f^t - f^{t-1}$ frame difference $f^t - \tilde{f}^{t-1}$

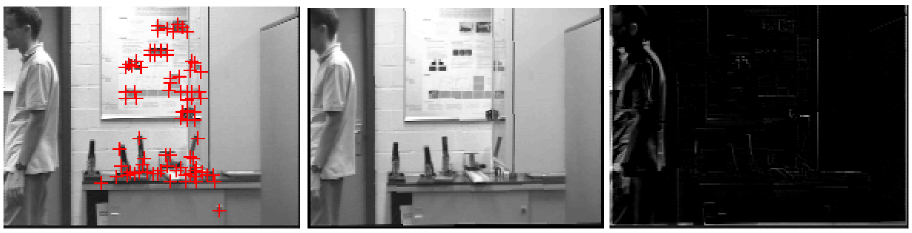
Figure 2.4: Feature tracking based CMEC algorithm in a static scene



frame f^t and detected features frame difference $f^t - f^{t-1}$ frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.5: Feature tracking based CMEC algorithm in a dynamic scene

The drawback of this method for camera motion estimation is that it works only in the case of well textured environment. Figures 2.6 and 2.7 show the results of the warping in a well and less textured scenes, respectively. We can see that in less textured environment (Figure 2.7) the estimated motion model does not represent the correct motion of the camera which leads to a distortion of the wrapped frame $\tilde{f}^{t-1}$.



frame f^t and detected features frame $\tilde{f}^{t-1}$ frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.6: Feature tracking based CMEC algorithm in a well textured scene

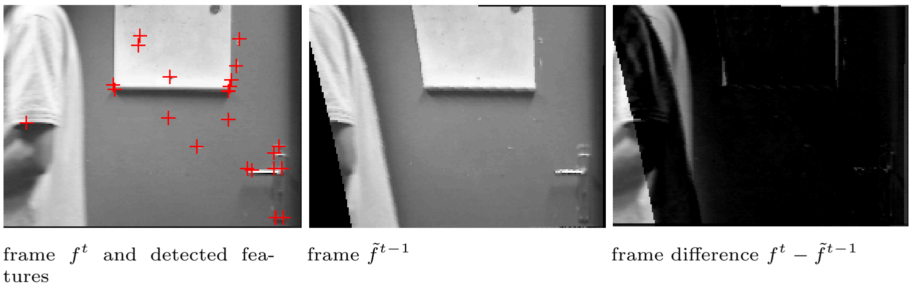

frame f^t and detected features
frame $\tilde{f}^{t-1}$
frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.7: Feature tracking based CMEC algorithm in a less textured scene

Figure 2.9 shows the mean square difference between successive frames (f^{t-1} and f^t) in an image sequence (figure 2.8) and the mean square difference between the current frame $f^t$and the compensated previous frame from camera motion $\tilde{f}^{t-1}$.

$$\text{diff1} = \sqrt{(\text{mean}((f^t - f^{t-1})^2))} \tag{2.52}$$

$$\text{diff2} = \sqrt{(\text{mean}((f^t - \tilde{f}^{t-1})^2))} \tag{2.53}$$

The error in diff1 include also the error due to the interpolation in the warping procedure (see section 2.2.4.3.3).

Figure 2.8: First and Last frames of the lab sequence

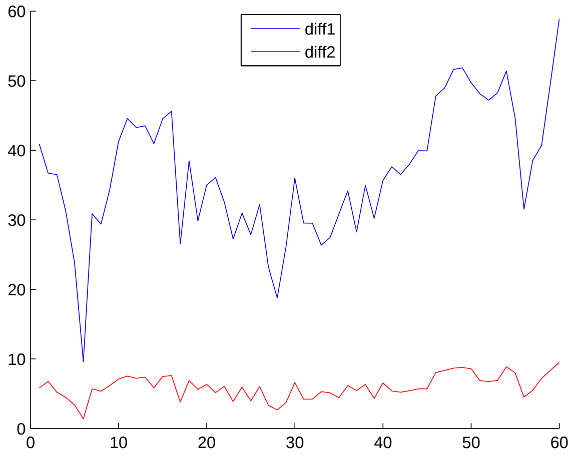


Figure 2.9: Mean Square Differences diff1 and diff2

2.3.1.2 Dense flow based approach

Figures 2.10, 2.11, and 2.12 show the results of the camera motion estimation and compensation based on dense motion in the case of static scene, dynamic scene, and less textured dynamic scene, respectively. We can see that, even we lost some information from the motion of the foreground, this method can robustly handle the camera motion (global motion). This method is the one we used in our implementation for motion segmentation with a moving observer.

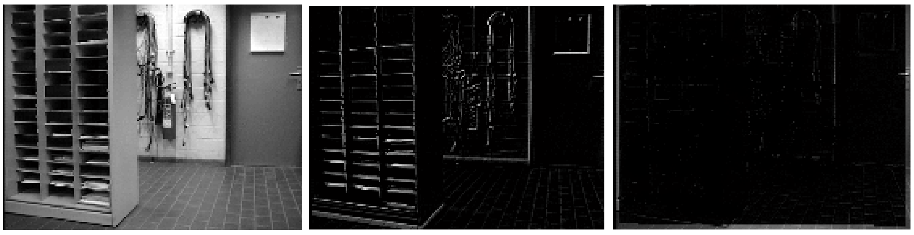

frame f^t
frame difference $f^t - f^{t-1}$
frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.10: Dense motion based CMEC algorithm in a static scene

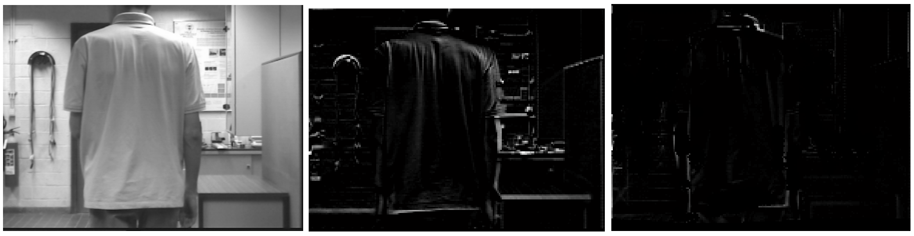

frame f^t
frame difference $f^t - f^{t-1}$
frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.11: Dense motion based CMEC algorithm in a dynamic scene

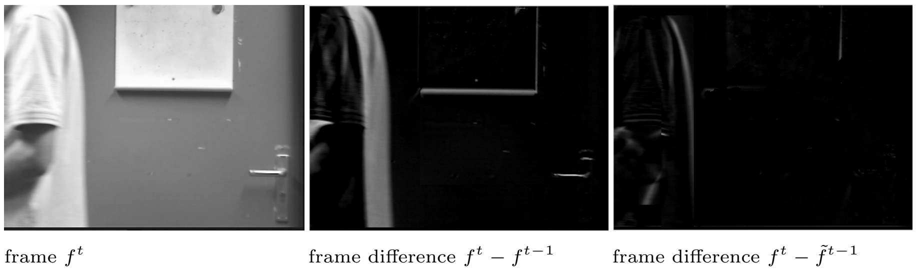

frame f^t
frame difference $f^t - f^{t-1}$
frame difference $f^t - \tilde{f}^{t-1}$

Figure 2.12: Dense motion based CMEC algorithm in a less textured dynamic scene

Figure 2.13 shows the mean square differences (equations (2.52) and (2.53)) in the image sequence offigure 2.8. We can see that the camera motion is well estimated compared to the feature tracking based estimation method.

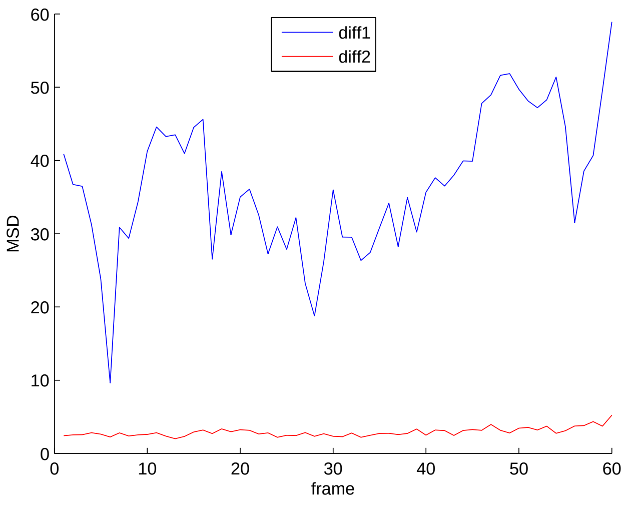


Figure 2.13: Mean Square Differences diff1 and diff2

The parameters $\mathbf{a} = \{a_1, ..., a_8\}$ of the projective model of the camera motion encode movement along all three spatial axes, zoom (i.e., scale in each of the image coordinates x and y), and rotation (including rotations due to panning, tilting, and movement about the optical axis) as follows:

a_3 :	Translation in X direction;
a_6 :	Translation in Y direction;
a_1, a_2, a_4, a_5 :	Scaling (zooming), Rotation, Shearing;
a_7, a_8 :	Chirping and keystoning.

Figure 2.14 shows the estimated parameters for the camera motion in the sequence of figure 2.8. We can see that only parameters representing translation

(a_3 and a_6) which are changing a lot.

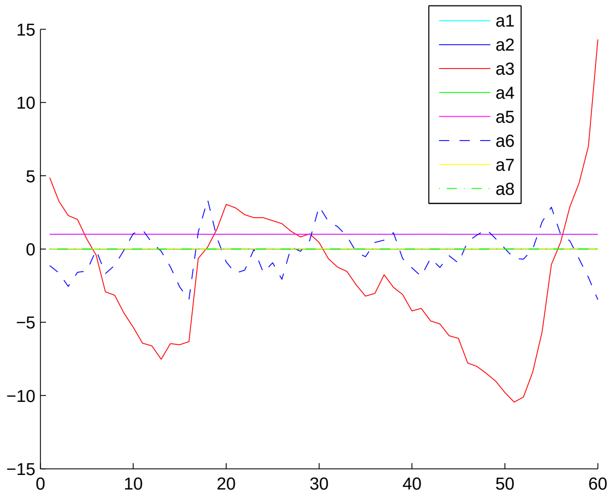


Figure 2.14: Camera Motion Parameters $\mathbf{a} = \{a_1, ..., a_8\}$

The method was tested also on another image sequence (figure 2.15) where there is more changing in zooming then in translation or rotation. The sequence contains also a moving object.

We can see from figures 2.16 and 2.17 that also in this case the motion estimation method based on dense motion performed well and the estimated parameters represent camera motion (big and constant zooming (a_1 and a_5) with a small change due to translation (a_3 and a_6)).

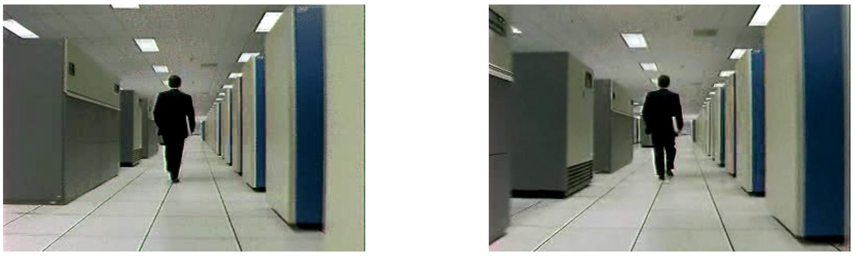


Figure 2.15: First and Last frames of the Office sequence

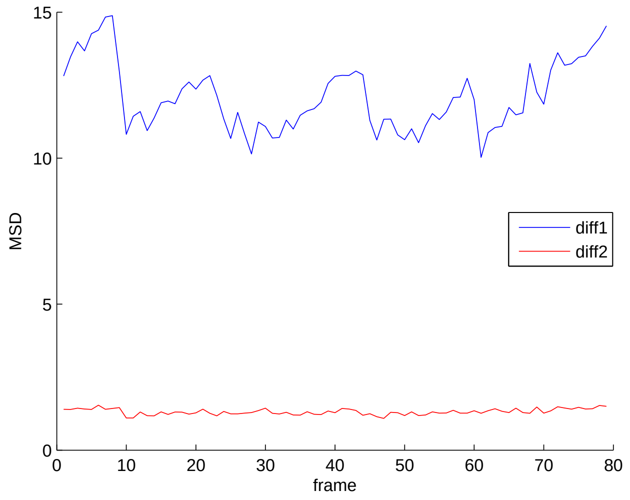


Figure 2.16: Mean Square Differences diff1 and diff2

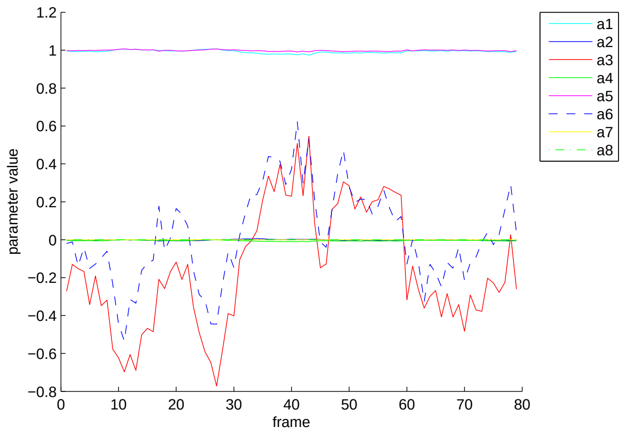


Figure 2.17: Camera Motion Parameters $\mathbf{a} = \{a_1, ..., a_8\}$

2.3.1.3 Camera motion compensation given its 3D motion

As seen in chapter 1, the SLAM process estimates the 3D camera motion. In this section we propose using the estimated 3D camera motion from the SLAM to derive the 2D camera motion parameters for its compensation.

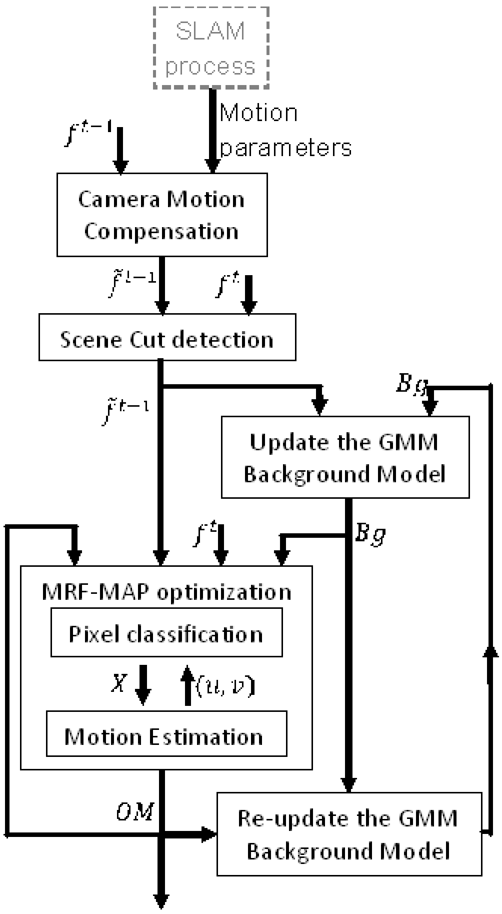


Figure 2.18: Algorithm for motion segmentation with a moving observer

The coordinate in the camera frame system of a point in the space (x^C, y^C, z^C) and the coordinates in the image frame system of the point image (x, y) are related by the perspective transformation:

$$x = f\frac{x^C}{z^C}, \qquad y = f\frac{y^C}{z^C} \tag{2.54}$$

where f is the camera focal length.

Let (x^C, y^C, z^C) be the coordinates in the camera frame system of a stationary point at time t and (x'^C, y'^C, z'^C) its corresponding coordinates at time t'. When the camera undergoes a 3D rotation and translation, the relationship between the 3D point coordinates before and after the camera motion is:

$$\begin{bmatrix} x'^C \\ y'^C \\ z'^C \end{bmatrix} = \begin{bmatrix} 1 & -w_z & w_y \\ w_z & 1 & -w_x \\ -w_y & w_x & 1 \end{bmatrix} \begin{bmatrix} x^C \\ y^C \\ z^C \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.55}$$

2.3.2 Scene cut detection

A scene cut detector consists in detecting highly uncorrelated consecutive frames. It evaluates whether or not the squared difference between the current original frame f^t and the compensated previous frame from camera motion $\tilde{f}^{t-1}$ exceeds a given threshold Th_{SC} (equation 2.56). The evaluation is only performed within the background regions of the previous frame.

$$\frac{1}{N_{Bg}} \sum_{s/l_s^t=0} \left(f_s^t - \tilde{f}_s^{t-1}\right)^2 \begin{array}{ll} > Th_{SC} \Rightarrow & \textit{scene cut} \\ \\ \leq Th_{SC} \Rightarrow & \textit{no scene cut} \end{array} \tag{2.56}$$

N_{Bg} denotes the number of background pixels, l_s^t is the value of the classification at site s and time t, and f_s^t and $\tilde{f}_s^{t-1}$ are the intensity at site (pixel) s of frames f^t and $\tilde{f}^{t-1}$, respectively.

In case of a scene cut between two consecutive frames, the camera motion can not be compensated correctly. Two cases are considered: - if the camera can retrieve cursorily its place and therefore the correct scene view. The few false frames are detected and canceled by the scene cut detector. Instead, in case where the gap between the last "correct frame" and the new ones persist for several seconds, the algorithm is re-initialized: the robot stops, takes time to detect moving object to avoid outliers in camera motion estimation when it starts moving. The first frame after the initialization of the algorithm is considered as the current background to be updated. In our implementation, the process decides to re-initialize after 7 to 9 scene cuts from the last "correct frame".

2.3.3 Gaussian Mixture Background Modeling

To model the background we use the Gaussian Mixture Model (GMM) introduced by Stauffer and Grimson [99] given by equation 2.11 (section 2.2.1.2.2).

In this model each pixel is modeled as a mixture of Gaussians. The number of normal distributions (Gaussians), K, is set as a parameter from the beginning.

In our implementation, the channels of the feature space are considered as independent ($\Sigma_{t,k} = \sigma_{t,k}^2 I$ with I the identity matrix) and the matching of the actual data with the distributions is done using the Mahalanobis distance (equation 2.57) instead of the Euclidean distance (equation 2.14):

$$d_s^t = (f_s^t - \mu_{s,k}^t)^T {\Sigma_{s,k}^t}^{-1} (f_s^t - \mu_{s,k}^t) = \frac{\| f_s^t - \mu_{s,k}^t \|^2}{(\sigma_{s,k}^t)^2} \tag{2.57}$$

The parameters of the matched component $\hat{k}$ are updated using equations 2.16.

Number of components

From our experiences, the number of components in the case of Gaussian Mixture modeling for background subtraction, is generally small and 2 components are enough, as there are only two classes (background, foreground), if the learning rates (α,ρ) and the background threshold are well chosen.

Figure 2.19 shows the segmentation results obtained with a GMM algorithm with 2 and 5 components applied on the sequence "02463C1AL" from USF-NIST [1] [204]. In this sequence, a person at the far right side of the camera's field of view is moving (walking) to the left side, turns toward the camera, and walks back to the right.

The parameters of the algorithms (determined empirically) are:
GMM with 2 Gaussians: $\alpha = 0.05, \rho = 0.0005$, and $T = 0.0001$.
GMM with 5 Gaussians: $\alpha = 0.01, \rho = 0.0075$, and $T = 0.6$.
The initial value for the standard deviation is $\sigma_0 = 22^2$ in both cases.

A GMM model with a number of components $K = 2$ and a good choice of learning parameters is faster then a GMM model with 5 components and it can guaranty a multi-modality of the system on a particular image sequence. However, a given set of learning parameters for a two GMM model can not be used for different image sequences. The two GMM algorithms described above have been tested on several image sequences and the obtained results demonstrate that a the GMM model with $K = 5$ can always adapt to these sequences which is not the case with the GMM model with $K = 2$ as it is shown in figure 2.20 [2].

[1]the sequence can be found at http://figment.csee.usf.edu/GaitBaseline
[2]the sequence can be found at http://www.cvg.cs.rdg.ac.uk/datasets/index.html

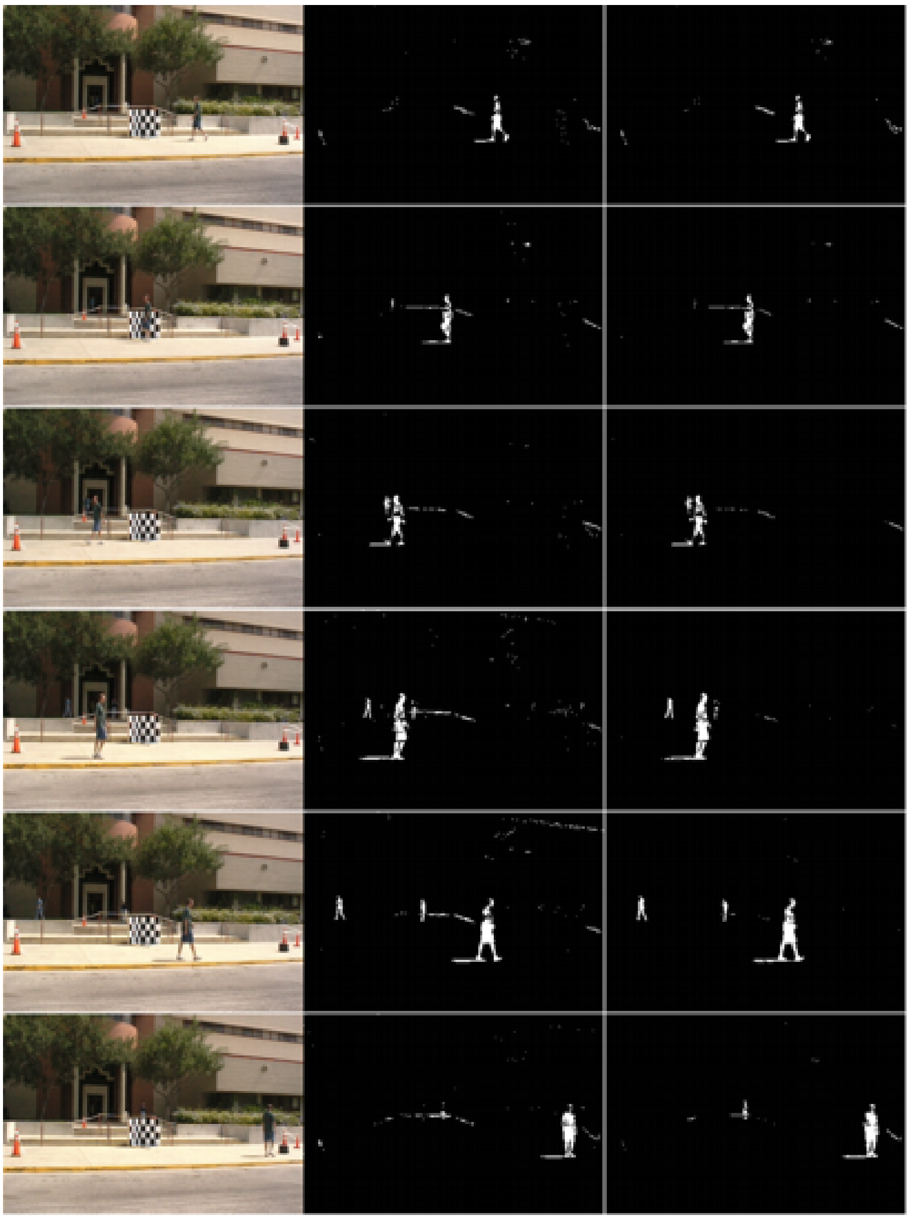
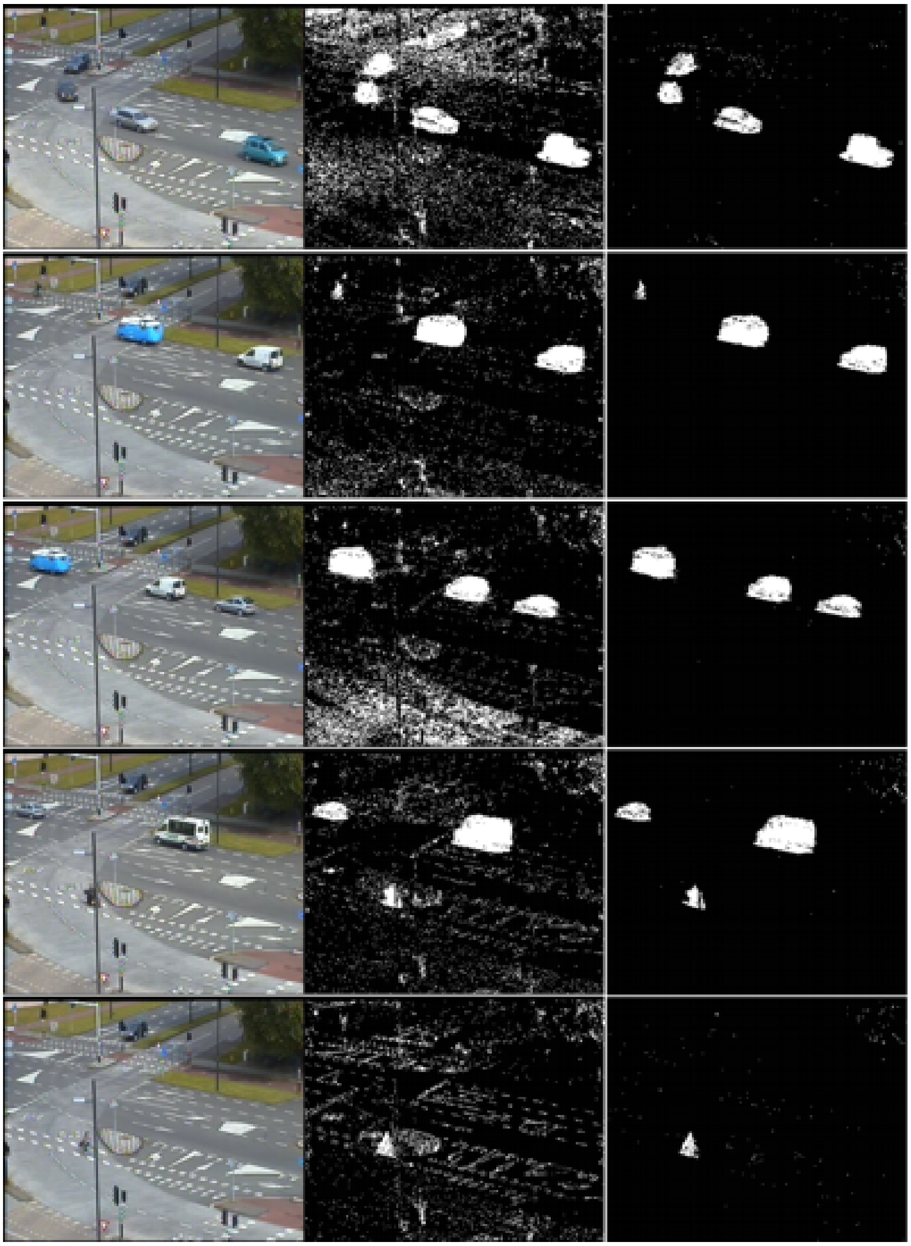


Figure 2.19: Segmentation with different number of components in the GMM model. (sequence ”02463C1AL” from USF-NIST)



a) Original frame b) segmentation with K=2 b) segmentation with K=5

Figure 2.20: Segmentation with different number of components in the GMM model.

2.3.4 MRF modeling for motion estimation and segmentation

GMM motion segmentation is pixel based, that is, each pixel is processed against its background model independently of other pixels in the image. Furthermore, there is no time dependency in this technique as the output of a subtraction process at time t has nothing to do with its output at time $t-1$. To take advantage of both the space and time dependencies that moving objects impose on the image pixels, we combine the segmentation and the motion estimation process in a Markov Random Field (MRF) framework.

Given the current frame, f^t, the compensated previous frame from camera motion, $\tilde{f}^{t-1}$, the background model, Bg, and the previous segmentation object mask, OM, we wish to compute the *maximum a posteriori probability (MAP)* estimate of the motion field, $\mathcal{U}^t$, and the segmentation label field L. We use the notations $\mathbf{u}^t = \{\mathbf{u}^t_s, s \in S\}$ and $l^t = \{l^t_s, s \in S\}$ for the realizations of $\mathcal{U}^t$ and L^t, respectively. S is the set of pixels' indices. In this work we are concerned with binary classification ; at each site (pixel) s, $s \in S$, the label value l_s can be either 0 (background) or 1 (foreground).

Using the Bayes rule, the posteriori *probability density function* (pdf) of $\mathbf{u}^t$ and l^t given f^t and $\tilde{f}^{t-1}$ can be expressed as:

$$p(\mathbf{u}^t, l^t \mid f^t, \tilde{f}^{t-1}) = \frac{p(f^t \mid \mathbf{u}^t, l^t, \tilde{f}^{t-1})p(\mathbf{u}^t \mid l^t, \tilde{f}^{t-1})p(l^t \mid \tilde{f}^{t-1})}{p(f^t \mid \tilde{f}^{t-1})} \tag{2.58}$$

The denominator is constant with respect to the unknowns. Thus the *MAP* estimate corresponds to the following:

$$(\hat{\mathbf{u}}^t, \hat{l}^t) = \max_{\mathbf{u}^t, l^t} \left(p(f^t \mid \mathbf{u}^t, l^t, \tilde{f}^{t-1})p(\mathbf{u}^t \mid l^t, \tilde{f}^{t-1})p(l^t \mid \tilde{f}^{t-1}) \right) \tag{2.59}$$

The conditional pdf $p(f^t \mid \mathbf{u}^t, l^t, \tilde{f}^{t-1})$ quantifies how well the motion field and motion segmentation estimates fit the current frame. As the color space and motion field are mutually independent (given the spatial information, we cannot predict the optical flow and the same, given the optical flow at a spatial location, we cannot predict its colors), the probability $p(f^t \mid \mathbf{u}^t, l^t, \tilde{f}^{t-1})$ is a product of two terms modelled by Gibbs distributions:

$$p(f^t \mid \mathbf{u}^t, l^t, \tilde{f}^{t-1}) = T_1(l^t, f^t, \tilde{f}^{t-1}).T_2(\mathbf{u}^t, f^t, \tilde{f}^{t-1}) \tag{2.60}$$

The term T_1 models a normally distributed noise around the site's color:

$$T_1(l^t, f^t, \tilde{f}^{t-1}) = \left(\prod_{s \in S} \frac{1}{(2\pi)^{\frac{3}{2}} \sigma^t_{s,\hat{k}}} \right) .e^{-U_1(l^t, f^t, \tilde{f}^{t-1})} \tag{2.61}$$

with

$$U_1(l^t, f^t, \tilde{f}^{t-1}) = \sum_{s \in S} \frac{\| f_s^t - \mu_{t,\hat{k}|_s} \|^2}{2(\sigma^t_{s,\hat{k}})^2} \tag{2.62}$$

is is the energy function.

$\sigma^t_{s,\hat{k}}$ and $\mu^t_{s,\hat{k}}$ are, respectively, the variance and the mean at time t of the matched component $\hat{k}$ at site s in the background GMM model.

The coefficient $\prod\limits_{s \in S} \frac{1}{(2\pi)^{\frac{3}{2}} \sigma^t_{s,\hat{k}}}$ is incorporated in the partition function Z.

