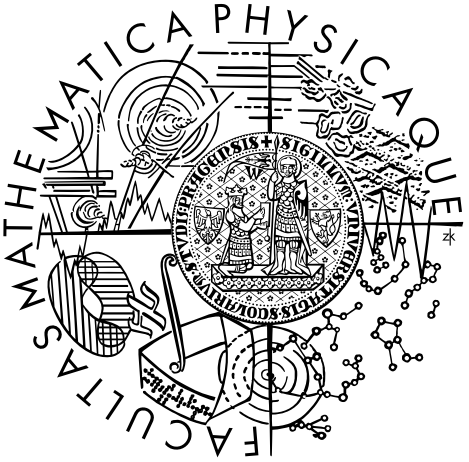
Charles University in Prague
Faculty of Mathematics and Physics

MASTER'S THESIS



Jan Benda

Navigation System for a Mobile Robot Based on Omni-directional Vision

Department of Software and Computer Science Education
Supervisor: Prof. Ing. Jan Flusser, DrSc.
Study Program: Computer Science

Acknowledgments:
I would like to thank my supervisor Jan Flusser for many helpful comments, especially in the initial phase, when the idea was formed. My acknowledgements also go to Zbyněk Winkler for the implementation of the MCL algorithm and to Ondřej Luks for constructing the test platform.

I declare that I wrote the master thesis on my own, using only the referenced sources. I approve lending of the thesis.

Prague, December 15^{th} 2005 Jan Benda

Contents

1 Introduction **8**
1.1 Preface 8

2 Problem Analysis **10**
2.1 Architectures 10
2.1.1 Behavioral 10
2.1.2 Deliberative 11
2.2 Sensors 13
2.2.1 Range-finders 13
2.2.2 Beacons 15
2.2.3 Vision 16
2.2.4 Omni-directional Vision 17
2.3 Approaches 18
2.3.1 Global Localization vs. Tracking 19
2.3.2 Geometrical vs. Topological Approach 20
2.3.3 Probabilistic vs. Analytical Approach 20
2.3.4 Feature-Based vs. Image-Based Evaluation 23
2.3.5 Natural and Artificial Landmarks 24
2.4 Related Work 25
2.4.1 Appearance-Based Approach 25
2.4.2 Feature-Based Approach 29
2.4.3 Contest Environment 31
2.5 Proposed Solution 36
2.5.1 Target Environment 36
2.5.2 Task Formulation 37
2.5.3 Selected Approach 38

3 Theory **40**
3.1 Omni-directional Vision 40
3.2 Color perception 42
3.3 Monte Carlo Localization 44

3.3.1 Prediction 44
3.3.2 Update 45

4 Implementation **47**
4.1 Filters and Chains 47
4.2 Pre-processing 48
4.2.1 Input Enhancement 48
4.2.2 Color Classification 49
4.2.3 Color Display 50
4.2.4 Image Rectification 50
4.3 Prediction 51
4.4 Correction 53

5 Experiments **56**
5.1 Simulated Environment 56
5.2 Target Environment 58

6 Conclusion **63**
6.1 Results 63
6.2 Future Research 64

Appendices **65**

A Playing Field Plan **65**

B Target Robotic Platform **67**

C Attachment CD Contents **69**

D Software Documentation **71**
D.1 Vision Framework Usage 71
D.2 Description of Vision Framework Files 73

List of Figures

2.1 Overview of the general control architectures 11
2.2 Overview of the sensor modalities used in mobile robot localization 13
2.3 The time-of-flight Swiss Ranger sensor 14
2.4 Omni-directional vision used as an affordable range-finder replacement 17
2.5 Classification of omni-vision based approaches 19
2.6 Power spectrum of the Fourier transform of cylindrical projection of an omni-directional image 26
2.7 Eigenvector images computed for a database of rotated images 27
2.8 Important edges in an omni-directional image and the respective distance transform 28
2.9 Minimum free space estimation 29
2.10 A light map of a museum ceiling 30
2.11 Feature-based matching against the image database 31
2.12 Bird's eye view of the RoboCup field 32
2.13 The process of estimating playing field position 33
2.14 Possible locations of a robot observing a relative bearing of two landmarks 34
2.15 Searching for geometrical features in an omni-directional image 34
2.16 Visual range-finder measurements from an omni-directional image 35
2.17 Playing field for the Eurobot^{open} 2005 contest 37
2.18 Overview of the vision-based localization process 38

3.1 Examples of Non-SVP and SVP projections 41
3.2 Schematic for the ground plane projection and rectification . . 42
3.3 Representation of landmark color classes in the `YCrCb` space . 43

4.1 `NormYUVFilter` overview 48
4.2 `ThreshBoundFilter` overview 49
4.3 `ColorFilter` overview 50

4.4 `ProjectFilter` overview 51
4.5 `ProjectFilter` overview 52
4.6 `SumFilter` overview 52
4.7 `VisualMclFilter` overview 54
4.8 Likelihood plot for different robot positions given the actual observations 55

5.1 Simulated tracking experiment 57
5.2 Average displacement error in the simulation experiment with respect to the particle count 57
5.3 Real-world tracking experiment 59
5.4 The trajectories estimated using the vision-based method rendered for the eight test video streams 59
5.5 Influence of particle count on accuracy of rotation tracking . . 60
5.6 Influence of particle count on the probability of successful pose tracking 61

A.1 The plan of the Eurobot^{open} 2005 contest playing field 66

B.1 The test robotic platform with the omni-directional sensor mounted on top 68

Název: Navigační systém pro mobilního robota založený na všesměrovém obrazu
Autor: Jan Benda
Katedra: Kabinet software a výuky informatiky
Vedoucí diplomové práce: Prof. Ing. Jan Flusser, DrSc.
e-mail vedoucího: flusser@utia.cas.cz
Abstrakt:

Diplomová práce se zabývá problémem lokalizace mobilního robota v dynamickém prostředí robotické soutěže. Představuje metodu založenou na částicových filtrech, která využívá jako jediný senzor všesměrovou kameru. Velkou výhodou tohoto přístupu je jeho snadná přenositelnost, protože tato metoda lokalizace nezávisí na žádném jiném systému mobilního robota. Také je velmi robustní vůči externím vlivům, např. kolizím s jinými roboty. Implementovaný algoritmus využívá rychlého barvného prahování, tabulkového převodu souřadnic, vizuální odometrie a Monte Carlo Lokalizace a zajišťuje robustní a spolehlivou lokalizaci v rámci soutěžního hřiště. Protože navržená metoda používá všesměrový obraz jak pro odhad pohybu, tak i pro zpřesnění průběžného odhadu polohy, snadno se vypořádá s nečekanými pohyby robota způsobenými vnějším zásahem.

Klíčová slova: mobilní robot, lokalizace, sledování polohy, všesměrové vidění

Title: Navigation System for a Mobile Robot Based on Omni-directional Vision
Author: Jan Benda
Department: Department of Software and Computer Science Education
Supervisor: Prof. Ing. Jan Flusser, DrSc.
Supervisor's e-mail address: flusser@utia.cas.cz
Abstract:

The thesis addresses the vast problem of mobile robot localization in the dynamic environment of a robotic contest. Method based on particle filters is developed, using only the image of a catadioptric visual sensor as its input. The advantage of this approach is an easy portability thanks to independence on other systems of the mobile robot and robustness to external influences such as robot collisions. This new method employs the composition of fast color thresholding, look-up coordinate transformation, vision-based motion prediction and Monte Carlo Localization to gain robust and reliable pose tracking using a color map of a delimited environment. Since the method uses visual data both to determine the relative motion and to verify the current location, it can cope with an unexpected events such as wheel slippage or collision.

Keywords: mobile robot, localization, real-time tracking, omni-directional vision

Chapter 1

Introduction

Mobile robot localization is one of the most important and challenging tasks in the domain of mobile robotics. A mobile robot performing any kind of operation on a specific place must be able to navigate in the environment and to identify the goal location with an accuracy required by the task. Whatever is the robot's mission, reliable position qualification either in geometrical or topological terms is a necessary condition. All the operations required for the robot's motion, such as path planning, path following and known obstacle avoidance, depend on the knowledge of the robot's location in the environment.

1.1 Preface

This thesis addresses the problem of geometrical localization for a mobile robot with three degrees of freedom, namely two for position on the ground plane and one for orientation. The task is to navigate a robot in a robotic competition, where it can benefit of a highly structured environment with well distinguishable visual navigation marks, but also faces a large amount of unexpected influences including numerous partial view occlusions and unexpected motion due to robot interaction. Fast and robust self-localization is crucial when a robot has to be able to execute its tasks effectively.

The localization problem in general can be described as determining and tracking the location of a mobile robot with respect to some global representation of space. This can be expressed either in qualitative or quantitative terms, by either identifying a previously visited position or giving a numerical representation of the robot state, respectively. For a known delimited environment of a robotic contest, where an *a priori* geometrical description is available, the latter is a more suitable option.

The operation field of the robot is limited and known — it is the playing

field of the Eurobot^{open} 2005 robotic contest. Although the prototype solution targets this specific environment, the proposed approach and the developed algorithms can be reused in different contest environments.

The presented work is divided into four chapters. Chapter 2 presents and further specifies the given task seeking for an optimal strategy. Several possibilities of non-visual and visual methods applicable for the target environment are discussed highlighting probabilistic approaches. The choice of omni-directional imaging is substantiated, examples of known applications are given, and a customized solution is presented. In Chapter 3, a theoretical background for omni-directional image formation, color perception and particle filtering is given. Chapter 4 describes the prototype implementation in detail. The experiments in Chapter 5 display the performance of the selected approach and describe the limits of the method and the problems open for future research.

Chapter 2

Problem Analysis

Ongoing research on mobile robotics provides numerous approaches to the localization problem. The selection of an appropriate method is a key task on the way to successful implementation. The following subsections will discuss the derivation of the proposed solution.

2.1 Architectures

Modern robotic applications can be roughly classified by the top-level organization of the control architecture as *behavioral* or *deliberative*. Every robotic problem decomposition starts with the selection of an appropriate overall design, which is chosen with respect to the complexity of the application. The architecture design is also tightly coupled with the conception of the *world model*, the internal representation of the operating environment. In this section, the architecture of the control system is discussed (see fig. 2.1).

2.1.1 Behavioral

The behavioral approach focuses on the actual interaction of the robot and the environment. Internal reasoning about the world is suppressed and the task is decomposed into required responses to external stimulations with associated priority levels. This method assumes that the world itself is the best possible model, thus resigns on any modelling and gives all the responsibility to the actual sensors. The obvious disadvantage of such approach is the necessity to possess sensors able to discriminate every considered event and the impossibility of any long-term planning. The basic *reactive* concept is stateless and is usually applied on very simple tasks, such as e.g. wall following. For tasks where appropriate mapping between sensor input and desired behavior exists, the optimal mapping may be effectively realized using methods

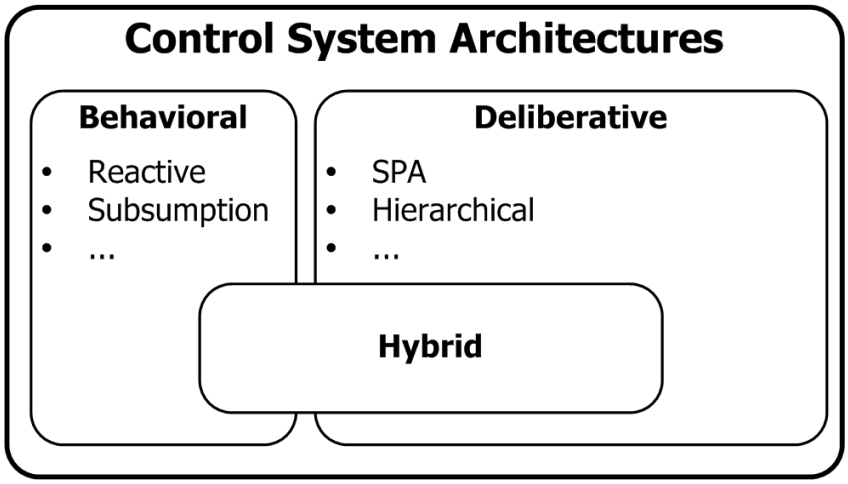


Figure 2.1: Overview of the general control architectures

of artificial intelligence such as *artificial neural networks* (ANN) and genetic optimization [11]. The mapping function may also be realized in the closed loop of a *continuous control* in the classical sense of control systems.

When a more complicated response is required, the *subsumption* [7] architecture may take place. This method uses a set of behavioral models arranged in a hierarchy of *levels of competence*, each of which directly maps sensations to actions. Lower levels have implicit priority over the higher ones, which enables e.g. the collision avoidance model to take control when appropriate. The incremental nature of this design makes it straightforward to implement. However, debugging such architecture may be problematic, as the final emergent behavior of subsumption architecture is the set of reactions that emerge from the various models triggered by the real-time conditions that control the individual behaviors.

Behavioral models may be advantageous in simple tasks where a sensing–response mapping exists and the advantage of evolutionary (ANN) approach may be taken; or when real-time response (such as using a closed loop controller) is required. These controllers may either implement the entire control system of a primitive application, e.g. line follower robots, or take part as low level modules in a *hierarchical system* (see below), e.g. to keep the robot on a planned path or avoid unexpected obstacles.

2.1.2 Deliberative

The deliberative approach is typically described in terms of *functional (horizontal) decomposition*. The world is processed and represented using a discrete set of actions, times, and events. The system consists of individual

modules operating in a deterministic serial fashion. In this architecture, commonly known as *Sense-Plan-Act* (SPA), the robot periodically executes a series of `'percept-model-plan-execute'` modules with the output of one module being input of the following. Although it is hard to reach real-time response of such system, functional examples exist [34].

To be able to both perform in real time and allow for complex model manipulation and reasoning, the system must be designed as *hierarchical* [11]. A typical design would include a persistent world model, which is periodically updated with the sensed data and used for reasoning. The individual modules in the hierarchical model are all required to have real-time response, i.e. to finish every execution cycle in a fixed period of time. In contrast with the classical SPA, these modules typically run concurrently, which allows different update rates of individual modules. This approach is well suited for vision-based navigation and reasoning that typically requires significant time for processing one captured image. Although this time may be kept constant, it will typically be much longer than the execution time of e.g. a low-level motion controller.

In a strict hierarchical system, the high-level layers such as mission planning and operational activity monitoring have full control over the layers that provide low-level control of the robot's sensors and actuators. This also defines the mechanism for propagation of information and control sequences. In a *blackboard* system [11], the communication between modules is provided by a neutral common information pool, shared by the individual computational processes. Fundamental to any blackboard-based system is a mechanism to provide efficient communication between the various computational agents and the blackboard. A mechanism that proves elegant, effective, and scalable is the *publish-subscribe* model. In this scenario, the actual world model variables reside in the individual agents providing the remaining modules with access to the contained variables using a common communication system. The common pool thus serves only for pairing the publishers containing the variables with the subscribers requesting access to the data. The key task is to distribute the world model so that inter-module communication is minimized.

Note that the global conception of the target control architecture falls beyond the scope of this thesis and is not described here. The thesis focuses on one fundamental part of the complex system, which is the localization module. The other essential components, such as the motion control and planning algorithms are not covered. The modular approach enables to design the individual components in an isolated manner. The localization module processes the output of sensoric subsystems and produces the position information via a defined interface.

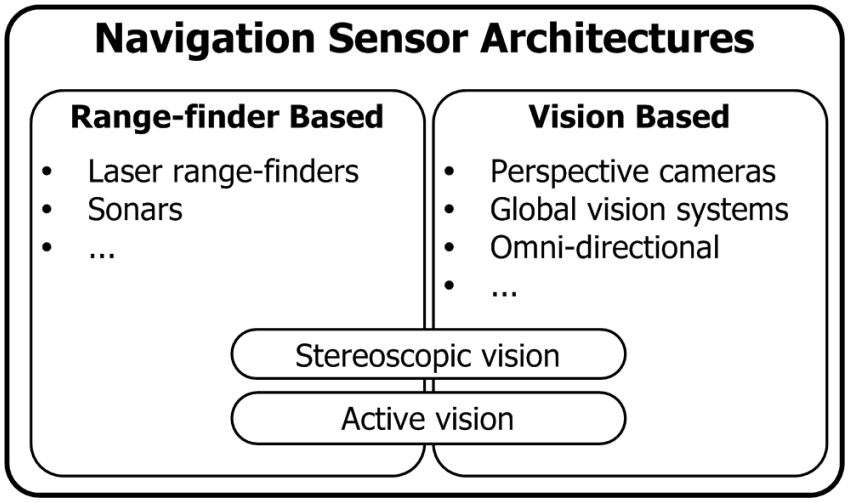


Figure 2.2: Overview of the sensor modalities used in mobile robot localization

2.2 Sensors

Sensing is a key prerequisite for the robot to be able to reason about the environment. All factors and events the robot is required to consider must be detectable by the robot's sensoric equipment. Apart from internal sensors such as odometers, accelerometers, inclinometers or gyroscopes, providing the robot with information about relative motion with respect to the environment, and tactile sensors providing the information about the nearest vicinity, long-range sensors are required to identify the surrounding environment, plan actions and even predict the development of the environment and the relative robot state (see fig. 2.2).