The second term in the likelihood, $T_2(\mathbf{u}^t, f^t, \tilde{f}^{t-1})$, models the quality to obtain the current frame from the previous one using the estimated motion field . It is modeled by a Gibbs distribution with the energy function:

$$U_2(\mathbf{u}^t, f^t, \tilde{f}^{t-1}) = \sum_{s \in S} \varepsilon_s^2 \tag{2.63}$$

where

$$\varepsilon_s = \| f^t(\mathbf{x}_s) - \tilde{f}^{t-1}(\mathbf{x}_s + \mathbf{u}_s^t) \| \tag{2.64}$$

is the displaced frame difference (DFD) at site s, $\mathbf{x}_s$ is the coordinate of the site s, $f^t(\mathbf{x}_s)$ is the intensity of the frame f^t at site s, and $\mathbf{u}_s$ is the motion field at this site.

T_2 is maximized when the motion field minimizes the DFD function, indicating that accurate optical flow estimates are obtained.

The second term in (2.59) is the conditional pdf of the displacement field given the motion field and the previous frame. Neglecting the dependence on the previous frame, this term is imposed only when the neighboring pixels share the same segmentation label to smooth the motion field. We model this pdf also by a Gibbs distribution with the energy function

$$U_3(\mathbf{u}^t \mid l^t) = \lambda \sum_{s \in S} \sum_{r \in N_s} \varepsilon'_{s,r} \delta\left(l_s^t - l_r^t\right) \tag{2.65}$$

Where

$$\varepsilon'_{s,r} = \| \mathbf{u}_s^t - \mathbf{u}_r^t \|^2 \tag{2.66}$$

denotes the squared displacement difference between neighboring sites s and r. N_s is the set of neighbors of the site s, δ is the Kronecker function and λ is a scalar used to control the emphasis of the constraint.

• Neglecting the dependence on the previous frame, $\tilde{f}^{t-1}$, the third term in (2.59) represents the a priori probability of the segmentation in order to encourage formation of contiguous regions. It is modelled by a Gibbs distribution with the energy function:

$$U_4(l^t) = \beta \sum_{s\in S} \sum_{r\in N_s} V\left(l_s^t, l_r^t\right) \tag{2.67}$$

Where β control the emphasis of this term and

$$V\left(l_s^t, l_r^t\right) = \begin{cases} -\frac{1}{1+\varepsilon'_{s,r}}, & \text{if } l_s^t = l_r^t \\ \\ +\frac{1}{1+\varepsilon'_{s,r}}, & \text{otherwise} \end{cases} \tag{2.68}$$

denotes a two-clique potential function.

In this function, two neighbors sites s and r are more likely classified to be part of the same object if they have the same motion.

In summary the a posteriori pdf (2.58) can be rewritten as:

$$p(\mathbf{u}^t, l^t \mid f^t, \tilde{f}^{t-1}) \propto$$

$$\exp\left\{-U_1(l^t, f^t, \tilde{f}^{t-1}) - U_2(\mathbf{u}^t, f^t, \tilde{f}^{t-1}) - U_3(\mathbf{u}^t, l^t) - U_4(l^t)\right\} \tag{2.69}$$

Substituting U_1, U_2, U_3, and U_4 with equations (2.62),(2.63),(2.65), and (2.67), respectively, the maximization of the a posteriori pdf $p(\mathbf{u}^t, l^t \mid f^t, \tilde{f}^{t-1})$ is equivalent to minimizing the energy function

$$U = U_1 + U_2 + U_3 + U_4 = \sum_{s\in S} \frac{\| f_s^t - \mu_{t,\hat{k}|_s} \|^2}{2(\sigma^t_{s,\hat{k}})^2} + \sum_{s\in S} \varepsilon_s^2 + \\ \lambda \sum_{s\in S} \sum_{r\in N_s} \varepsilon'_{s,r} \delta\left(l_s^t - l_r^t\right) + \beta \sum_{s\in S} \sum_{r\in N_s} V\left(l_s^t, l_r^t\right) \tag{2.70}$$

Where ε_s, $\varepsilon'_{s,r}$, and $V\left(l_s^t, l_r^t\right)$ are given in equations (2.64), (2.66), and (2.68), respectively.

2.3.5 Background model re-updating

After the MRF optimization of the motion segmentation, some of the pixels detected as belonging to the background using the GMM model became part of the foreground (false background detection) and others detected with the GMM model as belonging to the foreground became part of the background (false foreground detection). In this case the model of the background should be updated according to the finale result of motion segmentation. For this, we propose to reevaluate the parameters of the model corresponding to the changed pixels. The reevaluation should be done inversely to the learning process in GMM model.

- **False background detection**

The parameters of the matched component in the GMM background model are re-updated as follows:

$$\begin{aligned}\omega^{t+1}_{s,\hat{k}} &= \tfrac{1}{1-\alpha}(\omega^{t}_{s,\hat{k}} - \alpha)\\ \mu^{t+1}_{s,\hat{k}} &= \tfrac{1}{1-\rho}(\mu^{t}_{s,\hat{k}} - \rho f^t_s)\\ (\sigma^{t+1}_{s,\hat{k}})^2 &= \tfrac{1}{1-\rho}((\sigma^{t}_{s,\hat{k}})^2 + \rho \parallel f^t_s - \mu^{t}_{s,\hat{k}} \parallel^2)\end{aligned} \tag{2.71}$$

Where α is the same learning rate used in the learning phase of the GMM model (see section 2.2.1.2.2). $\omega^t_{s,\hat{k}}, \mu^t_{s,\hat{k}}$, and $\sigma^t_{s,\hat{k}}$ are, respectively, the weight, the mean, and the standard deviation of the matched model at changed pixels.

• **False foreground detection**

In this case we admit that we have surely a background pixel and then the parameters of the closest component ($\hat{k}$) to this pixel in the GMM background model are re-updated as follows:

$$\begin{aligned}\omega^{t+1}_{s,\hat{k}} &= (1-\alpha')\omega^{t}_{s,\hat{k}} - \alpha'\\ \mu^{t+1}_{s,\hat{k}} &= (1-\rho')\mu^{t}_{s,\hat{k}} + \rho' f^t_s\\ (\sigma^{t+1}_{s,\hat{k}})^2 &= (1-\rho')(\sigma^{t}_{s,\hat{k}})^2 + \rho' \parallel f^t_s - \mu^{t}_{s,\hat{k}} \parallel^2\end{aligned} \tag{2.72}$$

$\alpha' >> \alpha$ to have a fast forgetting of the past background model and $\rho' = \alpha'/\omega_{\hat{k}}$.

2.3.6 Implementation

The previously described approach has been implemented in matlab and tested on synthetic and real image sequences. To show the advantage of using the motion field in the segmentation process, a simplified version of the proposed algorithm has been implement as well.

2.3.6.1 Simplified Model

A simplified version of the global algorithm in which the posterior depends only on the previous segmentation, the current frame and the background model, without estimation of the motion field, can be defined as

$$p(l^t \mid f^t, \tilde{f}^{t-1}) = \frac{p(f^t \mid l^t, \tilde{f}^{t-1})p(l^t \mid \tilde{f}^{t-1})}{p(f^t)} \tag{2.73}$$

The maximization of the corresponding a posteriori pdf is then equivalent to the minimization of the energy function (neglecting in this case also the dependence on the pervious frame in the priori probability function)

$$U(l^t \mid f^t, \tilde{f}^{t-1}) = U_1(f^t \mid l^t, \tilde{f}^{t-1}) + U_2(l^t) \tag{2.74}$$

with

$$U_1(f^t \mid l^t, \tilde{f}^{t-1}) = \sum_{s \in S} \frac{\| f_s^t - \mu_{t,\hat{k}|_s} \|^2}{2(\sigma^t_{s,\hat{k}})^2} \tag{2.75}$$

and

$$U_2(l^t) = \beta \sum_{s \in S} \sum_{r \in N_s} V\left(l_s^t, l_r^t\right) + \gamma \sum_{s \in S} V'\left(l_s^t, l_s^{t-1}\right) \tag{2.76}$$

The first term in the prior energy function (2.76) is the spatial smoothing function as in the global algorithm with the following potential function

$$V\left(l_s^t, l_r^t\right) = \begin{cases} -1, & \text{if } l_s^t = l_r^t \\ \\ +1, & \text{otherwise} \end{cases} \tag{2.77}$$

whereas the second term in (2.76) is used to introduced a temporal continuity with the potential function

$$V'(l_s^t, l_s^{t-1}) = \begin{cases} -1, & l_s^t = l_s^{t-1} = 0; \\ \\ 0, & \text{otherwise;} \end{cases} \tag{2.78}$$

With this potential function, as we are interested in background estimation, we want that if a pixel was classified as background at time $t-1$, it would remain background at time t. Aside from that, we did not want to impose any other relationship on a pixel's segmentation.

2.3.6.1.1 Optimization For the simplified version of the algorithm, the optimization is done using the simulated annealing with metropolis algorithm as follows:

- Initialization (current solution, Temperature)
- Calculation of the $CurrentCost$
- Loop
  - $NewState$
  - Calculation of the $NewCost$
  - if $\Delta(CurrentCost - NewCost) \leq 0$ then
    - $CurrentState = NewState$
  - else
    - if $\exp \frac{(CurrentCost - Newcost)}{Temperature} > random\, number\, in\, [0,1]$ then
      - Accept $CurentState = NewState$

```
        * else
            · Reject
    – Decrease Temperature
• Exit Loop and stop when StopCriteriom
```

2.3.6.2 General algorithm

In the case of the global algorithm, the optimization of equation (2.3.4) with respect to all unknowns is a difficult problem. To this effect, we perform the minimization of the energy function U (2.3.4) by iterating over the following two steps:
1. Update the motion field $\mathcal{U}$, given the best estimate of the segmentation field, $L = l$. This step involves minimization of the energy function:

$$E_{\mathcal{U}} = \sum_{s \in S} \varepsilon_s^2 + \lambda \sum_{s \in S} \sum_{r \in N_s} \varepsilon'_{s,r} \delta \left(l_s^t - l_r^t \right) \tag{2.79}$$

which contains all terms in (2.3.4) that depend on $\mathbf{u}$. Where ε_s and $\varepsilon'_{s,r}$ are given with the equations (2.64) and (2.66), respectively.
2. Update the segmentation field L, assuming the motion field $\mathcal{U} = \mathbf{u}$ is given. This step involves the minimization of the energy function containing all terms that contain l as well as the spatial information:

$$E_L = \sum_{s \in S} \frac{\| f_s^t - \mu_{t,\hat{k}|_s} \|^2}{2(\sigma_{s,\hat{k}}^t)^2} + \beta \sum_{s \in S} \sum_{r \in N_s} V \left(l_s^t, l_r^t \right) \tag{2.80}$$

with $V \left(l_s^t, l_r^t \right)$ is given with equation (2.68).

At each step the optimization is carried out using the ICM algorithm. The optimization procedure stops when a (local) minimum is reached.

The parameters λ and β (or β and γ for the simplified version) in the proposed algorithm, control the weight of the potentials in the *MAP* estimation. These parameters have been determined empirically.

In our implementations, the motion field has been initialized using the *Lucas-Kanade* method for optical flow estimation (see section 2.2.4.3.1 for a description of this method).

2.3.7 Experimental results

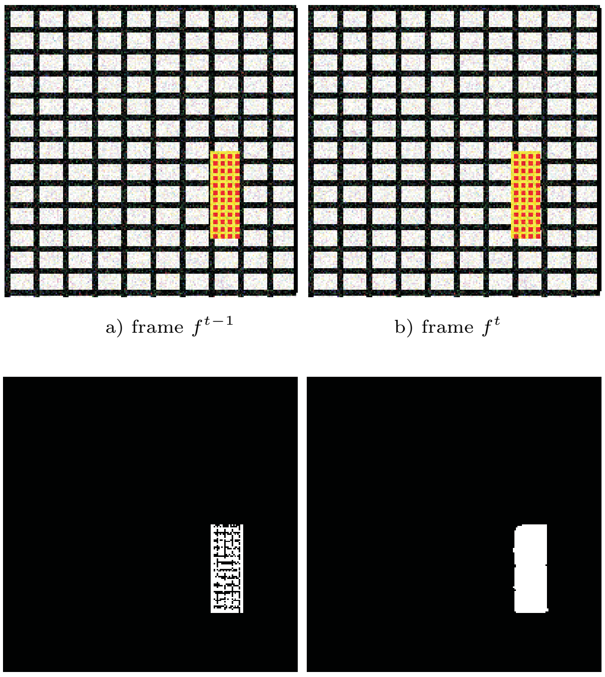
The above algorithms have been tested on synthetic scenes and real image sequences. For all our experiments, the used parameters of the background gaussian mixture model are:

number of components: $K = 5$
learning rate for the weighting: $\alpha = 0.01$

learning rate for the parameters: $\rho = 0.0075$
matching threshold: 4 times the standard deviation distance from the mean of the components.
the threshold for selecting the background components (modes): $Th = 0.8$

2.3.7.1 results using the simplified Model

Figure 2.21 shows the optimization of the motion segmentation using the MRF procedure on a synthetic image sequence. Fig2.21-a and Fig2.21-b are the frames f^{t-1} and f^{t} of the sequence, respectively. Fig2.21-c is the result of the GMM segmentation step and Fig2.21-d is the result after the MRF optimization. In Fig2.21-c and Fig2.21-d, the black pixels are background whereas the white pixels correspond to the foreground.


a) frame f^{t-1} b) frame f^{t}

c) GMM segmentation d) MRF optimization

Figure 2.21: Segmentation optimization in MRF procedure

Figures 2.22 and 2.23 show the results obtained with the simplified version of the proposed algorithm for motion segmentation in real scene sequences taken with a static and a moving camera, respectively. Each row shows the original frame, the foreground mask and the combination of the original frame and the mask.
From these figures one can notice that the results are acceptable in the case of still

camera, whereas in the case of moving camera, even if the foreground is detected, the contour is not smooth.

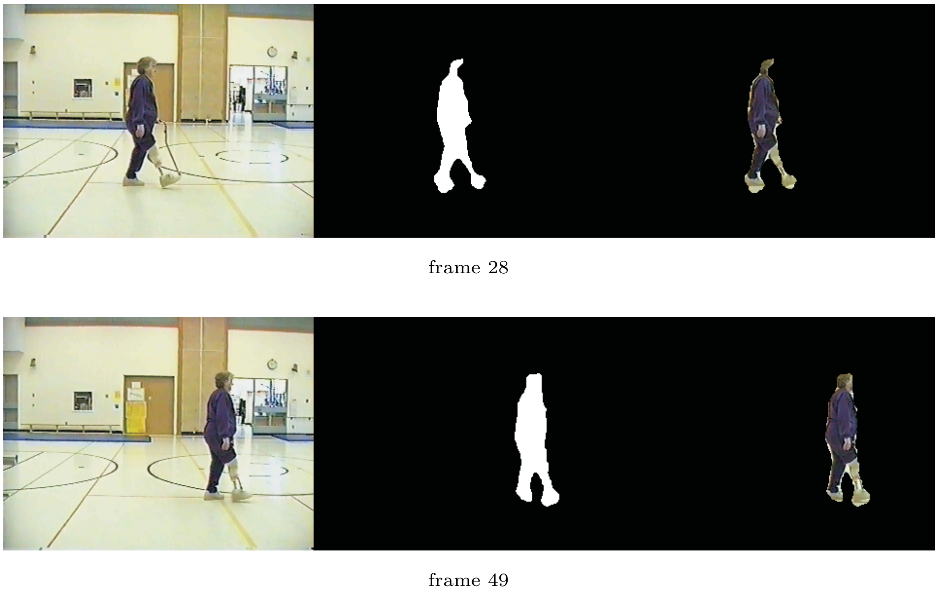


Figure 2.22: Simplified version of the algorithm in the case of a static camera.

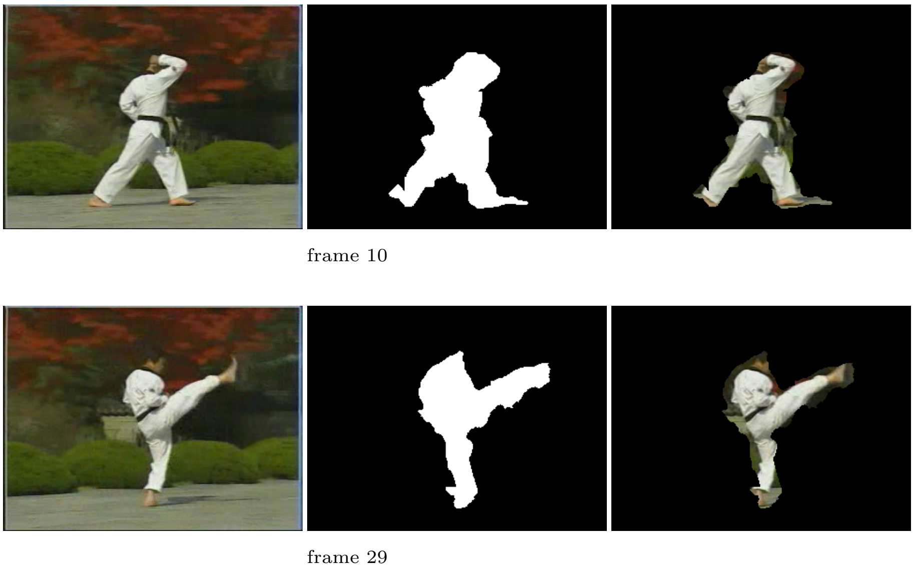


Figure 2.23: Simplified version of the algorithm in the case of a moving camera.

If there is an error in the the camera motion estimation and compensation, as in Figure 2.24 where suddenly a big part of the frame is detected as moving, this will be detected with the scene cut detection procedure, and therefore the current frame will not be used to update the background model.

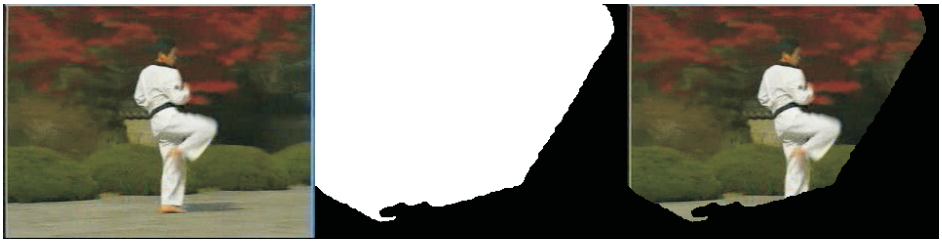


Figure 2.24: Error in motion estimation and compensation at frame 27

2.3.7.2 Results using the global algorithm

For the global algorithm, the used parameters for the MRF framework are $\lambda = 1.0$ and $\beta = 0.5$ in the case of a static camera, and $\lambda = 0.7$ and $\beta = 0.5$ in the case of a moving camera.

Figures 2.25 shows the results obtained in the case of a static camera. Comparing the first row of figure 2.25 with the results in figure 2.22, one can notice that by using the motion field in the MRF framework, we obtain more smoothed borders.

The third result in figure 2.25 presents a more complicate case where the motion field of the foreground is not a regular translation as the foreground is moving toward the camera.

Figures 2.26 and 2.27 show examples with a moving camera. In the example of figure 2.27, there are some false detections of some winding parts of the green plant and the trees' leaves. These false detections have been filtered based on their size, except the regions connected to the real foreground.

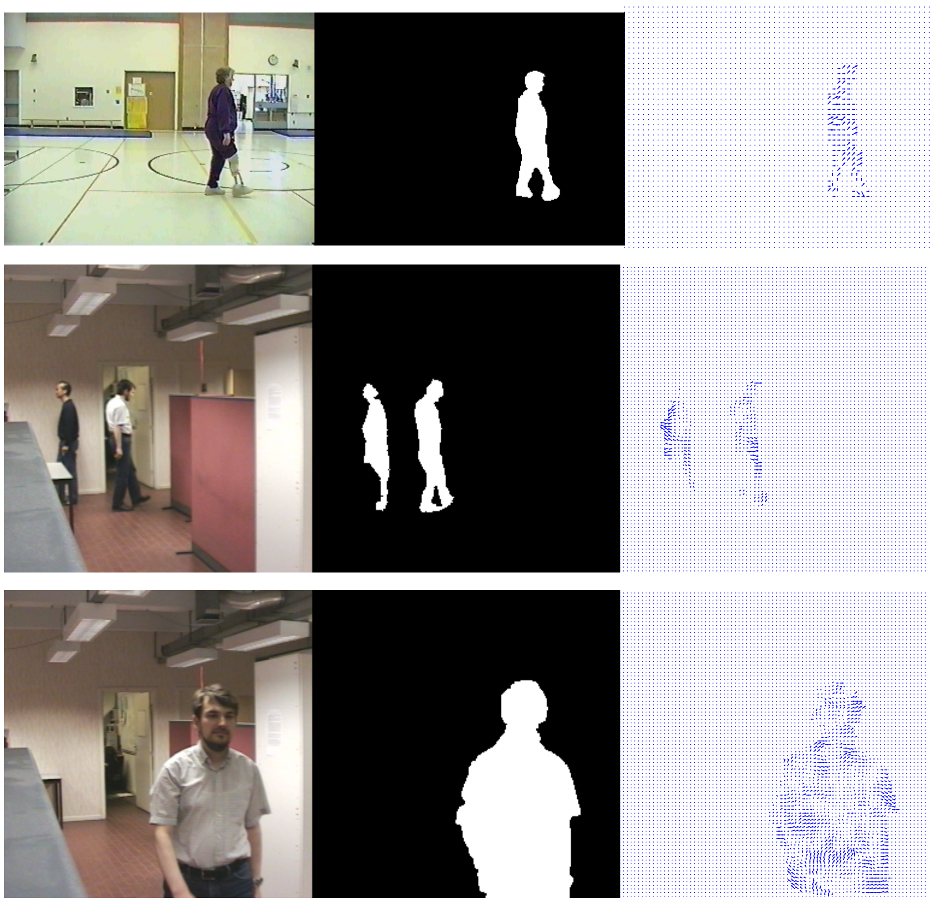

a)Original frame b) Foreground Mask c) Motion field

Figure 2.25: Motion segmentation with static camera

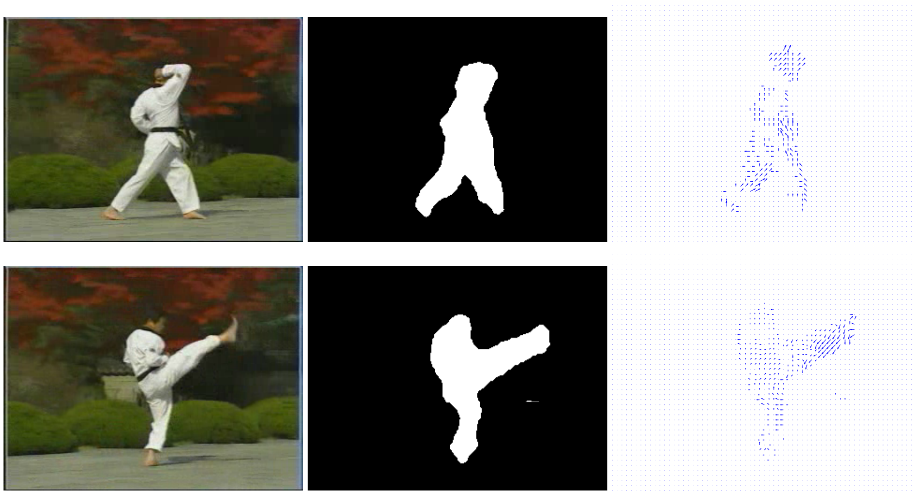


Figure 2.26: Motion segmentation with moving camera - first example.

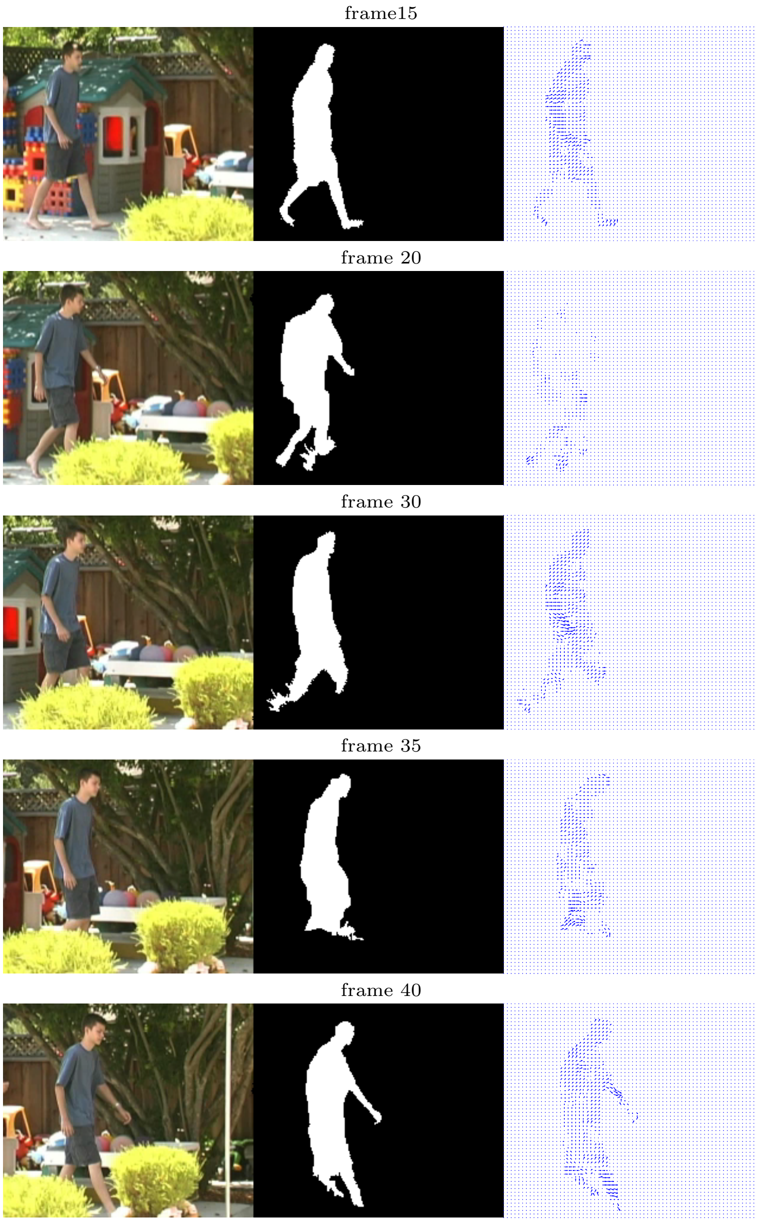


Figure 2.27: Motion segmentation with moving camera - second example.

2.3.7.3 Method evaluation

To show the advantage gained with the use of the *MRF* segmentation to re-update the *GMM* model, the method is compared to the classical GMM algorithm. Figure 2.28 shows that the learning in the proposed algorithm is faster compared to the classical GMM algorithm. The proposed algorithm needs only 25 frames to learn the part of the background which was covered by the foreground at the beginning, instead of the 42 frames needed by the classical GMM algorithm.

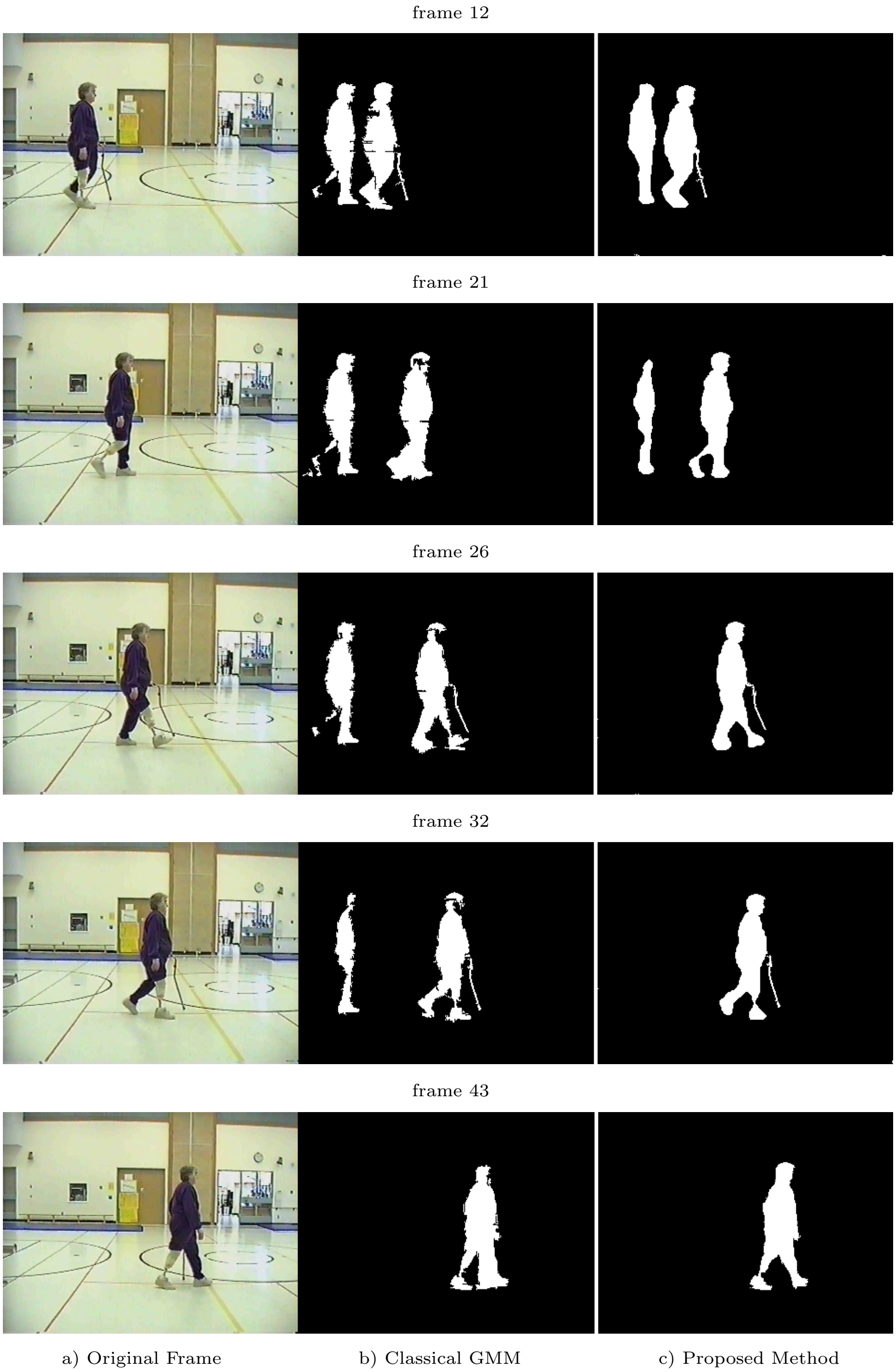


Figure 2.28: Background Model updating

For a qualitative comparison between the segmentation results (background / Foreground) obtained with the classical GMM algorithm and the proposed algorithm, we considered the classification, in time, of three pixels of the sequence of figure 2.28. The considered pixels are depicted in figure 2.29. Pixel 'P1' is located on the foreground on the first frame, then it becomes part of the background after 5 frames. Pixel 'P2' is a background pixel and becomes part of the foreground from frame 12 to frame 19. Whereas, pixel 'P3' is always part of the background. Each row in figure 2.30 represents the red values of the considered pixels in the original image sequence, the background model obtained with the classical GMM algorithm, and the background model obtained with the proposed algorithm.

The results demonstrate the stability of the proposed algorithm as it selects the more representative GMMs for the background model. Figure 2.30.a shows also the fast adaptation of background using the proposed algorithm.

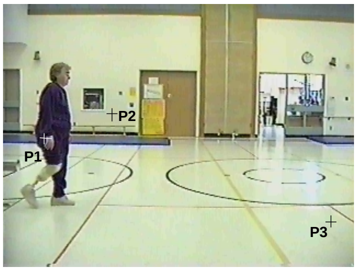
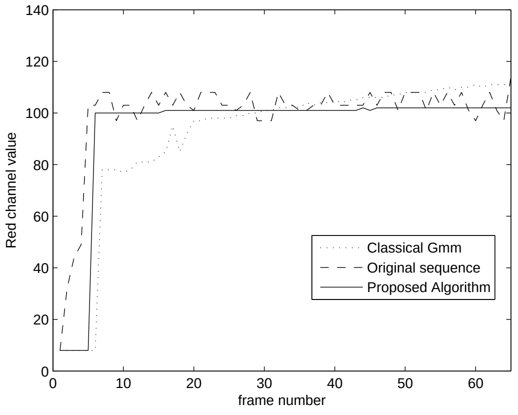
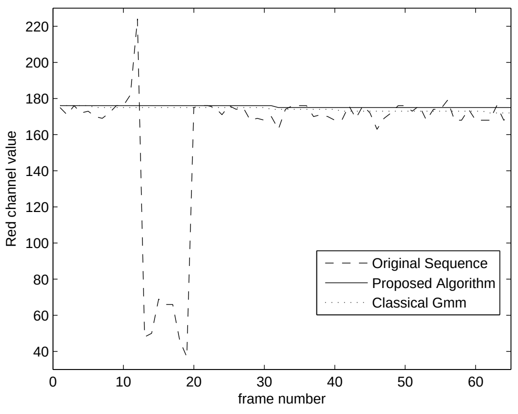


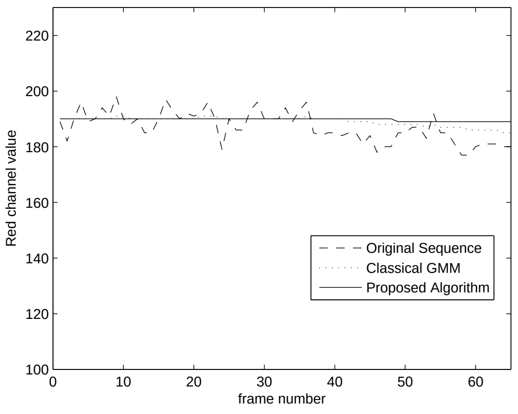
Figure 2.29: Ground truth pixels



a) pixel p1(39,123)



b) pixel p2(100,100)



c) pixel p3(300,200)

Figure 2.30: evolution of the background model at the ground truth pixels

For a further quantitative evaluation of the proposed global algorithm we estimated the percentage of the pixels belonging to the moving objects and are correctly assigned to the foreground (percentage of true foreground detection) and the percentage of the background pixels that are incorrectly classified as belonging to the foreground (percentage of false background detection). The ground truth used for the evaluations is obtained by segmenting manually some frames of the sequences.

Figure 2.31 shows the ground truth used for this evaluation.

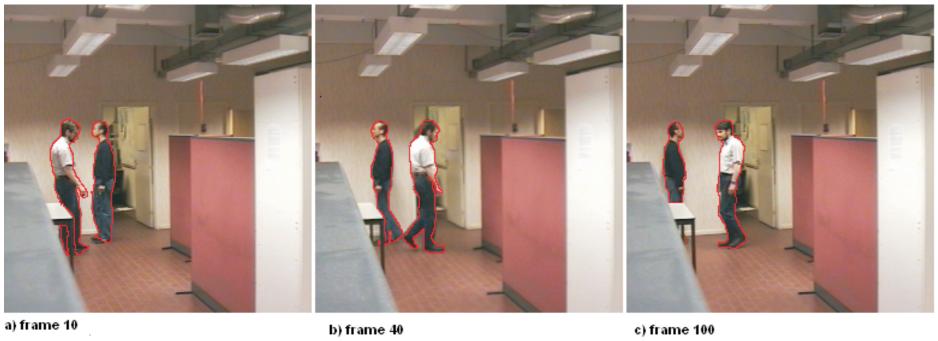


Figure 2.31: Ground-truth for quantitative evaluation of the proposed motion segmentation algorithm in the case of a still camera

Figures 2.32 and 2.33 represent the percentage of true foreground detection and the percentage of the false background detection at some frames of the sequence, respectively. The comparison is made after an initialization time. The lower percentage of true foreground detection is generally due to the less textured part of the foreground or when it stops moving for few moments. This last case appears with a big false background detection (in this sequences this problem appears each time the persons stop to change their direction).
The first part in both graphs correspond to the learning process.

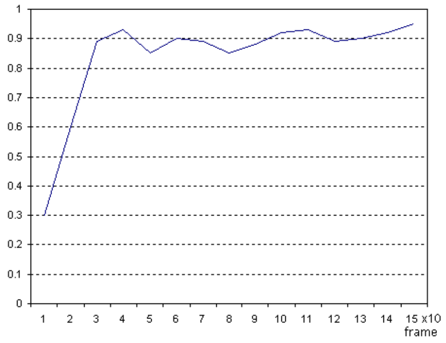


Figure 2.32: Percentage of true foreground detection

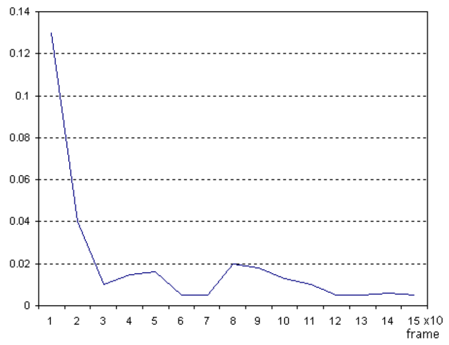


Figure 2.33: Percentage of False Background detection

The same experiences have been made on a sequence with a moving camera. The ground truth are shown in figure 2.34 and the percentage of true foreground detection and false background detection are given in figures 2.35 and 2.36, respectively.

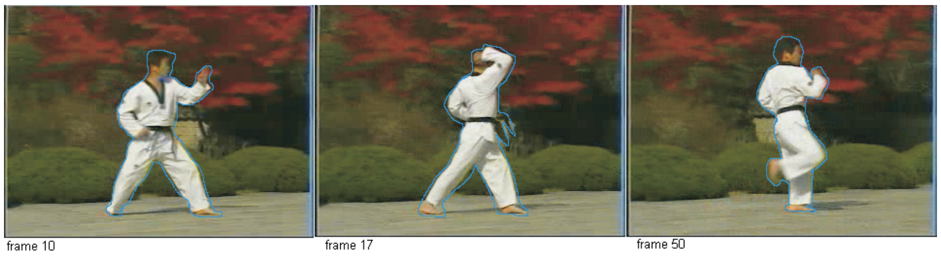


Figure 2.34: Ground-truth for quantitative evaluation of the proposed motion segmentation algorithm

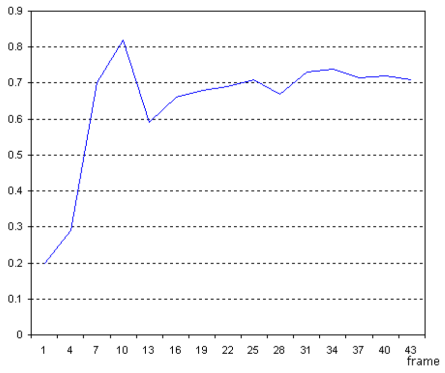


Figure 2.35: Percentage of true foreground detection

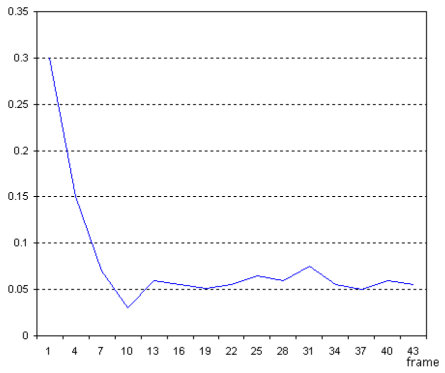


Figure 2.36: Percentage of False Background detection

2.3.8 Conclusion

In this chapter, an overview of the most important methods for motion segmentation has been presented. We proposed a new method for simultaneous motion estimation and segmentation of image sequences taken with a moving camera. In the proposed method, we model the space and time dependencies that moving objects impose on the image pixels as fields of random variables. Compared to the per-pixel background subtraction methods, our approach provides higher quality silhouettes of the foreground objects.

The feedback of the MRF optimization results to update the background model enables the acceleration of the learning of the new stationary regions and therefore avoids the fragmentation problem.

The moving object mask detected by the motion segmentation algorithm will be used in the following chapters for Simultaneous Localization And Mapping (SLAM) algorithm to exclude the moving part of the scene from the map by removing the outliers features, and for the path planning algorithms for obstacle avoidance.

Chapter 3

Navigation System

3.1 Introduction

In this chapter we present the proposed path planning system for mobile robot navigation. It is composed of a global path planning and a local path planning procedures.

The global path planning process, based on the global estimated map and the global positioning of the robot by the SLAM system (see chapter 1), calculates a path allowing the robot to reach a user defined goal. The output of this process is a set of intermediate goals between the current and final robot positions. If the global map is not known and no final goal is defined by the user, the global path planning process defines a set of arbitrary goals, based on the successive local maps, allowing a maximum exploration of the environment.

On the other hand, the local path planning process allows the robot to follow the planned path while avoiding unplanned obstacles in the SLAM system and detected moving obstacles using the motion segmentation system (see chapter 2). The position of the detected moving objects is predicted by a Kalman filter tracking procedure applied to the output object mask from the motion detection process and the used sensors for static obstacles detection are infrared and ultrasound sensors.

The local path planning procedure is based on two fuzzy logic controllers: a goal seeking controller and an obstacle avoidance controller. The goal seeking controller tries to find the optimal path to the intermediate goals (defined by the global path planning), while the obstacle avoidance controller has for mission to avoid obstacles. A command fusion scheme based on a conditioned activation for each controller arbitrates between the two behaviors.

The fuzzy logic controller for obstacle avoidance is equipped with reinforcement learning algorithm, which consists of a scalar reinforcement signal as a performance feedback from the environment enabling the navigator to tune itself.

3.2 Global Path Planning

In this section we present a method by which an autonomous robot in possession of a feature based representation of its surrounding can decide where to move so as to reach a user define goal or to best explore its environment. the algorithm we used in our work is the one developed by Bradley Hasegawa [240] in which the problem is formulated as a selective traveling salesman problem (S-TSP), then converted to an optimal constraint satisfaction problem where each point of interest is assigned a value, and each edge connecting points is assigned a cost, and solved using the Constraint Based A* algorithm to choose the most valuable and feasible ordered set of waypoints. Figure 3.1 represents the diagram of the method [240].

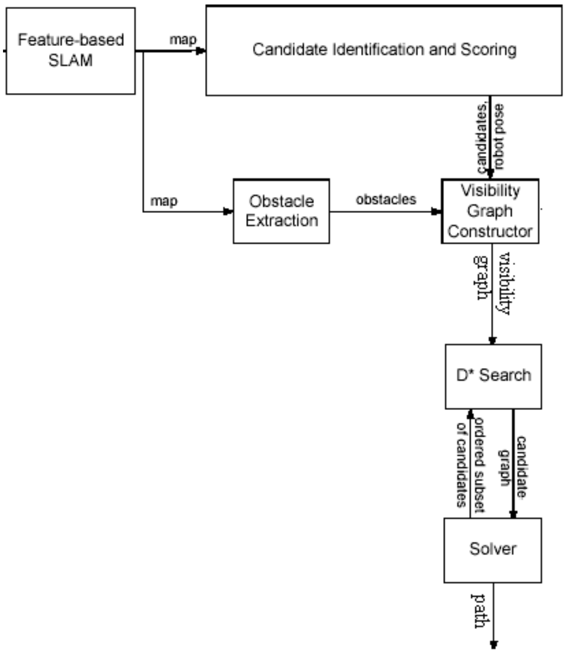


Figure 3.1: Diagram of the algorithm for path planning from a feature based map

The system uses the D* algorithm [241, 242] in order to search the visibility graph for the least cost path between every pair of candidates. D* performs an incremental search and creates the candidate graph. It saves its last calculated set of least cost paths. Then, when the map updates and a new visibility graph is built, the system tells D* what edges changed in the visibility graph. D* then searches the visibility graph only as much as it needs, in order to update its saved set of least cost paths, to accurately reflect the least cost paths through the visibility graph between every pair of candidates.

The candidate graph is passed to the solver module which returns an ordered subset of candidates to visit. The D* search module uses then its stored set of least cost paths to fill in the path between these candidates.

3.2.1 The Solver Module

The solver module takes a candidate graph as input and outputs an ordered subset of candidates to visit. There are four versions of the solver module: the greedy version, the full horizon version, the receding horizon version, and the fixed horizon version. The greedy version simply searches through all of the edges leading out of the vertex representing the robot for the edge with the lowest weight and returns the candidate at the other end of this edge. The full horizon version solves the TSP on the candidate graph and returns the resulting sequence of candidates. The receding horizon version solves the S-TSP on the candidate graph for the constant horizon length L and returns the resulting ordered subset of candidates. The fixed horizon version solves the S-TSP on the candidate graph for a horizon length of L-d, where L is a constant and d is the distance the robot has traveled since the last horizon, and returns the resulting ordered subset of candidates.

To solve the S-TSP, [240] formulates the problem as an Optimal Constraint Satisfaction Problem (OCSP). An OCSP consists of a set of variables with finite domains, a set of constraints which map each assignment to the variables to true or false, and a utility function that maps each assignment to the variables to a real number. A solution to an OCSP is an assignment to the variables that maximizes the utility function such that the constraints are satisfied.

In order to formulate the S-TSP as an OCSP, one variable is created for each candidate. Each variable can take the value of either 1 or 0. The candidate corresponding to a variable that is assigned to 1 is included in the ordered subset of candidates that is the solution to the S-TSP, while the candidate corresponding to a variable assigned to 0 is not included. Each variable also has its own utility function, called an attribute utility function. This function maps a variable assigned to 1 to the utility of the corresponding candidate and a variable assigned to 0 to zero. The utility of an assignment to the entire set of variables is equal to the sum of the values of the attribute utility functions of the individual variables. To describe the constraint, the sub-graph formed by removing every vertex corresponding to a candidate whose variable is assigned to 0 (and every edge including such a vertex) from the candidate graph is considered. The constraint over the OCSP variables is that the solution to the TSP on this sub-graph must have a length that is less than or equal to the horizon length L.

The S-TSP formulated as an OCSP is solved with the constraint-based A* algorithm [242]. Constraint-based A* is an efficient method based on A* search of enumerating the possible assignments to the variables from highest to lowest value of the utility function. To maximize the utility function for a partial assignment to the variables, it is sufficient to assign each of the unassigned variables to a value that maximizes its attribute utility function. Constraint-based A* takes advantage of this fact in order to efficiently find the next best full assignment to the variables, and in order to efficiently calculate an admissible heuristic during the search. In order to solve the S-TSP, constraint-based A* enumerates full assignments to the variables one at a time and checks the constraint for each assignment. The approach in [240] checks the constraint on an assignment by running the Concorde TSP solver on the subgraph corresponding to the assignment. The first full assignment that constraint-based A* finds is consistent must correspond to the subset of candidates in the solution to the STSP, since these assignments are generated in best first order.

3.2.1.1 Constraint-based A*

Constraint-based A* uses a form of state-space search to enumerate the variable assignments of an OCSP in best-first order. Constraint-based A* improves upon the efficiency of A* by exploiting a requirement that the utility function of an OCSP be mutually preferential independent (MPI). If each decision variable x_i has an attribute utility function $g_i(x_i)$ defined for it, and if the utility function for full assignments to the decision variables is a function of the values of the attribute utility functions, that is the utility function is of the form $G(g_1(x_1), g_2(x_2), , g_n(x_n))$, then the utility function for full assignments is a multiattribute utility function. An MPI utility function is a multiattribute utility function which can be maximized by maximizing the attribute utility of each decision variable independent of all of the other decision variables. For example, an additive utility function $G(g_1(x_1), g_2(x_2), , g_n(x_n)) = g_1(x_1) + g_2(x_2) + ... + g_n(x_n)$ is MPI because we can find the assignment that maximizes G by finding the value for x_1 that maximizes $g_1(x_1)$, the value for x_2 that maximizes $g_2(x_2)$, and so on. Constraint-based A* takes advantage of MPI utility functions in order to limit the expansion of each search tree node to only its best child, and to efficiently calculate an admissible heuristic at each node.

In the constraint-based A* framework, search states are partial or full assignments to the decision variables. In order to move from one state to the next, constraint-based A* finds a variable that has not been assigned in the current state and assigns it one of its possible values. Given state $\{x_1 = 0\}$, if we choose the next variable to be x_2, then $\{x_1 = 0\}$ can transition to $\{x_1 = 0, x_2 = 0\}$ or $\{x_1 = 0, x_2 = 1\}$. The initial state of the search tree is the state in which no decision variables have been assigned a value, and leaves of the search tree are states in which all of the decision variables have been assigned a value. The search proceeds by expanding the search node with the best estimated utility until it reaches a leaf.

3.2.2 Environment exploration

In case of environment exploration, the robot should try (i) to visit areas that are open and sparsely populated with features ; (ii) to stay away from areas that have already been visited ; (iii) to visit areas that are close to and reachable from the current position.

3.3 Local Path Planning

3.3.1 Fuzzy Logic and Fuzzy Control

Fuzzy Logic (FL) is based on the concept of fuzzy sets, proposed by Lotfy Zadeh in 1965, in which membership is expressed in varying degrees of truth. That is, FL is a multivalued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white, etc. Notions like rather warm or pretty cold can be formulated mathematically and processed by computers.

Fuzzy logic is a way of reasoning that can cope with uncertain, imprecise, or partial information. It is often confused with probability. The difference between probability and fuzzy logic is clear when we consider the underlying concept that each attempts to model. Probability is concerned with the undecidability in the outcome of clearly defined and randomly occurring events, while fuzzy logic is concerned with the ambiguity or undecidability inherent in the description of the event itself. Fuzziness is usually expressed as ambiguity rather than imprecision or uncertainty and remains a characteristic of perception as well as concept.

Fuzzy control is one of the important applications of fuzzy theory. It works rather different than conventional controllers; expert knowledge is used instead of differential equations to describe a system. This knowledge can be expressed in a very natural way using linguistic variables, which are described by fuzzy sets.

The basic configuration of a Fuzzy Logic Controller comprises four principal components [208, 209, 210, 211, 223]: Fuzzifier, Defuzzifier, decision making logic, and knowledge base component (Figure 3.2):

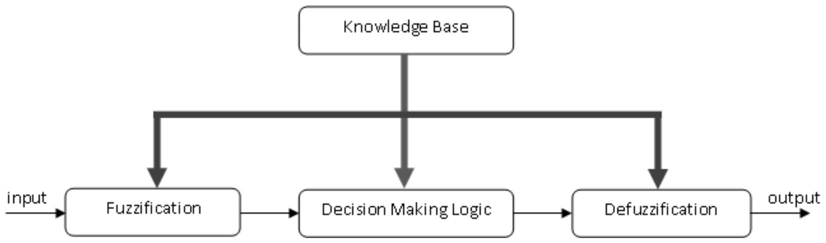


Figure 3.2: Fuzzy Controller block diagram

Input variables $x_i, i = 1, ..., n$ of the controller are state variables of the plant to be controlled. These variables are based on measurements at instant t and have somewhat uncertain values (for example distance between a robot and an obstacle, the speed of the robot, the direction of moving obstacles,...).

The controller should be able to determine control values $y_j, j = 1, ..., m$ (for example a rotation angle, change of the speed, ...) based on the current measurements and the prior knowledge of an expert. The prior knowledge is made available as a set of linguistic rules of the form:

$$R^r: \quad \textbf{IF } x_1 \text{ is } A_1^r \text{ AND/OR} \ldots \text{AND/OR } x_n \text{ is } A_n^r \textbf{ THEN } y_1 \text{ is } B_1^r, \ldots, y_m \text{ is } B_m^r$$

A_i^r (respectively B_j^r) are linguistic terms of input (respectively output) variables described by l membership functions $\mu_{i,1}, \ldots, \mu_{i,l}$ (respectively p membership functions $\eta_{j,1}, \ldots, \eta_{j,p}$) in their universes of discourse. AND and OR are fuzzy operators [208, 210].

The fuzzification interface transforms the input variables (crisp) to fuzzy sets or linguistic terms by determining their membership degree.

3.3.1.1 Knowladge Base

The knowledge base provides necessary definitions used to characterize the fuzzy control rules and the fuzzy data manipulations in the fuzzy logic controller. It contains the fuzzy sets defined on the input and output universes of discourse and a set of decision making rules.

The designer of a fuzzy controller is inevitably faced with two specific questions when building the set terms: (i) How does one determine the shape of the membership functions corresponding to the fuzzy sets? and (ii) How many sets are necessary?

According to fuzzy set theory, the choice of these parameters is subjective and there are few empirical rules for helping defining fuzzy sets [212]: (i) A fuzzy set should be sufficiently wide to allow measurement noise handling, and (ii) A certain amount of overlap between fuzzy sets is desirable. If there is a gap between two sets no rules will fire for values in the gap. Consequently the controller function is not defined. The most common shapes of membership functions are triangular, trapezoidal, and gaussian (bell curve) functions [212].

The designer of the fuzzy controller should also define the fuzzy operators for aggregation, activation, and accumulation which will be used in the inference engine.

3.3.1.2 Fuzzification

The first step in Fuzzy Logic Control is to fuzzify the input variables into linguistic variables linked to fuzzy sets. This step consists of determining the degree of membership of each input to the fuzzy sets via membership functions.

Example

For the fuzzification of the car speed value $x_0 = 70km/h$ the two membership functions μ_A and μ_B from Figure 3.3 can be used, which characterise a low and a medium speed fuzzy set, respectively. The given speed value of $x_0 = 70km/h$ belongs with a grade of $\mu_A(x_0) = 0.75$ to the fuzzy set 'low' and with a grade of $\mu_B(x_0) = 0.25$ to the fuzzy set 'medium'.

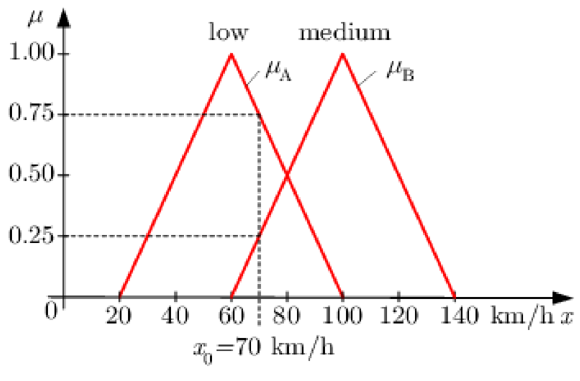


Figure 3.3: Example of fuzzification

3.3.1.3 Inference Engine

This part of a fuzzy controller combines the facts obtained from the fuzzification with the rule base and conducts the fuzzy reasoning process. It is done in tree steps [212]:

Aggregation: The aggregation operation is used when calculating the *degree of fulfillment* or *firing strength* α_r of the condition of a rule r. It combines the membership degrees of the input variables using the fuzzy operators (AND,OR). The commonly used operators for the fuzzy AND are *min* or *product* and for the fuzzy OR are *max* or *algebraic sum*.

Activation: The activation of a rule is the deduction of the conclusion, possibly reduced by its firing strength. It uses the min or product operators. Both operators work well, although the product operators results in a slightly smoother control as it preserves the initial shape of the output membership curve.

Accumulation: All activated conclusions are accumulated , using the max or sum operations.

The Inference engine is designed by its activation and accumulation operators and the most used inference engine mechanisms are the Max-Min and Max-product.

Example

For a fuzzy system with two inputs x_1 and x_2, one output y, and two fuzzy rules (the linguistic term S stands for 'small', M for 'medium', L for 'large', N for 'negative' and P for 'positive')

$$\begin{array}{l}(1)\ \text{IF } (x_1 = \text{P})\ \text{AND } (x_2 = \text{M})\ \text{THEN } (y = \text{M})\\ (2)\ \text{IF } (x_1 = \text{N})\ \text{OR } (x_2 = \text{S})\ \text{THEN } (y = \text{S})\end{array}$$

The inference is evaluated as follows: Each rule contains two premises, which are differently connected. In rule 1 the connective operation is the intersection, which can be performed by the min operation ($\mu^1 = \min(\mu_{A_{1,1}}(x_1), \mu_{A_{1,2}}(x_2))$, and in rule 2 the premise is a union of the two premises, which can be performed by the max operation ($\mu^2 = \max(\mu_{A_{2,1}}(x_1), \mu_{A_{2,2}}(x_2))$. $\mu_{A_{1,1}}, \mu_{A_{1,2}}, \mu_{A_{2,1}}$, and $\mu_{A_{2,2}}$ are the membership functions defining the fuzzy partitions of the inputs variables universes of discourses.

The membership functions of the conclusion of each rule is determined using the degree of fulfillment of the corresponding rule by applying either the Max-Min inference method or the Max-Prod inference method as shown in Figure 3.4.

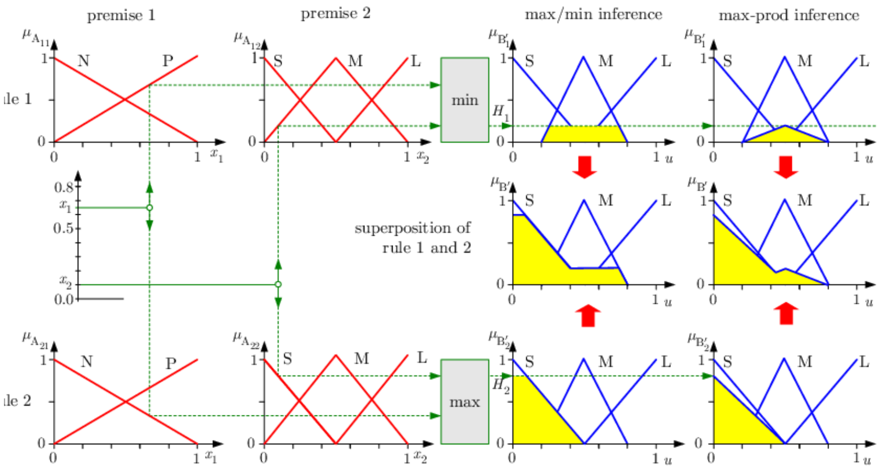


Figure 3.4: Example of the application with two premises with (a) max/min inference and (b) max-prod inference

3.3.1.4 Defuzzification

The resulting fuzzy set from the inference process must converted to a crisp value that can be sent to the controlled system as a control signal. This operation is called defuzzification. There are several defuzzification methods [208, 209, 210], the most commonly used are *maximum* and *center of gravity* techniques.

Maximum Defuzzification Technique [210, 220]

This method gives the output with the highest membership function (equation 3.1). This defuzzification technique is very fast but is only accurate for peaked output.

$$y_j^* / \quad \mu_{B'}(y^*) \geq \mu_{B'}(y), \text{ for all } y \in Y \tag{3.1}$$

where y^* is the defuzzified value, $\mu_{B'}(y)$ is the degree of membership of the output y to the fuzzy set B' (obtained after Accumulation) and Y is the output universe of discourse.

Centre of gravity (COG) technique [208, 210, 220]

This is the most commonly used technique for defuzzification and is also known as center of gravity or center of area (equation 3.2). This technique was developed by Sugeno in 1985. The disadvantage of this technique is that it is computationally difficult for complex membership functions.

$$y_j^* = \frac{\int \mu_{B'}(y)y}{\int \mu_{B'}(y)} \tag{3.2}$$

3.3.1.5 Takagi-Sugeno controller

In the rule based systems, fuzzy sets are used both in the premises and in the conclusions. This kind of inference is called Mamdani inference [211, 220, 223]. A modified inference scheme, developed by Takagi and Sugeno [217, 218, 219], represents the conclusions by functions of the inputs data:

$$R^r : \textbf{IF}\ x_1\, is\, A_1^r \text{ AND/OR} \ldots \text{AND/OR } x_n\, is\, A_n^r\ \textbf{THEN}\ y_j^r\ = F_j^r(x_1, ..., x_n),\ (j = 1..m)$$

The function F_j^r represents a direct mapping from the input space $X_1 \times X_2 ... \times X_n$ with the input values $x_1, ..., x_n$ to the output space Y.

The order-0 Takagi-Sugeno controller is a particular version where the output in the rule decision part is given by a real number:

$$R^r : \textbf{IF}\ x_1\, is\, A_1^r \text{ AND/OR} \ldots \text{AND/OR } x_n\, is\, A_n^r\ \textbf{THEN}\ y_j^r\ = c_j^r,\ (j = 1..m)$$

The final output is determined as a weighted mean value over all R rules according to:

$$y_j^* = \frac{\sum_r \mu^r F_j^r(x_1, ..., x_n)}{\sum_r \mu^r} \tag{3.3}$$

where μ^r is the degree of fulfillment of rule r calculated by combining input membership values using fuzzy AND and OR operators.

3.3.1.6 fuzzy logic implementation

Although there have been many successful applications of fuzzy control, control designers still need to face two major obstacles in implementing a fuzzy control. The first is the acquisition of fuzzy rules, and the second is the definition of optimal parameters of membership functions for the linguistic rules. However, there exists no guidelines for designing fuzzy controllers. Manual designs may require long period of trial and error and much input from experts are required. This provides the motivation for adaptive fuzzy control, where the focus is on the automatic on-line synthesis and tuning of fuzzy controller parameters. Several parameter identification techniques have been proposed for tuning fuzzy controllers. We can find methods based on supervised learning [212, 213, 214, 215, 216, 231], and others on unsupervised learning [231, 234, 235].

Obtaining effective and quality training data in the case of our application for mobile robot navigation in unknown environment is not an easy task and probably impossible to represent the varying environment. For these reasons, we opt for solving the problem of fuzzy controller adaptation using an unsupervised learning paradigm known as reinforcement learning (RL) [236, 237, 238] (see section 3.3.2).

3.3.1.7 Application of Fuzzy Logic for mobile robot navigation

The development of techniques based on fuzzy logic (FL) for autonomous navigation in real-world environments received considerable interest in the current research on robotics. The existent methods using FL for robot navigation are often hierarchical behavior-based methods [220, 221, 222, 223, 224, 225, 226], where a set of simple behavior processing units (such as "seek the goal", "ovoid obstacles",...,) are coordinated with an arbitration strategy and command fusion process (figure 3.5).

These behaviors work independently and are not activated all together, each behavior is only activated when needed. The *arbitration strategy* decides which behaviors should be activated depending on the state of the environment [227]. Several behaviors may also be simultaneously activated. In this case, *a command fusion* is needed to combine the results of activated behaviors.