2.2.1 Range-finders

Reasoning about the distant objects in the environment is typically based on sensing the energy emitted or reflected by the object surfaces. A wide class of *active sensors* is based on the principle of emitting energy into the environment and measuring the properties of the signal reflected back to the robot. The simplest application is to compare the quantity of the reflected signal in order to estimate distance to objects. In practical applications, the emitted signal is further modulated to enable discrimination of the reflected signal from the background noise. Such devices, usually implemented using infrared light or ultrasonic sound, suffer from the fact that different surfaces have different reflective characteristics and thus the distance can not be determined precisely. Moreover, mainly for the sonic sensors, the speed

Figure 2.3: The time-of-flight Swiss Ranger sensor that is able to provide real-time depth maps of the environment, reprinted from [49]

of distribution of the signal through the environment depends on environmental properties such as temperature and humidity. These simple sensors are thus often used as proximity detectors, kind of early-warning bumpers. More advanced approach considers additional features of the reflected signal, such as reflection angle for triangulation [4] or time-of-flight [11].

Using electromagnetic waves in conjunction with time-of-flight or phase-based measurements is a very robust and reliable technique. Both methods use the time delay of signal propagation. This is especially advantageous considering the high and relatively invariable speed of light. On the other hand, it poses significant requirements on signal processing, making such devices relatively expensive. Various applications exist, exploiting the characteristics of specific wavelengths, from long-range radars to high precision *laser range-finders* (LRF). These devices are usually able to take very frequent distance measurements along a defined ray. Therefore it is possible to extend the measurement to multiple directions by reflecting the measurement ray [1]. As mobile robots usually operate on a flat surface, one range-finder scanning a single horizontal plane is sufficient for navigation tasks. Some obstacles, however, such as furniture or steps require scanning in additional planes. More range-finders may be added or a single device may be pointed to different directions to get better coverage of the space in front of the robot. Advanced sensors providing direct real-time 3D imaging also exist (see fig. 2.3).

For modern robotic applications, the laser or sonar range-finders are very popular, although these sensors have a number of problems. They tend to be easily confused in highly dynamic environments, e.g. when the laser or sonar beams are blocked by people or other robots. The operation area must also be populated with objects detectable by the range-finders. The localization process using range-finder measurements requires that the operation field is surrounded by walls. Unfortunately, this is not the case for many robotic contests, as the operation fields for the contests are flat and may be placed

in an arbitrary indoor area. Localization in such environments using LRFs is feasible, but is far from being simple and inexpensive [20].

Additionally, feature estimates obtained from rangefinder measurements are generic, without any uniquely defining characteristics. This poses a difficult problem for solving the data association problem between the current measurements and the map, where a single feature observation might require comparison with the entire map. This is an important drawback comparing to vision-based sensors, which usually provide a great deal of contextual information that can constrain data association, and reduce the cost of feature matching [44].

2.2.2 Beacons

A tempting approach is to use a navigation system based on bearing or distance measurements towards a set of *beacons* — active devices whose location is known. A popular example is the *Global Positioning System* (GPS). A GPS receiver can be localized with the accuracy of tens of meters anywhere on the globe using 21 Earth-orbiting satellites. The accuracy may be further increased using *Differential GPS* (DGPS) systems, where a reference GPS receiver with known coordinates transmits the current relative error of the GPS system, although the gain in accuracy is negligible after the *selective availability* encryption for civilian receivers was removed [36]. A still more accurate form of DGPS called *Differential Carrier GPS* exists. This system uses the difference in carrier cycle measurements between the mobile and the reference receivers and can achieve accuracy of centimeters. Naturally, Differential Carrier GPS is substantially more costly than basic DGPS [11]. One of the great advantages of the GPS system is that is does not involve any other observation of the external environment. It is also a purely passive system that involves no energy output by the receiver, which makes it ideal for military applications. Unfortunately, for majority of robotic applications, the tradeoff between price and accuracy is unreachable. Moreover, for indoor mobile robots reliable GPS signal coverage is often missing.

Another possibility is to create a special beacon system customized for the indoor navigation task. It would even be applicable to a contest environment where several active or passive landmarks may be installed in known positions. Many possibilities exist, e.g. triangulation using time-of-flight measured distances of beacons emitting ultrasonic pulses or using relative bearing measurements towards light-reflecting landmarks. Unfortunately, such system would require expensive custom electro-mechanical design and manufacturing, which makes it unavailable for the considered application.

2.2.3 Vision

As described so far, the entire class of active sensors is not appropriate for the target environment. Rejecting the active emission of energy by the robot itself as well as by external beacons, one important class of sensors remains — the visual perception. Providing the robot with the ability to "see", i.e. process and interpret 2D images of the environment acquired by a camera, has many advantages as well as drawbacks. A positive aspect is that human-populated environments are vision-friendly as vision is the dominant sense used by people. Even in unmodified environments, the important features relevant to navigation, such as steps or door-posts, are made visually discriminable.

The major drawback and most significant challenge of a general computer vision task is the ambiguity of inverse mapping from the 2D perspective projection back into the 3D space. The most general approach, inspired by the human *stereoscopic vision*, employs two or more cameras and the relative shift of correspondent features to determine their distance from the observer. *Active vision* methods combining the active and passive approaches are sometimes used in scene reconstruction — a known pattern projected onto the observed scene helps to disambiguate the inverse perspective projection. For example the *light striping* technique, which uses a laser beam to highlight a plane, whose intersection with the surrounding objects is triangulated to obtain the depth estimation [11]. To avoid the computationally demanding scene reconstruction, constraints may be applied on the locations of the observed features, e.g. positioning them all on a single plane. The perspective projection of this subset of features then becomes a bijection. Concerning a contest environment, the distinctive markers on the floor represent exactly this class of features.

An important factor that led to the selection of visual approach for the given environment is that the operation field contains well structured landmarks discriminable by color. Moreover, due to their small vertical diversity, a minimal error is introduced assuming their location is on the ground-plane. Based on this assumption, the environment can be represented using a 2D map of ground color and the scene reconstruction may be simplified to a one-to-one function. Camera is also a perfect sensor for tracking proximate color landmarks as it provides high amount of data for a relatively low cost. On the other hand, a computationally efficient approach to the extraction of landmark information is difficult to implement. Appropriate choice of the visual sensor configuration may help to ease the task.

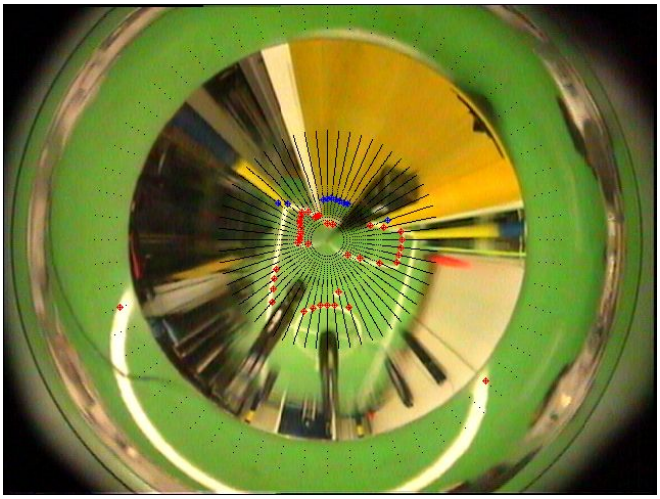


Figure 2.4: Omni-directional vision can be used as an affordable range-finder replacement. Here, distances to different visual landmarks determined by color transition are measured, reprinted from [31]

2.2.4 Omni-directional Vision

The feature that makes traditional perspective cameras hardly applicable to robot localization is the limited field of view and the emerging *viewpoint* and *occlusion* problems. A single camera may be occluded by an obstacle or the robot may be facing a direction where no appropriate landmarks are observable [46]. When the operation field is small, an overhead camera may be mounted monitoring the entire area, such as for example in the RoboCup Small Size League competition. Unfortunately, no suitable location for a global vision camera exists in the target environment.

To address the field-of-view problem, multiple-camera solutions, active panning, or fish-eye lenses have been proposed. An alternative is to use specially designed mirrors to bring a larger portion of the environment into the field of view of the camera (see fig. 2.4). Such configurations are called *catadioptric* as they employ both reflection and refraction in the process of image formation. The camera is typically mounted in the upwards direction with the mirror above it providing the robot with a 360° view of the environment [11]. Panoramic or omni-directional cameras have become popular for self-localization in recent years because of their relatively low cost and large field of view, which makes it possible to define features that are invariant to the robot's orientation [3].

One of the objectionable features of panoramic sensors is the relatively low spatial resolution. For applications that require both wide field of view and

high resolution, a high-resolution camera [9] or multi-camera solution [23] may be applied. However, for the majority of applications the resolution of a general camera is sufficient, besides the fact that a higher amount of data could be too complicated to process. Equipping the robot with an additional perspective camera providing detailed view of one particular area of interest [18] is a suitable compromise in many cases.

A customized mirror surface may also be designed and manufactured for a special application. An interesting design that uses the camera resolution effectively is a mirror with different scales for close and distant landmarks (see fig. 2.4). *Constant resolution* camera designs are also popular in many applications. It is a class of sensors that project linearly a measure in the world coordinates to a measure in the image ones. The most common constant resolution cameras designs are [17]:

- *horizontal* (CHR) — projects linearly the points on a plane perpendicular to the mirror axis,
- *vertical* (CVR) — projects linearly the points on a cylinder co-axial with the mirror surface, and
- *angular* (CAR) — projects linearly the points on a sphere centered at the camera's center of projection.

CVR cameras are sometimes referred to as *panoramic* and the CHR are called *bird eye's view* cameras. Although it is often possible to define mapping functions to reach desired image geometry, the *rectification* process always leads to a loss of information. On the other hand, custom mirror profiles are often hardly affordable. Section 3.1 presents additional information on the projection used in this thesis.

2.3 Approaches

There is a variety of possible approaches for localization using omnidirectional image. They can be roughly classified by the following aspects:

- the ability to globally localize the robot from scratch
- the internal representation of the current robot pose
- the method for maintaining the pose between individual measurements
- the nature of localization information extracted from the omnidirectional image

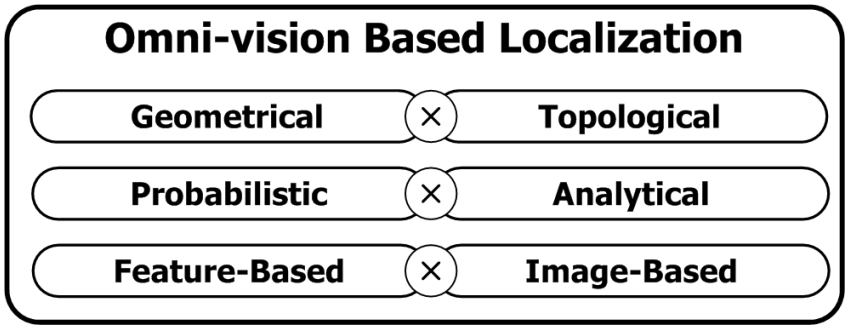


Figure 2.5: Classification of omni-vision based approaches

For each of these aspects, two representative classes of applications are selected and described (see fig. 2.5). In the real-world applications, combinations are often implemented to benefit from the positives of both sides. The following sub-sections aim at defining the classes and describing their advantages and drawbacks.

2.3.1 Global Localization vs. Tracking

When the robot is switched on without any prior information about its location, it faces the problem of a *global localization*. If it is provided with a map of the environment, it can make measurements and actions that lead to selection of the candidate locations and disambiguation between them. Once the location is known, either after a successful localization or from an external source, *tracking* may take place which benefits from the known kinematics of the robot and the proximity of the successive measurements. Pose tracking is a suitable approach for the target application as the initial location where the robot is started is known. An important precondition is that the localization process is robust enough to keep the correct location information all the time.

The task of global localization is often addressed in the *kidnapped robot problem* [48], as the problem of unknown location may reappear anytime during the pose tracking, especially when the robot is displaced artificially and in defiance of its kinematics. The pose tracking techniques are thus usually equipped with means to detect the complete loss of pose information and to enable re-localization. An alternative approach is to model the robot motion from the current observations, which eliminates the necessity to measure the ego-motion using dead reckoning and reduces the influence of external intervention [43] [6].

2.3.2 Geometrical vs. Topological Approach

There are two general principles to represent the robot's state within the world model. A continuous, *geometrical (metrical)*, approach describes the robot's location using numerical terms with respect to a selected coordinate system. The discrete one, *topological*, uses specific highlighted places to represent the positions and goals within the environment. This is better suited for applications involving human operators, as people preferably describe a goal as 'enter the second door on the left' rather than 'follow azimuth 244 for 7.29 meters, then azimuth 152 for 2.6 meters'. The metrical approach on the other hand is more suitable when the robot is required to follow a path within a specified margin [17].

In the topological approach, the world is described using a graph, where the nodes represent the known *reference locations*, and the edges correspond to the possible transitions between these nodes. The individual reference locations are recorded during the learning (mapping) phase and contain relevant information that can be used for the localization (such as metrical coordinates or semantic description of the place) together with the information that enables the robot to evaluate the similarity of current observations to the reference ones, and thus identify the closest node.

It is also important to note, that for majority of applications the topological navigation has to be combined with some kind of metrical navigation, as manual creation of a topological map is hardly feasible. Once the graph representation including metrical evaluation of the reference locations is recorded, a simple architecture (such as a reactive one, based only on dead reckoning) may be used to move between the individual nodes. A suitable scenario for the topological localization is an indoor office/hallway environment, which maps well to a graph representation [33].

For the target operation area, and generally for open environments where the robot can freely move in any direction and no locations with specific importance exist, the geometrical approach is more suitable. It is also useful when the robot has to follow precisely a given path, presented as a set of numerical coordinates. The selection of the reference coordinate system is straightforward for a limited known environment. For a rectangular operation field a Cartesian system aligned with the two axes of the field is often used.

2.3.3 Probabilistic vs. Analytical Approach

An ideal solution to the localization problem is to develop sensors able to *analytically* calculate the current position using only the most recent observations. This is achievable with e.g. beacon-based systems, where the

specific landmarks in the environment can be uniquely identified and precisely localized. In unmodified natural environments this is typically not the case. Even if the vision-based systems provide less ambiguity of the landmark observations compared to the rangefinder-based approaches, the problem of *perceptual aliasing* [33], i.e. existence of distinct but visually similar locations has to be considered.

In most cases, the uncertainty of the robot state and its development is modelled in the localization process. The restrictions based on the robot's kinematics help to eliminate the distant candidate locations, reduce the ambiguity, and focus the computation on the particular area of interest. The general specification of the probabilistic localization problem involves position estimates using dead-reckoning (for example with the robot's odometry) and iterative enhancement using environment map and sensor input. The system starts with an initial estimate of the robot's location in a configuration space (Euclidean space whose dimension is equal to the number of robot's degrees of freedom) given by a *probability density function* (PDF) of the robot's position *belief*. As the robot navigates through the environment, the PDF is updated using a probabilistic model of the robot's kinematics and sensors. The probabilistic model of the robot's motion is first applied to obtain a *predictive* PDF representing the possible development of the robot's state. Sensor data is then used in conjunction with map to produce a refined position estimate having increased probability density about the true position of the robot [11].

The existing probabilistic approaches differ in their internal representation of the belief and the associated update methods [10]. The *Kalman filter*, *Markov Localization* and the *particle filter-based methods* are some of the most popular approaches.

Kalman filtering [27] emerges when representing all the state and measurement densities by Gaussians. If all the motion and the measurement PDFs have normal distributions, and the initial state and measurement noise are also specified as Gaussians, then the density will remain Gaussian at all times. It can be shown analytically that the Kalman filter is the optimal solution to the probabilistic update rules under these assumptions [30]. The inherent problem in this approach is that only one pose hypothesis can be represented using a Gaussian density making the method in general unable to globally localize the robot or to recover from total localization failures [19].

This limitation has been addressed by two related families of algorithms: localization with multi-hypothesis Kalman filters and Markov localization. Multi-hypothesis Kalman filters represent beliefs using mixtures of Gaussians, which enables them to represent and track multiple, distinct hypotheses. However, to meet the Gaussian noise assumption inherent in the Kalman

filter, low-dimensional features have to be extracted from the sensor data, ignoring much of the information acquired by the robot's sensors.

Markov localization algorithms [14], in contrast, represent beliefs in a discretized manner over the space of all possible poses. There are different methods which can be roughly distinguished by the type of discretization used for the representation of the state space. In the *Topological Markov Localization*, the state space is organized according to the topological structure of the environment [45]. The coarse resolution of the state representation limits the accuracy of the position estimates. Topological approaches typically give only a rough sense as to where the robot is. In many applications, a more fine-grained position estimate is required, e.g. in environments with a simple topology but large open spaces, where accurate placement of the robot is needed.

To deal with multi-modal and non-Gaussian densities at a fine resolution, the *Grid-based Markov Localization* may take place [16]. A part of interest within the state space is discretized into a regular grid which is used as the basis for an approximation of the PDF by a piece-wise constant function. Methods that use this type of representation are powerful, but suffer from the disadvantages of computational overhead and a priori commitment to the size of the state space. In addition, the resolution and thereby also the precision at which they can represent the state has to be fixed beforehand [10]. Defining data structures enabling variable resolution representation of the state space, e.g. oct-trees, allow for effective utilization of memory and computation resources. However, the entire class of Markov-based approaches has been recently overcome by sampling-based systems.

The sampling-based representations are an obvious generalization of the Markov localization methods. Instead of discretizing the domain of the PDF into a fixed set of samples, it is sampled dynamically. The set of samples is drawn randomly from the PDF so that they concentrate in the regions with high likelihood. A duality between the samples and the represented density exists, which enables to apply existing update equations on the sampled PDF [47].

Particle-based representations have several key advantages over the foregoing approaches [15]:

1. In contrast to existing Kalman filtering based techniques, they are able to represent multi-modal distributions and thus can globally localize a robot.

2. They drastically reduce the amount of memory required compared to grid-based Markov localization and can integrate measurements at a considerably higher frequency.