Algorithms for mobile robot navigation based on fuzzy logic differ in the number and type of behaviors and the used combination strategies.

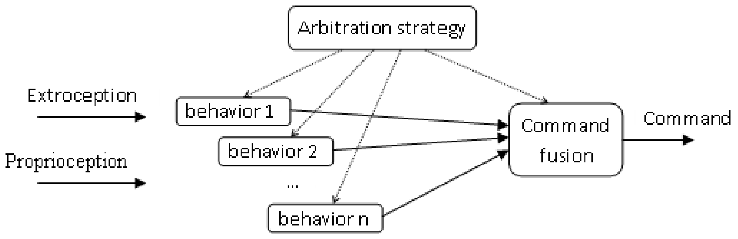


Figure 3.5: Behavior-based navigation strategy

The command fusion process can be a simple strategy as switching system, where only the behavior with high importance is used and other behaviors are ignored or summation process, where the resulting command is a weighted sum of the activated behaviors. It can be also more complicated strategy as fuzzy fusion process, where the output of activated behaviors are combined by a fuzzy operator [227, 228].

Other authors proposed methods based on adaptive fuzzy controllers for mobile robot navigation allowing the system to acquire knowledge and adapt it behavior by interacting with the environment. These methods have also an hierarchical architecture and depend on the used learning process [224, 229, 230, 231, 232, 233, 234, 235]. They can be classified into two classes: navigation methods with online learning and navigation methods with off-line learning.

Methods with off-line learning for mobile robot navigation are based, usually, on supervised learning algorithms, where a set of possible situations and corresponding decisions are presented to the robot [220, 224]. These techniques are well suited for some classes of environment. Where the robot is called to navigate in similar architecture without hard changing of the environment (new type of obstacles, moving obstacles,...).

In the other side, methods with online learning have the possibility to adapt themselves to new situations in their environment [229, 235], but the chose of the learning policy is not easy and therefor their utilization is limited to the environment where possible errors are tolerated, for example with navigation methods based on reinforcement learning, bumping or approaching an obstacle or take a false way during the learning should be tolerated.

3.3.2 Reinforcement learning for fuzzy controller

3.3.2.1 Reinforcement learning

Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions an agent ought to take in those states [236, 237]. The environment, in return, provides a reward which can be positive or negative. The reinforcement learning algorithm consists to find a policy for maximizing cumulative reward for the agent over the course of the problem.

The environment is typically formulated as a finite-state Markov decision process (MDP), and reinforcement learning algorithms for this context are highly related to dynamic programming techniques. State transition probabilities and reward probabilities in the MDP are typically stochastic but stationary over the course of the problem.

Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of unknown territory) and exploitation (of current knowledge). The exploration v.s. exploitation tradeoff in reinforcement learning has been mostly studied through the multi-armed bandit problem [236].

Formally, the basic reinforcement learning model consists of:

- a set of environment states $S = \{s_t\}$,
- a set of actions $A = \{a_t\}$, and
- a set of scalar "rewards" r_t in $\mathbb{R}$.

At each time t, the agent perceives its state $s_t \in S$ and the set of possible actions $A(s_t)$. It chooses an action $a_t \in A(s_t)$ and receives from the environment the new state s_{t+1} and a reward r_{t+1}. Based on these interactions, the reinforcement learning agent must develop a policy $\pi : S \rightarrow A$ which maximizes the quantity $R = \sum_t \mathrm{r}_t$ for MDPs which have a terminal state, or the quantity $R = \sum_t \gamma_t \mathrm{r}_t$ for MDPs without terminal states (where γ_t is some "future reward" discounting factor between 0 and 1).

After we have defined an appropriate return function to be maximised, we need to specify the algorithm that will be used to find the policy with the maximum return. There are two main approaches, the value function approach and the direct approach.

The direct approach entails the following two steps [236, 237]

1. For each possible policy, sample returns while following it.
2. Choose the policy with the largest expected return.

One problem with this is that the number of policies can be extremely large, or even infinite. Another is that returns might be stochastic, in which case a large number of samples will be required to accurately estimate the return of each policy.

The problems with the direct approach might be ameliorated if we assume some structure in the problem and somehow allow samples generated from one policy to influence the estimates made for another. Value function approaches do this by only maintaining a set of estimates of expected returns for one policy π (usually either the current or the optimal one). In such approaches one attempts to estimate either the expected return starting from state s and following π thereafter,

$$V(s_t) = E[R|s_t, \pi] \tag{3.4}$$

or the expected return when taking an action a_t in state s_t and following π thereafter,

$$Q(s_t, a_t) = E[R|s_t, \pi] \tag{3.5}$$

We can choose optimal actions by simply choosing the action with the highest value at each state. In order to do this using V, we must either have a model of the environment, in the form of probabilities $P(s_{t+1}|s_t, a)$ to transit from state s_t to s_{t+1} while applying the action $a_t = \pi(s)$, which allow us to calculate Q simply through:

$$Q(s_t, a_t) = \sum_{s_{t+1}} V(s_{t+1}) P(s_{t+1}|s_t, a_t) \tag{3.6}$$

or we can employ the so-called Actor-Critic methods [236], in which the model is split into two parts: the critic, which maintains the state value estimate V, and the actor, which is responsible for choosing the appropriate actions at each state.

Given a fixed policy π, Estimating $E[R|.]$ for $\gamma_t = 0$ is trivial, as one only has to average the immediate rewards. The most obvious way to do this for $\gamma_t > 0$ is to average the total return after each state.

The above methods can converge to the optimal policy. This is usually done by following a policy *pi* that is somehow derived from the current value estimates, i.e. by choosing the action with the highest evaluation most of the time, while still occasionally taking random actions in order to explore the space.

An alternative method to find the optimal policy is to search directly in the policy space. Policy space methods define the policy as a parameterized function $\pi(s_t, \theta_t)$ with parameters θ_t. Commonly, a gradient method is employed to adjust the parameters. However, the application of gradient methods is not trivial, since no gradient information is assumed.

3.3.2.2 Q learning

It exists several approaches for reinforcement learning without models. Some are based on policy iteration (see section 3), such as the Actor Critic Learning [239], and others on value iteration, such as Q-Learning or SARSA. The Q-Learning, proposed by Watkins [238], is perhaps the more popular of Reinforcement algorithms, for its simplicity.

In this algorithm, the agent observes the present state, s_t, and executes an action, a_t, according to the evaluation of the return that it makes at this stage. It updates its evaluation of the action's value while taking in account, a) the immediate reinforcement, r_t, and b) the estimated value of the new state, $V(s_{t+1})$, that is defined by:

$$V(s_{t+1}) = \max_{a_t} Q(s_{t+1}, a_t) \tag{3.7}$$

The updates are made by:

$$Q(s_{t+1}, a_t) = Q(s_t, a_t) + \beta\,(\mathrm{r}_t + \gamma V(s_{t+1}) - Q(s_t, a_t)) \tag{3.8}$$

β is a learning rate such that $\beta \to 0$ as $t \to \infty$.
This equation can be written:

$$Q(s_{t+1}, a_t) = (1-\beta) Q(s_t, a_t) + \beta\,(\mathrm{r}_t + \gamma V(s_{t+1})) \tag{3.9}$$

This update corresponds to the barycenter of the old and new evaluations, weighted by β.

The evaluations of the Q-values, are independent of the policy followed by the agent. The later can follow any policy, while continuing to construct correct evaluations of the action's value.

3.3.2.3 Fuzzy controller optimization using Q-learning

In this paragraph we consider the order-0 Takagi-Sugeno fuzzy controller FC, which is used in our mobile robot navigation. The controller has n inputs and one output variables and is described by a set of N fuzzy rules such as:

$$R^r : \textbf{IF } x_1 \; is \; A_1^r \text{ AND/OR} \ldots \text{AND/OR } x_n \; is \; A_n^r \textbf{ THEN } y^r = c^r, \qquad r = 1, .., N \tag{3.10}$$

We suppose a partition of input universes of discourses with triangular membership functions, $\mu_{A_i^r}$, and the conjunction in the premise part of the rules is the product operator. These constraints are not restrictive but allow model simplification.

Let $\mathbf{x_t} = (x_1, ..., x_n)$ be an input vector. The output $y(\mathbf{x}) = FC(\mathbf{x})$ is then given by:

$$y(\mathbf{x}) = \frac{\sum\limits_r \mu^r(\mathbf{x}).c^r}{\sum\limits_r \mu^r(\mathbf{x})} \tag{3.11}$$

where $\mu^r(\mathbf{x}) = \prod\limits_i \mu_{A_i^r}(x_i)$ is the degree of fulfillment of rule r using the product for the AND conjunction.

If we consider strong fuzzy set partitions (50% overlapping of the fuzzy sets) then, $\sum\limits_r \mu^r(\mathbf{x}) = 1,\ \forall \mathbf{x} \in \mathbf{X}$ and

$$y(\mathbf{x}) = \sum_r \mu^r(\mathbf{x}).c^r \tag{3.12}$$

We call the premise part of a rule r: $\mathbf{s}^r = (x_1\, is\, A_1^r$ AND ... AND $x_n\, is\, A_n^r)$ a state of the system. For the FC adaptation using a Q-learning algorithm we consider a 0-order Takagi-Sugeno fuzzy controller where each rule r has K possible conclusions a_k^r with quality factor q_k^r ($k = 0..K$). a is used for rule conclusions instead of c in Eq(3.21) for a coherent notation with reinforcement learning algorithm.

$$\begin{array}{lrl} R^r : \textbf{If } \mathbf{x}\, is\, \mathbf{s}^r \textbf{ then} & & y = a_1^r \textbf{ with } q_1^r \\ & \textbf{or} & y = a_2^r \textbf{ with } q_2^r \\ & \textbf{or} & \quad ... \\ & \textbf{or} & y = a_K^r \textbf{ with } q_K^r \end{array}$$

Using Eq(3.12) for an input vector $\mathbf{x}_t$ at time t, one can define the inferred action, $A(\mathbf{x}_t)$, and its associated quality, $Q(\mathbf{x}_t, A(\mathbf{x}_t))$, by:

$$A(\mathbf{x}_t) = \sum_r \mu^r(\mathbf{x}_t).a_{k(r)}^r \tag{3.13}$$

$$Q(\mathbf{x}_t, A(\mathbf{x}_t)) = \sum_r \mu^r(\mathbf{x}_t).q_{k^r}^r \tag{3.14}$$

with k^r being the subscript of the chosen conclusion for the rule r.

Conclusion Selection

During the exploration phase of the learning algorithm, the K possible conclusions are not modified as in supervised learning. They are analyzed and their q-values are updated. In the exploitation phase of the fuzzy controller, a greedy policy will take the best conclusion among the proposed ones; their initial choice is therefore important. If there is no knowledge about the process to control, the K possible conclusions are equally distributed in the output universe of discourse. The q-values are initialized to zero to give the same chance for all possible conclusions.

Let $\hat{q}_t^r$ be the maximum q-value for the rule r at time t:

$$\hat{q}_t^r = \max_k(q_{k,t}^r) \tag{3.15}$$

The value of the new state $\mathbf{x}_{t+1}$ after application of action $A(\mathbf{x}_t)$ corresponding to the maximum q-value at state $\mathbf{x}_t$ is then (see equation 3.7)

$$V(\mathbf{x}_{t+1}) = \sum_r \mu^r(\mathbf{x}_t).\hat{q}_t^r \tag{3.16}$$

and the updates are given by:

$$\Delta q_{k,t}^r = \begin{cases} \beta\left(\mathrm{r}_t + \gamma V(s_{t+1}) - Q(s_t, a_t)\right)\mu^r(\mathbf{x}_t) & \textit{if } k = k^r \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

with r_t is the system reward factor and β and γ are the learning parameters (see paragraph 3.3.2.2).

3.3.3 Proposed Navigation Algorithm

The purpose of the proposed local navigation algorithm is to follow the global path defined by the global path planing algorithm while avoiding the unplanned obstacles in the SLAM algorithm (moving obstacles). The inputs of the navigation algorithm are the global robot position, the speed of the robot, the global path as a set of intermediate target points, the data from ultrasonic and infrared sensors, and the path and position of the moving obstacles detected by the motion segmentation algorithm (chapter 2) and tracked by a Kalman filter (see 3.3.3.1.2 of this chapter).

The proposed navigation system include two behaviors implemented by fuzzy logic controllers: Goal Seeking behavior and Obstacle Avoiding behavior. The behaviors are coordinated by a conditioned activation fusion scheme (figure 3.3.3).

The proposed algorithm has been adapted and implemented for a car-like robot "ROBUDEM" equipped with sonar sensors and a vision camera.

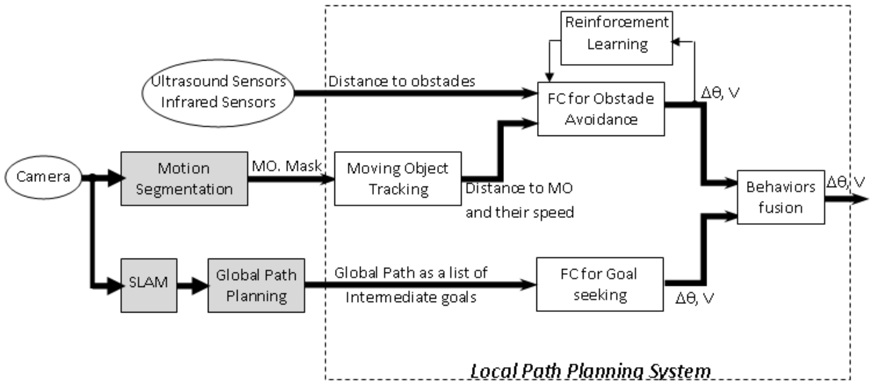


Figure 3.6: Local Navigation system

3.3.3.1 Sensory data

3.3.3.1.1 Data from US and Infrared sensors

The "ROBUDEM" robot is equipped with eight sonar sensors grouped in 4 pairs as shown in figure 3.7.

For robot navigation the sensors are grouped in five groups (figure 3.7).

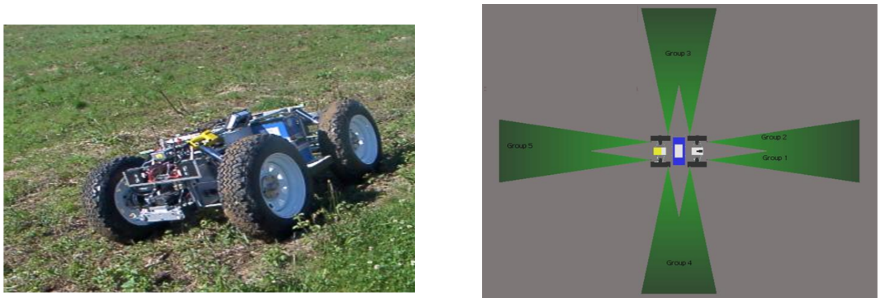


Figure 3.7: Groups of Sensors

The distance d_i measured by the ith sensor group is expressed as:
$For \quad i = 1, ..., 5,$

$$d_i = min\{d_{imax}, min_j\{\{d_{ij}\}\}; \quad j = 1..2 \tag{3.18}$$

where d_{ij} is the distance measured by the jth sensor of the sensor group i.

3.3.3.1.2 Data from vision system

The position with respect to the camera's optic axis and the speed of the detected moving objects (see chapter 2) is estimated by a Kalman filter based tracking procedure. Whereas the position of these objects with respect to the robot (depth position) is estimated by epipolar geometry applied to detected features on the moving object.

Moving Object Tracking

A moving obstacle is defined by a bounding box around the estimated object mask and its position is the center of the bounding box.The estimation of the 2D position and speed of moving object from the motion segmentation blobs could be quite noisy. In our application we use a Kalman filter tracking procedure to estimate to true 2D position and speed of the moving objects. The size of the bounding is defined only from the segmentation process and is not estimated by the Kalman filter.

To use Kalman filter for object tracking we assume that the motion of the objects is almost constant over frames. The commonly used model for 2D tracking can be found in [56]. In this model, the state vector is the position (x, y) of the center of the moving object bounding box. In our application, this model is augmented by the speeds u and v along the X and Y axis, respectively, of the tracked objects.

The dynamic model giving the system state at time t function of the state at $t - 1$ is:

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathcal{N}(0, \mathbf{Q}_t) \tag{3.19}$$

where the state transition matrix is derived from the theory of motion under constant speed which can be expressed with equations:

$$\begin{aligned} p_t &= p_{t-1} + v_t \Delta t \\ v_t &= v_{t-1} \end{aligned}$$

p, v, and Δt are position, velocity and time step, respectively.

$\mathbf{F}_t$ is given then by:

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The process noise is assumed to be drawn from a zero mean multivariate normal distribution with covariance $\mathbf{Q}_t$, $\mathcal{N}(0, \mathbf{Q}_t)$.

$$\mathbf{Q}_t = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ q^2 & 0 \\ 0 & q^2 \end{bmatrix}$$

At time t an observation (or measurement) $\mathbf{z}_t$ of the true state $\mathbf{x}_t$ is made according to

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathrm{v}_t \tag{3.20}$$

where $\mathbf{H}_t$ is the observation model which maps the true state space into the observed space and v_t is the observation noise which is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_t$

$$\mathrm{v}_t \sim \mathcal{N}(0, \mathbf{R}_t)$$

The measurements are the positions of the blobs center in the object mask and are obtained from the motion segmentation process, so

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The initial state, and the noise vectors at each step $\{x_0, w_1, ..., \mathrm{w}_t, \mathrm{v}_1 ... \mathrm{v}_t\}$ are all assumed to be mutually independent.

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. The state of the filter is represented by two variables:

$\hat{\mathbf{x}}_{t|t}$ the estimate of the state at time t;
$\mathbf{P}_{t|t}$ the error covariance matrix (a measure of the estimated accuracy of the state estimate).

Each time a new moving object is detected, we initiate a new Kalman Filter and use the measured position of the detected blob as the expected value of the position coordinates for the initial state. From a single frame, we do not know the velocity of the moving object, but since it could be traveling in any direction we suppose that it initial velocity is 0. For the covariance matrix, there is no reason to suppose that the position and the velocity are correlated, so P_0 is block diagonal matrix. The uncertainty about the initial position coordinates is really the same as our measurement error (i.e., how really we can localize the blob). In our implementation we used an accuracy error equal to a few pixels for each coordinate (3 or 4 pixels) and for the velocity accuracy an error of 15 to 20 pixels is used.

$$\mathbf{x}_0 = \begin{bmatrix} x_0 \\ y_0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{P}_0 = \begin{bmatrix} 4^2 & 0 & 0 & 0 \\ 0 & 4^2 & 0 & 0 \\ 0 & 0 & 20^2 & 0 \\ 0 & 0 & 0 & 20^2 \end{bmatrix}$$

The Kalman filter has two distinct phases (see appendix C): Predict and Update. The predict phase uses the estimate from the previous timestep to produce an estimate of the current state. In the update phase, measurement information from the current timestep is used to refine this prediction to arrive at a new, (hopefully) more accurate estimate.

3.3.3.1.2.1 Depth estimation

To estimate the position of the detected moving object toward the robot (depth of the objects), the epipolar geometry procedure is applied on the detected features on these objects (see appendix A for more details).

For more navigation security, the considered distance "robot-moving object" is the smallest estimated distance corresponding for the closest feature.

3.3.3.1.3 results of the tracking process Figure 3.8 and 3.9 show results for the tracking procedure. The track is represented by a surrounding box around the moving object and a path of the box center at each time-lapse.

In the case of 3.9, each time the moving objects meet, an object is considered as lost and the two objects are merged in one big object, and when the objects separate, an new object is created.

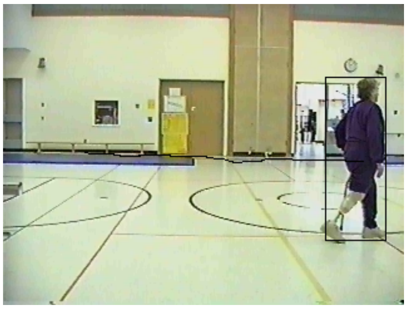


Figure 3.8: Tracking of moving objects in the scene

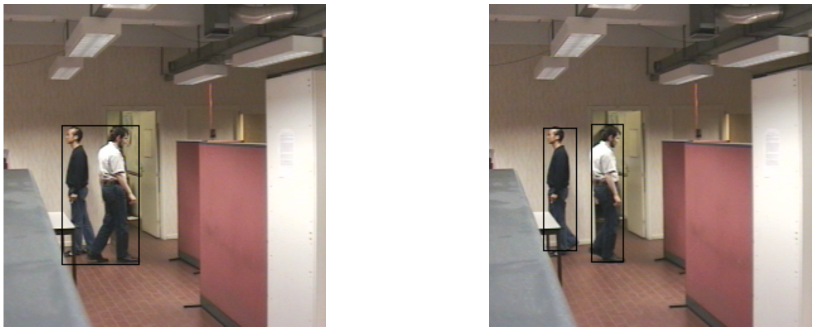


Figure 3.9: Tracking of moving objects in the scene

3.3.3.2 Goal seeking behavior

This fuzzy controller calculates the turn angle allowing the robot to reach the intermediate targets defined by the global path planing process. The model of the controller is based on the distance of the robot to the goal and the goal direction. The fuzzy logic controller is of a "Sugeno" type [217, 218, 219] described by a set of N fuzzy rules:

$$R^r : \textbf{IF} \ x_1 \, is \, A_1^r \ \text{AND} \ x_2 \, is \, A_2^r \ \textbf{THEN} \, y^r = c^r, \qquad r = 1, .., N \tag{3.21}$$

The inputs of the goal seeker are the goal distance (x_1) and the goal direction (x_2). The goal distance ranges between [0,7] and the goal direction between [$-180\,°$,$180\,°$]. A_1^r and A_2^r are the fuzzy partitions defined on the input universes of discourse (see next paragraph)
The output y of the controller is the turn angle of the robot, which ranges between $-180\,°$ and $180\,°$.

3.3.3.2.1 Fuzzification of inputs and output variables
Input 1: Goal direction

The direction of the goal is fuzzified into 11 Gaussian fuzzy sets:

- Back Right (BR): The center of this fuzzy set is $-180\,°$ and its variance is $19\,°$.
- Oblique Back Right (OBR): The center of this fuzzy set is $-135\,°$ and its variance is $19\,°$.
- Right (R): The center of this fuzzy set is $-90\,°$ and its variance is $19\,°$.
- Oblique Front Right (OFR): The center of this fuzzy set is $-45\,°$ and its variance is $9.5\,°$.
- Front Right (FR): The center of this fuzzy set is $-22.5\,°$ and its variance is $9.5\,°$.
- Front (F): The center of this fuzzy set is $0\,°$ and its variance is $9.5\,°$.
- Front Left (FL): The center of this fuzzy set is $22.5\,°$ and its variance is $9.5\,°$.
- Oblique Front Left (OFL): The center of this fuzzy set is $45\,°$ and its variance is $9.5\,°$.
- Left (L): The center of this fuzzy set is $90\,°$ and its variance is $19\,°$.
- Oblique Back Left (OBL): The center of this fuzzy set is $135\,°$ and its variance is $19\,°$.
- Back Left (BL): The center of this fuzzy set is $180\,°$ and its variance is $19\,°$.

The membership functions of the first input variable ”goal direction” are depicted in Figure (3.10).

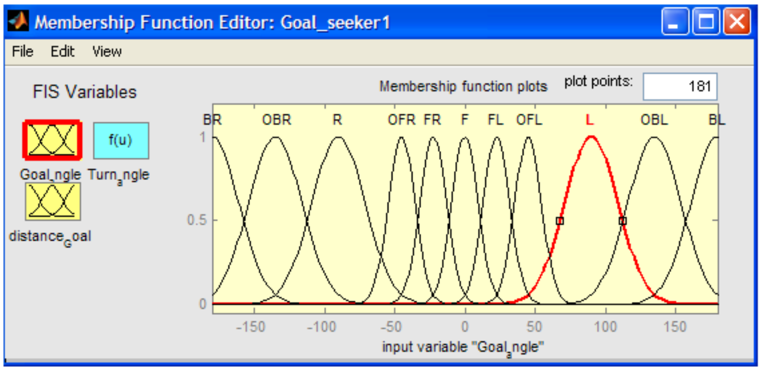

Membership Function Editor: Goal_seeker1
File Edit View
FIS Variables
Goal_angle
Turn_angle
f(u)
distance_Goal
Membership function plots
plot points: 181
BR OBR R OFR FR F FL OFL L OBL BL
1
0.5
0
-150 -100 -50 0 50 100 150
input variable "Goal_angle"

Figure 3.10: Membership functions for the goal angle

Input 2: Goal distance

The distance of the goal to the robot is fuzzified into four Gaussian fuzzy sets:

- Very Near (VN): the center of this fuzzy set is 1 and its variance is 0.525.
- Near (NR): the center of this fuzzy set is 2 and its variance is 0.525.
- Medium (M): the center of this fuzzy set is 3.5 and its variance is 0.525.
- Far (FR): the center of this fuzzy set is 5 and its variance is 0.525.

The membership functions of the second input variable ”goal distance” are depicted in Figure (3.11).

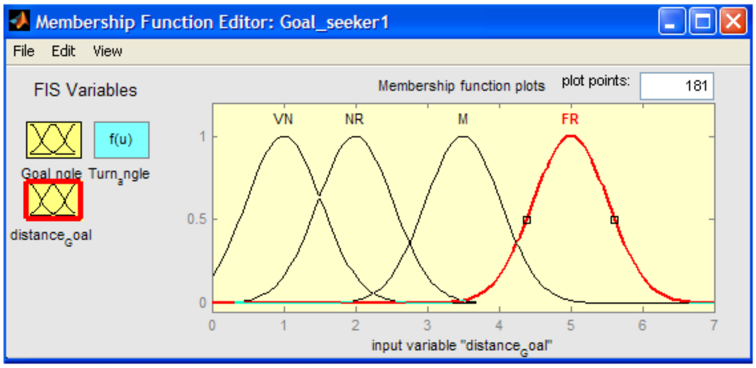

Membership Function Editor: Goal_seeker1
File
Edit
View
FIS Variables
f(u)
Goal ngle
Turn ngle
distance oal
Membership function plots
plot points: 181
VN
NR
M
FR
input variable "distance oal"

Figure 3.11: Membership functions for the goal distance

Output: Turn angle

Since we want our robot to choose a smooth path to its goal, we have opted for a fuzzy distribution of the turn angle such that it covers 360 °.
Since the model of our fuzzy controller is a "Sugeno", the output membership functions are of a constant type.
The turn angle of the robot is fuzzified into 11 constant fuzzy sets:

- Back Right (BR): The turn angle is $-180\,°$.
- Oblique Back Right (OBR): The turn angle is $-135\,°$.
- Right (R): The turn angle is $-90\,°$.
- Oblique Front Right (OFR): The turn angle is $-45\,°$.
- Front Right (FR): The turn angle is $-22.5\,°$.
- Front (F): The turn angle is $0\,°$.
- Front Left (FL): The turn angle is $22.5\,°$.
- Oblique Front Left (OFL): The turn angle is $45\,°$.
- Left (L): The turn angle is $90\,°$.
- Oblique Back Left (OBL): The turn angle is $135\,°$.
- Back Left (BL): The turn angle is $180\,°$.

3.3.3.2.2 Fuzzy Inference Engine and Fuzzy Rule Base

Each fuzzy rule corresponds to a fuzzy relation. The product-operator chosen is "Min".
The goal seeker functions as follows: If the target is located on the left side of the robot, then it has to reach it by turning left and vice-versa. The fuzzy rule base of the goal seeker is presented in Table (3.1):

G_ang\G_dist	FR	M	NR	VN
BR	OBR	OBR	BR	BR
OBR	R	R	OBR	OBR
R	OFR	OFR	R	R
OFR	FR	FR	OFR	OFR
FR	F	FR	FR	FR
F	F	F	F	F
FL	F	FL	FL	FL
OFL	FL	FL	OFL	OFL
L	OFL	L	L	L
OBL	L	L	OBL	OBL
BL	OBL	OBL	BL	BL

Table 3.1: Inference Table for the Goal Seeking Behavior

Figure (3.12) shows a graphical representation of dependency of the turn angle on the goal distance and the goal direction.

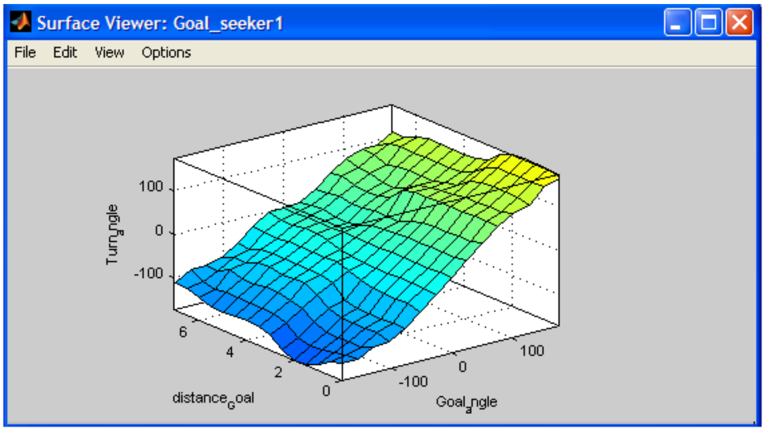


Figure 3.12: Turn angle as a function of goal distance and goal direction

We can see from this figure that the turn angle increases with the increase of the goal direction. This is logical since the robot has to reach the goal direction to get to its target.

It's clear also that the dependency of the turn on the goal direction is more important than its dependency on the goal distance. If the distance is very short, then the turn angle increases rapidly with the goal direction and if this distance is long, the dependency of the turn angle on the goal direction is less important.

3.3.3.3 Obstacle Avoidance behavior

The 0-order Takagi-Sugeno (eq.3.22) fuzzy controller used for obstacle avoidance is based on the distances to the obstacle detected by the sonar and infrared sensors as well as the estimated distance to the detected moving obstacles.

$$\begin{aligned} R^r : \textbf{IF}\ x_1\, is\, A_1^r \text{ AND } x_2\, is\, A_2^r \text{ AND } \ldots \text{ AND } x_8\, is\, A_8^r \\ \textbf{THEN}\ y_1^r = c_1^r, \text{ AND } y_2^r = c_2^r \\ r = 1, .., N \end{aligned} \tag{3.22}$$

The inputs $x_1, x_2, ..., x_5$ are distance to obstacles detected by the ultrasound and infrared sensors, x_6 is the goal direction, x_7 is the estimated distance to moving obstacles by the tracking process and x_8 is the direction of the moving obstacles. The output of the controller are the change of orientation and the change of speed to apply to the robot.

This fuzzy controller is equipped with an online reinforcement learning algorithm. The actions correspond to the possible scalar outputs of the controller and the states correspond to the different combinations of the input fuzzy variables of the controller.

To help convergence of the learning algorithm, the K rule conclusions take into account the evident decisions. This means if an obstacle is detected on the left side of the robot, the decision of the navigation process is to turn to the right an vice-versa. Then, all possible conclusions are considered as turn to the right (or turn to the left if the obstacle is on the right side of the robot).

In the same idea, if the robot has the choice between two directions, the rule decisions should favor the direction to the goal. This means that most of them are defined in the direction of the goal and few of them are left in the other side to avoid convergence of the algorithm to local minima.

The reinforcement learning algorithm is as follows (see section 3.3.2.3):

1. Initialize Q(s,a) to 0 for all state s and action a
2. Perceive current state s
3. Choose an action a according to action value function
4. carry out action a in the environment. Let the next state be s' and the reward be r.

5. Update action value function from s, a, s', and r:

$$Q_{t+1}(s,a) = (1-\alpha_t)Q_t(s,a) + \beta_t(r + \gamma \max_{a' \in A} Q_t(s',a'))$$

   where α_t is a learning rate parameter and γ is a fixed discounting factor between 0 and 1.

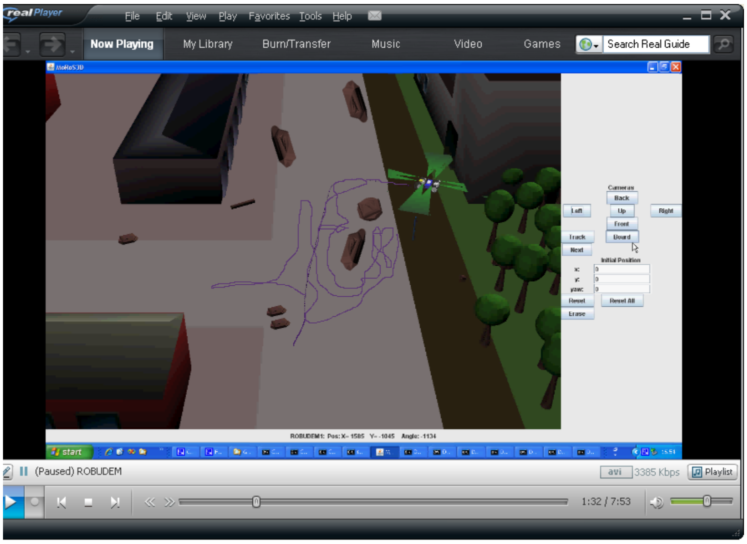
6. Return to 2

3.3.3.4 Command Fusion

The two previously described behaviors are independent. The go to goal behavior is activated if no obstacle is detected in its way. But when the readings from front sensors go below a given threshold (adapted from experience), the avoid obstacles is activated. In this case the readings from lateral and rear sensors are used to help decision.

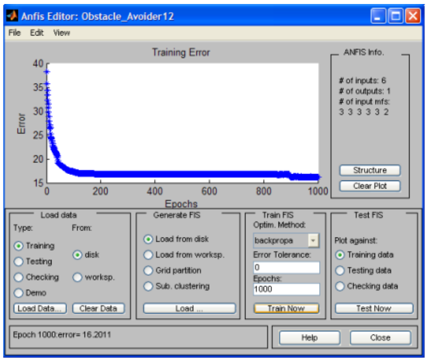
3.3.4 Experimental results

The simulation of the robot navigational behavior is done with simulator MoRoS3D. A simulator as such increases safety when developing and testing algorithms. In MoRoS3D a robot can be placed in a 3D environment and interact with that environment in a manner similar to that of the robot in the real physical situation. Although MoRoS3D visualizes the entire surroundings of the robot, the robot software only "sees" the information it collects through its sensors, just like with a physical robot. The MoRoS3D simulator provides simple interaction with the user and offers different virtual cameras including on-board and tracking ones. Simple distance sensors, such as Laser, US and IR, are simulated. Sensor simulation is actually a geometrical problem that comes to calculating intersections between shapes. Figure 3.13 shows the training of the path planning system in the a realistic environment. The evolution of the training error is also represented with respect to the number of training epochs (figure 3.14). We can see that the user defined 'smooth' path is almost found by the learning process after few training epochs.



ROBUDEM1: Pos: X=-1585 Y=-1045 Angle: -1134

Figure 3.13: Training of the path planing system



ANFIS Info.
# of inputs: 6
# of outputs: 1
# of input mfs:
3 3 3 3 3 2

Load data
Type: [✓ Training] Testing Checking Demo
From: [✓ disk] worksp.

Generate FIS
[✓ Load from disk] Load from worksp. Grid partition Sub. clustering

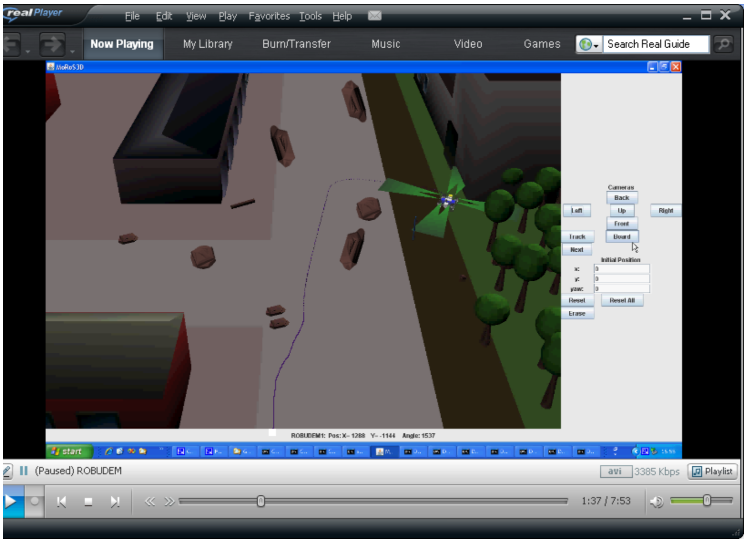
Train FIS
Optim. Method: backpropa
Error Tolerance: 0
Epochs: 1000

Test FIS
Plot against: [✓ Training data] Testing data Checking data

Epoch 1000:error= 16.2011

Figure 3.14: Training error compared to a chosen smooth trajectory

Figure 3.15 shows the results of the developed path planning system in some realistic cases.



ROBUDEM1: Pos: X- 1288 Y- -1144 Angle: 1537

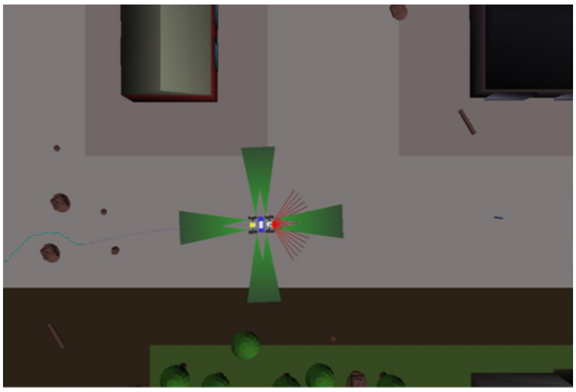


Figure 3.15: Results of the navigation strategy, presented in the MoRoS3D multi robot simulator

3.4 Conclusion

In this chapter we presented an adaptive navigation solution for a mobile robot. While following a predefined path by the global path planing process based on the environment model (SLAM output), the robot uses an unsupervised learning approach (reinforcement learning)to adept itself to navigate around the unmodelled obstacles and improve its performance. The approach adopts a behaviorist design and a strategy for multi-behaviors coordination. The inputs of navigation process are ultrasound data and the position and size of the detected moving objects. A Kalman filter is used to predict the motion of the moving objects.

Final Conclusions and Future Work

This work has introduced three tasks needed for mobile robot navigation relying on a single on-board camera as sensory input: Detection of moving object and estimation of their motion and position ; feature based simultaneous localization and mapping ; and learning based path planing. The path planing process uses also ultrasound sensors for obstacle detection.

The main contributions for the proposed algorithm for moving objects detection and motion estimation are:

- The combination of the Gaussian Mixture Model (GMM) background subtraction approach and a Maximum a Posteriori Probability Markov Random Field (MAP-MRF) framework to solve the problem. This has enabled exploiting the simplicity and capability of the GMM approach to adapt to illumination changes and small motions in the scene and the advantages of spatio-temporal dependencies that moving objects impose on pixels and the interdependence of motion and segmentation fields. Tacking into account the spatial information (background model) estimated with the GMM model, as well as the simultaneous estimation of the motion while segmenting has permitted a good detection of the moving objects and avoided artifacts and blurring caused with motion compensation.

- A feedback of the MRF segmentation results is used to re-update the background model. This has resulted in accelerating the learning of the new stationary regions and therefore avoiding the fragmentation problem.

As a future work we can study the automatic estimation of the number of components in the GMM model as well as the automatic learning of the MRF model parameters.

The proposed algorithm for simultaneous localization and mapping is a solution for vision based SLAM problem in large environment. The approach builds several size limited local maps and combine them into a global map. For a robust matching the algorithm uses a product of three parameters: the Mahalanobis distance between measurements and their predictions, the Euclidean distance between the descriptor vectors of the features, and the distance of the feature to the induced epipolar line (epipolar constraint). This allows using the advantage of looking for feature matching based on the prediction of their position based on the system model and the advantage of the space-scale invariance parameters. Another contribution of the proposed SLAM algorithm is the use of the epipolar geometry principle to estimate the 3D position of the features. To avoid introducing outlier features in the built map, the algorithm uses the segmentation algorithm to detected the moving objects and therefore eliminates the associated features.

An extension for the proposed SLAM algorithm could be the fusion of the data from other sensors as the inertial sensors or robot wheel encoders for a more robust and fast localization and mapping.

The built feature based map is exploited by a global path planning algorithm of B. Hasegawa [240] to generate the safe path.

The estimated 3D position of the detected moving objects and the data from ultrasound sensors are used by an adaptive fuzzy controller for obstacle avoidance. The output of this obstacle avoidance behavior is fused with the output of a goal seeking behavior for a safe navigation of the mobile robot. The multi-behavior coordination.

Appendices

Appendix A

Imaging - Fundamental Definitions

This appendix introduces the basic geometric concepts of multiple-view computer vision. The focus is on geometric models of perspective cameras, and the constraints and properties such models generate when the same 3D scene is observed by multiple cameras (multiple views).

A.1 Image Formation

An image is created by projecting the 3D scene on a 2D image plane. The drop from three-dimensional world to a two-dimensional image is a projection process in which one dimension is lost. The usual way for modeling this process is by central projection in which a ray from a point in space is drawn from a 3D world point through a fixed point in space, the center of projection. This ray will intersect a specific plane in space chosen as the image plane. The image plane is located at the distance of the focal length from the origin of the 3D axis along the Z-direction, and it is perpendicular to it. The complete scene is located at positive Z-ordinates and we view the image with viewing direction on negative Z-direction.

Let $I(x, y)$ be the image intensity at time t at the image point (x, y). The intersection of the ray with the image plane represents the image of the point. This model is in accord with a simple model of a camera, in which a ray of light from a point in the world passes through the lens of a camera and impinges on a film or digital device, producing an image of the point. Ignoring such effects as focus and lens thickness, a reasonable approximation is that all the rays pass through a single point, the center of the lens.

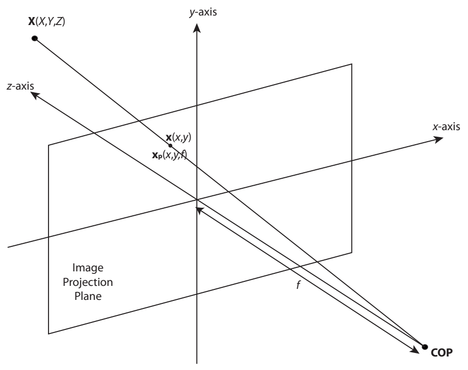


Figure A.1: The Perspective Projection

In order to analyze the mapping process, it is advisable to first define the projective space $\mathbb{P}^n$. The Euclidean space $\mathbb{R}^n$ can be extended to a projective space $\mathbb{P}^n$ by representing points as homogeneous vectors. In this text, we denote the homogeneous counterpart of vector $\mathbf{x}$ as $\tilde{\mathbf{x}}$. A linear transformation of Euclidean space $\mathbb{R}^n$ is represented by matrix multiplication applied to the coordinates of the point. In just the same way a projective transformation of projective space $\mathbb{P}^n$ is a mapping of the homogeneous coordinates representing a point, in which the coordinate vector is multiplied by a non-singular matrix.
Central projection is then simply a mapping from $\mathbb{P}^3$ to $\mathbb{P}^2$. To describe this mapping, three coordinate systems need to be taken into account: the camera, image and world coordinate system.

Consider a point in $\mathbb{P}^3$ in the camera coordinate system, written in terms of homogeneous coordinates $\tilde{\mathbf{X}}_c(X_c, Y_c, Z_c, T)^T$. We can now see that the set of all points $\tilde{\mathbf{X}}(X, Y, Z, T)^T$ for fixed X, Y and Z, but varying T, form a single ray passing through the point center of projection. As a result, all these points map onto the same point, thus the final coordinate of $\tilde{\mathbf{X}}(X, Y, Z, T)^T$ is irrelevant to where the point is imaged. In fact, the image point is the point in $\mathbb{P}^2$ with homogeneous coordinates $\tilde{\mathbf{x}}_c(x_c, y_c, f)^T$, as defined by the projection equation

$$\begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}, \tag{A-1}$$

with f the focal length of the camera lens.
The mapping may thus be represented in its most simple form by a mapping of 3D homogeneous coordinates, represented by a 3×4 matrix $\mathbf{P}_0$ with the block structure $\mathbf{P}_0 = [\mathbf{I}_{3\times3} \,|\mathbf{0}_{3\times1}]$, where $\mathbf{I}_{3\times3}$ is the identity matrix and $\mathbf{0}_{3\times1}$ is a zero 3-vector.

In the image coordinate system, the mapping from $\tilde{\mathbf{x}}_c(x_c, y_c, f)^T$ to image coordinates is described. This mapping takes into account different centers of projection (x_0, y_0), non-square pixels and skewed coordinate axes. As such, it encompasses all the internal camera parameters. This mapping can be expressed in terms of matrix multiplication as:

$$\tilde{\mathbf{x}}_i = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \alpha_x \cot(\theta) & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix}, \tag{A-2}$$

where (x_0, y_0) is the coordinate of the principle point or image center, α_x and α_y denote the scaling in the x and y direction and θ is the angle between the axes, which is in general equal to $\pi/2$. The matrix $\mathbf{K}$ is an upper triangular matrix which provides the transformation between an image point and a ray in Euclidean 3-space. It encompasses all internal camera parameters and is called the *camera calibration matrix*. Throughout this work, we will assume that the cameras are calibrated, which means that $\mathbf{K}$ is known.

As a last step of projection, the description of the transformation between the camera and the world coordinate system is required. Changing coordinates in space is equivalent to multiplication by a 4×4 matrix:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \tag{A-3}$$

with $\mathbf{R}$ the rotation matrix and $\mathbf{t}$ the translation vector.

Concatenating the expressions A-3, A-2 and A-1, it is clear that the most general image projection can be represented by an arbitrary 3×4 matrix of rank 3, acting on the homogeneous coordinates of the point in $\mathbb{P}^3$ mapping it to the imaged point in $\mathbb{P}^2$:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \mathbf{K}\left[\mathbf{R}|\,\mathbf{t}\right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{A-4}$$

It thus turns out that the most general imaging projection is represented by an arbitrary 3×4 matrix of rank 3, acting on the homogeneous coordinates of the point in $\mathbb{P}^3$ mapping it to the imaged point in $\mathbb{P}^2$:

$$\tilde{\mathbf{x}} = \mathbf{P}\tilde{\mathbf{X}}, \tag{A-5}$$

with:

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\,\mathbf{t}\right] \tag{A-6}$$

This matrix $\mathbf{P}$ is known as the camera matrix. It expresses the action of a projective camera on a point in space in terms of a linear mapping of homogeneous coordinates.

When returning to non-homogeneous coordinates for a camera based in the origin and ignoring non-square pixel aspect ratios (this means $\mathbf{P} = [\mathbf{I}_{3\times3}\,|\mathbf{0}_3\,]$), it can be observed that to map a 3D point $X = (X, Y, Z)$ to the image coordinates $\mathbf{x} = (x, y, f)$, the following perspective projection equations can be written:

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{A-7}$$

in which x and y are the image coordinates. In order to reduce the complexity of some equations and for numerical stability, we can parameterize the depth by a proximity factor $d = \frac{1}{Z}$.

A.2 Two-View Image Geometry

A.2.1 Two-View Geometry described by the Fundamental MAtrix

The geometry between two views is called the *epipolar geometry.* This geometry depends on the internal parameters and relative position of the two cameras. The fundamental matrix $\mathbf{F}$ encapsulates this intrinsic geometry and was introduced by Faugeras in [49] and Hartley in [50]. It is a 3×3 matrix of rank 2. The fundamental matrix describes the relationship between matching points: if a point $\tilde{\mathbf{X}}$ is imaged as $\tilde{\mathbf{x}}$ in the first view, and $\tilde{\mathbf{x}}'$ in the second, then the image points must satisfy the relation $\tilde{\mathbf{x}}'^T \mathbf{F} \tilde{\mathbf{x}} = 0$. In this section, the epipolar geometry is described and the fundamental matrix is derived. The fundamental matrix is independent of scene structure. However, it can be computed from correspondences of imaged scene points alone, without requiring knowledge of the cameras internal parameters or relative pose.

To describe this mapping, first the geometric entities involved in epipolar geometry are introduced in figure A.2. Here, the epipole $\tilde{\mathbf{e}}$ is the point of intersection of the line joining the camera centers (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera center of the other view. It is also the vanishing point of the baseline (translation) direction. An epipolar plane is a plane containing the baseline. There is a one-parameter family of epipolar planes. An epipolar line is the intersection of an epipolar plane with the image plane. The epipolar line corresponding to $\tilde{\mathbf{x}}$ is the image in the second view of the ray back-projected from $\tilde{\mathbf{x}}$. Any point $\tilde{\mathbf{x}}'$ in the second image matching the point $\tilde{\mathbf{x}}$ must lie on the epipolar line $\tilde{\mathbf{l}}'$. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

The mapping from a point in one image to a corresponding epipolar line in the other image may be decomposed into two steps. In the first step, the point $\tilde{\mathbf{x}}$ is mapped to some point $\tilde{\mathbf{x}}'$ in the other image lying on the epipolar line $\tilde{\mathbf{l}}'$. This point $\tilde{\mathbf{x}}'$ is a potential match for the point $\tilde{\mathbf{x}}$. In the second step, the epipolar line $\tilde{\mathbf{l}}'$ is obtained as the line joining $\tilde{\mathbf{x}}'$ to the epipole $\tilde{\mathbf{e}}'$. Figure A.2 illustrates this mapping process.

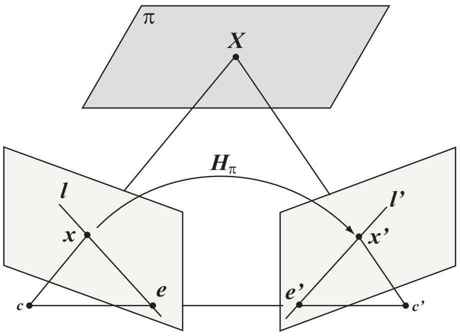


Figure A.2: The mapping process from one camera according to the epipolar geometry

Consider a plane π in space not passing through either of the two camera centers. The ray through the first camera center corresponding to the point $\tilde{\mathbf{x}}$ meets the plane π in a point $\tilde{\mathbf{X}}$. This point $\tilde{\mathbf{X}}$ is then projected to a point $\tilde{\mathbf{x}}'$ in the second image. This procedure is known as transfer via the plane π. Since $\tilde{\mathbf{X}}$ lies on the ray corresponding to $\tilde{\mathbf{x}}$, the projected point $\tilde{\mathbf{x}}'$ must lie on the epipolar line $\tilde{\mathbf{l}}'$ corresponding to the image of this ray, as illustrated in A.2. The points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ are both images of the 3D point $\tilde{\mathbf{X}}$ lying on a plane. The set of all such points $\tilde{\mathbf{x}}_i$ in the first image and the corresponding points $\tilde{\mathbf{x}}'_i$ in the second image are projectively equivalent, since they are each projectively equivalent to the planar point set $\tilde{\mathbf{X}}'$. Thus there is a 2D homography $\mathbf{H}_\pi$ mapping each $\tilde{\mathbf{x}}_i$ to $\tilde{\mathbf{x}}'_i$.

Given the point $\tilde{\mathbf{x}}'$ the epipolar line $\tilde{\mathbf{l}}'$ passing through $\tilde{\mathbf{x}}'$ and the epipole $\tilde{\mathbf{e}}'$ can be written as $\tilde{\mathbf{l}}' = \tilde{\mathbf{e}}' \times \tilde{\mathbf{x}}' = [\tilde{\mathbf{e}}']_\times \tilde{\mathbf{x}}'$ ($[\tilde{\mathbf{e}}']_\times$ being the skew-symmetric matrix form of $\tilde{\mathbf{e}}'$). Since $\tilde{\mathbf{x}}'$ may be written as $\tilde{\mathbf{x}}' = \mathbf{H}_\pi \tilde{\mathbf{x}}$, we have:

$$\tilde{\mathbf{l}}' = [\tilde{\mathbf{e}}']_\times \mathbf{H}_\pi \tilde{\mathbf{x}} = \mathbf{F}\tilde{\mathbf{x}}, \tag{A-8}$$

where we define $\mathbf{F} = [\tilde{\mathbf{e}}']_\times \mathbf{H}_\pi$ as the fundamental matrix. Since $[\tilde{\mathbf{e}}']_\times$ has rank 2 and $\mathbf{H}_\pi$ rank 3, $\mathbf{F}$ is a matrix of rank 2, which is logic as $\mathbf{F}$ represents a mapping from a 2-dimensional onto a 1-dimensional projective space.

The fundamental matrix satisfies the condition that for any pair of corresponding points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ in the two images

$$\tilde{\mathbf{x}}'^T \mathbf{F} \tilde{\mathbf{x}} = 0 \tag{A-9}$$

This is true, because if points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ correspond, then $\tilde{\mathbf{x}}'$ lies on the epipolar line $\tilde{\mathbf{l}}' = \mathbf{F}\tilde{\mathbf{x}}$ corresponding to the point $\tilde{\mathbf{x}}$. In other words $0 = \tilde{\mathbf{x}}'^T \tilde{\mathbf{l}}' = \tilde{\mathbf{x}}'^T \mathbf{F} \tilde{\mathbf{x}}$. Conversely, if image points satisfy the relation $\tilde{\mathbf{x}}'^T \mathbf{F} \tilde{\mathbf{x}} = 0$ then the rays defined by these points are coplanar. This is a necessary condition for points to correspond. The importance of the relation A-10 is that it gives a way of characterizing the fundamental matrix without reference to the camera matrices, i.e. only in terms of corresponding image points. This enables $\mathbf{F}$ to be computed from image correspondences alone.

A.2.1.1 Estimating F: the eight-point algorithm

If a number of point correspondences $\tilde{\mathbf{x}} \leftrightarrow \tilde{\mathbf{x}}'$ is given, we can use equation (A-10) to compute the unknown matrix F.

Each point correspondence gives rise to one linear equation in the unknown entries of **F**. From a set of n point correspondences, we obtain a $n \times 9$ coefficient matrix A by stacking up one equation for each correspondence.

- In general A will have rank 8 and the solution is the 1-dimensional right null-space of A.

- The fundamental matrix **F** is computed by solving the resulting linear system of equations, for $n > 8$.

- If the data are not exact and more than 8 points are used, the rank of A will be 9 and a least-squares solution is sought.

- The least-squares solution for **F** is the singular vector corresponding to the smallest singular value of A.

- This method does not explicitly enforce F to be singular, so it must be done a posteriori.

- Replace **F** by $\mathbf{F}'$ such that $\det \mathbf{F}' = 0$, by forcing to zero the least singular value.

- It can be shown that $\mathbf{F}'$ is the closest singular matrix to **F** in Frobenius norm.

- Geometrically, the singularity constraint ensures that the epipolar lines meet in a common epipole

A.2.1.2 Estimating F using RANSAC algorithm

Automatic methods for finding point correspondence make mistakes very often in practice, thus introducing outliers in the point correspondences for the fundamental matrix estimation. It is well known that least-square estimation is very sensitive to outliers, therefore several robust estimation techniques have been applied to overcome the outlier problem. Among them, random sampling consensus (RANSAC) technique has been widely used in many computer vision problems. The basic idea of using RANSAC for fundamental matrix estimation is as follows: randomly selecting a number of minimal subsets of point correspondences to determine the fundamental matrix for each subset, and then find the best fundamental matrix that is most consistent with the entire set of point correspondences.

Given a set of point correspondences, denoted by $\mathbb{X} = (\tilde{\mathbf{x}}_k, \tilde{\mathbf{x}}'_k) \mid k = 1, ..., n$, the RANSAC algorithm for fundamental matrix estimation consists of the following steps:

1. Randomly select a number of subsets of eight point correspondences from the entire set $\mathbb{X}$.

2. For each subset, indexed by j, compute the corresponding fundamental matrix $\mathbf{F}_j$ by using a linear estimation algorithm (previous describe algorithm).

3. For each estimate $\mathbf{F}_j$, compute the residue r_j for each point correspondence and count the total number of consistent correspondences, i.e. $r_j^2 < \sigma^2$, where σ^2 is a predefined constant.

4. Keep the fundamental matrix F_j that yields the most consistent correspondences.

5. Refine the fundamental matrix estimation by applying the linear estimation algorithm or the M-estimator on the set of consistent point correspondences only.

Assume the outlier percentage in the entire correspondence set $\mathbb{X}$ is τ. Thus, the total number of randomly selected subsets N required in RANSAC to achieve a probability P that at least one selected subset does not contain any outlier is given by

$$N = \frac{log(1-P)}{log[1-(1-\tau)^q]} \tag{A-10}$$

where q is the size of the minimal subset, i.e 8 in this case.

A.2.2 Two-View Geometry described by the Essential Matrix

The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates. Historically, the essential matrix was introduced by Longuet-Higgins in [243] before the fundamental matrix, and the fundamental matrix may be thought of as the generalization of the essential matrix in which the inessential assumption of calibrated cameras is removed. The essential matrix has fewer degrees of freedom, and additional properties, compared to the fundamental matrix.

Consider a camera matrix decomposed as $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$, and let $\tilde{\mathbf{x}} = \mathbf{P}\tilde{\mathbf{X}}$ be a point in the image. If the calibration matrix $\mathbf{K}$ is known, then we may apply its inverse to the point $\tilde{\mathbf{x}}$ to obtain the point $\widehat{\mathbf{x}} = \mathbf{K}^{-1}\tilde{\mathbf{x}}$. Then $\widehat{\mathbf{x}} = [\mathbf{R}|\mathbf{t}]\tilde{\mathbf{X}}$, where $\widehat{\mathbf{x}}$ is the image point expressed in normalized coordinates. It may be thought of as the image of the point $\tilde{\mathbf{X}}$ with respect to a camera $[\mathbf{R}|\mathbf{t}]$ having the identity matrix $\mathbf{I}$ as calibration matrix. The camera matrix $\mathbf{K}^{-1}\mathbf{P} = [\mathbf{R}\,|\,\mathbf{t}\,]$ is called a normalized camera matrix, the effect of the known calibration matrix having been removed. Now, consider a pair of normalized camera matrices $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = [\mathbf{R}|\mathbf{t}]$. The fundamental matrix corresponding to the pair of normalized cameras is customarily called the essential matrix and has the form:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \mathbf{R}\left[\mathbf{R}^T\mathbf{t}\right]_\times \tag{A-11}$$

The essential matrix can then be defined as:

$$\widehat{\mathbf{x}}'^T\mathbf{E}\widehat{\mathbf{x}} = 0 \tag{A-12}$$

in terms of the normalized image coordinates for corresponding points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$. Substituting for $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{x}}'$ gives $\tilde{\mathbf{x}}'^T\mathbf{K}'^{-1T}\mathbf{E}\mathbf{K}^{-1}\tilde{\mathbf{x}} = 0$. Comparing this with the relation $\tilde{\mathbf{x}}'^T\mathbf{F}\tilde{\mathbf{x}} = 0$ for the fundamental matrix, it follows that the relationship between the fundamental and essential matrices is:

$$\mathbf{E} = \mathbf{K}'^T\mathbf{F}\mathbf{K} \tag{A-13}$$

This relationship shows that once the camera calibration matrix $\mathbf{K}$ is known, the essential matrix can be calculated from the fundamental matrix.

The essential matrix holds all the information about the "external" calibration parameters: rotation and translation between the two camera frames.

$$\mathbf{E} = \mathbf{R}\,[\mathbf{t}]_\times\,, \tag{A-14}$$

with $[\mathbf{t}]_\times$ the skew-symmetric matrix form of the translation vector.

A.2.3 Triangulation

Given the camera matrices P and P', let $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ be two corresponding points satisfying the epipolar constraint $\tilde{\mathbf{x}}'^T\mathbf{F}\tilde{\mathbf{x}} = 0$. It follows that $\tilde{\mathbf{x}}'$ lies on the epipolar line $\mathbf{F}\tilde{\mathbf{x}}$ and so the two rays back-projected from image points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ lie in a common epipolar plane. Since they lie in the same plane, they will intersect at some point. This point is the reconstructed 3D scene point $\tilde{\mathbf{X}}$.

Analytically, the reconstructed 3D point $\tilde{\mathbf{X}}$ can be found by solving for depth parameter Z or Z' in the following equation:

$$\tilde{\mathbf{e}} = Z\tilde{\mathbf{x}} - Z'\tilde{\mathbf{x}}' \tag{A-15}$$

The depths Z and Z' are unknown. Both encode the position of $\tilde{\mathbf{X}}$ in space, as Z is the depth of $\tilde{\mathbf{X}}$ whit respect to the camera at the first view and Z' is the depth of M with respect to the camera at the second view.

The three points $\tilde{\mathbf{x}}$, $\tilde{\mathbf{e}}$ and $\tilde{\mathbf{x}}'$ are known and are collinear, so we can solve for Z using the following expression:

$$Z = \frac{(\tilde{\mathbf{e}} \times \tilde{\mathbf{x}}').(\tilde{\mathbf{x}} \times \tilde{\mathbf{x}}')}{\parallel \tilde{\mathbf{x}} \times \tilde{\mathbf{x}}' \parallel^2} \tag{A-16}$$

A.3 Three-View Geometry

The trifocal tensor approach is an extension to the case of three views of the two-view geometry description. This approach maintains a similar projective geometry spirit and has been proposed and developed by Sashua [244], Hartley [245] and Faugeras [246]. The trifocal tensor is a $3 \times 3 \times 3$ array of numbers that relate the coordinates of corresponding points or lines in three views. Just as the fundamental matrix is determined by the two camera matrices, and determines them up to projective transformation, so in three views, the trifocal tensor is determined by the three camera matrices, and in turn determines them, again up to projective transformation.

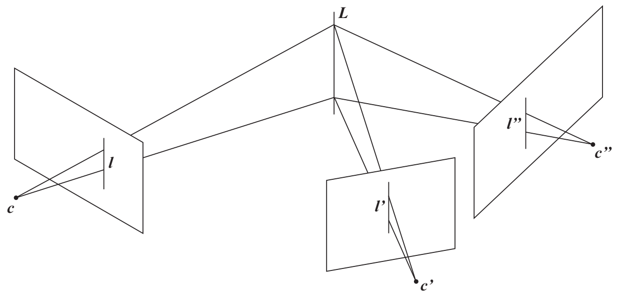


Figure A.3: A line in 3-space is imaged as the corresponding triplet $\mathbf{l}$,$\mathbf{l}'$,$\mathbf{l}''$ in three views indicated by their centers, $\mathbf{c}$,$\mathbf{c}'$,$\mathbf{c}''$, and image planes.