3. They are more accurate than Markov localization with a fixed cell size, as the state represented in the samples is not discretized.

4. They are also much easier to implement.

Monte Carlo Localization (MCL) [10] with a constant number of samples is the probabilistic algorithm selected in this thesis. A fixed set of samples is initialized at the starting location of the robot and then undergoes a periodic sequence of prediction, correction and resampling phases. The entire algorithm is described in more detail in Section 3.3.

2.3.4 Feature-Based vs. Image-Based Evaluation

One popular method for omnivision-based localization uses a similarity measure defined directly over the space of original omni-directional images to evaluate the current observation. This *appearance-based* approach involves a database of images captured at the reference locations and searches for the most similar one. There are two important issues concerning this matching. Firstly, a compression is required to make the image comparison computationally effective. Secondly, rotational invariance should be involved into the image comparison, otherwise multiple rotations have to be recorded for the same reference location.

Various methods have been developed. The *principal component analysis* (PCA) addresses the first issue reducing the dimensionality of the problem. The omni-directional images are treated as `n`-dimensional vectors where `n` is the number of image pixels. An orthonormal basis of the reference image database is computed, which allows space-saving representation of every image of the training set using only a few parameters corresponding to the most discriminative vectors of the basis. PCA offers means for calculating an optimal basis using singular value decomposition (SVD) of the covariance matrix, representing the images in a low-dimensional subspace that is an optimal linear approximation of the original set in the least squares sense. The dimension of the feature subspace is usually set between 10 and 15 for a grayscale omni-directional image. Higher number of dimensions leads to better discriminability but increases the risk of over-learning. Another suitable approach employs a cylindrical projection of the omni-directional image. The advantage of this projection is in that rotation of the camera corresponds to a translation of the projected image — the azimuth of a scene point equals to the horizontal coordinate in the rectified image. Selected components from the Fourier transform of the projected image thus provide both image compression and rotational invariance.

Although the image-based approach looks temptingly feasible, straightforward and natural, there are several issues. One is the problem of perceptual aliasing, e.g. in long hallways with doors similar one to another. Other fundamental problem is the uneven influence of illumination changes and partial occlusions on the representation of the captured image. Finally, the creation of the database may be extremely tedious if performed manually or require additional metrical mapping or map geometry restoration technique if it is to be automatized.

Viable real-world applications extract some kind of features contained in the image of the environment. The discussion on the usable landmarks is given a respective section.

2.3.5 Natural and Artificial Landmarks

A *landmark* is generally any feature of the environment that can be repeatedly identified by the robot as being relevant. There is a broad range of landmarks considering both their artificiality and relevance for the localization task. Objects ranging from purpose-built beacons and reflective or contractive markings to edges or corners in a human-built environment or just distinct contours in a natural outdoor environment, all these can be used to determine relevant position information. Also the amount of obtained information differs from precise triangulation using uniquely identified beacons to relative motion information from tracking a distinctive feature among several successive images.

The definition of landmarks used by the specific implementation is typically imposed by the restrictions of the target environment. Special applications posing severe requirements on robustness and reliability usually allow arbitrary modification of the environment in order to simplify the application design. This class of tasks is considered to be solved and is typically addressed by engineers rather than robotics researchers. Current research highlights the ability to navigate in unmodified and unknown environments allowing the robots to extend their operation range beyond the robotic labs and workshops. Similarly, the robotic competitions tend to gradually complicate the navigation task by reducing the amount of artificial landmarks and limitations.

There are currently three principal domains for autonomous robot navigation: the exploration devices designed for operation in hardly accessible locations, the robots designed in the context of academic research and the robots involved in robotic competitions. All these domains involve a slightly different approach to the ultimate task of unrestricted autonomous operation based on the actual priorities.

It is the space program that engages the most robust and sophisticated applications, but various terrestrial missions as e.g. land mine elimination or rescue operations [29] also require reliable devices. It is obvious that no prior modification of the environment is applicable. The general localization in natural unmodified outdoor environment still remains a very hard problem and a large portion of human operator intervention stays involved.

An important class of research applications targets natural indoor environments. The scope of such projects is to develop fully autonomous devices able to navigate inside any building. Besides the fact that the considered motion may be restricted to three degrees of freedom (within a single floor), an unmodified indoor environment also contains a large number of interesting landmarks, such as e.g. windows, door posts and other edges, moreover typically rectangular. The landmarks most commonly used in the indoor environments are vertical edges, as they project aptly especially on omni-directional sensors, and the more general *SIFT features*, a class of image features that are invariant to image scale, rotation and translation as well as to illumination changes and affine or 3D projection.

Last but not the least, the domain of robotic competitions motivates the robotic science. The participants are presented rules, which are restrictive in a manner to encourage research and favor new solutions while leaving considerable space for creativity and improvements. The operation environment is pre-defined and typically contains multiple kinds of artificial landmarks. The focus is on real-time response and reasonable robustness in order to accomplish the given task. The robotic competitions are very important for the development of robotics as the concurrent evolution of the game rules and the game tactics pushes the horizons in a reasonable pace.

2.4 Related Work

Surprisingly little work exists on the omni-directional vision applied to the task of robot localization; although it can be seen that once this approach is visited by a researcher it is rarely abandoned. Mainly the principle of full sensoric coverage of the robot's surroundings is very advantageous.

The annotated works are grouped by the top-level approach, i.e. appearance or feature based. Much of the feature-based localization work is being done for the RoboCup Middle Size League. These applications are given a proper subsection.

2.4.1 Appearance-Based Approach

Back in the 1996, *Ishiguro and Tsuji* [25] were one of the first to examine

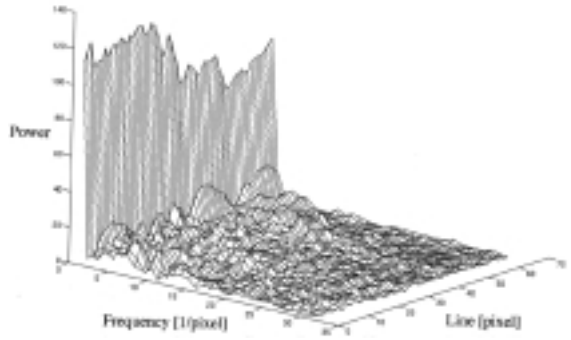


Figure 2.6: Power spectrum of the Fourier transform of cylindrical projection of an omni-directional image, reprinted from [25]

the potential of omnivision-based localization. They calculate the Fourier coefficients to represent cylindrical projections of omni-directional images in a lower-dimensional subspace (see fig. 2.6). The coefficients indicate the features of global views at the reference points and the phase components indicate the orientation of the views. However, they only address place recognition by indexing and do not try to generate an interpolated model. The representation used is not robust, since the Fourier transform is inherently a non-robust transformation. An occlusion in the image influences the frequency spectra in a non-predictable way and therefore arbitrarily changes the coefficients.

Aihara et al. [2] were probably the first to use the panoramic eigenspace approach using PCA to represent the image database. In order to avoid the problem of rotation of the sensor around the optical axis, they use row-autocorrelated transforms of cylindrical panoramic images. The approach suffers from less accurate results for images acquired on novel positions, since by autocorrelating the images some of the information is lost. Moreover, the process of autocorrelating the image is non-robust, meaning that any occlusion in the image may result in an erroneous localization.

Many authors came with their own propositions on the solution of the rotational invariance problem. *Pajdla and Hlaváč* [38] proposed to estimate a reference orientation from images alone with the *zero phase representation* (ZPR). ZPR, in contrast to autocorrelation, tends to preserve the original image content while at the same time achieving rotational independence, as it orients images by zeroing the phase of the first harmonic of the Fourier transform of the image. The experiments indicate that images taken at nearby positions tend to have the same reference orientation, which enables to interpolate between the relevant reference locations. The method is, however, sensitive to variations in the scene, since it operates only with a single fre-

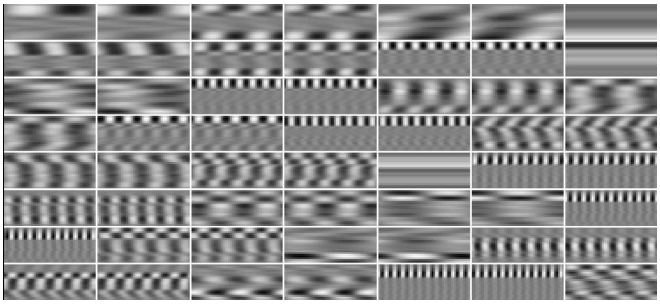


Figure 2.7: Eigenvector images computed for a database of rotated images, reprinted from [26]

quency using a global transform. Generally all methods that apply a global transform to the images result inherently into non-robust eigenspace representations.

An alternative approach is to include several possible rotations for each reference location into the database (see fig. 2.7) as presented by *Jogan and Leonardis* [26]. The system proves to be robust especially with respect to the amount of occlusion as expected. However, this method can not reliably deal with major changes in illumination. The results obtained with this method proved reliable up to 60% occlusion that yields to a mean localization error of approximately 60 cm, which is the resolution of the training set. The entire matching process requires several seconds to complete.

It is possible to consider the recent robot's state to lessen the computational demands of the matching process. *Menegatti et al.* [33] propose to use the Monte Carlo Localization to track the robot's position and reduce the amount of reference locations considered. This allows the system to reach real-time performance, while preserving the ability to globally localize the robot using a hierarchical search within the entire database [32]. The proposed solution spreads in each step 10% of the samples to the best matching locations (reference locations with least dissimilarity to the current observation). This amount is experimentally proved to enable the algorithm to quickly converge in case of kidnapping while not importantly drifting the probability distribution in a common case of perceptual aliasing.

The authors of [33] use so called *Fourier components* as the feature vector for storing and comparing the images. This signature is a subset of Fourier coefficients computed for individual rows of a cylindrical projection of a grayscale omni-directional image. The major advantage of the Fourier coefficients is that they naturally provide rotational invariance. Another benefit is the reduction of memory consumption, as this signature occupies only 0.25% of the space required by raw 640×480 24 bit images.

Figure 2.8: Important edges in an omni-directional image and the respective distance transform, reprinted from [17]

The obvious disadvantage of this method is the necessity to create the reference image database. The total of 500 reference images together with relative metrical information had to be taken in a corridor approximately 100 meters long. The major advantage is the possibility of quick matching against all possible locations to address the kidnapped robot problem effectively. This solution is best suited for environments with little color information, possible but not periodical perceptual aliasing, and relatively few possible reference locations, such as are e.g. the corridors of a large public building.

An efficient solution to the illumination problem is presented in [17]. Instead of treating the original omni-directional images, an edge-based representation is created using a distance transform (see fig. 2.8). A computationally effective method is derived using an eigenspace approximation of the Hausdorff distance [22]. However, the Hausdorff based matching is only applied in locations where illumination can change significantly. A method based on PCA is applied otherwise as it is more informative.

One of the most complex methods addressing the image-based similarity measure is given by *Paletta et al.* [39]. The authors robustified the panoramic eigenspace approach by applying Bayesian reasoning over local image appearances. They achieved robustness and handled rotation by dividing the panoramic image into overlapping vertical strips representing unidirectional camera views. Distributions of sector images in eigenspace are represented by mixture of Gaussians to provide a posterior distribution over potential locations. However, the local windowing introduces ambiguities that have later to be resolved through Bayesian probabilistic framework. This solution displays the limits of the image-based approach as it abandons the similarity measures based on appearance of the entire image. The sector-wise approach is evidently biased towards feature-based image treatment. It can be seen that for practical applications, specific features have to be addressed in the omni-directional image. For fully general solutions with no specific domain

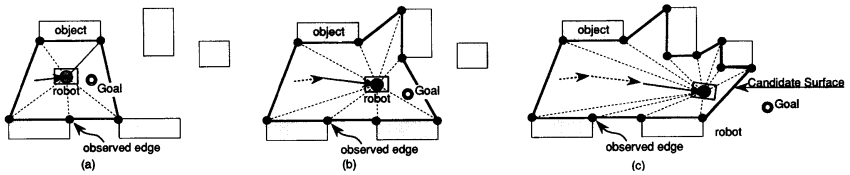


Figure 2.9: Minimum free space estimation, reprinted from [51]

of operation, generic invariant features should be taken into account instead of aspiring for appearance-based invariance.

2.4.2 Feature-Based Approach

Returning back to the 1995, one of the first practical experiments with omni-directional navigation already uses a feature based approach. In the work of *Yagi et al.* [51] a *Conic Projection Image Sensor* (COPIS) detects and tracks vertical edges in the environment. The sensor contains a conical mirror, which projects the vertical edges to radial lines in the image, and a TV camera, which sends the images wirelessly to a control computer. Because the COPIS sensor is omni-directional and the environment is constructed to be populated with vertical lines, a large number of azimuth readings are obtained surrounding the robot. The system is capable both of improving the rough location estimate using a map of known features and of detecting and recording a new feature. A simple sort algorithm is used to maintain the list of known edges. The locations of unknown obstacles are estimated by monitoring loci of azimuth angles of the vertical edges. The coordinates of new edges are corrected using triangulation from successive readings. A large error occurs in the front region of the robot when the robot moves straight ahead, but the error diminishes when the robot changes direction.

Known vertical edges are used to define the *minimum free space* (see fig. 2.9), a wire occupancy map of the environment created connecting the adjacent vertical edges and verifying the *candidate surfaces* using an acoustic sensor. An ultrasonic sensor is used if and only if the robot is required to pass through a candidate surface to verify if it is a real one. This is one of the major drawbacks of edge-based navigation — the configuration of object surfaces can only be estimated and need to be verified otherwise.

The image analysis was implemented on an image processor and was able to process about 1 fps including the time needed for communication. The speed of the robot was about 5 cm per second and the localization error in an artificial box-like test environment ranged from 3 to 7 cm.

In the work of *Dellaert et al.* from 1999 [10] the idea of *Monte Carlo Localization* (MCL) crystallized. This localization technique was presented as

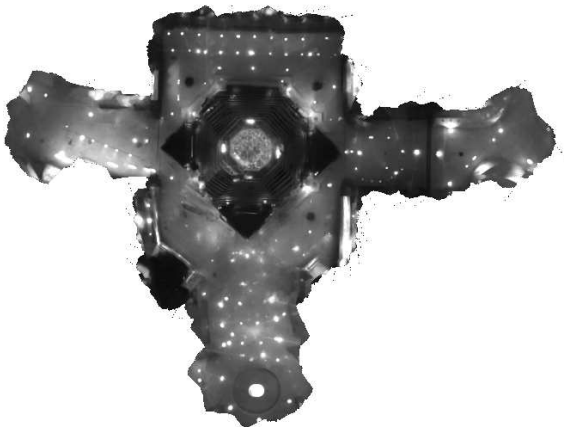


Figure 2.10: A light map of a museum ceiling used for vision-based update, reprinted from [10]

a novel application of a known Monte Carlo method originally developed for object tracking. The MCL applies the *conditional density propagation* [24] algorithm to the problem of tracking the motion of the entire camera platform. The motivation for a particle-based approach was the high ambiguity of the sensor used for the localization which yielded to a complicated measurement PDF disabling the traditional approaches.

The testing platform — a museum tour guide robot — used the odometry information to produce the predictive PDF. A light map of the ceiling (see fig. 2.10) was compared with the average brightness value of a small area in the middle of an upwards-mounted camera image. This little information provides surprisingly good results when combined with a robust probabilistic algorithm. One of the drawbacks of such simple sensor is that in case of global localization, the disambiguation had to be performed using random motion of the robot, during which the robot had no information about its location. Also, the precision of the method was limited because of high perceptual aliasing of the single-dimensional sensor. It is obvious, that much more information could be harvested from a camera image, which may help to overcome these limitations.

Andreasson et al. [3] presented a method that combines the advantageous properties of a particle filter, an omni-directional sensor and the SIFT features. The system is thus capable of matching generic features extracted from local interest points in the image (see fig. 2.11). This results into a universal and very robust system that can reliably navigate a mobile robot in a large, populated indoor environment, able to face kidnapping and occlusion of up to 90% of the robot's field of view.

The matching used in the application employs a *Modified Scale-Invariant*

Figure 2.11: Feature-based matching of an omni-directional image against the image database, reprinted from [3]

Feature detector (MSIFT), which selects landmarks that are invariant to image scale, rotation and translation as well as to illumination changes and affine or 3D projection. The map of the environment contains 100 strongest interest points recorded for each reference location, each of which is associated with a feature descriptor built from image gradient orientations in the neighborhood of the interest point. Only several closest reference locations are tested for match with every particle using the similarity of the features descriptors to match the interest points. The new weight of the particle is based on the number of interest points that match between the current image and the corresponding database image. To estimate the rotation of the robot a histogram of relative rotations of the matched interest points is used.

The mean localization error ranged from 0.5 m to 2 m in the experiments with the span of database images being 0.5 m. The principal drawback of this method is a relatively high computational cost as the feature extraction, matching and update costs about 2 seconds on a 2 GHz desktop computer which makes this approach currently inapplicable for real-time localization of an autonomous robot.

2.4.3 Contest Environment

The following works all address the environment of the RoboCup Middle Size League soccer field [13]. Although the approaches vary a lot, the omni-directional cameras already have a stable place in this competition. This is mainly due to the removal of fixed surrounding walls from the rules since 2001, which makes it harder to apply range-finder based methods. Moreover, the game elements are strongly color-coded, which favors the usage of vision-based systems.

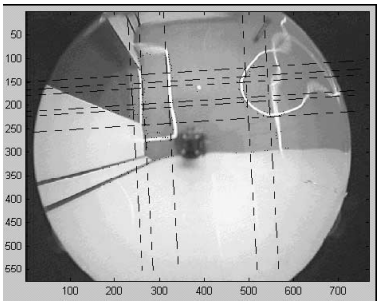


Figure 2.12: Bird's eye view of the RoboCup field with the estimated pairs of line segments highlighted, reprinted from [28]

Marques and Lima [28] propose to extract the geometrical features from a bird eye's perspective of the environment. They describe a general approach for tracking straight line segments on the floor and an application to analytical localization on a RoboCup soccer field. The method benefits from a mirror profile that realizes a constant horizontal resolution projection and thus preserves the geometry of the ground plane(see fig. 2.12).

The *transition pixels* denoting interesting features in the acquired image are processed using *Hough transform* (HT) [21] to detect straight linear patterns of the floor. A fixed count of most distinct straight lines is picked and all pairs made out of these lines are classified for relevance using *a priori* knowledge of the environment geometrics. This step is used to highlight pairs of parallel lines enclosing an expected gap. The common direction of the most relevant pair is used to pick the remaining parallel lines. These selected lines are then correlated with a corresponding set of model features to find the best match. The selection and correlation process is then repeated with the remaining non-parallel lines. Two sets of six and five parallel lines of a RoboCup soccer field were used in the practical application. Typical position errors ranged from 0 to 10 cm.

This approach is functional and robust mainly due to the integrative fashion of the HT. On the other hand, ignoring the continuity of the robot state throws away much of the information reusable in the next localization step. This may especially prove derogative in real-world environment with considerable distortion and occlusion influences. The analytical approach risks sudden fallouts, which could be avoided using an uncertainty-modelling approach.

Sekimori et al. [41] present an analytical approach to the localization problem on a RoboCup field based on a single tracked landmark — the green rectangular shape of the entire playing field floor. The method starts with image pre-processing that extracts a convex hull of the floor projection.

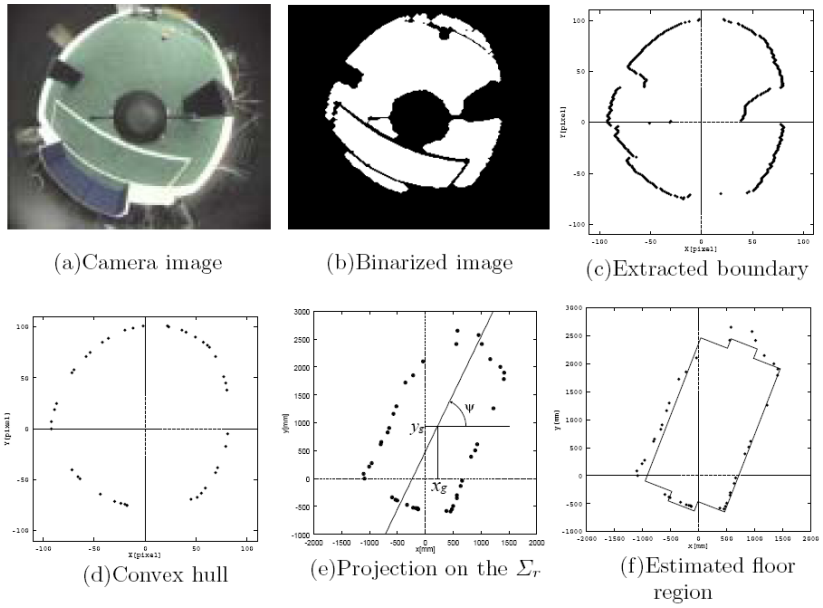


Figure 2.13: The process of estimating playing field position using the green floor segment, reprinted from [41]

This reduces the effect of occlusions on the floor shape. The floor outline is then projected back to the coordinate system of the ground plane and geometric moments of the shape are calculated to obtain a rough match with the field model. Least-squares optimization is then used for best fit with the model (see fig. 2.13). The rotation and shift of the observed landmark define the current robot's coordinates with the only ambiguity of field central symmetry. If the color of goal is observed it solves this ambiguity, otherwise the most recent solution is chosen.

Extending the initial idea using probabilistic refinement, a robust solution is achieved [42]. The global position information can be easily reformulated in a form suitable for the Kalman filter. The self position estimation is updated by dead reckoning using odometry and the belief is improved by the vision-based self-position observation described above. The Kalman filtering helps to improve the localization performance, which would otherwise suffer from quantization errors, occlusions and noise in the images.

The resulting method seems to be robust and reliable. The only shortcoming may be the necessity to always fit the entire field into the view of the omni-directional camera. This greatly reduces the resolution of the sensor but there are means to address this issue if required [31].

An interesting hybrid of the analytical and probabilistic approaches is presented in [35]. The method uses ambiguous landmark observations of

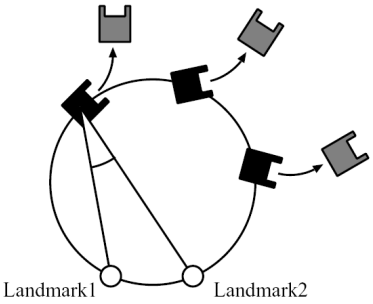


Figure 2.14: Possible locations of a robot observing a relative bearing of two landmarks, reprinted from [35]

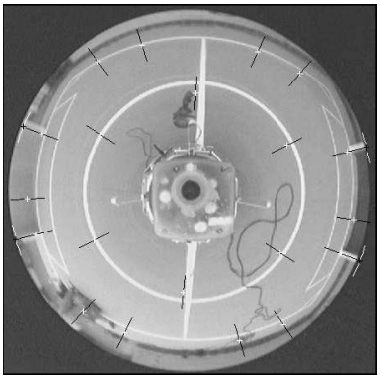


Figure 2.15: Searching for geometrical features in an omni-directional image, reprinted from [50]

the field corners to draw a sampled set of position estimates (see fig. 2.14), each of which is weighted with subsequent landmark observations. Instead of estimating the position using a definite triangulation from three or more landmarks, only two are selected. This results into a set of possible positions located on a circle determined by the relative bearing of the two landmarks and their order [11]. The circle is then sampled into a fixed count of position estimates each having an error value accumulator. This counter is zero at the beginning and increases over time as different landmarks are observed and the robot moves. The relative bearing of two landmarks and the direction angle to an individual landmark is compared with the predicted values for each sample. The difference between the observed and the predicted value is added to the error counter. The sample with the lowest accumulated observation error is used as the current position. When the total error exceeds a given threshold, a new pair of landmarks is selected. Several independent localization processes are executed to increase robustness.

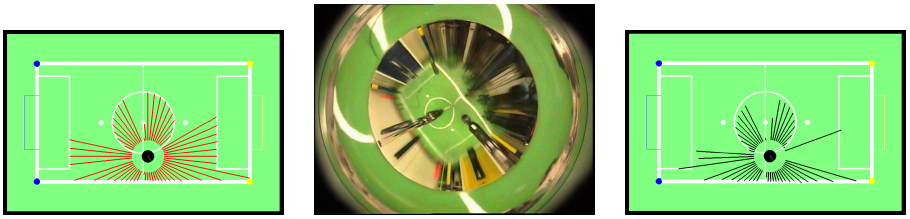


Figure 2.16: Visual range-finder measurements from an omni-directional image, reprinted from [31]

Same as in [28], the features selected by *Wolf and Pinz* [50] are the white lines drawn on the playing field floor. A particle filter is used to track the position estimate using the robot's odometry and a geometrical correction method based on omni-directional image of the floor. Several important locations selected *a priori* in the field model are transformed into the image coordinates for every sample and the omni-directional image is searched for color transitions corresponding to these landmarks. The difference between the expected coordinate and the true position of the edge is used for weighting the samples. The selected approach requires a geometrical description of the markings and a complicated evaluation of the match between the estimated pose and the current observations. On the other hand, the evaluation can be performed very fast. The major drawback of the selected method is the inability to deal with the kidnapped robot problem as the update algorithm only searches for expected landmarks in a restricted area thus failing to observe any landmarks at all if the position estimate error exceeds a threshold.

A very good application addressing the localization problem on the RoboCup field is presented by *Menegatti et al.* [31]. A custom mirror profile is used that performs two projections of the playing field at a different scale. The inner portion provides a bird's eye overview of the entire playing field, while the outer rim serves for precise low-distance measurements. The authors define a paradigm of visual range-finder: they extract the color transition information along several rays in the omni-directional image to detect for the distances to visual markers of the RoboCup field. This approach enables to combine the existing methods developed for range-finder based navigation with the rich information provided by the visual sensor.

Monte Carlo Localization is used to track the location of the robot. To address the kidnapped robot problem, random samples are inserted to enable re-localization. The system is shown to perform active disambiguation in 6 update steps corresponding to a distance of 4 meters traveled. This solution is very good but inapplicable to the task of this thesis for at least two reasons.

Firstly, the target environment has a periodic color pattern, which would make a visual range-finder very ambiguous. Secondly, such custom mirror profile is difficult to obtain.

2.5 Proposed Solution

Firstly, it has to be noted that a customized approach must be derived for a very specific target environment to make the solution effective in terms of development and computational costs. The applications noted above are all performing well in their own environments but are rarely general enough to be simply reproduced. On the other hand, selecting a too complex and generic approach would make the resulting application hardly competitive in the terms of processing speed.

The scope of this thesis is to design and implement a localization module for a mobile robot operating in the Eurobot^{open} 2005 contest environment [12]. The following sections first specify the available features of the given environment, followed by the presentation of the prototype solution and its contribution.

2.5.1 Target Environment

The operation field of the Eurobot^{open} 2005 contest is a rectangular area of approximately 2.1×3.6 meters consisting of two fields with a brown-beige 30×30 cm checkerboard pattern separated by a blue ditch and surrounded by a white border. The central ditch is 60 cm wide and is 3.6 cm below the level of the playing field floor. It is separated from the two floors by a glossy white line. The construction of the target robotic platform enables it only to cross the ditch using one of the two beige bridges, positioned randomly before the start of a match. The detailed description of the playing field can be found in the Appendix A.

Visual features usable for the localization include the checkerboard pattern, the blue ditch with the randomly positioned beige bridges and the white borders denoting the limits of the checkerboard fields and the ditch.

The localization task is complicated by the fact that the playing field is populated with additional elements that tend to confuse the visual sensor by occluding the expected landmarks and creating false ones. These are namely the game elements, such as are the 15 green and 15 red skittles and two black bowls. Moreover, the solution must expect very frequent influences from other robots operating on the playing field — physical robot interactions are very frequent in this contest.

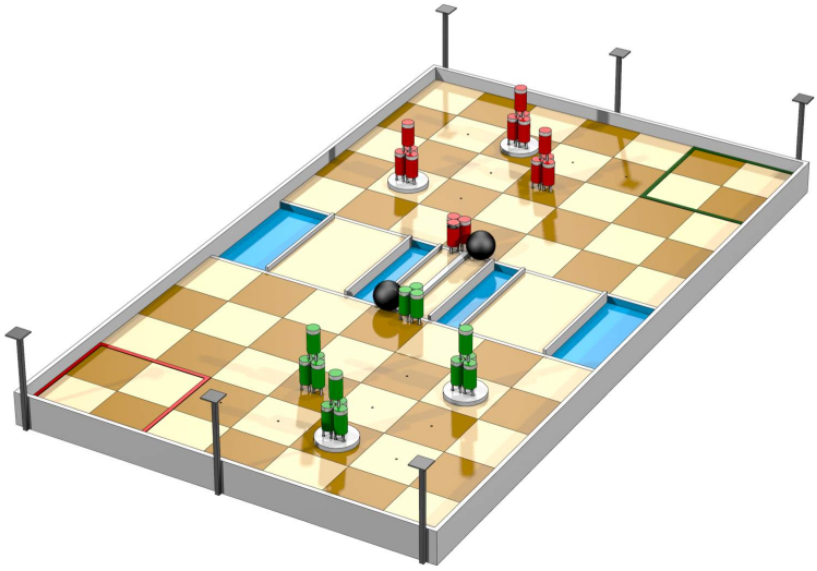


Figure 2.17: Playing field for the Eurobot^{open} 2005 contest

2.5.2 Task Formulation

The most common implementations of a probabilistic localization use the information from odometry to predict the current pose of the robot. However, there is a shortcoming in relying on the odometry data as the wheel slippage introduces an unmodeled noise into the system. Often, especially in robotic competitions, the robot moves in an unexpected direction because of collision with another moving object. Such motion, not considered in the prediction phase, results in the so called *kidnapped robot problem* [48]. Therefore a method will be proposed that uses the image data for both the motion estimation and the weight updates. Compared to a hardware odometry based approach this method can deal with any kind of robot's motion.

The aim of the thesis is to develop a localization module that would enable the target robotic platform to localize itself within the Eurobot^{open} 2005 contest environment using solely the information provided by a visual sensor. The only input to the algorithm is a video stream of an omni-directional camera with a hyperbolic mirror that provides a panoramic view of the contest environment. The task of the localization module is to track the position of the moving camera platform given the initial coordinates within the environment independently on the robot's actions and inputs.

The advantage of the resulting method is that it can be used on any other robotic platform with the only requirement that the same visual sensor

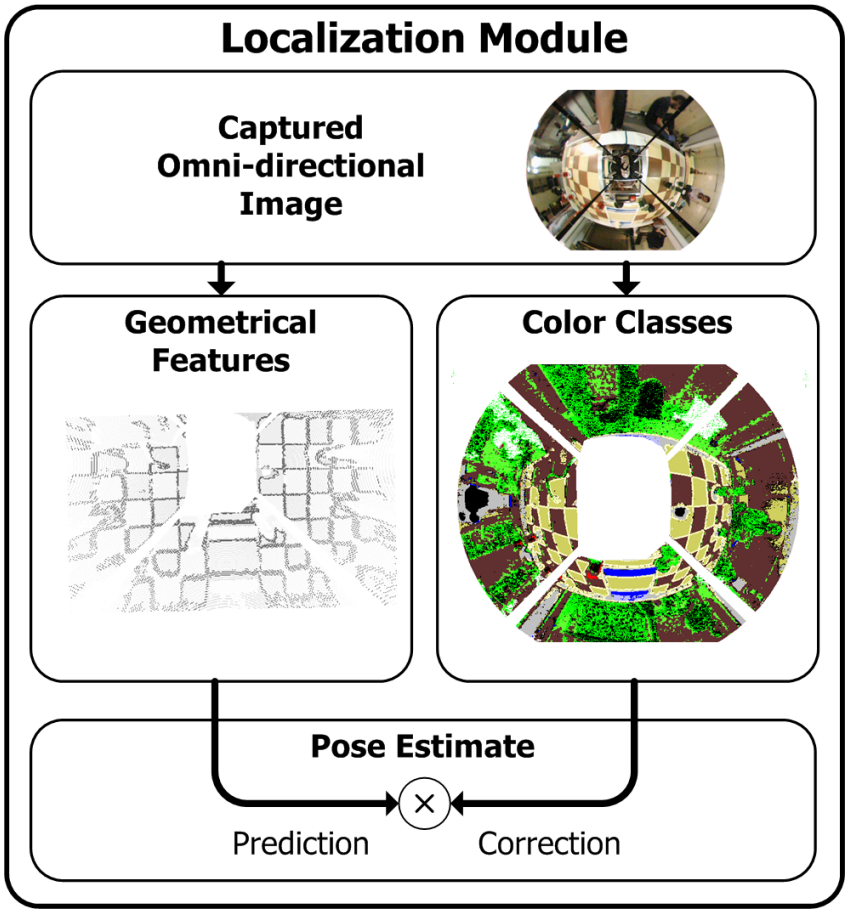


Figure 2.18: Overview of the vision-based localization process

is used. It is also possible to experiment with an image-based replacement of hardware odometry and to hopefully overcome some of its limitations. Moreover, only the recorded video stream is needed for development and off-line evaluation of the method.

2.5.3 Selected Approach

An overview of the selected method is given here (see fig. 2.18). The detailed description of the prototype implementation is given in Chapter 4 and the theoretical background for the implemented algorithms is described in Chapter 3.

A sampling-based probabilistic algorithm was chosen to track the current position of the camera platform. A *visual odometry module* predicts the motion of the camera and the precise localization is achieved using a *visual correction module* extracting landmark features from the omni-directional

image. A bird's eye view transform is applied on the acquired image to rectify the geometry of the rectangular playing field landmarks.

Instead of trying to evaluate the relative position of the entire playing field, which would be very hard due to the fact that the playing field surroundings are not defined, easily detectable features of the playing field are extracted and tracked to determine the relative motion of the robot. A probabilistic update is then applied to refine this position estimate.

The odometry module tracks the boundaries of the playing field landmarks. A relative position within one checkerboard square is estimated by detecting the orientation and shift of the rectangular grid formed by the individual checkerboard squares in the rectified image. This relative information is continually tracked to get the global position estimate. Because of the periodicity of the basic pattern, an ambiguity in the motion estimation has to be solved imposing an upper bound on the robot's speed.

To correct for position estimate errors induced by the image noise, occlusions, reflections, and especially distortions caused by the camera tilt, the raw accumulated position estimate is periodically corrected in the probabilistic framework using a map of the playing field colors and a sparse pixel-wise comparison with the color-classified omni-directional image.

This is an alternative approach opposing the most common modalities consisting of precise odometry and range-finder sensors. One of the motivations is to examine the possibilities of the visual odometry and to offer an affordable option to the costly sensors used in current robotic applications. The major benefit of the probabilistic approach is that additional sensors can be easily integrated into the existing framework to increase precision and robustness. This would enable e.g. to enhance the developed solution with hardware odometry if available on the target platform or to replace the visual odometry at all if no suitable landmarks exist within an alternative environment.

Chapter 3

Theory

This chapter presents a theoretical background for the omni-directional image formation, color classification and probabilistic localization algorithms employed in the prototype implementation.