Consider a line $\mathbf{L}$ in 3D space, which is projected onto three cameras, resulting in 3 lines $\mathbf{l}$, $\mathbf{l}'$ and $\mathbf{l}''$ in image space, as illustrated by Figure A.3. These lines are obviously inter-related. The trifocal tensor expresses this relation by mapping lines 2 images to a line in the remaining image. According to the three-view geometry model, the incidence relation for the i^{th} coordinate l_i of $\mathbf{l}$ can be written as:

$$l_i = \mathbf{l}'^T \mathcal{T}_i \mathbf{l}'' \qquad \text{(A-17)}$$

By definition, the set of three matrices $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ constitute the trifocal tensor in matrix notation. In tensor notation, the basic incidence relation A-17 becomes:

$$l_i = l'_j l''_k \mathcal{T}_i^{jk} \qquad \text{(A-18)}$$

By defining the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ as the i^{th} columns of the camera matrices for the three views, the three-view trifocal tensor formulation can also be written as:

$$\mathcal{T}_i^{jk} = \mathbf{a}_i^j \mathbf{b}_4^k - \mathbf{a}_4^j \mathbf{b}_i^k, \qquad \text{(A-19)}$$

where $\mathbf{a}_4$ and $\mathbf{b}_4$ are the epipoles in views two and three respectively, arising from the first camera.

As with the fundamental matrix, once the trifocal tensor is known, it is possible to extract the three camera matrices from it, and thereby obtain a reconstruction of the scene points and lines. As ever, this reconstruction is unique only up to a 3D projective transformation; it is a projective reconstruction.

It is straightforward to compute the fundamental matrices $\mathbf{F}_{21}$ and $\mathbf{F}_{31}$ between the first and the other views from the trifocal tensor:

$$\mathbf{F}_{21} = [\mathbf{e}']_\times [\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]\,\mathbf{e}''; \qquad \mathbf{F}_{31} = [\mathbf{e}'']_\times \left[\mathcal{T}_1^T, \mathcal{T}_2^T, \mathcal{T}_3^T\right] \mathbf{e}' \qquad \text{(A-20)}$$

To retrieve the camera matrices, the first camera may be chosen as $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$. Since $\mathbf{F}_{21}$ is known from equation A-20, it is possible to derive the form of the second camera as:

$$\mathbf{P}' = [[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]\,\mathbf{e}''|\mathbf{e}'] \qquad \text{(A-21)}$$

The third camera cannot be chosen independently of the projective frame of the first two. It turns out that $\mathbf{P}''$ can be written as:

$$\mathbf{P}'' = \left[\left(\mathbf{e}''\mathbf{e}''^T - I \right) \left[\mathcal{T}_1^T, \mathcal{T}_2^T, \mathcal{T}_3^T \right] \mathbf{e}' | \mathbf{e}'' \right] \tag{A-22}$$

This decomposition shows that the trifocal tensor may be computed from the three camera matrices, and that conversely the three camera matrices may be computed, up to projective equivalence, from the trifocal tensor. Thus, the trifocal tensor completely captures the three cameras up to projective equivalence and we are able to generalize the method for two views to three views. There are several advantages to using such a three-view method for reconstruction.

- It is possible to use a mixture of line and point correspondences to compute the projective reconstruction. With two views, only point correspondences can be used.
- Using three views gives greater stability to the reconstruction, and avoids unstable configurations that may occur using only two views for the reconstruction.

Appendix B

Visual Features

B.1 Feature detection

The navigation map is built with features present in the environment that can be detected by the external sensor. Recognizable features are essential for SLAM since the algorithm will not operate correctly in a featureless environment.

There are two types of features, primitive and high-level features. Primitive features are defined as features which can be described by simple geometrical definitions. Common examples are points and lines. While the high level features, referred to in the literature as landmarks, are more complex and can, therefore, be intrinsically more distinctive for the environment. These landmarks should be invariant to geometric transformation, eg. doors, objects and the feature extraction can be seen as a filter which permits the handling of noise from sensors and dynamics in the environment.

In the field of robot navigation, the mostly used primitive features are corners and SIFT (Scale Invariant Feature Transform) features. In the following we will give an overview on these primitive features extraction methods.

B.1.1 Corner detection Methods

Corners are local image features characterized by locations where variations of intensity function f in both X and Y directions are high. In mathematical definition this means both partial derivatives f_x and f_y are large. The quality of a corner detector is often judged based on its ability to detect the same corner in multiple images, which are similar but not identical, for example having different lighting, translation, rotation and other transforms.

Different corner detectors exist, and they can be classified in tree main groups: edge-relation methods, topology methods, and autocorrelation methods. The detectors that do not fit into one of these categories are discussed in a separate paragraph.

B.1.1.1 Edge-Relation Methods

Kitchen and Rosenfeld [191] were the first to apply differential geometry operators to corner detection. They proposed a cornerness measure for each pixel based on the change of gradient direction (second-order derivatives) along an edge contour weighted by the local gradient magnitude. Corners are then identified by the local maximum of this measure, which can be done in a computationally efficient manner by applying non-maximum suppression to the gradient magnitudes before weighting the second-order derivatives. This corner detector suffers from sensitivity as it relies on second-order derivative terms and has been shown to have a poor repeatability rate and localization.

A popular corner detector designed for motion estimation was proposed by Wang and Brady [191]. They applied differential geometry operators to detect corners based on the measurement of surface curvature, but derived a simplified cornerness measure suitable for real-time applications while improving performance relative to the Kitchen and Rosenfeld operator.

B.1.1.2 Topology Methods

Beaudet [192] developed one of the first corner detectors and the insights gained from his approach are exploited by many other corner detectors. The Beaudet operator is a rotationally invariant measure of cornerness given by the determinant of the Hessian matrix. Since the Hessian matrix involves the computation of second-order derivatives this operator is sensitive to noise. In addition, it has been shown to be unstable in scale space. This approach can be viewed as looking for high curvature edges by calculating image Gaussian curvature (i.e. the product of two principle curvatures).

Deriche extended Beaudet's operator by applying it at multiple scales. Lines are drawn between the corresponding corners at each scale and the intersection of these lines with the nearest zero crossing of the Laplacian image are defined as the position of the corner. This method improves the localization relative to the Beaudet approach, but the Deriche operator still suffers from sensitivity to noise.

B.1.1.3 Autocorrelation Methods

The Moravec operator [193] considers a local window in the image and determines the average change of intensity resulting from shifting the window by a small amount in various directions. This operation is repeated for each pixel position which is assigned an interest value equal to the minimum change produced by these shifts. Corners are the local maximum of the interest values. This operator has been shown to be sensitive to noise along strong edges because only the minimum intensity change is considered for each pixel position.

Many of the weaknesses of the Moravec operator are addressed by the Harris operator [191, 194]. This operator estimates the measure of local autocorrelation using first-order derivatives which is suggested by performing an analytic expansion of the Moravec operator. The response is isotropic as the variation of the autocorrelation over different orientations can be calculated from the principle curvatures of the local autocorrelation. The Harris operator is generally considered the best operator with respect to detecting true corners, but has poor localization.

The Forstner [195] operator uses a similar measure of cornerness to the Harris operator (although, how this measure is calculated differs greatly) and uses local statistics to calculate the selection threshold. The result is better localization at the cost of increased computation.

B.1.1.4 Other Methods

Another corner detector algorithm based on brightness comparisons within a circular mask has been proposed by Smith and Brady [191]. SUSAN (Smallest Univalue Segment Assimilating Nucleus) assumes that within a relatively small circular region pixels belonging to a given object will have relatively uniform brightness. The algorithm computes the number of pixels with similar brightness to the pixel at the center of the mask (the nucleus of the mask). These pixels are called the USAN (Univalue Segment Assimilating Nucleus) of the mask. Corners are detected by applying the mask to all pixels in the image and then finding the local minima in this new USAN map. This corner detector is robust to noise (no spatial derivatives are computed), fast to compute, but only has an average repeatability rate.

Trajkovic and Hedley [191] have proposed a corner detector that uses the same intuition used in the SUSAN operator. For a given point in the image, the variation in brightness along all lines passing through the point are considered. At corners the variation in brightness will be high for all lines. The repeatability rate of this algorithm is not as strong as the Harris operator, but it is one of the fastest available corner detectors.

The use of corners for camera motion estimation works well for small motion, but will fail if there are large scale or viewpoint changes between the images. This is because the corner detectors used are not scale-invariant, and the correlation measures are not invariant to viewpoint, scale and illumination change. The first problem is addressed by scale-space theory, which has proposed feature detectors with automatic scale selection [178]. In particular, scale-space interest point detectors have been shown to have much greater repeatability than their fixed scale equivalents [198]. The second problem suggests the need for local descriptors of image regions that are invariant to the imaging process.
Geometrical invariance can be acheived by assuming that the regions to be matched are locally planar, and by describing them in a manner which is invariant under homographies. Many authors use feature descriptors which are invariant under special cases of this group e.g. similarities or affinities. The well known approach is Lowe's SIFT features [198] , which uses a characteristic scale and orientation

at interest points to form similarity invariant descriptors.

B.1.1.5 Harris corner detector

Harris corner detector is one of the most frequently used operator in many application areas, as motion tracking, stereo matching, and image database retrieval. It is relatively simple but still efficient and reliable.
This corner detector is based on a local structure matrix (tensor):

$$C_{str} = \mathcal{G}(r,\sigma) \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \tag{A-1}$$

with $\mathcal{G}(r,\sigma)$ is a smoothing Gaussian filter of a selected size σ. f_x and f_y are the spatial derivatives and f_t is the temporal derivative of the intensity function f. denoting in (A-1) the smoothing by $\widehat{ff}$, we have:

$$C_{str} = \begin{bmatrix} \widehat{f_x^2} & \widehat{f_x f_y} \\ \widehat{f_x f_y} & \widehat{f_y^2} \end{bmatrix} \tag{A-2}$$

The local structure matrix C_{str} is:

- Symmetric and therefore it can be diagonalized by rotation of the coordinate axes. The diagonal entries are the two eigenvalues λ_1 and λ_2:

$$C_{str} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{A-3}$$

- Positive definite which means the eigenvalues are nonnegative ($\lambda_1 \geq 0, \lambda_2 \geq 0$).

Assume $\lambda_1 \geq \lambda_2 \geq 0$, a corner is a location where the smaller eigenvalue, λ_2, is large enough.
In Harris method a corner is detected at location $\mathbf{x}$ when

$$H(\mathbf{x}) > Thr \tag{A-4}$$

Where $H(\mathbf{x})$ is a measure of corner strength:

$$\begin{aligned} H(\mathbf{x}) &= detC_{str} - \alpha(traceC_{str}) \\ &= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \end{aligned} \tag{A-5}$$

α is a parameter and $H \geq 0$ if $0 \leq \alpha \leq 0.25$.

Usually, Thr is set close to zero and fixed, while α is a variable parameter.

- Larger $\alpha \Rightarrow$ smaller H $\Rightarrow$ less sensitive detector: less corners detected (Figure B.1.a).
- Smaller $\alpha \Rightarrow$ larger H $\Rightarrow$ more sensitive detector: more corners detected (Figure B.1.b).

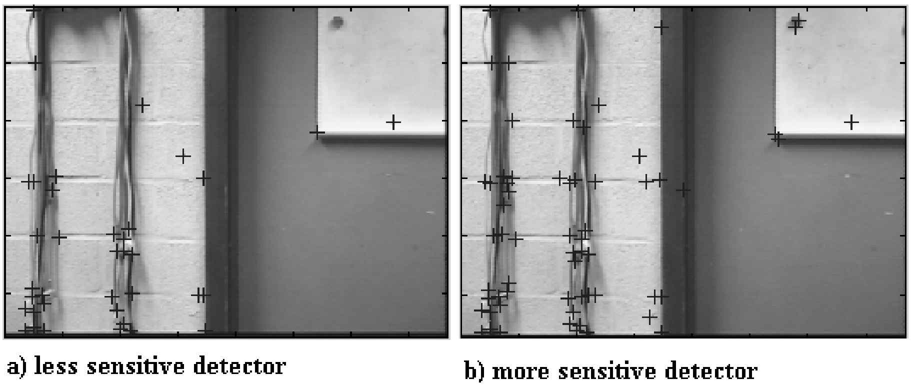


Figure B.1: Harris corner detector

B.1.2 SIFT feature detector

The SIFT approach [178], for image feature generation, takes an image and transforms it into a collection of local feature vectors. Each of these feature vectors is invariant to any scaling, rotation or translation of the image. To extract these features the SIFT algorithm applies a 4 stage filtering approach:

1. Detection of the keypoints candidates as extrema of Difference-of-Gaussian (DOG) function in the image scale space.
2. Localization of the keypoints as extrema of the second order Taylor expansion of DOG function.
3. Assignment of orientation according to the major gradient direction around each keypoint at the selected scale.
4. Calculation of the local keypoint descriptors based on the set of surrounding image gradients.

Scale-Space Extrema Detection

This stage of the filtering attempts to identify those locations and scales that are identifiable from different views of the same object. This can be efficiently achieved using a "scale space" function. The scale space is defined by the function:

$$L(x, y, \sigma) = G(x, y, \sigma) * f(x, y) \tag{A-6}$$

Where $f(x, y)$ is the input image, $*$ is the convolution operator, and $G(x, y, \sigma)$ is a variable-scale Gaussian

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Difference of Gaussians is then used to detect stable keypoint locations in the scale-space by locating local extrema of the difference $D(x, y, \sigma)$ between two Gaussian-blurred images, one with scale k times the other:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{A-7}$$

The difference-of-Gaussian filter is in effect a tunable bandpass filter. To detect the local maxima and minima of $D(x, y, \sigma)$ each point is compared with its 8 neighbours at the same scale, and its 9 neighbours up and down one scale. If this value is the minimum or maximum of all these points then this point is an extrema and it is selected as a candidate interest point (called keypoint in the SIFT framework).

Keypoint Localistaion

The location of the keypoints corresponds to the extremum $\bar{\mathbf{x}}$ of the second order Taylor expansion of the scale space function $D(x, y, \sigma)$ (eq.A-8), having the origin at the sample point.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{A-8}$$

where $\mathbf{x} = (x, y, \sigma)^T$ is the offset from the sample point. The location $\bar{\mathbf{x}}$ is determined by taking the derivative of equation (A-8) with respect to $\mathbf{x}$ and setting it to zero, giving

$$\bar{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{A-9}$$

Local extrema with low contrast are rejected because they are sensitive to noise, and keypoints that correspond to edges are also discarded.

Orientation Assignment

This step aims to assign a consistent orientation to the keypoints based on local image properties. The approach taken to find an orientation is:

- The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner.
- Compute gradient magnitude, m

$$m(x, y) = \sqrt{(L(x+1, y, \sigma) - L(x-1, y, \sigma))^2 + (L(x, y+1, \sigma) - L(x, y-1, \sigma))^2}$$

- Compute orientation, θ

$$\theta(x, y) = atan\left(\frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)}\right)$$

- Form an orientation histogram from gradient orientations of sample points in the neighborhood of the keypoint. The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with a σ that is 1.5 times the scale of the keypoint.

- Peaks in the histogram correspond to dominant orientations. A separate keypoint is created for the direction corresponding to the histogram maximum, and any other direction within 80% of the maximum value.

- Some points will be assigned multiple orientations

- Fit a parabola to the 3 histogram values closest to each peak to interpolate the peaks position

Keypoint Descriptor

Once a keypoint orientation has been selected, the feature descriptor is computed as a set of orientation histograms on a 4×4 pixel neighborhood. The orientation histograms are relative to the keypoint orientation.

Like before, the contribution of each pixel is weighted by the gradient magnitude, and by a Gaussian with $\sigma = 1.5$ times the scale of the keypoint.

Histograms contain 8 bins each, and each descriptor contains an array of 4 histograms around the keypoint. This leads to a SIFT feature vector with $4 \times 4 \times 8 = 128$ elements. This vector is normalized to enhance invariance to changes in illumination. Figure B.2 shows an example of a 2×2 descriptor array computed from 8×8 set of samples.

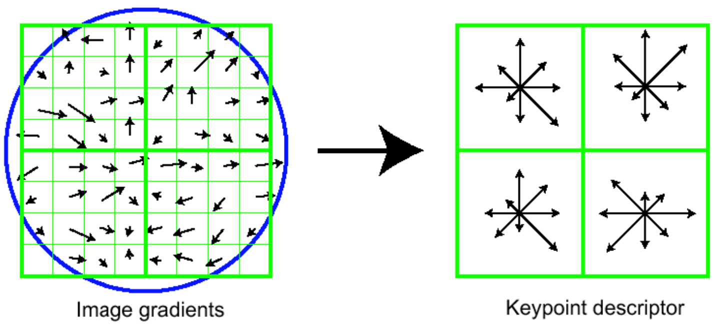


Figure B.2: SIFT feature descriptor (from [46])

Figure B.3 shows an example on feature detection using the above described algorithm. The arrows indicate scale and orientation and their starting points are the features' position.

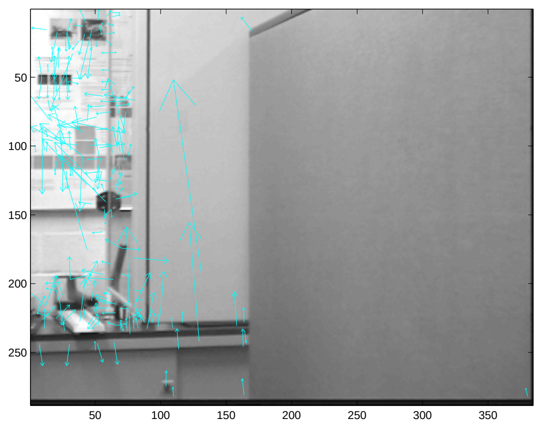


Figure B.3: features detected using the SIFT algorithm

B.2 Pyramidal Lucas-Kanade feature tracker

In this section the Pyramidal implementation of the Lucas Kanade feature tracker will be briefly summarized based on the description of [197].

Let f and f' be two 2D grayscaled images. The two quantities $f_s = f(\mathbf{x}_s) = f(x_s, y_s)$ and $f'_s = f'(\mathbf{x}_s) = f'(x_s, y_s)$ are then the grayscale value of the two images at the location $\mathbf{x}_s = [x_s \quad y_s]^T$ of the site (pixel) s.

Consider a selected feature s at location $\mathbf{x}_s = [x_s \quad y_s]^T$ on the first image f. The goal of feature tracking is to find the location $\mathbf{x}_r = \mathbf{x}_s + \mathbf{u}_s = [x_s + u_{s,x} \quad y_s + u_{s,y}]^T$ of the site r on the second image f' such as f_s and f'_r are "similar". The vector $\mathbf{u}_s = [u_{s,x} \quad u_{s,y}]^T$ is the image velocity at site s, also known as the optical flow at site s and $u_{s,x}$ and $u_{s,y}$ are its X- and Y-components, respectively.

Let ω_x and ω_y two integers. We define the image velocity $\mathbf{u}_s$ at site s as being the vector that minimizes the residual function e at site s defined as follows:

$$e_s = \sum_{x_i = x_s - \omega_x}^{x_s + \omega_x} \sum_{y_i = y_s - \omega_y}^{y_s + \omega_y} \left(f(x_i, y_i) - f'(x_i + u_{s,x}, y_i + u_{s,y}) \right)^2 \tag{A-10}$$

So the similarity function is measured on a image neighborhood of size $(2\omega_x + 1) \times (2\omega_y + 1)$. This neighborhood is called the integration window. Typical values for ω_x and ω_y are between 2 and 7 pixels.

A small integration window would be preferable in order not to "smooth out" the details contained in the images. That is especially required for closest moving objects. And at the other side to handle large motions, it is intuitively preferable to pick a large integration window. There is therefore a natural tradeoff when choosing the integration window size. In attempt to solve this problem, a modified version of the Lucas-Kanade tracking algorithm with pyramidal implementation has been used.

In this algorithm the following equation is used to construct recursively the pyramidal representations of an image f:

$$\begin{aligned} f^J(x_s, y_s) = {} & \tfrac{1}{4} f^{J-1}(2x_s, 2y_s) + \\ & \tfrac{1}{8} \big(f^{J-1}(2x_s - 1, 2y_s) + f^{J-1}(2x_s + 1, 2y_s) + \\ & \quad f^{J-1}(2x_s, 2y_s - 1) + f^{J-1}(2x_s, 2y_s + 1) \big) + \\ & \tfrac{1}{16} \big(f^{J-1}(2x_s - 1, 2y_s - 1) + f^{J-1}(2x_s + 1, 2y_s + 1) + \\ & \quad f^{J-1}(2x_s - 1, 2y_s + 1) + f^{J-1}(2x_s + 1, 2y_s - 1) \big) \end{aligned} \tag{A-11}$$

This equation is only defined for values of x_s and y_s such that $0 \le 2x_s \le n_x^{J-1} - 1$ and $0 \le 2y_s \le n_y^{J-1} - 1$ with J the level of the image f and n_x^J and n_y^J the with and height of f^J, respectively.

Therefore, the with n_x^J and the height n_y^J of f^J are the largest integers that satisfy the two conditions:

$$n_x^J \leq \frac{n_x^{J-1}+1}{2}$$
$$n_y^J \leq \frac{n_y^{J-1}+1}{2}$$

From the pyramid representation equation (A-11) the corresponding coordinates $\mathbf{x}_s^J$ of a point s are given by:

$$\mathbf{x}_s^J = \frac{\mathbf{x}_s^{J-1}}{2^J}$$

The overall pyramidal tracking algorithm proceeds as follows: first, the displacement (optical flow) is computed at the deepest pyramid level, say J_m. Then the result of computation is propagated to the upper level $J_m - 1$ in a form of an initial guess for the displacement. This guess is then refined at this level. The result is then propagated to next upper level, say $J_m - 2$ and so on until the original image (level 0) is reached.

In order to compute the optical flow at site s and level J, it is necessary to find the residual pixel displacement vector $\mathbf{u}_s^J = [u_{s,x}^J \quad u_{s,y}^J]^T$ that minimizes the new image matching error function ε^J:

$$\varepsilon^J(\mathbf{u}_s^J) = \sum_{x_i=x_s^J-\omega_x}^{x_s^J+\omega_x} \sum_{y_i=y_s^J-\omega_y}^{y_s^J+\omega_y} \left(f^J(x_i,y_i) - f'^J(x_i + {}^*u_{s,x}^J + u_{s,x}^J, y_i + {}^*u_{s,y}^J + u_{s,y}^J)\right)^2 \tag{A-12}$$

Where ${}^*\mathbf{u}_s^J = [{}^*u_{s,x}^J \quad {}^*u_{s,y}^J]$ is the initial guess for optical flow at site s and level J available from the computations done from level J_m to level $J+1$. The propagation of the initial guess for optical flow from level J to the next level $J-1$ is done by:

$${}^*\mathbf{u}_s^{J-1} = 2({}^*\mathbf{u}_s^J + \mathbf{u}_s^J) \tag{A-13}$$

with ${}^*\mathbf{u}_s^{J_m} = [0 \quad 0]^T$
The integration window is of constant size $(2\omega_x + 1) \times (2\omega_y + 1)$ for all values of J.
The residual flow vector $\mathbf{u}_s^J = [u_{s,x}^J \quad u_{s,y}^J]^T$ is computed through a standard Lucas-Kanade step.
At the minimum, the first derivative of ε with respect to $\mathbf{u}$ is zero:

$$\left.\frac{\partial \varepsilon(\mathbf{u}_s)}{\partial \mathbf{u}_s}\right|_{\mathbf{u}_s=\hat{\mathbf{u}}_s} = [0 \quad 0]$$

We obtain:

$$\hat{\mathbf{u}}_s = G_s^{-1} b_s \tag{A-14}$$

With

$$G_s = \sum_{x_i = x_s^J - \omega_x}^{x_s^J + \omega_x} \sum_{y_i = y_s^J - \omega_y}^{y_s^J + \omega_y} \begin{bmatrix} f_{s,x_i}^2 & f_{s,x_i} f_{s,y_i} \\ f_{s,x_i} f_{s,y_i} & f_{s,y_i}^2 \end{bmatrix}$$

and

$$b = \sum_{x_i = x_s^J - \omega_x}^{x_s^J + \omega_x} \sum_{y_i = y_s^J - \omega_y}^{y_s^J + \omega_y} \begin{bmatrix} f_{s,t} f_{s,x_i} \\ f_{s,t} f_{s,y_i} \end{bmatrix}$$

$f_{s,x}$ and $f_{s,y}$ are the spatial derivatives and $f_{s,t}$ is temporal derivative at site s.

This is possible only if the matrix G is invertible. In practice the estimation of the displacement $\mathbf{u}_s$ is done by an iterative approximation (Newton-Raphson iteration scheme)
The final optical flow $\mathbf{u}_s$ is given by:

$$\mathbf{u}_s = {}^*\mathbf{u}_s^0 + \mathbf{u}_s^0 \tag{A-15}$$

This solution can be expressed in the following form:

$$\mathbf{u}_s = \sum_{J=0}^{J_m} 2^J \mathbf{u}_s^J \tag{A-16}$$

summary of the algorithm

Initialization of pyramidal guess: ${}^*\mathbf{u}^{J_m} = [{}^*u_x^{J_m} \quad {}^*u_y^{J_m}] = [0 \qquad 0]^T$

for $J = J_m$ ***down to*** 0 ***with step of*** -1

Location of a point s on image f^J: $\mathbf{x}_s^J = \frac{\mathbf{x}_s^{J-1}}{2^J}$

Derivatives of f^J with respect to x and y at location $\mathbf{x}_s = (x_s, y_s)$ of the point s:

$$f_x(x_s, y_s) = \frac{1}{2}(f^J(x_s + 1, y_s) - f^J(x_s - 1, y_s))$$

$$f_y(x_s, y_s) = \frac{1}{2}(f^J(x_s, y_s + 1) - f^J(x_s, y_s - 1))$$

Spatial gradient matrix:

$$G_s = \sum_{x_i = x_s^J - \omega_x}^{x_s^J + \omega_x} \sum_{y_i = y_s^J - \omega_y}^{y_s^J + \omega_y} \begin{bmatrix} f_{s,x_i}^2 & f_{s,x_i} f_{s,y_i} \\ f_{s,x_i} f_{s,y_i} & f_{s,y_i}^2 \end{bmatrix}$$

```
    Initialization of iterative Lucas-Kanade method: $\mathbf{u}_s = [0 \quad 0]^T$
    for $k = 1$ to $K$ with step of $1$
        Image difference at point $s$:
            $f_{s,t} = f^J(x_i, y_i) - f'^J(x_i + {}^*u^J_{s,x} + u^J_{s,x}, y_i + {}^*u^J_{s,y} + u^J_{s,y})$
        Image mismatch vector:
            $b = \sum_{x_i = x_s^J - \omega_x}^{x_s^J + \omega_x} \sum_{y_i = y_s^J - \omega_y}^{y_s^J + \omega_y} \begin{bmatrix} f_{s,t} f_{s,x_i} \\ f_{s,t} f_{s,y_i} \end{bmatrix}$
        Optical flow (Lucas-Kanade):
            $\hat{\mathbf{u}}_s = G_s^{-1} b_s$
        Guess for next iteration:
            $\mathbf{u}_s = {}^*\mathbf{u}_s + \mathbf{u}_s$
    end of for-loop on k
    Guess for next level $J - 1$:
        ${}^*\mathbf{u}_s^{J-1} = 2({}^*\mathbf{u}_s^J + \mathbf{u}_s^J)$
end of for-loop on $J$
Final optical flow vector:
    $\mathbf{u}_s = {}^*\mathbf{u}_s^0 + \mathbf{u}_s^0$
Location of point $s$ on image $f'$:
    $\mathbf{x'}_s = \mathbf{x}_s + \mathbf{u}_s$
```

The clear advantage of a pyramidal implementation is that each residual optical flow vector $\mathbf{u}_s^J$ can be kept very small while computing a large overall pixel displacement vector $\mathbf{u}_s$. Assuming that each elementary optical flow computation step can handle pixel motions up to $\mathbf{u}_{s,max}$, then the overall pixel motion that the pyramidal implementation can handle becomes $\mathbf{u}_{s,maxfinal} = (2^{J_m+1} - 1)\mathbf{u}_{s,max}$. For example, for a pyramidal depth of $J_m = 3$, this means a maximum pixel displacement gain of 15. This enables large pixel motions, while keeping the size of the integration window relatively small.

• **Declaring a feature "lost" during the tracking process**
There are two possibilities where a feature can be declared as lost. The first case is when the point falls outside of the image and the other case is when the image patch around the tracked point varies too much between image f and f' (the point disappears due to occlusion for example). For this, a point is declared as

lost if its final cost function $\varepsilon(\mathbf{u}_s)$ is large than a given threshold. This measure permit to use a high sensitive corner detector.

An another case are the areas with repetitive structures (several feature in a small area), the tracking can easily switch to a similar structure in the neighborhood and this errors would be difficult to detect.

B.3 Feature Matching

Feature matching known also as data association is the process of matching observations that are received by the filter to the features to which they correspond.Given that the state estimation process relies on generating statistical estimates of the locations of features in the environment, statistical methods are used for establishing these associations. The most common method for associating observations to features in the map relies on nearest neighbor techniques [53, 83, 84]. A Nearest Neighbor association is taken in this case to be the closest association in a statistical sense.

A key weakness of the SLAM filtering is its dependence on correct association of observed features to mapped ones. A miss-association results in an inconsistent reduction of estimated uncertainty while increasing the actual error. A single incorrect assignment may cause the filter to diverge (i.e., fail irretrievably). In situations with high feature clutter and large location uncertainties, it becomes necessary to devise a means of ensuring correct association. Although this problem has been addressed using complex data association methods as multiple hypothesis approaches [31, 32] or joint compatibility criteria [33, 85], it still requires further research to add the robustness required for cluttered outdoor environments.

B.3.1 Features Matching techniques

In this subsection we will describe some algorithms that have been used widely for feature point matching.

B.3.1.1 Cross-Correlation Matching

The simplest and most common technique for matching feature points in images f_1 and f_2 is to use a cross-correlation (CC) of the image content [33]. Around each feature point a small window of size $\omega \times \omega$ is selected and the two image patches are correlated against each other.

$$\mathrm{CC}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sum\limits_{\mathbf{x}_i \in (N(\mathbf{x}_1), N(\mathbf{x}_2))} f_1(\mathbf{x}_i).f_2(\mathbf{x}_i)}{\sqrt{\sum\limits_{\mathbf{x}_i \in N(\mathbf{x}_1)} f_1^2(\mathbf{x}_i)}.\sqrt{\sum\limits_{\mathbf{x}_i \in N(\mathbf{x}_2)} f_2^2(\mathbf{x}_i)}} \tag{A-17}$$

where $N(\mathbf{x}_1)$ and $N(\mathbf{x}_2)$ are the set of neighboring pixels lying inside the window ω around the feature points $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively.

The correlation A-17 is evaluated for each possible feature match - that is every possible combination of features in image f_1 with features in image f_2. After correlation, the maximum value for each feature point is selected. The algorithm can be accelerated by limiting the search range for correlations to a small search radius.