3.1 Omni-directional Vision

The projection model used in this thesis consists of the *perspective camera projection* combined with a *curved reflexive surface*.

In the perspective projection model, the three-dimensional feature points are projecting onto an image plane with perspective rays originating at the *center of projection* (COP), which would lie within the physical camera. The origin of the coordinate system is taken to be the COP and the *focal length*, f is the distance from the COP to the image plane along the *optical axis*, which is aligned with the $\vec{z}$ axis.

The perspective projection of a point $(x, y, z)^T$ in the world coordinates to a point $(u, v)^T$ in the image coordinates is performed by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \frac{f}{z}$$

In a *catadioptric* configuration, the camera is combined with a curved mirror to increase the field-of-view of the sensor. The mirror profile is a rotationally symmetrical surface described by a *surface function* $z(x)$ — the surface is constructed by rotating the graph of $z(x)$ around the optical axis. Depending on the shape of the surface equation, the mirror profiles can be roughly classified as *single viewpoint* (SVP), e.g. hyperbolic, and *non-single viewpoint* (Non-SVP), e.g. spherical or conic (see fig. 3.1). The advantage of a SVP profile is that in certain camera configuration it reflects the rays passing the camera COP such that they all seem to originate in a single point

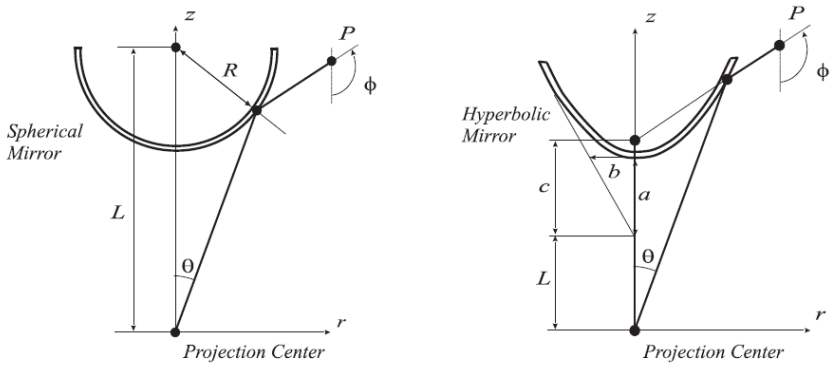


Figure 3.1: Examples of Non-SVP and SVP projections, reprinted from [17]

inside the mirror. For example a hyperbolic surface reflects a ray passing one of the foci of the hyperboloid onto a line passing the other focus. This feature of the hyperbolic mirror is used later in the image rectification. More on SVP profiles can be found in [5].

Since the geometrical markings used for the localization are all rectangular and the hyperbolic mirror profile used in the prototype application does not provide a bird's eye view projection, image rectification is required to recover the original shape of the landmarks. However, the single-viewpoint feature of the hyperbolic mirror and the alignment of the camera COP with one of the mirror's foci (the focus F in fig. 3.2) enable the derivation of the following equations.

The image rectification process recovers the landmark shape using a pixel-wise projection from the image plane coordinate system to the ground plane coordinate system. Since the catadioptric projection is rotationally symmetrical, the rectification can be redefined in the cylindrical coordinates aligned with the optical axis of the camera-mirror system. Putting the angular coordinate aside for a while, the rectification becomes a 2-dimensional problem: the ray passing through the camera COP is described by the angle ρ — the deviation from the optical axis.

The mirror profile is now given by a hyperbolic equation $z^2/a^2 - t^2/b^2 = 1$, which yields the surface function $z(t) = a\sqrt{1 + t^2/b^2}$. The coefficients a and b define the shape of the hyperbola and are assumed to be known. For a given radius ρ defining a point in the image coordinates, the task is to find the intersection point of the corresponding ray with the hyperbolic curve $[t, z(t)]$. This is done by solving for an unknown t the equation:

$$\frac{e + z(t)}{t} = \frac{f}{\rho}$$

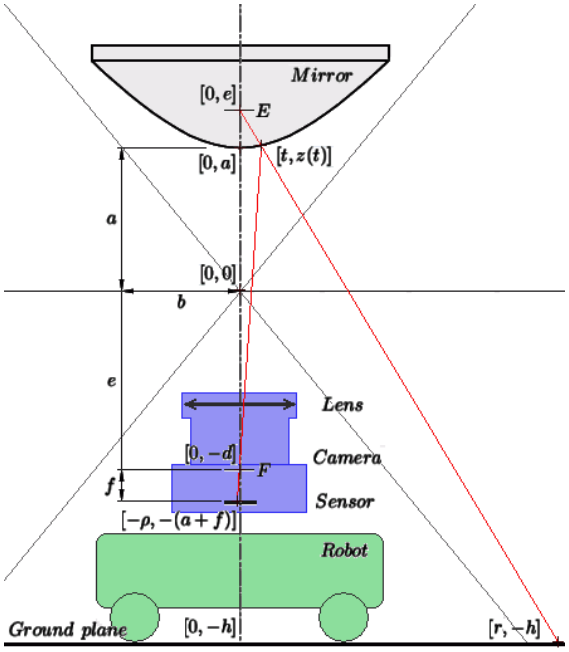


Figure 3.2: Schematic for the ground plane projection and rectification

The solution of this equation is:

$$t = \frac{2ab^2\rho f}{b^2 f^2 - a^2\rho^2}$$

Because of the single-viewpoint feature of the hyperbolic profile, the point of intersection with the ground plane $[r, -h]$ can be evaluated using triangle similarity. This is because the reflected ray is on a line connecting the inner mirror focus $[0, e]$ and the reflection point $[t, z(t)]$ (see fig. 3.2).

Note that in the implementation the transformation is pre-calculated and stored in a look-up table to increase the execution speed.

3.2 Color perception

Fast and robust color classification is an important part of an application detecting visual landmarks determined by their color. A special algorithm exists that addresses the classification issue for robotic applications where the processing speed is an important factor [8].

Objects uniformly painted by the same color are not represented by an equal RGB color when captured by a camera. This is due to the effect of illumination on the hue and brightness of the reflected light. It is mainly

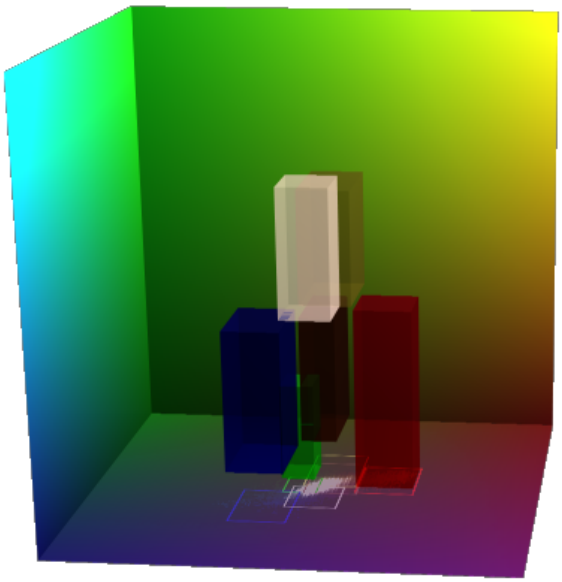


Figure 3.3: Representation of the landmark color classes using rectangular thresholds in the `YCrCb` space

the brightness that varies widely for the object's pixels. This is why the captured image is first transformed into the `YCrCb` color space, which uses one coordinate to express brightness and the two others for blue and red chromaticity respectively. The `YCrCb` space enables to approximate the color classes for many robotic applications with rectangular blocks (see fig. 3.3). This is because the artificial colors of landmarks can be usually delimited in the `CrCb` plane and the `Y` coordinate is left to compensate for the influence of illumination. Allowing a wide span in the brightness channel reduces the effect of object shape, orientation and position relative to the camera and light sources on the color classification.

The rectangular shape of the color classes is advantageous for the fast classification algorithm — it can classify each pixel using only 2 bitwise `AND` operations, whereas a naive approach would require 5 conditional `AND` operations in the worst case:

```
if ((Y >= Ylowerthresh) AND (Y <= Yupperthresh)
    AND (U >= Ulowerthresh) AND (U <= Uupperthresh)
    AND (V >= Vlowerthresh) AND (V <= Vupperthresh))
    pixel_color = color_class;
```

If the task was to decide whether a given number ranging from 0 to 255 is within a pre-defined interval, one can define a lookup table representing the characteristic function of the interval. Since the three-dimensional thresh-

old can be represented as a product of three interval-defined subspaces, the classification of one pixel can be performed using 3 lookups:

```
pixel_in_class = YClass[Y] AND UClass[U] AND VClass[V];
```

For best performance, all the color classes can be evaluated at once. If the lookup tables are made of 32-bit integers, 32 color classes can be stored, one in each bit. The bitwise `AND` identifies all the color classes a pixel belongs to making the respective bit set.

3.3 Monte Carlo Localization

In a probabilistic localization framework [10], the robot state at a current time-step k is estimated given knowledge about the initial state and all *measurements* $Z_k = \{\mathbf{z}_k, i = 1..k\}$ up to the current time. The robot *state* is given by a vector $\mathbf{x}_k = [x, y, \theta]^T$ comprising the position and orientation of the robot. The estimation problem is an instance of the Bayesian filtering problem where the posterior density $p(\mathbf{x}_k|Z_k)$ of the current state conditioned on all measurements is constructed. In the Bayesian approach, this *probability density function* (PDF) holds all the knowledge that is available about the state $\mathbf{x}_k$. In sampling based methods, the density is approximated by a set of N random weighted samples or *particles* $S_k = \{(s_k^i, w_k^i); i = 1..N\}$ drawn from it. Each s_k^i stands for one possible state of the robot and the w_k^i expresses the probability of the current state being s_k^i. The current position can be estimated from this representation, e.g. as a weighted average of all samples.

To localize the robot, the goal is to recursively compute at each time-step k the density $p(\mathbf{x}_k|Z_k)$ working with the set of samples S_k that is drawn from it. This is done in two phases, the *prediction* and the *update*.

3.3.1 Prediction

In the first phase a *motion model* is used to predict the current position of the robot in the form of a predictive PDF $p(\mathbf{x}_k|Z^{k-1})$, taking only motion into account. The current state $\mathbf{x}_k$ is assumed to depend only on the previous state $\mathbf{x}_{k-1}$ and a known control input $\mathbf{u}_{k-1}$. The motion model is thus specified as a conditional density $p(\mathbf{x}_k|\mathbf{x}_{k-1}; \mathbf{u}_{k-1})$. The predictive density over $\mathbf{x}_k$ is then obtained by integration:

$$p(\mathbf{x}_k|Z^{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}; \mathbf{u}_{k-1})\, p(\mathbf{x}_{k-1}|Z^{k-1})\, d\mathbf{x}_{k-1}$$

In the implementation of a particle filter, the motion model is applied to each of the particles s^i_{k-1} computed in the previous iteration obtaining a new set S'_k that approximates a random sample from the predictive density $p(\mathbf{x}_k|Z^{k-1})$:

```
for each particle s^i_{k-1}:
    draw one sample s'^i_k from p(x_k|s^i_{k-1}; u_{k-1})
```

3.3.2 Update

In the second phase the position estimate obtained in the form of a predictive PDF is updated using a measurement model of the current sensor readings to obtain the posterior PDF $p(\mathbf{x}_k|Z^k)$. The measurement $\mathbf{z}_k$ is assumed to be conditionally independent of earlier measurements Z^{k-1} given $\mathbf{x}_k$. The measurement model is expressed in terms of a likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$. This term denotes the likelihood of the state $\mathbf{x}_k$ given that $\mathbf{z}_k$ was observed. The posterior density over $\mathbf{x}_k$ is obtained using Bayes' theorem:

$$p(\mathbf{x}_k|Z^k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|Z^{k-1})}{p(\mathbf{z}_k|Z^{k-1})}$$

For a particle filter, this update consists of two steps. First the weights are corrected according to the measurement $\mathbf{z}_k$. Then the particle set is resampled to represent the new density $p(\mathbf{x}_k|Z^k)$.

All the samples in the predictive set S'_k are updated by the likelihood of s'^i_k given $\mathbf{z}_k$, i.e. weighted by:

$$w^i_k = p(\mathbf{z}_k|s'^i_k)\, w^i_{k-1}$$

The new set S_k is then obtained by resampling from this weighted set. All the weights are first normalized so that $\sum_i w^i_k = N$. The resampling then reproduces the samples s'^i_k having a high weight associated with them replacing those with a low weight. This addresses the problem of a continual degeneration of the sampled PDF. The sample set must be updated to remove insignificant samples and give place to the promising ones. In doing so a new set S_k is obtained that approximates a random sample from $p(\mathbf{x}_k|Z^k)$:

```
for i = 1..N:
    if w^i_k > 2 draw two S_k samples from p(x_k|s'^i_k)
    else if w^i_k < 0.5 drop the sample s'^i_k
    else if |S_k| < N draw one S_k sample from p(x_k|s'^i_k)
```

After the update phase the process is repeated recursively.

At time $k = 0$ the knowledge about the initial state $\mathbf{x}_0$ is assumed to be available in the form of a density $p(\mathbf{x}_0)$. The particle filter starts with a random set of samples $S_0 = s_0^i$ drawn from this prior density. In the pose tracking task, the density $p(\mathbf{x}_0)$ represents a narrow distribution around the known initial pose.

Chapter 4

Implementation

This chapter describes a prototype implementation of the probabilistic localization module presented in Section 2.5. The implementation consists not only of the resulting module but creates a complex framework for testing and tuning various image-processing modules. See the Appendix D.1 for the list of features of the application accompanying this thesis.

For the purpose of examining various image processing methods, the abstraction of an *image filter* is used. This is generally a piece of code that takes one image at its input and produces one at the output. The individual filters are aggregated into *filter chains*, sequences of filters aimed at a particular task. The following sections give an overview of the filters used in the localization process.

4.1 Filters and Chains

An image filter provides one level of processing of the input video stream. Each filter is defined as a descendant of the abstract class `ImageFilter`. This class provides the principal ability to group into chains and additional services such as import and export of settings and saving the output to a file. It also allocates memory for the output image and can display the filter's output in a window. One method, `Process`, is abstract and is left to be implemented by the specific filter classes.

Individual filters are grouped into filter chains that aggregate the image filtering for a particular purpose. Each chain is derived from the `FilterChain` class and must redefine the method `Start` that adds and initializes the individual filters. The framework can switch between different chains, each of which may provide a different view of the input video. Each chain also defines, which of the contained filters will have an output window. It allows

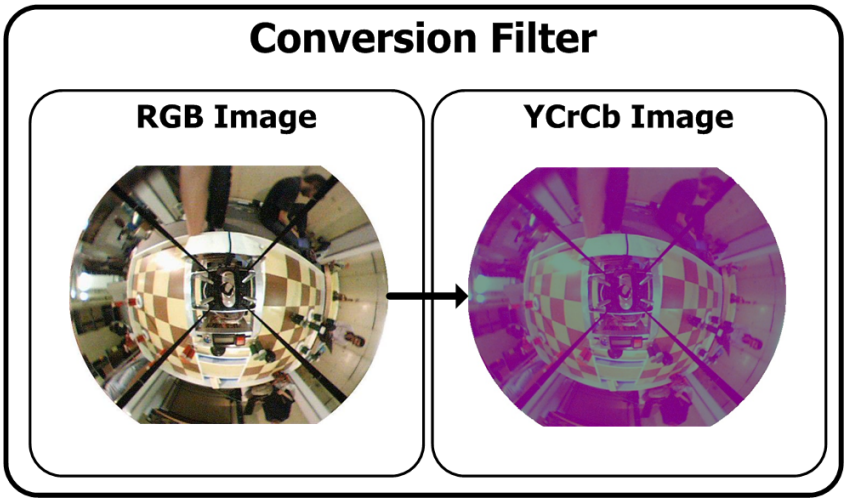


Figure 4.1: `NormYUVFilter` enhances the input images and transforms it into the `YCrCb` color space. Note that the output image is displayed in false colors for illustration reasons as the image itself does not change, only the base it is enumerated to. The `Y`, `Cr`, and `Cb` channels are displayed as green, red and blue respectively.

a chain to hide the filters with uninteresting or hardly readable output and make the framework less cluttered.

Many filters also provide access to their configuration variables via trackbars displayed in the respective windows. A user may change these variables and store them in a configuration file, which is automatically read at each startup.

Additional filter chains allow tuning of the filters involved in the localization chain. The following subsections describe the individual filters in more detail.

4.2 Pre-processing

There is a number of helper filters in the framework that can be used to enhance the input video and to experiment with its properties. Only those used in the localization chain are described here.

4.2.1 Input Enhancement

The input video is first normalized so that the mean and standard deviation of the individual color channels are constant. This helps to robustify the following process with respect to the changes in illumination reducing the

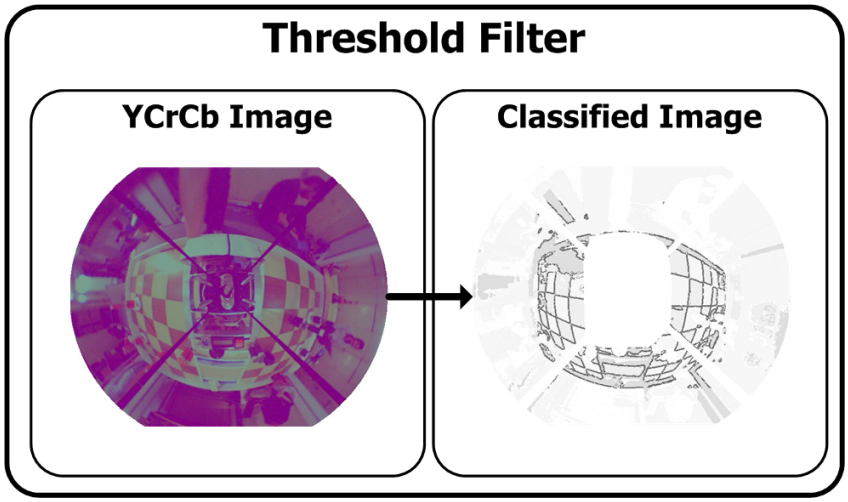


Figure 4.2: `ThreshBoundFilter` classifies the input color image into color classes and detects the important playing field boundaries

influence of color and intensity of the ambient light on the tint and brightness of the observed visual marks. The transformation is global, i.e. applied uniformly on all image pixels. Without this pre-processing the subsequent color classification is less reliable.

In one step of the localization chain the robot's silhouette is also masked out to remove confusing data from the processed image.

The input is also transformed from the `RGB` color space of the camera output to the `YCrCb` color space used in the color classification. For performance reasons, both these transforms are performed at once by the `NormYUVFilter` class (see fig. 4.1).

4.2.2 Color Classification

The color classification extracts the landmark information from the input image by applying a 3-dimensional threshold to the input image for each of the six defined color classes (see Section 3.2). The colors defined in the contest environment are red and green for the game elements, blue, brown and beige for the playing field and white for its borders. Each of these classes is represented by one bit of the 8-bit output image. One more bit is used to store the information about the playing field boundaries, which is later used for the motion prediction. The inner playing field boundaries of the individual checkerboard squares as well as the outer field boundaries of the checkerboard and the white border are marked by decorating the pixels

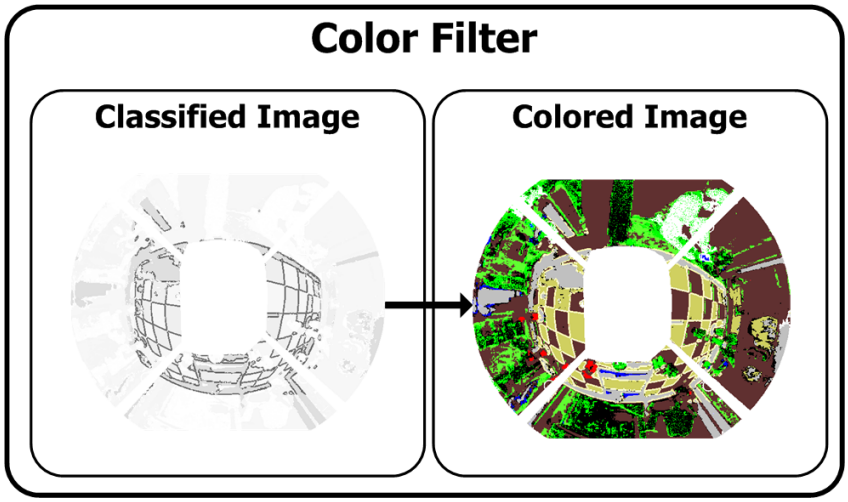


Figure 4.3: `ColorFilter` visualizes the classification by displaying the individual color classes in schematic colors

that lay in a close neighborhood of two of these three color classes with the boundary bit.

To improve performance, two operations are again performed at once. The input image is classified and searched for color class boundaries at the same time (see fig. 4.2). It is important to note that throughout the localization process, only the thresholded input image is used, never the original one. This is enabled by the discriminative painting of the color landmarks and allows better performance of the whole process.

4.2.3 Color Display

One of the helper filters that is only used to visualize information for the user is the `ColorFilter` (see fig. 4.3). It paints the classified image with schematic colors to make the color classification better readable. This filter is used e.g. in the `ClassEditChain` that enables to interactively tune the classification thresholds.

4.2.4 Image Rectification

As the floor markings form a periodical orthogonal pattern, it is straightforward to begin the processing with a projection of the camera image to the ground plane. This approach yields an image where straight lines correspond to straight lines on the floor, which significantly simplifies the following process. The `ProjectFilter` (see fig. 4.4) uses several user-defined

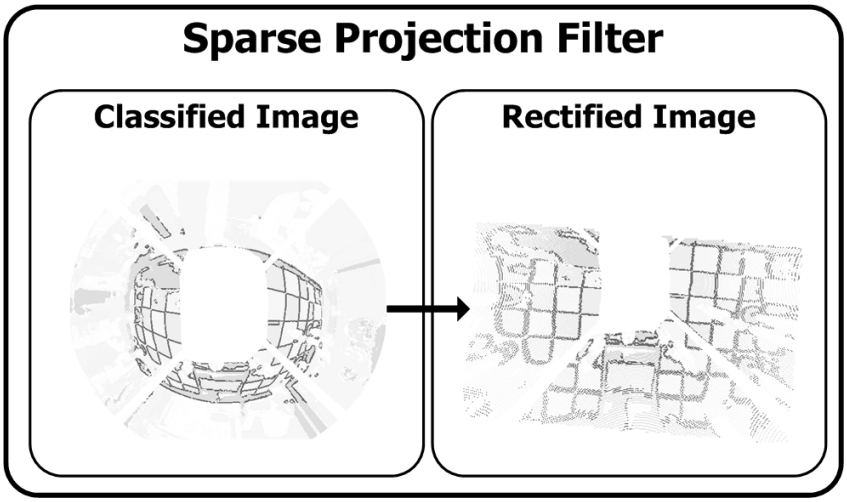


Figure 4.4: `ProjectFilter` rectifies the input image into ground plane coordinates given the parameters of the mirror geometry

parameters, such as the mirror radius and position within the input image to recover the geometry of the floor markings using the transform described in Section 3.1. Its descendant `MaskProjectFilter` is used to mask out the robot's silhouette that could confuse some of the following filters. Another filter, `FieldMaskFilter`, is used to further limit the projection only to the current estimated bounds of the playing field relative to the camera.

Note that the transform used in the localization chain is sparse, in terms that it only moves the individual pixels of the input image to the transformed coordinates and does not perform any interpolation. This is because an interpolation would not add any new information and would uselessly waste computational resources because no subsequent filter requires a dense input image. However, there is an `InverseProjectFilter` that calculates the inverse projection for the destination pixels to get a dense rectified image using the nearest neighbor interpolation (see fig. 4.5). This output is meant for evaluation of the transform parameters by the framework user and is provided by the `ProjectPreviewChain`.

4.3 Prediction

After the image is rectified, the boundaries extracted in the enhancement phase form several parallel lines in two perpendicular directions. The task is to determine the rotation and relative shift of these lines with respect to the camera. If all the lines in the rectified image were either horizontal or vertical, computing the column sums would result in sharp local maxima cor-

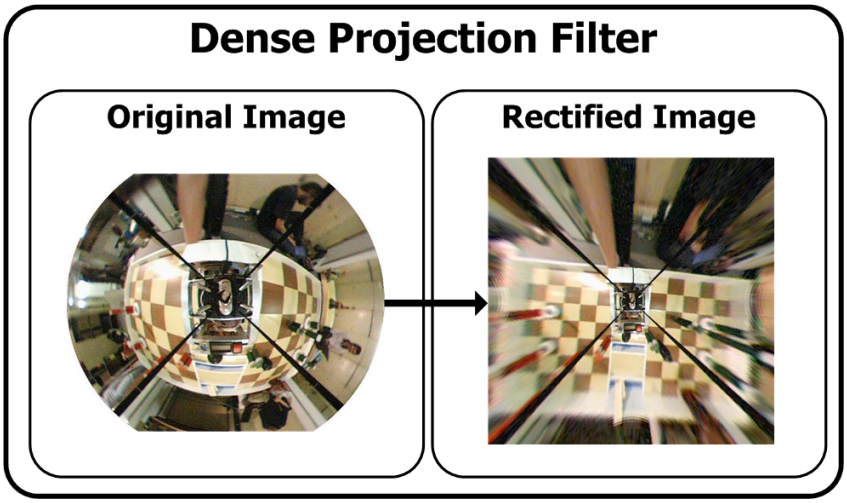


Figure 4.5: `ProjectFilter` rectifies the input image into ground plane coordinates given the parameters of the mirror geometry

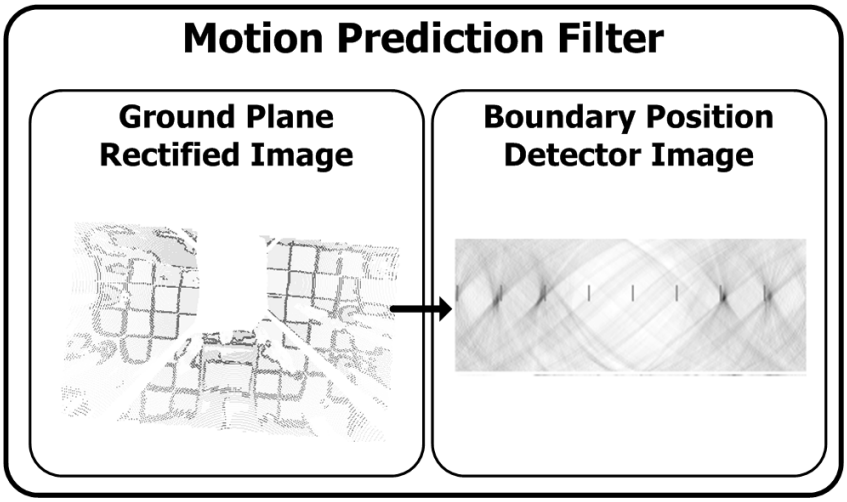


Figure 4.6: `SumFilter` predicts the robot's motion from the evolution of the detected field boundaries

responding to the vertical lines. If neither of the two boundary orientations is perfectly vertical, these local maxima will blur. But if the image was first skewed so that some of the boundaries become vertical, the local maxima would reappear. Thus, if a fixed number of skew angles is tried out, the one that best matches the true orientation of the vertical lines will have the sharpest peaks in the scanline sum. The `SumFilter` examines 90 different orientations, to estimate the robot's relative orientation with a resolution of one angular degree as the projected boundary image is $\frac{\pi}{2}$-periodical.

Analogical approach may be used for the offset of the perpendicular boundaries but as the best skew angle is already known only one sum (in the appropriate direction) is performed.

An important feature of the pose estimation process is the invariance to the playing field surroundings. This is required because the area beyond the playing field borders is not defined and can easily contain similar landmarks that would confuse the pose estimation. This problem is solved by clipping the projected view to the extent of the estimated playing field location. As long as the estimated pose matches the true one, all the landmarks beyond the borders are ignored.

The drawback of the boundary-based approach is that it only provides location within one checkerboard square. If all the boundaries were always detectable, counting the peaks on both sides of the robot would provide an absolute location. This unfortunately is not true, because several boundaries may be occluded by the robot itself, as well as other objects randomly placed on the playing field. An average of the peak offsets is used instead to estimate the robot's offset within the square-size period.

The relative position within one checkerboard square is estimated and this information is aggregated recursively to provide the rough estimate of the robot's location. This estimate is filtered using a simple exponential averaging to reduce the effect of temporary distortions on the predicted motion. The change of the filtered estimate then enters the MCL framework in the terms of a control input $\mathbf{u}_k$ (see Section 3.3.1). A random noise term is also added to deal with prediction errors.

4.4 Correction

The final image filter uses the thresholded image again but now only several random pixels are picked instead of processing the entire image. This adds a complementary information to the process that was up to now based only on edge information — the color of landmark surfaces is used to refine the position estimate within the MCL framework.

For each sample, a fixed count of random observations is picked from the

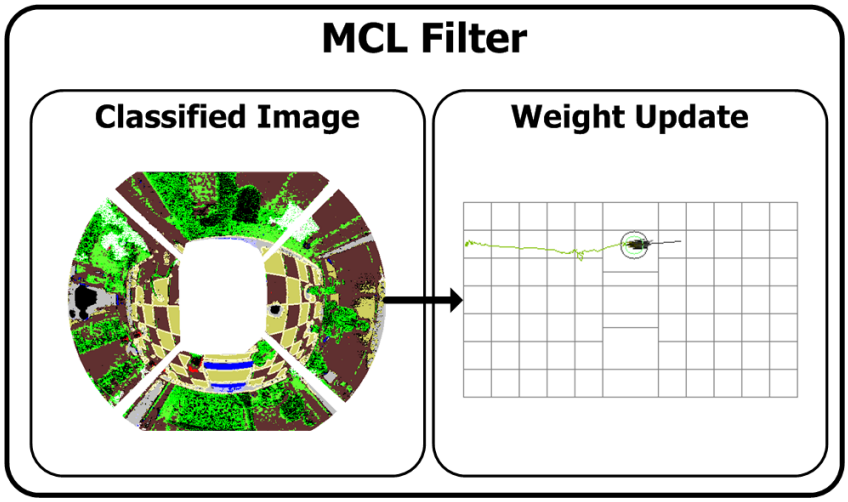


Figure 4.7: `VisualMclFilter` updates the MCL samples with the classified visual information and displays the localization progress

thresholded image, transformed to the playing field coordinates using the pose represented by the sample and verified against a known map of playing field colors. A match with the map increases the weight of the sample a mismatch decreases it and a neutral observation (a pixel not belonging to any of the landmark color classes) leaves the weight intact within the MCL update (see Section3.3.2). The weight update highlights the samples around the true robot's pose as the observations for outlying samples do not match the color map and cause a quick decrease of the associated weights.

The pixel-wise observation method helps to increase the robustness of the entire process as virtually any pixel in the source image can be used for the weight update. If a small portion of pixels represents false observations it only affects a small subset of all the samples. This reduces the influence of image distortions, occlusions and local color misclassifications on the overall performance. Each of the approximately 90 000 possible observations contained in the input image may be used in the correction phase. The individual possible observations are stored in a lookup table in the form of a mapping from the source image coordinates to the respective ground-plane projection. The evaluation is thus very fast as it only requires one lookup and a transformation from the sample-centric coordinates to the world ones.

Similarly to the prediction phase, the update must also be insensitive to the playing field surroundings. This is done by simply ignoring the observations that fall outside the color map. The map contains all the area of the playing field and a small margin to enable matching against the playing field borders.

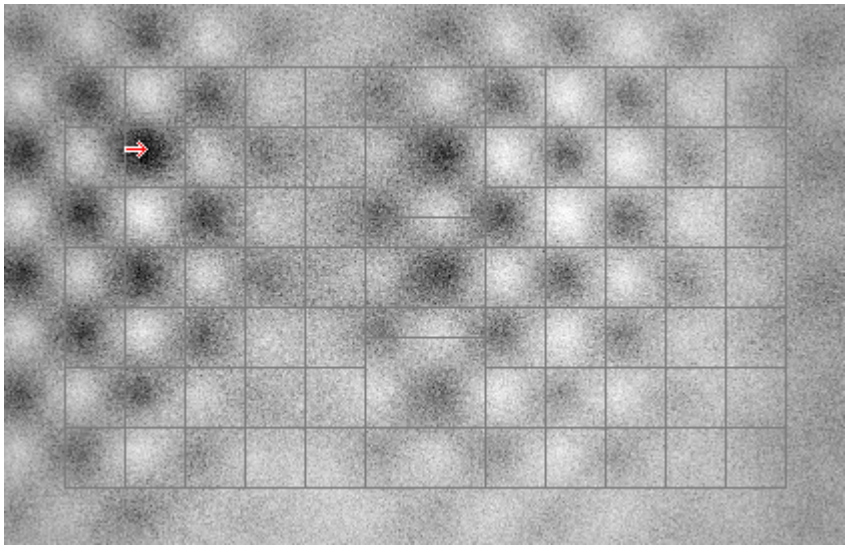


Figure 4.8: A likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$ evaluated for different robot positions given the set of 90 000 pixel-wise observations, 20 of which are randomly selected per update. Darker colors denote higher values. The true robot pose is shown by a red arrow

The major complication posed by the checkerboard playing field on the localization module is the periodicity that allows the pose estimate to drift to a diagonally adjacent square producing very similar observations. To compensate for this ambiguity, the observations of the ditch and the playing field border are given higher weights. The probability distribution of the measurement model $p(\mathbf{z}_k|\mathbf{x}_k)$ induced by the pixel-wise observation evaluation is shown in Figure 4.8. It is a probabilistic plot rendered by averaging a large number of updates for each of the different positions drawn in the image. The orientation of the robot is fixed and corresponds to the horizontal direction in the plot. The true position of the robot is marked by an arrow. As expected, the probabilistic update (in asymptotic terms) reaches significant local maxima at the true robot pose but also at the diagonally adjacent squares.

Chapter 5

Experiments

To verify the usefulness of the proposed approach, an extensive amount of tests was conducted with the prototype implementation. The experiments were performed both in simulated conditions and in real-world environment. The results of these two experiment scenarios are given in the following sections. All the experiments were performed off-line, using only the video stream representing an omni-directional view of the operation field.

Ten different sizes of the particle set were always used for the pose estimation. All the experiments were repeated several times with 50, 100, 150, 200, 250, 300, 400, 500, 750, and 1000 particles to test for the influence of the size of the sample set on the accuracy and robustness of the method. Note that the computational burden increases linearly depending on the particle count taking the constant overhead of the image pre-processing into account. The processing speed varied from 20 fps for 50 samples to 13 fps for 1000 samples on a Celeron-M at 1300 MHz.

Fast update rate is important not only for accuracy reasons. The visual odometry also requires that the updates occur frequently enough to resolve the ambiguity in the motion of the periodic pattern. The maximum forward speed must allow two consecutive observations within one checkerboard square to determine the direction of the motion. A similar condition applies to the angular speed. A processing pace of 15 fps imposes an upper bound of approximately $2\,ms^{-1}$ on the forward speed and $4\pi\,s^{-1}$ on the angular one, which is far enough for a mobile robot acting on a playing field less than 4 m long.