B.3.1.2 Mean Square Error Matching

Another algorithm for estimating feature matches in images was originally proposed in [29]. It uses not only the intensity values -i.e. gray level values -but takes all three RGB color channels into account. For comparison small image patches are created again, based on a given window size w. Then the difference between two pixel values is evaluated based on the Euclidian distance in RGB color space. Finally, a summation over all pixels in this image patch and normalization with the window size is carried out.

The mathematical description gives rise to a mean square error (MSE) formulation (evaluated in color space) between the two image patches. Image values are now 3-tuples taken from the RGB color space and the distance is evaluated with the Euclidian norm:

$$MSE(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{w^2} \sum_{\mathbf{x}_i \in (N(\mathbf{x}_1), N(\mathbf{x}_2))} \| f_1(\mathbf{x}_i) - f_2(\mathbf{x}_i) \| \tag{A-18}$$

where $N(\mathbf{x}_1)$ and $N(\mathbf{x}_2$ are the set of neighboring pixels as before.

Selection of best matches and the symmetry constraint has to be applied as already described in the preceding sections. Since this is an error and not a similarity measure, the minimum value -and not the maximum as above -has to be used for matching features.

The drawback of this approach is the high impact of small variations in illumination. So it can only be useful in a rigid hardware environment, where all lighting, exposure and posing conditions can be controlled precisely. In addition, its application is generally limited to small baseline applications and the results in section four are therefore very unsatisfactory. A variation to this approach using a different color space, where luminance and colors are separated from each other -i.e. the L*a*b color space -is straightforward, but results are still not convincing [30].

B.3.1.3 Iterative Closest Point Matching

This is a geometry-based method for feature matching. The idea is to find a transformation T that minimizes the geometric distance

$$D = \min_{T} \sum_{\mathbf{x}_1, \mathbf{x}_2} d(\mathbf{x}_1, \mathbf{x}_2) \tag{A-19}$$

between all feature points $\mathbf{x}_1$ of the first image and their nearest neighbor feature point $\mathbf{x}_2$ of the second image.

Appendix C

Kalman Filtering

The Kalman filter, introduced by Rudolph Emil Kalman in the 1950s, is a recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. It's recursive in the way that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.

C.1 Discrete Kalman Filter

The basic Kalman Filter addresses the general problem of estimating the state $\mathbf{x} \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_{t-1} \tag{A-1}$$

where $\mathbf{A}$ is the state transition model which is applied to the previous state $\mathbf{x}_{t-1}$; $\mathbf{B}$ is the control-input model which is applied to the control vector $\mathbf{u}_{t-1}$; $\mathbf{w}_t$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance $\mathbf{Q}_t$, $\mathbf{w}_t = \mathcal{N}(0, \mathbf{Q}_t)$.

At time t an observation (or measurement) $\mathbf{z}_t$ of the true state $\mathbf{x}_t$ is made according to

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t \tag{A-2}$$

where $\mathbf{H}$ is the observation model which maps the true state space into the observed space and $\mathbf{v}_t$ is the observation noise which is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_t$, $\mathbf{v}_t = \mathcal{N}(0, \mathbf{R}_t)$.

The initial state, and the noise vectors at each step $\mathbf{x}_0$, $\mathbf{w}_1$, ..., $\mathbf{w}_t$, $\mathbf{v}_1$ ... $\mathbf{w}_t$ are all assumed to be mutually independent.

The Kalman filter has two distinct phases: Predict and Update.

Prediction The predict phase uses the state estimate from the previous time-step $\hat{\mathbf{x}}_{t-1|t-1}$ to produce an estimate of the state at the current time-step $\hat{\mathbf{x}}_{t|t-1}$

and an estimate of the error covariance matrix $\mathbf{P}_{t|t}$ which is a measure of the estimated accuracy of the state estimate.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{A}\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}\mathbf{u}_{t-1} \tag{A-3}$$
$$\mathbf{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^T + \mathbf{Q}_{t-1} \tag{A-4}$$

Update In the update phase, measurement information at the current time-step is used to refine the prediction to arrive at a new, (hopefully) more accurate state estimate.

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\tilde{\mathbf{e}}_t \tag{A-5}$$
$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t|t-1} \tag{A-6}$$

where

$\tilde{\mathbf{e}}_t = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_{t\|t-1}$	`is the innovation residual`
$\mathbf{K}_t = \mathbf{P}_{t\|t-1}\mathbf{H}^T\mathbf{S}_t^{-1}$	`is the Kalman gain`
$\mathbf{S}_t = \mathbf{H}\mathbf{P}_{t\|t-1}\mathbf{H}^T + \mathbf{R}_t$	`is the innovation covariance`

C.2 Extended Kalam Filter

The previously described Kalman filter addresses the estimation of the state vector in a linear model of a dynamical system. If, however, the model is non linear, we may extend the use of Kalman filtering through a linearisation procedure. The resulting filter is referred to as the extended Kalman Filter (EKF).

Consider a nonlinear dynamical system described by the state-space model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{w}_{t-1}) \tag{A-7}$$
$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{v}_t) \tag{A-8}$$

where, as before, the random variables $\mathbf{w}_t = \mathcal{N}(0, \mathbf{Q}_t)$ and $\mathbf{v}_t = \mathcal{N}(0, \mathbf{R}_t)$ represent the process and measurement noise. Here, however, the functional f denotes a nonlinear transition matrix function that is possibly time-variant. Likewise, the functional h denotes a nonlinear measurement matrix that may be time-variant, too.

The basic idea of the extended Kalman filter is to linearize the state-space model of Eqs. (A-7) and (A-8) at each time instant around the most recent state estimate, $\hat{\mathbf{x}}_{t-1|t-1}$. This is done by computing the Jacobians of f and h:

$$\mathbf{F}_t = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t} \tag{A-9}$$

$$\mathbf{H}_t = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t-1|t-1}} \tag{A-10}$$

Once a linear model is obtained, the standard Kalman filter equations are applied using the computed $\mathbf{F}_t$ and $\mathbf{H}_t$ as the state transition and observation matrices, respectively.

Prediction

$$\hat{\mathbf{x}}_{t|t-1} = f(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t) \tag{A-11}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^T + \mathbf{Q}_t \tag{A-12}$$

Update

$$\tilde{\mathbf{e}}_t = \mathbf{z}_t - h(\hat{\mathbf{x}}_{t|t-1}) \tag{A-13}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}^T \mathbf{S}_t^{-1} \tag{A-14}$$

$$\mathbf{S}_t = \mathbf{H} \mathbf{P}_{t|t-1} \mathbf{H}^T + \mathbf{R}_t \tag{A-15}$$

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \tilde{\mathbf{e}}_t \tag{A-16}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}) \mathbf{P}_{t|t-1} \tag{A-17}$$

Bibliography

[1] A. J. Davison, Y. G. Cid, N. Kita, *Real-time 3D SLAM with wide-angle vision*, In Intelligent Autonomous Vehicles, Lisboa-Portugal, July 2004.

[2] J. Folkesson, P. Jensfelt, H. Christensen, *Graphical SLAM using vision and the measurement subspace*, In IEEE/JRS -Intl Conf. on Intelligent Robotics and Systems (IROS), Edmundton-Canada, August, 2005.

[3] D. Wolf, G.S. Sukhatme, *Online Simultaneous Localization and Mapping in Dynamic Environments*, Proceedings of the Intl. Conf. on Robotics and Automation ICRA New Orleans, Louisiana, April, 2004.

[4] S. Se, D. Lowe, J. Little, *Local and Global Localization for Mobile Robots using Visual Landmarks*, Proceedings of the International Conference on Intelligent Robots and Systems, Maui, Hawaii, USA, Oct. 29 - Nov. 03, 2001, pp.414-420.

[5] J. A. Castellanos, J. Neira, J. D. Tards, *Multisensor fusion for simultaneous localization and map building*, IEEE Trans on Robotics and Automation, December 2001, Vol.17, N.6, pp.908-914.

[6] F. Andrade-Cetto, A. Sanfelin, *Concurrent Map Building and Localization with landmark validation*, 16th Intenational Conference on Pattern Recognition ICPR'02, 2002, vol.2.

[7] J. W. Fenwick, P. M. Newman, J. J. Leonard, *Cooperative Concurrent Mapping and Localization*, Proceedings of the 2002 IEEE International Conference on Robotics and Automation, May 2002, Washington, USA, pp.1810-1817.

[8] S. Thrun, D. Fox, W. Burgard, *A probabilistic approach to concurrent Mapping and Localization for Mobile Robots* , Machine Learning, 1998, Vol.31, N.1-3, pp.29-53.

[9] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, *MonoSLAM: Real-Time Single Camera SLAM*, IEEE Transaction on Pattern Analysis and Machine Intelligence, JUNE 2007, Vol.29, N.6, pp.1052-1067.

[10] C. Stachniss, W. Burgard, *Mobile Robot Mapping and Localization in Non-Static Environments*, AAAI, 2005, pp.1324-1329.

[11] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, S. Thrun, *Learning hierarchical object maps of non-stationary environments with mobile robots*, In Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI), 2002.

[12] D. Hahnel, R. Triebel, W. Burgard, S. Thrun, *Map Building with Mobile Robots in Dynamic Environments*, IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, September 14-19, 2003, vol.2, pp.1557-1563.

[13] `http://en.wikipedia.org/wiki/Geodetic_system`

[14] P. Jensfelt, emphApproches to mobile robot localization in indoor environment, PhD thesis, Departement of Signals, Sensors and Systems, Royal Instiut of Technology, Stockholm, Sweden, 2001.

[15] S. Se, D.G. Lowe, J. Little, *Global localization using distinctive visual features*, International Conference on Intelligent Robots and Systems, IROS, Lausanne, Switzerland, 2002, pp.226-231.

[16] A. Chiusoyz, P. Favaroy, H. Jiny, S. Soatto, *3-D Motion and Structure from 2-D Motion Causally Integrated over Time: Implementation*, Proceedings of the 6th European Conference on Computer Vision-Part II, June 26-July 01, 2000, pp.734-750.

[17] I. Bailey, *Mobile robot localisation and maping in extensive outdoor environments.* PhD thesis, Australian Centre for Field Robotics, University of Sydney, Australia, August 2002.

[18] N. Tomatis, *Hybrid, Metric-Topological, Mobile Robot Navigation*, PhD thesis, Ecole Polytechnique de Lausanne, Switzerland, 2001.

[19] B. Limketkai, R. Biswas, S. Thrun, *Learning Occupancy Grids of Non-Stationary Objects with Mobile Robots*, ISER, 2002

[20] D. Kortenkamp, T. Weymouth, *Topological mapping for mobile robots using a combination of sonar and vision sensing* In Proceedings of the Twelfth National Conference om Artificial Intelligence. AAAI Press/MIT Press, July 1994.

[21] C. Wahlgren, T. Duckett, *Topological Map Building for Mobile Robots Using Omnidirectional Vision*, in Proc of the Third Swedish Workshop on Autonomous Robotics, Stockholm, 2005.

[22] J. I. Nieto, J. E. Guivant, E. Nebot, *The HYbrid Metric Maps (HYMMs): A Novel Map Representation for DenseSLAM*, IEEE ICRA, New Orleans, USA, April 26 - May 1, 2004.

[23] J. Nieto, J. Guivant, E. Nebot, *A Novel Hybrid Map Representation for DenseSLAM in Unstructured Large Environments*,Proc. IEEE-International Conference on Robotics and Automation, New Orleans LA, USA, April 26 - May 01, 2003.

[24] N. Tomatis,I. Nourbakhsh, R. Siegwart, *Hybrid Simultaneous Localization and Map Building: a Natural Integration of Topological and Metric*, Robotics and Autonomous Systems, 2003, Vol.44, pp.3-14.

[25] B. Limketkai, L. Liao, D. Fox, *Relational Object Maps for Mobile Robots*, Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-05).

[26] A. Elfes, *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1989.

[27] H. M. Barber, M. A. Z. Izquierdo, A. G. Skarmeta, *World Modelling with Local Uncertainty for Robot Navigation*, 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU02) , Annecy (Francia), 2002.

[28] A. A. Makarenko, S. B. Williams, H. F. Durrant-Whyte, *Decentralized Certainty Grid Maps*,IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), 2003, Vol.3, pp.3258-3263.

[29] A. Koschan, V. Rodehorst, *Dense depth maps by active color illumination and image pyramids*, Adv. Comput. Vison, Springer, Wien., 1997, pp.137-148.

[30] O. Rusch, C. Ruwwe, U. Zolzer *An evaluation of feature matching algorithms for maritime images*, 6th IASTED International Conference on Visualization, Imaging and Image Processing (VIIP), Palma de Mallorca, Spain, August 28-30, 2006.

[31] N.M. Kwok, G. Dissanayake, *An efficient multiple hypothesis filter for bearing-only SLAM*, In Proc. of the International Conference on Intelligent Robots and Systems, 2004, IROS 2004, September 28- October 2, 2004, Vol. 1, pp.736-741.

[32] D. C.K. Yuen, B. A. MacDonald, *Theoretical Considerations of Multiple Particle Filters for Simultaneous Localisation and Map-Building*, Lecture Notes in Computer Science, 2004, Vol.3213, pp.203-209.

[33] J. Neira, J. D. Tards *Data Association in Stochastic Mapping Using the Joint Compatibility Test*, IEEE Trans. on Robotics and Automation, 2001, Vol.17, pp.890-897.

[34] T. Bailey, *Constrained initialisation for bearing-only slam*, in IEEE International Conference on Robotics and Automation, Taipei, Taiwan, September 2003.

[35] N. M. Kwok, G. Dissanayake, *Bearing-only SLAM in Indoor Environments Using a Modiffied Particle Filter*, Proc. of the Australasian Conference on Robotics & Automation, Brisbane, Australia, 2003, pp.1-8.

[36] T. Fitzgibbons, E. Nebot, *Bearing-Only SLAM using Colour-based Feature Tracking*, Proc. of the Australasian Conference on Robotics & Automation, Auckland, Australia, 2002, pp.234-239.

[37] J. Sol, T. Lemaire, M. Devy, S. Lacroix, A. Monin, *Delayed vs Undelayed Landmark Initialization for Bearing Only SLAM*, In Proceedings of the the IEEE Int. Conf. on Robotics and Automation workshop on SLAM, Barcelona, Spain, April 2005.

[38] J. A. Castellanos, J. Neira, J. D. Tardos, *Map Building and SLAM Algorithms*, in S. S. Ge and F. L. Lewis (Eds), Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications, Series in Control Engineering, CRC, Taylor & Francis Group, May 2006, pp.335-371.

[39] T. Lemaire and S. Lacroix, *Monocular-vision based SLAM using line segments*, in Proceedings of the International Conference on Robotics and Automation, Roma, Italy, 2007.

[40] T. Lemaire, C. Berger, I. Jung, S. Lacroix, *Vision-Based SLAM: Stereo and Monocular Approaches*, International Journal of Computer Vision, vol. 74, no. 3, 2007, pp.343-364.

[41] C. Stachniss, G. Grisetti, W. Burgard, *Recovering particle diversity in a rao-blackwellized particle filter for slam after actively closing loops*, In IEEE International Conference on Robotics and Automation, pp.667-672, 2005.

[42] M. Montemerlo, S. Thrun, D. Koller, B.Wegbreit, *Fastslam: A factored solution to the simultaneous localization and mapping problem.*, In National Conference on Artificial Intelligence, pp.593-598, Vancouver, BC, July 2002.

[43] J.D. Trados, J. Neira, P. Newman, J. Leonard, *Robust mapping and localization in indoor environments using sonar data*, International Journal of Robotics Research, 2002, N. 21, pp.311-330.

[44] S. B. Williams, *Efficient Solutions to Autonomous Mapping and Navigation Problems*, PhD thesis, Australian Centre for Field Robotics, University of Sydney, Australia, September 2001.

[45] C. Estrada, J. Neira, J. D. Tardos, *Hierarchical SLAM: real-time accurate mapping of large environments*, IEEE Transactions on Robotics, 2005, Vol.21, N.4, pp.588-596.

[46] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 2004, Vol.2, N.60, pp.91-110.

[47] K. Mikolajczyk, C. Schmid. *A performance evaluation of local descriptors*, Proceedings of Computer Vision and Pattern Recognition, 2003.

[48] M Deans, M. Hebert,*Experimental comparison of techniques for localization and mapping using a bearing-only sensor*, In Seventh International Symposium on Experimental Robotics, Honolulu, Hawaii, December 2000, Vol. 271, pp. 395- 404.

[49] O. D. Faugeras, *What can be seen in three dimensions with an uncalibrated stereo rig*, In ECCV '92: Proceedings of the Second European Conference on Computer Vision, pp.563-578, London, UK, 1992. Springer-Verlag.

[50] R. I. Hartley, *Estimation of relative camera positions for uncalibrated cameras*, In ECCV '92: Proceedings of the Second European Conference on Computer Vision, pp.579-587, London, UK, 1992. Springer-Verlag.

[51] D. Nistr, *Preemptive RANSAC for live structure and motion estimation*, In Proc. IEEE International Conference on Computer Vision, ICCV'2003, 2003, pp.199-206.

———

[52] L. Feng, J. Borenstein, H. R. Everett, *Where am I? : Sensors and Methods for Autonomous Mobile Robot Positioning*, technical report UM-MEAM-94-21, University of Michigan, December 1994.

[53] T. Bar-Shalom, T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, 1988.

[54] R. Sim, P. Elinas, M. Griffin, J. J. Little, *Vision-based SLAM using the rao-blackwellised particle filter*, in Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, 2005.

[55] J. B. Folkesson, H.Christensen,*Robust SLAM*,IAV-2004,Lisboa, Portugal, July 5-7, 2004,

[56] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. The MIT Press, September 2005.

[57] A.R. Jimenez, F. Seco, R. Ceres, L. Calderon,*Absolute Localization using Active Beacons: A survey and IAI-CSIC contributions*

[58] J. Borenstein, H. Everett, L. Feng, D. Wehe, *Mobile robot positioning: Sensors and techniques*, Journal of Robotic Systems 1997, Vol.14, N.4, pp.231-249.

[59] S. Livatino, C. B. Madsen, *Acquisition and Recognition of Visual Landmarks for Autonomous Robot Navigation*, SIRS'00 - 8th International Symposium on Intelligent Robotic System, July 18-20, 2000, pp.269-280.

[60] S.Se, D. Lowe, J. Little, *Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks*, in The International Journal of Robotics Research, August 2002, Vol.21, No.8, pp.735-758.

[61] J. Guivant, F. Masson, E. Nebot,*Simultaneous localization and map building using natural features and absolute information*, Journal of robotics and autonomous systems, August 31, 2002, Vol.40, pp.79-90.

[62] J. Neir, M. Isabel Ribeiro, J. Domingo Tardos, *Mobile Robot Localization and Map Building using Monocular Vision*, 5th Int. Symp. on Intelligent Robotic Systems, Stockholm, Sweden, July 8-11, 1997, pp.275-284.

[63] S. Kumar, F. Ramos, B. Upcroft, H. Durrant-Whyte, *A statistical framework for natural feature representation*, In IEEE/JRS- Intl. Conf. on Intelligent Robotics and Systems, Edmundton-Canada, August 2005, pp.1-6.

[64] R. Negenborn, *Robot Localization and Kalman Filters*, Master tehsis, Institute of Information and Computing Sciences, University of Utrecht, the Netherlands, 2003.

[65] D. Fox, *Markov Localization- A Probabilistic Framework for Mobile Robot Localization and Navigation*, PhD thesis, Institute of Computer Science III, University of Bonn, Germany, December, 1998.

[66] Schaffer, G., Gonzalez, J., and Stentz, A., *Comparison of Two Range-based Pose Estimators for a Mobile Robot*, Proceedings of the SPIE Conference on Mobile Robots, Boston, MA, November 18-20, 1992, pp.661-667.

[67] J. Neira, J. D.Tardos, J. A.Castellanos, *Linear time vehicle relocation in slam*, In IEEE- ICRA, Taiwan, September 2003.

[68] J. J. Leonard , R. J. Rikoski, P. M. Newman, M. Bosse, *Mapping partially observable features from multiple uncertain vantage points*, Intl Jour of Robotics Research, 2002.

[69] H. Jin, P. Favaro, S. Soatto, *Real-time feature tracking and outlier rejection with changes inillumination*, Computer Vision, 2001. ICCV'2001. 2001, Vol. 1, pp.684-689

[70] J. E. Guivant, E. M. Nebot, J. Nieto, F. Masson, *Navigation and mapping in large unstructured environments*, IJRR, April 2004, Vol.23, N.4.

[71] B. Kuipers, Y.T. Byun, *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, Journal of Robotics and Autonomous Systems, 1991, Vol.8, pp.47-63.

[72] Jens-Steffen Gutmann, *Markov-Kalman Localization for Mobile Robots*, Proceedings of the 16 th International Conference on Pattern Recognition, ICPR02, 2002, Vol.2.

[73] J. Vermaak, S. J. Godsill, P. Perez, *Monte Carlo Filtering for Multi-Target Tracking and Data Association*, IEEE 2004.

[74] N. Karlssonet, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, M. E. Munich, *The vSLAM Algorithm for Robust Localization and Mapping*, In Proceedings of Int. Conf. on Robotics and Automation ICRA 2005.

[75] J. Shi, C. Thomasi, *Good Features to track*, in Proc. of IEEE Conference on computer Vision and Pattern Recognition, 1994, pp.593-600.

[76] C. F. Olson, *A General Method for Geometric Feature Matching and Model Extraction*, International Journal of Computer Vision, October 2001, Vol.1, N.45, pp.39-54.

[77] J. Meltzer, R. Gupta, S. Ming-Hsuan Yang Soatto, *Simultaneous Localization and Mapping using Multiple View Feature Descriptors*, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004), September 28-October 2 2004, Vol.2, pp.1550-1555.

[78] C. Cohen, F. Koss, *A comprehensive study of three-object triangulation*, In SPIE Mobile Robots VII, 1992.

[79] W. L. Roque, D. Doering, *Trajectory planning for lab robots based on global vision and Voronoi roadmaps*, Robotica, 2005, Vol.23, pp.467-477.

[80] Y. Hayashi, S. Tohyama, H. Fujiyoshi, *Mosaic-Based Global Vision System for Small Size Robot League*, RoboCup 2005, Lecture Notes in Computer Science, 2006, pp.593-601.

[81] D.C. K. Yuen, B. A. MacDonald, *Vision-based localization algorithm based on landmark matching, triangulation, reconstruction, and comparison*, IEEE Transaction on Robotics, April 2005, Vol. 21, N. 2, pp.217-226.

[82] P. M. Newman,*On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*,PhD thesis, Australian Centre for Field Robotics, University of Sydney, Australia, March 1999.

[83] J. Neira, J. D. Tards,*Data Association in Stochastic Mapping Using the Joint Compatibility Test*, IEEE Trans. on Robotics and Automation, December 2001, Vol. 17, N. 6, pp.890-897.

[84] T. L. Song, J. Ryu, *A Probabilistic Nearst Neighbor Filter for Target Tracking in a Clutter Environment*, AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, Texas, August 11-14, 2003.

[85] J. J. Leonard, P. M. Newman, R. J. Rikoski, J. Neira, J. D. Tards, *Towards Robust Data Association and Feature Modeling for Concurrent Mapping and Localization*, Springer Tracts in Advanced Robotics, 2003, Vol.6, pp.7-20.

[86] H. Neemuchwala, A. O. Hero, and P.L. Carson, *Image matching using alpha-entropy measures and entropic graphs*, Signal Processing, Special Issue on Content-based Visual Information Retrieval, 2005, Vol. 85, pp.277-296.

[87] Yamauchi, B., Schultz, A., and Adams, *Integrating exploration and localization for mobile robots*, Adaptive Behavior, 1999, Vol.7, N.2, pp.217-230.

[88] S. Livatino, C.B. Madsen, *Autonomous robot navigation with automatic learning of visual landmarks*, Symposium on Intelligent Robotics Systems (SIRS), 1999.

[89] T. Goedem, *Visual Navigation*, PhD thesis, Katholieke Universiteit Leuven, ESAT-PSI-VISICS, Leuven, December 19, 2006.

[90] D. Cremers, A. Yuille, *A Generative Model Based Approach to Motion Segmentation*, DAGM 2003, 2003, pp.313-320.

[91] C. Ridder, O. Munkelt, H. Kirchner, *Adaptive Background estimation and foreground detection using Kalman filtering*, Proceedings of International Conference on recent Advances in Mechatronics (ICRAM), 1995, pp.193-199.

[92] M. Pic, L. Berthouze, T. Kurita, *Adaptive Background estimation: Computing a pixel-wise learning rate from local confidence and global correlation values*, IEICE trans. INF. & SYST., January 2004, Vol. E87.

[93] A. Robels-Kelly, E. R. Hankock, *An EM-like Algorithm for motion Segmentation via Eigen decomposition*, In Proceedings of the British Machine Vision Conference, 2001, pp.123-132.

[94] G. Gordon, T. Darrell, M. Harville, J. Woodfill, *Background Estimation and removal based range and color*, IEEE computer Society Conference on vision and pattern recognition, June 1999.

[95] H. S. Sawhney, S. Ayer, *Compact Representations of video through dominant and multiple motion estimation*, IEEE trans. On pattern analysis and machine intelligence, August 1996, Vol.18, N.8.

[96] J. W. McCandless, *detection of Aircraft in video sequences using a predective OF*, Optical Engineering, 1999, Vol.3, pp.523-530.

[97] C. Fowlkes, S. Belongie, J. Malik, *Efficient Spatiotemporal Grouping Using the Nystrom Method*, CVPR'01, Kaui, , December 2001, Vol.I, pp.231-238.

[98] S.C. Han, J.W. Woods, *Adaptive Coding of Moving Objects for Very Low Bit-rates*, IEEE journal on selected areas in communications, January 1998, Vol.16, pp.55-70.

[99] C. Stauffer, *Learning patterns of activity using real-time tracking*, IEEE trans. On pattern analysis and machine intelligence, August 2000, Vol.22, N.8.

[100] P.M.Q. Aguiar, J. M. F. Moura, *Maximum Likelihood Estimation of the Template of a Rigide Moving Object*, International Workshop on Energy Minimization methods in computer vision and pattern recognition, Sep 2001.

[101] K. Gerald, *Motion-based Segmentation and Classification of Video Objects*, PhD thesis, 2002.

[102] M. Ben Ezra, S. Peleg, B. Rousso, *motion segmentation using convergence properties*, APRA Image Understanding Workshop, November 1994.

[103] S.S. Beauchemin, J. L. Barron, *on discontinuous optical flow*, Journal of mathematical Imaging and Vision, 1998.

[104] J. Y. A. Wang, E. H. Adelson, *Representing Moving Images with layers*, IEEE transaction on Image Processing, 3(5), May 1994, pp.625-638.

[105] M. Smith, *Reviews of OF-Motion Segmentation -Edge Finding - Corner Finding*, technical report TR97SMS1, 1997.

[106] R. Vidal, S. Sastry, *Segmentation of dynamic scenes from image intensities*, IEEE workshop on Vision and Motion computing, December 2002, pp.44-49.

[107] D. W. Murray, B. F. Buxton, *Scene segementation from visual motion using global optimization*, IEEE Trans Pattern Analysis and Machine Intelligence, vol.9, n.2, March 1987, pp.220-228.

[108] J. Y. A. Wang, E. H. Adelson, *spatio-temporal segmentation of video data*, SPIE, Image and Video Processing II, San Jose, February 1994, Vol. 2182.

[109] P.A. Smith, *Edge based motion segmentation*, PhD thesis, Cambridge University, August 2001.

[110] M. C. Allmen, *Image sequence description using spatiotemporal flow curves: Toward Motion based recognition*, PhD thesis, University of Wisconsin - Madison, Computer Science Department, 1991.

[111] K. Y. Wong, *Tracking Based Motion Segmentation under Relaxed Statistical Assumptions*, technical report CS-2003-09, October 14, 2003.

[112] D. Cremers, S. Soatto, *Variational Space-Time Motion Segmentation*, 9th IEEE international Conference on computer Vision ICCV 2003, October 13 - 16, 2003, Vol.2.

[113] G. Sudhir, J. C. M. Lee, *Video Annotation by motion interpretation using Optical Flow streams*, Technical report HKUST-C396-16, July 13,1997.

[114] A. Elgammal, D. Harwood, L. Davis, *Non-parametric Model for Background Subtraction*, In Proceedings of the 6th European Conference on Computer Vision, June 26 - July 01, 2000, Part II, pp.751-767.

[115] Y. Ren, C. Chua, Y. Ho, *Statistical background modeling for non-stationary camera*,Pattern Recognition Letters, 2003, Vol.24, pp.183-196.

[116] P.H.S.Torr, *Motion Segmentation and Outlier Detection*, PhD thesis, University of Oxford, UK, December 1995.

[117] S.M. Smith, *ASSET-2/Real Time motion segmentation and object tracking*, Technical report TR95SMS2b, 1995

[118] R. Mech, J. Stauder, *2D shape estimation for moving objects considering a moving camera and cast shadows*, VCIP'99, part of Electronic Imaging, January 23-29, 1999, San Jose, USA.

[119] R. Mech, M. Wollborn, *for 2D shape estimation of moving objects in video sequence considering a moving camera*, Signal Processing, 1998, Vol.66, No.2, pp.203-217.

[120] R. Mech, M. Wollborn *A Noise Robust Method for 2D shape estimation of moving objects in video sequence considering a moving camera*,Workshop on Image Analysis for multimedia Interactive Services, 1997, Louvaine la Neuve, Belgium, June 24-25, 1997.

[121] R. Mech, *Robust 2D shape Estimation of moving objects considering spatial and temporal coherency in one Map detection rule*, ICIP 2000.

[122] S. Moon, Y. Park, Y. Yoon, K. Kim, C. Lee, *A global motion estimation scheme using inter-relations of motion vectors os symmetrical triple points*, Korean J. Math. Sciences, 2002, Vol.9, N.1, pp.35-45.

[123] D. Demirdjian, R. Horaud, *A projective Framework for Scene Segmentation in presence of Moving Objects*,IEEE Conference on Computer Vision and Pattern Recognition, June 1999, Vol.1, pp.2-8.

[124] P. A. Smith, *Edge-based Motion Segmentation*, PhD thesis, University of Cambridge, August 2001

[125] G. Khne, S. Richter, M. Beier, *Motion-based segmentation and contour-based classification of video objects* . ACM Multimedia, 2001, pp.41-50.

[126] D. Farin, P. H. N. de With, W. Effelsberg, *A segmentation system with model assisted completion of video objects*,SPIE Proc. Visual Communications and Image Processing (VCIP03). 5150, 1, 8-11 July 2003, Lugano (CH), 2003, pp.366-377.

[127] N. Paragios, G. Tziritas, *Adaptive Detection and Localization of Moving Objects in Image Sequences*, Signal Processing: Image Communication, 1992, Vol.4, pp.457-476.

[128] S. Y. Chien, S.Y.Ma, L.G. Chen, *An efficient video segmentation algorithm for real time MPEG-4 camera system* in Proceedings of Visual Communication and Image Processing (VCIP2000), 2000.

[129] Y. Sugaya, K. Kanatani, *Automatic camera model selection for multibody motion segmentation* , Momoires of the faculty of Engineering, Okayama University, Novembre 2002, Vol.37, N.1, pp.45-54.

[130] T. Meir, K.N. Ngan, *Automatic video sequence segmentation using object tracking*, IEEE TENCON'97, Brisbane, Australia, December 1997.