5.1 Simulated Environment

A series of simulated experiments was performed to test the prototype implementation in idealized conditions and to enable evaluation of the maximum

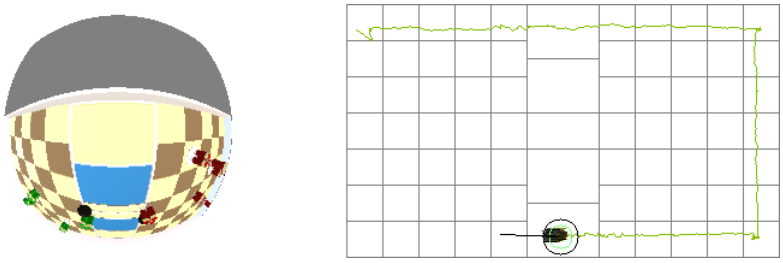
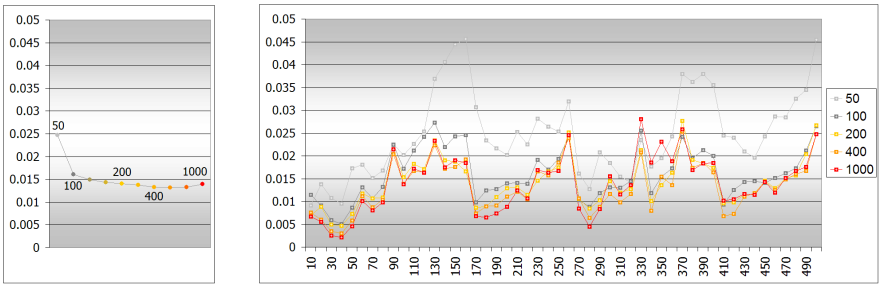


Figure 5.1: Simulated tracking experiment



(a) Batch error

(b) Average error per simulation step

Figure 5.2: Average displacement error in meters in the simulation experiment with respect to the particle count calculated (a) for the entire batch of 50 experiments and (b) for the individual localization steps of the simulated experiment

accuracy that can be reached with the proposed method. A 3-D model of the playing field was constructed and a simulated video stream was rendered along a rectangular trajectory. The model was created to mimic real-world conditions as truly as possible including occlusions caused by individual game elements and light reflections.

A rectangular trajectory circling the entire playing field at 20 cm from the border was rendered to verify the accuracy of the method with different sizes of the particle sets (see Section 3.3). Perpendicular distance from the original trajectory was measured in every tenth localization step. The number of observations per update was fixed to 20 for each sample. A random displacement realized by a uniform distribution $[-50\,mm, 50\,mm]$ was applied to each sample to compensate for inaccuracies in the motion prediction phase. An experiment batch of 50 tests was executed for each of the ten different particle counts.

Figure 5.2(b) displays the course of the displacement error averaged for the experiments within a batch. The results of only 5 representative batches are displayed. The development of the average localization error (computed

from all the 2500 localization steps recorded in every batch) is shown in Figure 5.2(a). The displayed value denotes average localization error in meters. The average error falls quickly towards 400 particles. After that it does not decrease significantly. The best value was reached using 500 particles and it was approximately 13 mm, which is very good.

The biggest local error is reached when passing from one playing field side to the other, i.e. when crossing the ditch. It shows up that one small simplification can have important influence on the local performance. The actual simplification included in the localization algorithm is that all the checkerboard boundaries form a rectangular pattern with a period of 30 cm. In fact, the ditch is separated from the two checkerboards by a 22 mm wide white line, which makes the boundaries detected on the different sides of the ditch misaligned by 4.4 cm. This error is unfortunately inherent and is unreplaceable because the motion prediction merges all the detected peaks into one assuming the 30 cm period. The relative position estimate thus slowly drifts from the coordinate system of one field to the other as the influence of the distant boundaries diminishes. In a real-world environment this error is negligible compared to other distortions but in the simulated case it is noticeable.

As can be seen from the charts the experiments conducted with the simulated environment proved that with too small particle sets the localization is not reliable. Using too many samples is generally not deteriorative although their update may consume too much computational resources. However, from a certain size the sample set provides a good coverage of the represented PDF and the results do not dramatically improve when adding more samples. In this application, the limit particle set contains around 500 particles.

5.2 Target Environment

For the following experiments the original playing field for the Eurobot^{open} 2005 Czech National Cup [12] was used. The test robotic platform was equipped with a catadioptric sensor whose hyperbolic mirror was placed 45 cm above the ground. Its field-of-view covers the entire operation field. The image was captured using an off-the-shelf web-camera connected to USB and providing thirty 640×480 uncompressed images per second. The robot was manually pushed along the playing field and the omni-directional video stream was recorded. As noted in Section 2.5, this video stream is sufficient for the application to be able to reconstruct the original robot's trajectory.

The prototype application was tested off-line on seven recorded video streams (see fig.5.4). The average length of a test stream is 40 s, which corresponds to 1200 localization steps. The localization process was repeated 70

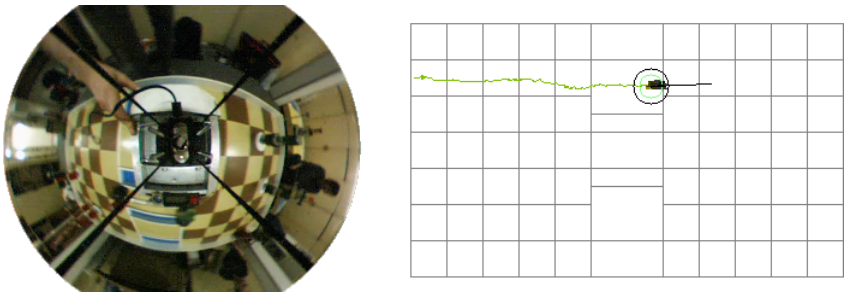


Figure 5.3: Real-world tracking experiment

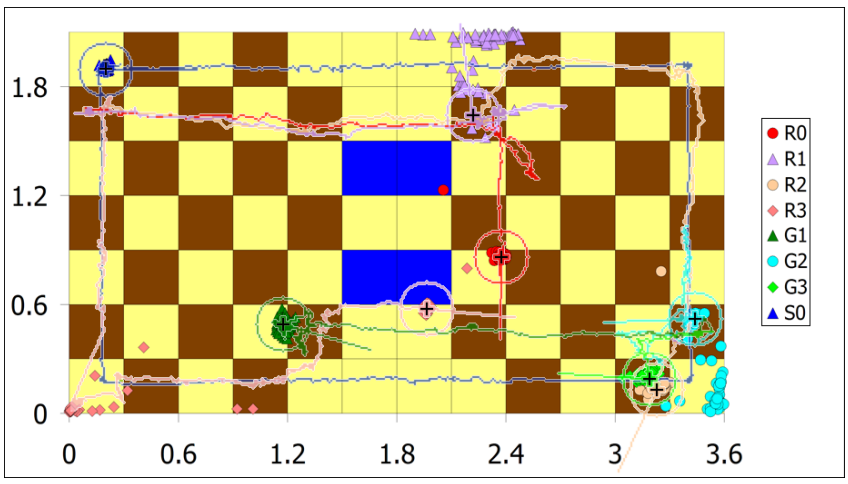


Figure 5.4: The trajectories estimated using the vision-based method rendered for the eight test video streams

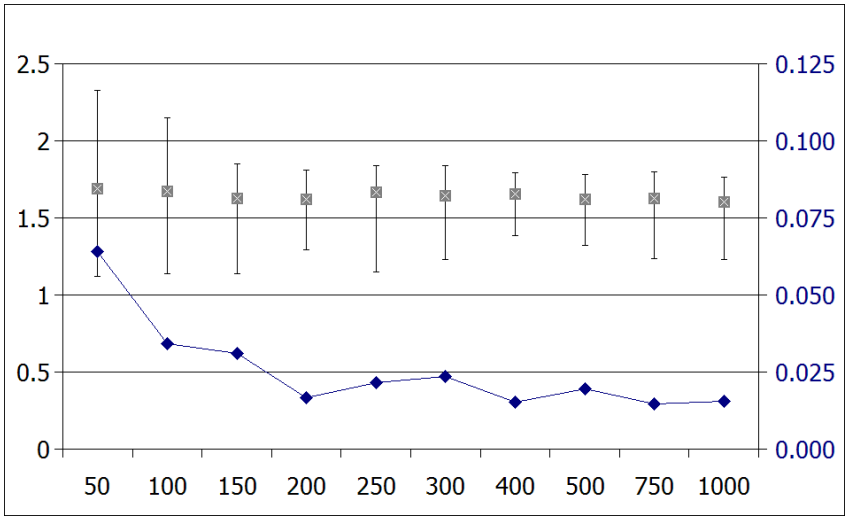


Figure 5.5: Influence of particle count on accuracy of rotation tracking; gray — final estimated orientation in radians, blue — the divergence of the final orientation

times (using a different initialization of the pseudo-random number generator) and the final estimated pose was evaluated to determine the probability of a complete tracking failure for the given video stream. Note that no ground truth information is available to be able to verify the accuracy in the real-world conditions. However, due to the periodic pattern of the playing field a translational error greater than 15 cm or rotational greater than 45° leads to a total failure.

Generally, the proposed method performs very well. Five of the eight test videos, namely `R0`, `R2`, `G1`, `G3`, and the simulated `S0`, produced in total only 3 erroneous results out of a total of 350 conducted experiments. There are three sample videos that pose some problems to the localization module. This is mainly because of significant image distortion at some step that leads to misinterpretation of the robot's motion and a false prediction. The rest of this section describes, how the proposed method deals with such errors.

The `R0` test video ends with a rotation of 270° around the robot's vertical axis near to an edge of the playing field. The fact that few playing field boundaries are visible and that the entire image is blurred by the rotation results to big oscillations in the motion prediction. Figure 5.5 shows the average final orientation of the robot estimated for a given particle count. The maximum and minimum values are shown by black vertical braces. The divergence of the final orientation is drawn in blue.

Similar to the results discussed above, the divergence of the final rotation continually falls down to a local minimum at 400 samples and then does

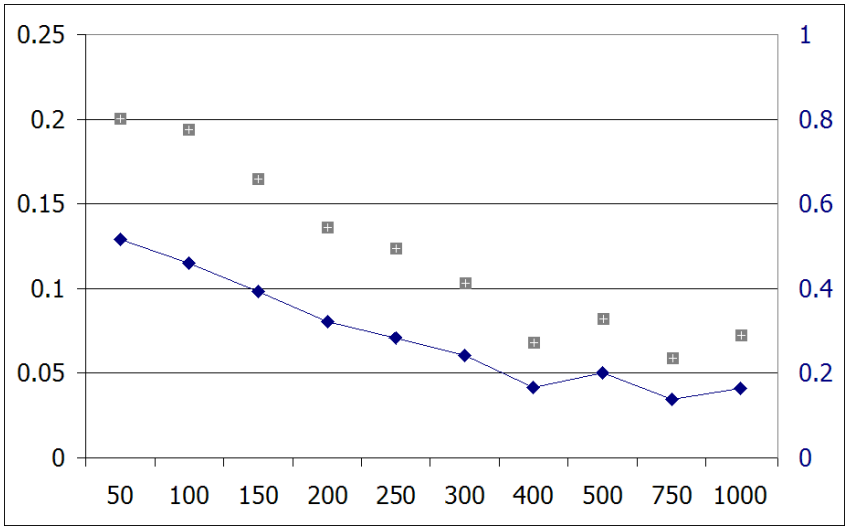


Figure 5.6: Influence of particle count on the probability of successful pose tracking; gray — average displacement at the final step in meters, blue — probability of total tracking failure i.e. returning a final pose more than 30 cm away from the true one

not further improve. This accords to the previously stated fact that with a particle set of this size the represented PDF is well covered.

This experiment shows that if the motion prediction fails during fast rotation of the robot it is very hard to recover. However, if the robot was controlled on-line, it could have detected the loss of credibility of the pose estimate and reduce the angular speed. In the off-line experiment, the video stream goes on and the pose changes too quickly to be recovered by the update method. Once the difference between the true pose and the estimated one exceeds the period of the field pattern the robot is lost.

The `G2` video shows an example of an omni-directional stream damaged by light reflections. The robot is maneuvered within an area close to the start location, where only a small portion of the playing field is displayed in good resolution. Moreover, the field is occluded by stacked game objects and finally a large portion of the color landmarks is destroyed by a reflecting light source. In this complicated setup the motion prediction fails to produce accurate estimate and the correction often picks misclassified pixels.

In such scenario a higher number of particles is an obvious advantage. The bigger the particle set the higher the probability a correctly classified pixel is used in the update. Figure 5.6 displays the influence of the size of the sample set on the probability of tracking failure and error in the final pose. Both these measures diminish as the particle count rises. A total of 580 tests was

executed within which the probability of tracking failure varied from 51% (for 50 particles) to 14% (750 particles). The average displacement error ranged from 200 mm (50 particles) to 59 mm (750 particles). Surprisingly, the configuration of 400 samples again performed very well.

In all the executed experiments the method showed to be robust with respect to the lighting, occlusions, and image blurs and distortions caused by the robot's motion and tilt. The weakest point shows to be the pose change estimation that can be fooled when too little of the playing field is drawn in good resolution. In such cases the motion prediction can produce a false estimate. This happens mainly near the corners of the playing field, especially when a part of the landmarks is further occluded by game elements or damaged by reflecting lights. The correction phase usually deals with such errors.

The average probability of correct tracking for all eight test videos is 79.7% using 400 particles. As noted above, this means that in four experiments out of five the maximum error never exceeded 15 cm, which is good.

Chapter 6

Conclusion

This thesis has given a thorough overview of the task of vision-based localization of a mobile robot within the dynamic environment of a robotic contest. It presented a complete solution that can track a mobile robot anywhere within the limited contest field using no other than visual sensors, performing real-time visual pose tracking on a general hardware. Since the large field of view is important to avoid lack of traceable features in the camera image, a solution using catadioptric omni-directional visual sensor was developed. The method is best suited for localization in a highly dynamic environment of robotic competitions taking place on delimited operation field painted with an easily detectable color pattern.

6.1 Results

A method for probabilistic ego-motion tracking based on edge and color features of the playing field reconstructed from an omni-directional image was developed. A prototype implementation using Monte Carlo Localization was presented and was shown to perform very well in the target environment of the Eurobot^{open} 2005 robotic contest. Instead of using an expensive range-finder sensor with ambiguous readings, an affordable combination of an off-the-shelf camera and a hyperbolic mirror was used to produce a reliable position estimate. The proposed method was able to reconstruct the original robot's trajectory reaching an average accuracy of approximately 15 mm. The results of both simulated and real-world experiments display the usability of the approach and propose several new challenges.

One of the scopes of this thesis was to evaluate whether a vision-based odometry that does not suffer from the problem of unmodeled displacement can completely replace a hardware one. Contrary to the odometry based on counting the revolutions of wheels that can slip and fail to detect sideways

motion caused by collision, the visual odometry detects all the robot's motion up to a certain speed. This is especially important in situations, where many moving objects may influence the robot's intended trajectory. On the other hand, the visual odometry, as well as the hardware one, produces erroneous measurements. The important fact is that the problematic scenarios of the two methods do not overlap. A combination of both methods could produce exquisite results.

The major contribution of the thesis is a new vision-based update method for the probabilistic framework that enables to use virtually any pixel of the input omni-directional image to enhance the pose estimation. The update method only requires a color map of the environment, which is very easy to define for the playing field of a robotic contest. This new approach decreases the influence of local misclassification as different observation items are selected for each of the particles. Each of the approximately 90 000 possible observations contained in the input image can be used. The evaluation is very fast, because the observation transforms are pre-calculated.

The important advantage of the omnidirectional sensor is that the image of entire operation field is available regardless to the robot's pose. Additionally, the evaluation function does not require a specific background beyond the border of the operation field. The localization process is insensitive to the surrounding of the playing field, which is not defined.

6.2 Future Research

Further work will focus on merging the hardware odometry information with the vision-based motion prediction to remove the limitation on the speed of intended motion. Future enhancements may consider a redefinition of the motion estimation to enable the method to perform in operation fields without a periodic rectangular pattern.

A global update sensor may also be added to enable relocalizing the robot in case of a tracking failure.

Finally, interesting results could be obtained implementing a feedback channel that would affect the robot's actions depending on the immediate reliability of the estimated pose. The robot can react by reducing speed or performing actions that may help the localization process to recover using reliable landmarks.

Appendix A

Playing Field Plan

The playing field for the Eurobot^{open} 2005 contest [12] is a rectangular area 3.644 m long and 2.1 m wide consisting of two fields painted with a brown-beige checkerboard pattern separated by a blue ditch that can be passed using three bridges (see fig. A.1). Each of the two fields consists of 7×5 tiles measuring 30×30 cm. The ditch is 60 cm wide and is separated from the two fields by a white stripe 22 mm wide. A white border 7 cm high is encircling the playing field. It is assumed to lay outside of the playing field dimensions.

The starting locations of the two robots are in the two opposite corners. The playing field contains following game elements and obstacles, all placed respecting a central symmetry:

- 15 green and 15 red skittles (painted wooden cylinders) initially stacked into 4 piles,
- 4 white skittle stands placed symmetrically under two and two of the skittle piles, and
- 2 black bowls placed on the ends of the central bridge.

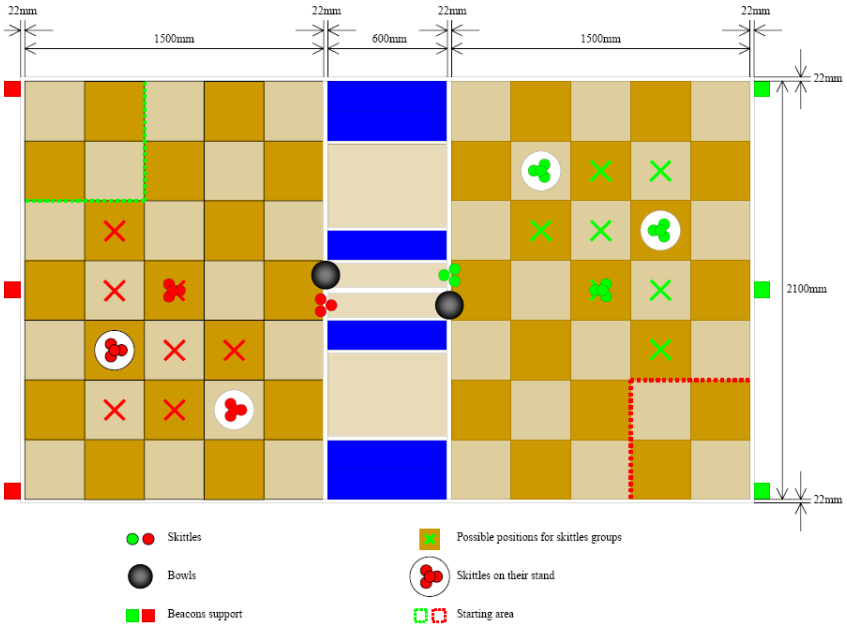


Figure A.1: The plan of the Eurobot^{open} 2005 contest playing field

Appendix B

Target Robotic Platform

The test robotic platform (see fig. B.1) used in the experiments is a differentially driven robot with the propelled wheels having 70 mm in diameter mounted in back and two steerable wheels mounted in the two front corners. The approximate dimensions of the robot are 30×20×32 cm.

The robot is equipped with a web-camera and a hyperbolic mirror mounted 45 cm above ground. The mirror is a hyperbolic glass mirror `H3G` from Neovision [37] and is described by the surface equation:

$$\frac{z^2}{789,3274} - \frac{x^2+y^2}{548,1440} = 1$$

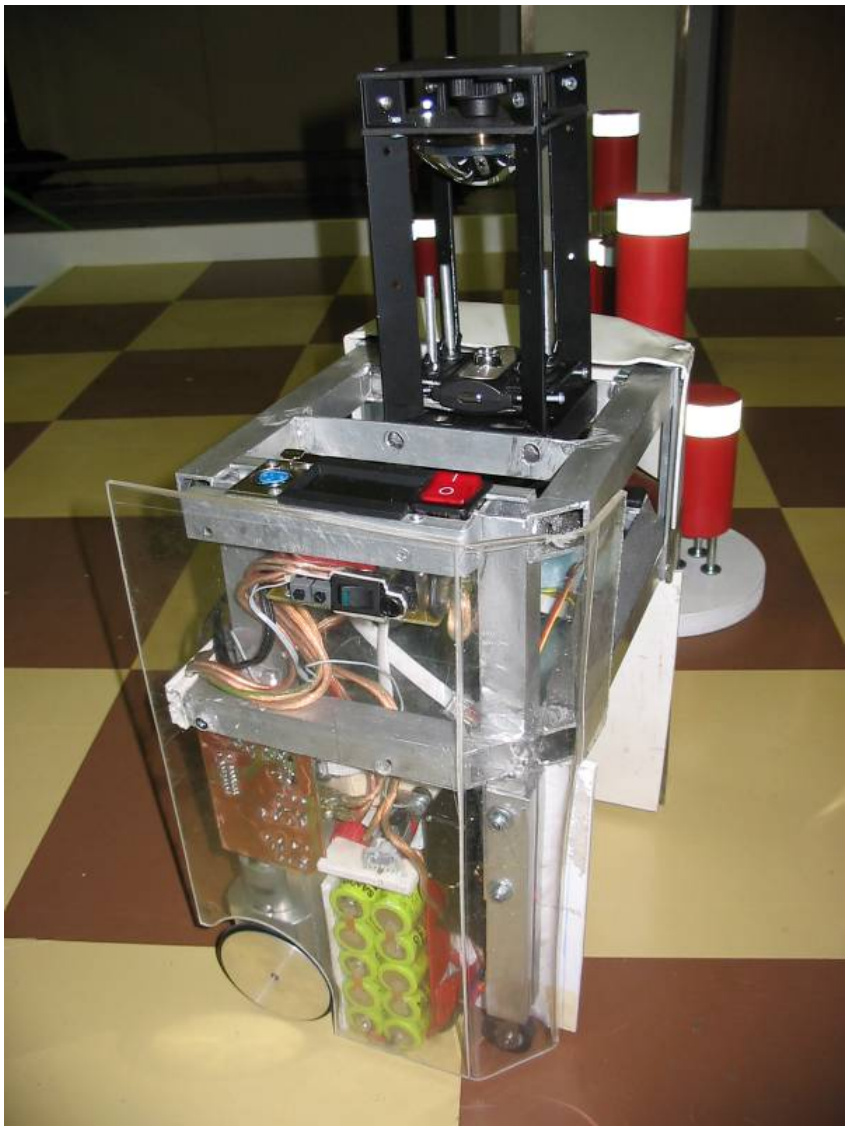


Figure B.1: The test robotic platform with the omni-directional sensor mounted on top

Appendix C

Attachment CD Contents

The following directories are present on the CD disk:

- `bin` — Contains all the binaries necessary to run the vision framework.
- `bonus` — Contains two videos about the Eurobot^{open}contest.
- `OpenCV` — Contains a redistribution of the Intel Open Source Computer Vision Library.
- `src` — Contains the complete source files of the application and a project file that can be opened with Visual Studio .NET 2003.
- `tracking` — Contains 9 output videos rendered by the application using the included sample video streams. They are stored both in AVI and QuickTime formats and may be immediately played back.
- `vids` — Contains 8 sample video streams for the vision framework in the AVI format.

The directory '`vids`' contains 8 sample videos that can be used to test the prototype application. A batch file 'vision.cmd' is contained in the '`vids`' directory. It accepts the name of a sample video as the first argument and executes the file '`vision.exe`' in the '`bin`' directory with the appropriate settings.

Even simpler, one of the eight batch files in the CD root can be used to execute the application with one of the sample videos. The following batch files are present:

- `S0.cmd` — Shows a rectangular path rendered in a simulated environment.
- `R0.cmd` — The robot follows an L-like trajectory across the field.

- `R1.cmd` — The robot goes straight and then rotates by 270°.
- `R2.cmd` — The robot goes around the field to the opposite corner.
- `R3.cmd` — The robot goes close to the start border and passes the far bridge.
- `G1.cmd` — The robot passes a bridge and turns by 180°.
- `G2.cmd` — The robot follows a complicated path near the starting location.
- `G3.cmd` — The robot is pulled sideways.

Appendix D

Software Documentation

The following sections document the prototype application provided at the attachment CD.

D.1 Vision Framework Usage

Once the vision framework is run (e.g. using one of the batch files in the CD root) a preview window opens and a list of available commands is printed to the console. These commands enable viewing different filter chains contained in the framework and to manipulate with the input video. They also enable to save the current chain output to disc.

The commands are as follows:

- `0-5,9` — Selects the displayed filter chain:
- `0` — `Preview` Chain (displays only the preview of the video)
- `1` — `Project Preview` Chain (shows the ground-rectified projection)
- `2` — `Color Class Edit` Chain (displays the color classification)
- `3` — `Boundary Detector` Chain (shows detected checkerboard boundaries)
- `4` — `Video MCL` Chain (displays a pose tracking window)
- `5` — `Combined MCL` Chain (displays more tracking windows for a detailed look onto the localization process; note that the localization process executes if and only if one of this two chains is selected)
- `9` — `Threshold Gradient` Chain (shows a histogram of the classified image)

- `SPACE` — Pause/Resume (toggles input video playback)
- `R` — Restarts the input video
- `+` — Advances 1 s forward in the input video stream
- `-` — Returns 1 s back in the input video stream
- `[` — Skips 1 frame back in the input video stream
- `]` — Skips 1 frame forward in the input video stream
- `D` — Dumps all the current chain images to disk
- `C` — Capture (continually dumps all windows)
- `V` — Video (dumps the chain output filter)
- `S` — Stop capture (stops recording)
- `I` — Imports the filter settings from disk (implicitly `Filters.txt`)
- `E` — Exports the filter settings to disk (implicitly `Filters.txt`)
- `M` — Marks (toggles display of calibration marks on the preview image)
- `F` — Freeze (stops the video stream and counts FPS on the current frame)
- `Q` — Query performance (displays the average processing speed)
- `x` — Exit (the program terminates when the video stream ends)
- `ESC` — EXIT (the program ends immediately)

Some filters also have interactive windows. These are namely the 'Preview' window (0: `Preview` Chain) that enables to change the mirror center and radius using the middle and right mouse buttons respectively and the 'Colored' window (2: `Color Class Edit` Chain) that enables to add and remove pixels to/from the current color class using left mouse button and the Ctrl/Alt key modifiers respectively.

The 'VisualMCL' window of the (5: `CombinedMcl` Chain) enables to replace the estimated robot pose using the right mouse button.

Note that the application can be executed directly from the CD. The batch files in the CD root can be used to open and test the individual sample video streams. When exporting data from the framework, these are stored in the subdirectories of `C:/temp/vision`.

D.2 Description of Vision Framework Files

The directory '`src`' on the CD contains the following files:

- Capture.h (cpp) — Handles input of the video stream including seeking.
- FilterChain.h — Aggregates the individual filters into chains that can be executed by the framework.
- Framework.cpp — Main source of the application. Performs handling of the command-line arguments and interactive commands on chains.
- ImageFilter.h (cpp) — Defines the base class for all the image filters. Handles allocation, reading of settings, output to a window and saving output to a file.
- Localize.h (cpp) — Implements the persistent state estimation used in the tracking.
- MclFilter.h (cpp) — Image filters used for the particle update from the classified image.
- PreviewFilter.h(cpp) — A set of simple filters that serve for visualization.
- ProjectFilter.h(cpp) — A set of projection filters that perform the image rectification.
- StdAfx.h (cpp) — A precompiled header file.
- SumFilter.h (cpp) — Performs the visual odometry by tracking periodic straight lines.
- ThresholdFilter.h(cpp) — Classifies the input image using a fast 3D thresholding.
- Transform.h (cpp) — Implements the mirror reflection transforms used in the rectification.
- Vision.sln (vcproj) — The project files for the Visual Studio .NET 2003.

Note that the project can be opened and compiled from the CD. However, intermediate files as well as the compiled binary are placed in `C:/temp/vision`. If you want to remove these files later you must do it manually.

The OpenCV library [40] is required for the project to compile. A complete package (redistributed under the terms of Intel License Agreement For Open Source Computer Vision Library) is included in the OpenCV directory.

The path to the OpenCV library files must be set in the project configuration, section `Linker|General`. It is set to `../OpenCV/lib` for the solution saved on the CD. If the solution files are moved to other location, this path must be reset accordingly.

Bibliography

[1] SICK AG. Lms 200 indoor laser measurement system. http://ecatalog.sick.com/Products/ProductFinder/product.aspx?pid=9168.

[2] H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localisation using omnidirectional images. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 297–299. IEEE Computer Society Press, Silver Spring, MD, 1998.

[3] H. Andreasson, A. Treptow, and T. Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Proc. of the ICRA-2005, IEEE International Conference on Robotics & Automation*, 2005.

[4] Acroname Articles. Demystifing the Sharp IR rangers. http://www.acroname.com/robotics/info/articles/sharp/sharp.html.

[5] Simon Baker and Shree K. Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2):1 – 22, 1999.

[6] J. Benda and Z. Winkler. Tracking the absolute position of a mobile robot using vision-based monte carlo localization. In *Proc. of the 1st International Conference on Dextrous Autonomous Robots and Humanoids (DARH)*, 2005.

[7] R.A. Brooks. A robust layered control system for a mobile robot. *Artificial Intelligence at MIT Expanding Frontiers*, 2:3–27, 1990.

[8] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2000.

[9] RemoteReality Corporation. Omnialert camera. http://www.remotereality.com/vtprod/oa_camera.html.

[10] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[11] G. Dudek and M. Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, New York, NY, USA, 2000.

[12] Eurobot^{open}Competition. Eurobot assosiation. http://eurobot.org/.

[13] The RoboCup Federation. Robocup official site. http://www.robocup.org/.

[14] D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Naviagation*. PhD thesis, University of Bonn, Germany, 1998.

[15] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pages 343–349, 1999.

[16] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots, 1998.

[17] J. Gaspar, N. Winters, E. Grossmann, and J. Santos-Victor. Toward robot perception using omnidirectional vision. *In Innovations in Machine Intelligence and Robot Perception*, 2004.

[18] H.-M. Gross, A. Koenig, H.-J. Boehme, and Ch. Schroeter. Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store. In *Proc. 2002 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS 2002)*, pages 256–262, 2002.

[19] J.-S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.

[20] D. Haehnel, D. Burgard, W. Fox, and Thrun S. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[21] P.V.C. Hough. Machine analysis of bubble chamber pictures. In *Proc. of the International Conference on High Energy Accelerators and Instrumentation*. CERN, 1959.

[22] D. Huttenlocher, R. Lilien, and C. Olsen. View-based recognition using an eigenspace approximation to the hausdorff measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):951–956, 1999.

[23] Point Grey Research Inc. Ladybug camera. http://www.ptgrey.com/products/ladybug2/index.html.

[24] M. Isard and A. Blake. Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[25] H. Ishiguro and S. Tsuji. Image-based memory of environment, 1996.

[26] M. Jogan and A. Leonardis. Robust localization using an omnidirectional appearance-based subspace model of environment. *Robotics and Autonomous Systems, Elsevier Science*, 45(1):51–72, 2003.

[27] R.E. Kalman and R. Bucy. A new approach to linear filtering and prediction problems. In *Trans. ASME, Journal of Basic Engineering*, 1960.

[28] C. Marques and P. Lima. A localization method for a soccer robot using a vision-based omni-directional sensor. *RoboCup 2000 Book*, 2001.

[29] R. Matoušek and L. Žalud. Project orpheus. http://www.orpheus-project.cz/.

[30] P. Maybeck. *Stochastic Models, Estimation and Control*, volume Vol. 1. Academic Press, 1979.

[31] E. Menegatti, A. Pretto, and E. Pagello. A new omnidirectional vision sensor for monte-carlo localization. In *Proc. of the 2004 Int. RoboCup Symposium CD-ROM*, 2004.

[32] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Hierarchical image-based localisation for mobile robots with monte-carlo localisation. In *Proc. of 1st European Conference on Mobile Robots ECMR'03 September 2003*, 2003.

[33] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based monte-carlo localisation with omnidirectional images. *Robotics and Autonomous Systems, Elsevier Volume 48, Issue 1*, pages 17–30, 2004.

[34] H. P. Moravec. The stanford cart and the cmu rover. *IEEE*, 71(7):872–884, 1983.

[35] A. Motomura, T. Matsuoka, and T. Hasegawa. Self-localization method using two landmarks and dead reckoning for autonomous mobile soccer robots. In *RoboCup 2003*, pages 526–533, 2003.

[36] National Oceanic National Geodetic Survey and Atmospheric Administration (NOAA). Removal of GPS selective availability. http://www.ngs.noaa.gov/FGCS/info/sans_SA/.

[37] Neovision. H3S hyperbolic mirror. http://neovision.cz/prods/panoramic/h3s.html.

[38] T. Pajdla and V. Hlaváč. Image-based self-localization by means of zero phase representation in panoramic images. In *In Proceedings of the 2nd International Conference on Advanced Pattern Recognition volume 2013 of Lecture Notes in Computer Science*, pages 24–33, 2003.

[39] L. Paletta, S. Frintrop, and J. Hertzberg. Robust localization using context in omnidirectional imaging. In *Proc. ICRA 2001*, pages 2072–2077, 2001.

[40] Intel Research. Open source computer vision library. http://www.intel.com/research/mrl/research/opencv/.

[41] D. Sekimori, T. Usui, Y. Masutani, and F. Miyazaki. High-speed obstacle avoidance and self-localization for mobile robots based on omni-directional imaging of floor region. In *RoboCup 2001: Robot Soccer World Cup V*, pages 204–213, London, UK, 2002. Springer-Verlag.

[42] D. Sekimori, T. Usui, Y. Masutani, and F. Miyazaki. Evaluation of self-localization performance for a local vision robot in the small size league. In *Lecture Notes in Computer Science*, volume 2752, pages 41–52, 2003.

[43] R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based SLAM using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.

[44] R. Sim, M. Griffin, A. Shyr, and J.J. Little. Scalable real-time vision-based SLAM for planetary rovers. In *Proceedings of the IEEE IROS Workshop on Robot Vision for Space Applications*, 2005.

[45] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1995.

[46] J. Sladek. Detekce orientačních bodů pro mobiln roboty. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2003.

[47] A.F.M. Smith and A.E. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician*, 46(2):84–88, 1992.

[48] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[49] J.W. Weingarten, G. Gruener, and R. Siegwart. A state-of-the-art 3d sensor for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[50] J. Wolf and A. Pinz. Particle filter for self localization using panoramic vision. *ÖGAI Journal*, 22(4):8–15, March 2003.

[51] Y. Yagi, Y. Nishizawa, and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor copis. *IEEE Trans. Robotics and Automation*, 11(5):634–648, 1995.