[131] N.D. Vitoria Lobo, J. K. Tsotsos, *Computing Egomotion and detecting independent motion from image motion using collinear points*, Computer vision and image understanding, July 1996, Vol.64, N.1, pp.21-52.

[132] C. Silvia, J. Santos-Victor, *Direct Egomotion Estimation*, International Conference on Pattern Recognition, ICPR96, Austria, February 1996.

[133] M.B.Van Leeuwen, F.C.A. Groen, *Egomotion Estimation for Traffic Applications*,Technical report, Computer Science Institute, University of Amsterdam, The Netherlands, July 2001.

[134] B. Zitova, J. Flusser, *Estimation of camera planar motion from blurred images*, Proceedings of the IEEE International Conference on Image Processing. ICIP'02. Rochester, 2002, pp.329-332.

[135] H. S. Sawhney, Y. G. R. Kumar, *Independent Motion Detection in 3D Scenes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, Vol. 22, N.10, pp.1191-1199.

[136] J. Ruiz-del-Solar, P.A. Vallejos, *Motion detection and tracking for an AIBO robot using camera motion compensation and Kalman Filtering*, 8th International Workshop on RoboCup 2004, Lecture Notes in Artificial Intelligence, Springer, Lisbon, Portugal.

[137] M.B.Van Leeuwen, F.C.A. Groen, *Motion Estimation with a Mobile Camera for traffic Applications*, Proc. IEEE Instrumentation and Measurement Conference, May 2000, pp.354-359.

[138] B. Heisele, *Motion-based object detection and tracking in color image sequences*, Fourth Asian Conference on Computer Vision, Taipei, 2000, pp.1028-1033.

[139] D. Demirdjian, R. Horaud, *Motion-Egomotion discrimination and motion segmentation from image pair streams*, Computer Vision and Image Understanding, April 2000, Vol. 78, N.1, pp.53-68.

[140] J.S. Lee, K.Y. Rhee, S.D. Kim, *Moving Target Algorithm based on the confidence measure of motion vectors*, IEEE International conference on Image processing, Thessaloniki, Greece, 2001, Vol. 1, pp.369-372.

[141] M. Betke, E. Haritaoglu, L. S. Davis, *Multiple Vehicle Detection and tracking in hard Real time*, IEEE Symposium on Intelligent Vehicles, Tokyo, Japan, September, 1996, pp.351-356.

[142] J.M. Odobez and P. Bouthemy, *Separation of moving regions from background in an image sequence acquired with a mobile camera*, Appeared as Chapter (Chapter 8, pp.283-311) in : Video Data Compression for Multimedia Computing, H. H. Li, S. Sun, and H. Derin (eds.), Kluwer Academic Publisher, 1997.

[143] V. Mezaris, I. Kompastsiaris, N.V. Boulgouris, M. G. Strintzis, *Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing and retrieval*, IEEE Trans. Circuits and Systems for Video Technology, May 2004, Vol. 14, N.5.

[144] W. J. Maclean, A. D. Jepson, R.C. Frecker, *Recovery of egomotion and Segmentation of independent object using EM*, Proceedings of the 5th British Machine Vision Conference, 1994, pp.13-16.

[145] J. Hornegger, C. Tomasi, *Representation Issues in the ML Estimation of Camera Motion*, In Proceedings of the 7th International Conference on Computer Vision (ICCV), Corfu, Greece, September 1999, pp.914-919.

[146] G. Lefaix,E. Marchand, P. Bouthemy, *Motion-based Obstacle Detection and Tracking for Car Driving Assistance*, IAPR Int. Conf. on Pattern Recognition, ICPR'2002, Quebec, Canada, August 2002, Vol.4, pp.74-77.

[147] E. Hayman, J.O. Eklundh, *Statistical Background Subtraction for a mobile observer*, In the Proceedings of the 9th International Conference on Computer Vision, Nice, France, 2003, pp.67-74.

[148] W. J. Maclean, *Recovery of Egomotion and segmentation of independent object motion using the EM algorithm*, PhD thesis, University of Toronto, 1996.

[149] R. Feghali, A. Mitiche, *Tracking with simultaneous camera motion subtraction by level set spatio-temporal surface Evolution*, ICIP'03, 2003, Vol. III, pp.929-932.

[150] J. Konrad, M. Ristivojevic, *Video Segmentation and occlusion detection over multiple frames*, SPIE symposium on Electronic Imaging, Image and Video Communications and Processing, Santa Clara, USA, January 20-24,2003.

[151] B. Jung, G. S. Sukhatme, *Detecting Moving Objects using a single camera on a mobile robot in an outdoor environment*8th conference on intelligent autonomous systems, Amsterdam, The Netherlands, March 10-13, 2004, pp.980-987.

[152] S. Fejes, L. S. Davis, *Detection of independent motion using directional motion estimation*, Computer Vision and Image Understanding, May 1999, Vol.74, N.2, pp.101-120.

[153] R. Dahyot, P. Charbonnier, F. Heitz, *Unsupervised statistical detection of changing objects in camera-in-motion video*, ICIP-2001, International Conference on Image processing, Thessaloniki, Greece, October 7-10, 2001.

[154] T. Y. Tian, C. Tomasi, D. J. Heeger, *Comparison of approaches to egomotion computation*, Proceedings of computer Vision and pattern recognition, 1996, pp 315-320.

[155] Y. Ma, *A differential geometric approach to computer vision and its applications in control*, PhD thesis, University of California at Berkeley, 2000.

[156] R. Mech, *Robust 2D shape Estimation of moving objects considering spatial and temporal coherency in one Map detection rule*, Proc. International Conference on Image Processing, ICIP'2000, Vancouver, Canada, September 2000.

[157] C.Zhang, S.C. Chen, M.L. Shyn, S. Peeta, *Adaptive Background learning for vehicle detection and spatio-temporal tracking*, ICICS-PCM 2003, December 15-18, 2003.

[158] M.M. Chang, A. M. Tekalp, M.I.Sezan, *Simultaneous Motion Estimation and Segmentation*,IEEE transaction on Image Processing, 1997, Vol.6, N.9, pp.1326-1333.

[159] D.J. Thirde, G.A. Jones,J. Flack, *Spatio-temporal semantic object segmentation using probabilistic sub-object regions*, in Proceedings of the 14th British Machine Vision Conference, BMVC03, Norwich, UK, September 911, 2003.

[160] P. Wayne Power, J. A. Schoonees, *Understanding Background Mixture Models for Forground Segmentation*, Proceedings Image and Vision Computing New Zealand 2002, Auckland, New Zealand, November 26-28, 2002.

[161] Z. Zivkovic, *Motion Detection and Object Tracking in Image Sequences*, PhD thesis, University of Twente, Netherlands, June 2003.

[162] J. Migdal, W. E. L. Grimson, *Background Subtraction Using Markov Thresholds*, Proceedings of the IEEE workshop on Motion and Video Computing WACV/Motion'05.

[163] Y. Zhou, W. Xu, H. Tao, Y. Gong, *Background Segmentation Using Spatial-Temporal Multi-Resolution MRF*, Proceedings of the IEEE workshop on Motion and Video Computing WACV/Motion'05.

[164] N. Paragios, G. Tziritas, *Detection and location of moving objects using deterministic relaxation Algorithms*, ICS-FORTH TR 155, Heraklion, Greece, December 1995.

[165] S. Khan, M. Shah *Object Based Segmentation of Video using Color, Motion and Spatial Information*,Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR'01.

[166] Y. X. Tong, Y. S. Tang, Z. H. Wen, *Region Tracking via HMMF in Joint Feature Spatial Space*, Proceedings of the IEEE workshop on Motion and Video Computing WACV/Motion'05.

[167] I. Patras, R.L. Lagendijk, *Video Segmentation by MAP Labeling of Watershed Segments*, IEEE Transactions on Pattern Analysis and Machine Intelligence, March 2001, Vol.23, N.3, pp.326-332.

[168] Y. Tsaig, A. Averbuch, *A Region based MRF Model for Unsupervised Segmentation of Moving Objects in Image Sequences*, In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, December 2001, VO.1, pp.889-896.

[169] Y. Tsaig, A. Averbuch, *Automatic Segmentation of Moving Objects in Video Sequences; A Region Labeling Approach*,IEEE Trans. Circuits Syst. Video Techn., July 2002, No.7, pp.597-612. .

[170] R. Cucchiara, A. Prati, R. Vezzani, *Real-Time motion Segmentation from moving cameras*, in Real-Time Imaging, 2004, Vol. 10, N.3, pp.127-143.

[171] M. Gelgon, P. Bouthemy, *A region level motion based graph representation and labeling for tracking a spatial image partitions*, Pattern Recognition, April 2000, Vol.33, N.4, pp.725-740.

[172] L. Vincent, P. Soille, *Watershed in Digital Spaces: An efficient Algorithm based on Immersion Simulations*, IEEE trans. Pattern Analysis and Machine Intelligence, June 1991, Vol.13, N.6, pp.583-589.

[173] F. Dufaux, J. Konrad, *Efficient, Robust, and Fast Global Motion Estimation for Video Coding*, IEEE Trans. on Image Processing, March 2000, Vol.9, N. 3, pp.497-501.

[174] R. Adams, L. Dischof, *Seeded region growing*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1994, Vol.16, N.6, pp.641-647.

[175] P.B. Chou, C.M. Brown, *The theory and practisse of Bayesian image modelling*, Int. J. Comput. Vision 1990, Vol.4, pp.185-210.

[176] Y.Wang, T. Tan, K.F. Loe, *Video Segmentation Based on Graphical Models*, Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03).

[177] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, *Motion compensated interframe coding of video conferencing*, in Proc. Nat. Telecommun. Conf., New Orleans, LA, December 1981, pp.G5.3.1-G.5.3.5.

[178] D. Lowe, *Object Recognition from Local Scale-Invariant Features*, In Proceedings of the International Conference on Computer Vision, Corfu, Greece, September 1999, pp.1150-1157.

[179] S. Kumar, M. Hebert, *Discriminative Fields for Modeling Spacial Dependencies in Natural Images*, Advances in Neural Information Processing Systems, NIPS 16, 2004.

[180] S. Kumar, M. Hebert, *Approximate Parameter Learning in Discriminative Fields*, Snowbird Learning Workshop, Utah, 2004.

[181] J. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, In Proc. Int. Conf. on Machine Learning, 2001.

[182] L. Younes, *Synchronous random fields and image restoration*,IEEE Transactions on Pattern Analysis and Machine Intelligence, April 1998, Vol.20, N.4, pp.380-390

[183] J. Zerubia, R. Chellappa, *Mean field approximation using Compound Gauss-Markov Random Field for edge detection and image estimation*, IEEE Trans. Neural Networks, 1993, Vol.8, pp.703-709.

[184] C.A.Bouman, M. Shapiro, *A Multiscale Random Field Model for Bayesian Image Segmentation*, IEEE Trans. Image Proc. 1994, Vol.3, pp.162-177.

[185] T.Sziranyi, J.Zerubia, *Markov Random Field Image Segmentation using Cellular Neural Network*, IEEE Tr. Circuits and Systems I., 1997, Vol.44, pp.86-89.

[186] D. Geiger, F. Girosi, *Parallel and Deterministic Algorithms from MRFs: Surface Reconstruction*, IEEE Transactions on Pattern Analysis and Machine Intelligence, May 1991, Vol.13 , N. 5, pp.401-412

[187] J.M. Letang, P. Bouthemy, F. Rebuffel, *Robust motion detection with temporal decompostionand statistical regularization*, report, october 1995, IRISA, Rennes University, France.

[188] J.W. Modestino, J. Zhang, *A markov Random Field model-based approach to image interpretation*, IEEE Trans. Pattern Analysis and Machine Intelligence, June 1992, Vol.14, N.6, pp.606-615.

[189] I.A.L. Manolis, A. A. Antonis,, *Vision-Based Camera Motion Recovery for Augmented Reality*,Proceedings of the Computer Graphics International, CGI'04, 2004.

[190] A. Katartzis, H. Sahli, *A Model-Based Approach to the Automatic Extraction of Linear Features from Airborne Images, using Mathematical Morphology and MRF theory*, Technical Report: IRIS-TR-0062, VUB-ETRO department, 2000.

[191] B. Johansson, *multiscale curvature detection in computer vision* Thesis, Department of Electrical Engineering Linkopings universitet, Sweden, 2001.

[192] G. Olague, B. Hernndez, *Flexible Model-based Multi-corner Detector for Accurate Measurements and Recognition.* 16th International Conference on Pattern Recognition. Qubec, Canada. August 11-15, 2002, Vol.2, pp.578-583.

[193] H. P. Moravec, *Visual Mapping by a Robot Rover*, International Joint Conference on Artificial Intelligence, 1979, pp.598-600.

[194] C. Harris, M. Stephens, *A Combined Corner and Edge Detector*, Proc. Alvey Vision Conf., Univ. Manchester, 1988, pp.147-151.

[195] C. S. Kenney, M. Zuliani, B.S. Manjunath, *An axiomatic approach to corner detection*, Computer Vision and Pattern Recognition, CVPR 2005. June 20-25, 2005, Vol.1, pp.191-197.

[196] C. Tomasi, T. Kanade, *Detection and tracking of point features*, Tech. Rept. CMU-CS-91132. Pittsburgh, CarnegieMellon U. School of Computer Science, 1991.

[197] JY. Bouguet, *Pyramidal implementation of the Lukas Kanade feature tracker. Description of the algorithm*, (OpenCV) Intel Corporation, Microprocessor Research Labs, 2000. http://www.intel.com/research/mrl/research/opencv/.

[198] K.P. Karmann, A. von Brandt, *Moving Object Recognition Using an Adaptive Background Memory*, in Proc. Time-varying, Image Processing and Moving Object Recognition, Amsterdam, 1990, pp.297-307.

[199] D. Koller, J. Weber, J. Malik, *Robust multiple car tracking with occlusion reasoning*, in ICCV 1, 1994, pp.189-196.

[200] Z. Zhu, G. Xu, E. M. Riseman, A. R. Hanson, *Fast Construction of Dynamic and Multi-Resolution* 360^o *Panoramas from Video Sequences*,Accepted to Image & Vision Computing Journal.

[201] Z. Zivkovic, *Improved Adaptive Gaussian Mixture Model for Background Subtraction*, Proc. Intl Conf. Pattern Recognition, 2004, vol. 2, pp.28-31.

[202] C.E. Priebe, D.J.Marchette, *Adaptive mixture density estimation*, Pattern Recognition, 1993, Vol.26, N.5 pp.771785.

[203] M. Figueiredo, A.K. Jain, *Unsupervised learning of finite mixture models*, IEEE Transaction on Pattern Analysis and Machine Intelligence, 2002, Vol.24, N.3, pp.381-396.

[204] J. Phillips, S. Sarkar, I. Robledo, P. Grother, K. Bowyer, *Baseline results for the challenge problem of human id using gait analysis*, In Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, 2002.

[205] V. Cerny, *A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm*, Journal of Optimization Theory and Applications, 1985, Vol.45, pp.41-51

[206] D. Farin, P. H. N. de With, *Evaluation of a feature-based global motion estimation system*, in Visual Communications and Image Processing, July 2005, Beijing, China, vol.5960 p. 1331-1342.

[207] J.P. Laimond, S. Sekhvat, F. Lamraux , *Guidlines in Nonholonomic Motion Planing*, Lectures Notes in Control and Information Sciences 229, Springer, 1998, pp343-397.

[208] M. Oussalah , H. T. Nguyen, V. Kreinovich , *A New Derivation of Centroid Defuzzification* , Technical report, Centre for Software Reliability, University Northampton Squares, UK, 2001.

[209] Nurcahyo, G.W., Shamsuddin, S.M., Alias, R.A., Mohd. Noor Md. Sap, *Selection of Defuzzification Method to Obtain Crisp Value for Representing Uncertain Data in a Modified Sweep Algorithm*, JCS&T, 2003, Vol.3, N.2, pp.22-28.

[210] A. V. Patel, *Simplest Fuzzy PI Controllers under Various Defuzzification Methods*,International Journal of Computational Cognition, March, 2005, Vol.3, N.1, pp.21-34.

[211] T. A. Runkler, *Extended Defuzzification Methods and Their Properties*, IEEE Transactions, 1996, pp.694-700.

[212] L.X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[213] L.X. Wang, *Stable adaptive fuzzy control of nonlinear systems*, IEEE Trans. Fuzzy System 1, 1993.

[214] S. Lin , Y. Chen, *Design of adaptive fuzzy sliding mode control for nonlinear system control*, the 3rd IEEE conference fuzzy system, Orlando, FL, June, 1994, Vol.1. pp.35-9.

[215] F.J. Lin, S.L. Chiu, *Adaptive fuzzy sliding-mode control for PM synchronous servo motor drives*, IEE Proc. Control Theor. Appl., 1998, Vol.1, pp.6372.

[216] T. Chai, S. Tong, *Fuzzy direct adaptive control for a class of nonlinear systems*, Fuzzy Sets Syst. 103, 1999, pp.379387.

[217] R. Pytelkova, P. Husek, *Stability Analysis of Discrete-Time Takagi-Sugeno Fuzzy Systems* , International Conference on Computational Science, ICCS 2002, Amsterdam, The Netherlands, April 21-24, 2002.

[218] S. Lee, G.G. Yen, *Analysis of Takagi-Sugeno Fuzzy Models in System Identification for Model-Based Control*, Journal of Control and Intelligent Systems, 2004, Vol.32, N.2, pp.69-79.

[219] L.K. Wong, F.H.F. Leung,P.K.S. Tam, *Design of fuzzy logic controllers for Takagi-Sugeno fuzzy model based system with guaranteed performance*, International Journal of Approximate Reasoning, May 2002, Vol.30, N.1, pp.41-55.

[220] H. Bahri, *Fuzzy Logic Based Robot Path Planning*, Master Thesis, ETRO, VUB, 2006.

[221] Alessandro Saffiotti, *The Uses of Fuzzy Logic in Autonomous Robot Navigation: a catalogue raisonné*, 1997, Technical Report TR/IRIDIA/97-6.

[222] A. Saffiotti, Z. Wasik, *Using Hierarchical Fuzzy Behaviors in the RoboCup Domain*, Center for Applied Autonomous Sensor Systems, Orbero University, 2003.

[223] S. Tongchai, S. Suksakulchai, D. M. Wilkes, N. Sarkar, *Sonar Behavior-Based Fuzzy Logic Control for a Mobile Robot*, 2000, Intelligent Robotics Laboratory, School of Engineering Vanderbilt University Nashville.

[224] P. Rusu, Emil M. Petriu, T. E. Whalen, A. Cornell, H. J. W. Spoelder, *Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation*, 2003, IEEE Transactions on Instrumentation and Measurement, Vol.52, N.4, pp.1335-1340.

[225] H. Seraji, A. Howard, *Behavior-based robot navigation on challenging terrain: A fuzzy logic approach*, IEEE Trans. Robotics Automat, 2002, Vol.18, N.3, pp.308-321.

[226] S. X. Yang, L. Hao, L., M. Meng, *Fuzzy control of a behaviour-based mobile robot*, The 12th IEEE Internat. Conf. on Fuzzy Systems, FUZZ'03, May, 2003, Vol.1, pp.319-324.

[227] S. Parasuraman, V. Ganapathy, Bijan Shirinzadeh, *Behavior Coordination and Selection using Situation Context-Dependent Method for Behavior based Robot Navigation using AI Techniques for Real world Environments* Proceeding of the 5th Asian Control IEEE conference ASCC2004, Melbourne, Australia, July, 2004.

[228] B. K. Quek, J. X. Xu, P. Vadakkepat, *Developing a Fuzzy Action-Selection Strategy for Mobile Robots* The Second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003),Singapore, December 15-18, 2003.

[229] N. H. C. Yung, Cang Ye, *An Intelligent Mobile Vehicle Navigator Based on Fuzzy Logic and Reinforcement Learning*, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, 1999, Vol. 29, N.2, pp.314-321.

[230] P. Webb, C. Fayad, C. Breitenbach, *The Integration of an Optimised Fuzzy Logic Navigation Algorithm into a Semi Autonomous robot Control System*, The International Workshop on Recent Advances in Mobile Robots, Leicester, UK, June 29, 2000.

[231] C.Fayad, P. Webb, *Optimized Fuzzy Logic Based on Algorithm for a Mobile Robot Collision Avoidance in an Unknown Environment*, 7th European Congress on Intelligent Techniques & Soft Computing, Aachen, Germany, September 13-16, 1999.

[232] M. Sreekumar, T. Nagarajan, M. Singaperumal, *Design of Supervisory Adaptive Fuzzy Controllers for Intelligent Robotic Vehicles*, The 2nd Indian International Conference on Artificial Intelligence, Pune, India, December 20-22, 2005.

[233] S. Zein-Sabatto, A. Sekmen, P. Koseeyaporn, and S. Colombano, *Evolutionary Membership Adaptation for Mobile Robot Fuzzy Intelligent Behaviors*, The IASTED Conference on Control and Applications, Montreal, Canada, May 24-26, 2006.

[234] D. Gu, H. Hu, L. Spacek, *Learning Fuzzy Logic Controller for Reactive Robot Behaviours*, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Kobe, Japan, July 20-24, 2003.

[235] N.H.C. Yung, Y. Cang, *An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning*, IEEE Transactions on Systems, Man and Cybernetics, Part B, April, 1999, Vol.29, N.2, pp.314-321.

[236] L. P. Kaelbling, M. L. Littman, A. W. Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research, 1996, Vol.4, pp.237-285.

[237] R. S. Sutton, A. G. Barto, *Reinforcement Learning: in introduction*, (online book) MIT Press, 1998,

[238] C. Watkins, Dayan P., *Q-learning*, Machine Learning, 1992, Vol.8, pp.279-292.

[239] Sutton, R.S., McAllester, D., Singh, S., Mansour, Y. *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, Advances in Neural Information Processing Systems, 2000, Vol.12, pp.1057-1063.

[240] B. Hasegawa, *Continuous Observation Planning for Autonomous Exploration*, M.Eng. Thesis,Massachusetts Institute of Technology, 2004.

[241] Stentz, Anthony, *The Focussed D* Algorithm for Real-Time Replanning*, In Proceedings of the International Joint Conference on Artificial Intelligence pp.16521659,1995.

[242] D. Wooden, *Graph-based Path Planning for Mobile Robots*, PhD Thesis, Georgia Institute of Technology, 2006

[243] H. C. Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, Nature, 293(5828):133-135, September 1981.

[244] A. Shashua, M. Werman, *On the trilinear tensor of three perspective views and its underlying geometry*, In Proc. of the International Conference on Computer Vision, Boston MA, June 1995.

[245] R. Hartley, *Lines and points in three views: A unified approach*, In ARPA94, pp II:1009-1016, 1994.

[246] O. Faugeras, T. Papadopoulo, *A nonlinear method for estimating the projective geometry of 3 views*, In ICCV '98: Proceedings of the Sixth International Conference on Computer Vision, pp 477, Washington, DC, USA, 1998. IEEE Computer Society.

Abstract

In this Thesis we present a navigation solution for a mobile robot. The proposed system uses vision input to enable the robot to build a feature based map of its environment, localize efficiently itself without any artificial markers or other modification, detect and track the moving objects, and navigate without colliding with obstacles.

The Simultaneous Localization and Mapping (SLAM) process is tackled as a stochastic problem using Extended Kalman Filter (EKF). Our contribution consists in building a global map of the environment based on several local maps. The SIFT features are used in our implementation and their descriptors are used as a matching constraint. The 3D initialization of the features is based on the visual geometry theory.

To avoid using outlier features, a motion segmentation and estimation (MSE) process is used to detect the moving part of the scene. Subsequently, during map building, the detected features on the moving parts are excluded. The MSE process consists in camera motion estimation and compensation, scene cut or strong camera motion detection, background Gaussian Mixture Model (GMM) model update, and a Maximum a Posteriori Probability Markov Random Field (MAPMRF) framework to detect the moving objects in the scene and estimate their motion. Two methods for camera motion estimation and compensation are used, one uses the 2D projection of the 3D motion estimated in the SLAM process and the other method uses the dense motion analysis proposed by Dufaux and Konrad.

We considered also the case, where the robot is equipped with other localization sensors such as inertial navigation system (INS), wheel encoders, and a global positioning system (GPS). Two solutions are considered. The first consists in integrating the data from the INS and encoders data in the dynamic model of the vehicle to estimate its motion in the SLAM process and using the GPS data for geo-localizing the robot and the built map. The second solution consists in using the data from the INS and encoders in a Kalman filter to correct the GPS data, the estimated linear and angular velocities by this filter are used as prediction in the SLAM filter and the output of this one are used to update the dynamics estimation in the integration filter. This solution will increase the accuracy and the robustness of the positioning during the outage of the GPS system and allows a SLAM in less featured environments.

The estimated map is used in a path planning process to generate a list of waypoints allowing the robot to reach the user defined goals. The robot uses then a navigation process to follow the planned path and avoiding the unplanned obstacles or moving objects in the scene. For this procedure, added to vision, infrared and ultrasound sensors are used for obstacles detection.

The navigation procedure is based on two fuzzy logic controllers: a goal seeking controller and an obstacle avoidance controller. The goal seeking controller tries to find the path to the intermediate waypoints, while the obstacle avoidance controller has for mission to avoid obstacles. The 3D position of the moving obstacles is estimated using the epipolar geometry applied to the detected features on the moving objects. A command fusion scheme based on a conditioned activation for each controller arbitrates between the two behaviors. The reinforcement learning algorithm is used to adapt the obstacle avoidance fuzzy controller.

Keywords: Mobile robots, Adaptive learning, Motion Segmentation, Motion Estimation, Simultaneous Localization and Mapping, Fuzzy Logic, Reinforcement Learning

Résumé

Cette thèse introduit une approche pour la navigation de robot mobile en utilisant la vision par mono camera pour la construction d'une carte de l'environnement, la localisation du robot dans la carte construite, et la navigation sécurisée du robot dans son environnement. La localisation et la cartographie de l'environnement sont traitées simultanément par le processus SLAM sous forme d'un problème stochastique en utilisant le filtre de Kalman étendu. Notre contribution consiste dans la construction d'une carte globale de l'environnement base sur plusieurs cartes locales pour simplifier la complexité du problème. Les points caractéristiques utilisés pour modéliser l'environnement sont de type SIFT et leur profondeur est estimée par la théorie de la géométrie visuelle.

Pour éviter d'utiliser les points caractéristiques associés aux objets mobiles dans le modèle de l'environnement nous utilisons un processus de segmentation et d'estimation de mouvement dans les séquences vidéo. Ce processus estime le mouvement de la camera pour l'´eliminer, détecte les coupures de scène ou grand mouvement dans la séquence vidéo, met jour le modèle GMM représentant le fond de la scène, et détecte les objets en mouvement et estime leur mouvement par un estimateur du maximum posteriori du Champs aléatoires de Markov.

Le mouvement de la camera est estimé par deux approches: la première par la projection du mouvement 3D estimé par le processus du SLAM et la deuxième par la technique de l'analyse du mouvement dans le plan de l'image proposée par Dufaux et Konrad. Nous avons considéré aussi le cas où le robot est équipé avec d'autres capteurs de localisation tels que les systèmes inertiels INS, les encodeurs des roues, et les systèmes de positionnement global GPS. Deux solutions sont proposées. La première consiste intégrer les données de l'INS et des encodeurs dans le modèle dynamique du véhicule pour estimer son mouvement dans le processus du SLAM et utiliser les données du GPS pour géo-localiser le robot et la carte construite. Tandis que dans la deuxième approche les données de l'INS et des encodeurs sont utilisées dans un filtre de Kalman pour corriger les données du GPS et les vitesses linéaires et angulaires estimées par le filtre sont utilisées comme prédiction au filtre du SLAM. Cette dernière solution augmente la précision et la robustesse du positionnement durant l'absence du signal GPS et permet un SLAM dans des espaces moins texturés.

La carte construite de l'environnement est utilisée pour une planification globale des trajectoires libre. Ces derniers sont définis sous formes d'une liste de points but. Le robot utilise ensuite un processus de navigation pour suivre les chemins planifiés et éviter les obstacles non représenté dans la carte et les objets en mouvement. Pour cette tâche de navigation le robot utilise en plus de la vision par camera, des capteurs ultrason et infrarouge. Le processus de navigation est bas´e sur les contrôleurs en logique floue adaptés par l'apprentissage par renforcement.

Mots clés : Robots Mobiles, Apprentissage Adaptatif, Segmentation de mouvement, Estimation de Mouvement, Localisation et cartographie simultanées, Logique Floue, Apprentissage par renforcement.

ملخص :

في هته الرسالة نقدم حلا للملاحة للروبوت المتحرك. النظام المقترح يستخدم الرؤية بكاميرا أحادية لتمكين الروبوت من بناء خريطة لمحيطه, تحديد موقعه داخل الخريطة دون استعمال علامات اصطناعية أو أي تعديل آخر, كشف وتعقب الأجسام المتحركة, و التنقل بدون حوادث .

بناء الخرائط و التموقع (SLAM) يتم دراسته كتسلسل عشوائي باستعمال مرشح كالمان الموسع (Extended Kalman Filter) . الطريقة المقترحة تقوم على بناء خريطة المحيط مكونة من عدة خرائط محلية ممثلة بنقاط مميزة من نوع SIFT بثلاث أبعاد3D . البُعد الداخلي لنقاط SIFT يتم حسابه بطريقة الهندسة البصرية .

لتجنب استخدام المميزات المنتمية للأجسام المتحركة, نقترح طريقة لتجزئة الحركة و تقديرها. الطريقة المقترحة تستعمل نمودج GMM لتمثيل الخلفية (background) للموقع و MAP-MRF لتحديد الأجسام المتحركة.

الرسالة تأخذ أيضا بعين الاعتبار الروبوت المجهزة بأنظمة أخرى للتموقع مثلGPS, INS و encoders. في هته الحالة يمكن استعمال إحدى الطريقتين التاليتين: الأولى تقوم على دمج البيانات من INS و encoders في التمثيل الديناميكي للروبوت لتقييم حركته داخل عملية SLAM و استعمال معطيات GPS لتحديد الموقع العالمي (Geo localization) للروبوت و الخريطة المبنية. الطريقة الثانية تتمثل في استعمال المعطيات من الINS وeucoders داخل مرشح كالمان لتصحيح معطيات الGPS. السرعات الخطية و الدورانية المحسوبة بالمرشح تستعمل كتقدير داخل SLAM في حين تصحيح المرشح يتم عن طريق نتائج ال SLAM . هذه الطريقة تسمح بزيادة دقة و متانة التموقع خلال انقطاعات نظام الGPS و إمكانية استعمال ال SLAM في أماكن بعلامات مميزة قليلة .

الخريطة المبنية تستعمل لتمكين الروبوت من التحرك داخل محيطه للوصول إلى المواقع المطلوبة. لتجنب العطبات الجديدة في الموقع, الروبوت يستعمل, بالزيادة إلى الكاميرا, أجهزة الاستشعار Infrared وultrasound.

التحرك يتم بطريقة التحكم بالمنطق الغامض المكيف(Adaptive fuzzy logique) باستعمال Reinforcement leurning.

كلمات البحث

الروبوت المتحرك، التعلم بالتكييف، تجزئة الحركة، تقدير الحركة، بناء الخرائط و التموقع، المنطق الغامض، التعلم بالتعزيز