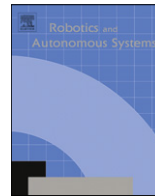




ELSEVIER

Contents lists available at ScienceDirect

## Robotics and Autonomous Systems

journal homepage: [www.elsevier.com/locate/robot](http://www.elsevier.com/locate/robot)

## An appearance-based visual compass for mobile robots

J. Sturm<sup>a,\*</sup>, A. Visser<sup>b,\*\*</sup><sup>a</sup> Lehrstuhl Autonome Intelligente Systeme, Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany<sup>b</sup> Intelligent Systems Laboratory Amsterdam, Informatica Instituut, Universiteit van Amsterdam, The Netherlands

## ARTICLE INFO

## Article history:

Received 5 September 2007

Received in revised form

1 October 2008

Accepted 9 October 2008

Available online xxx

## Keywords:

Appearance-based

Mobile robot localization

Active vision

Machine learning

## ABSTRACT

Localization is one of the most important basic skills of a mobile robot. Most approaches, however, still rely either on special sensors or require artificial environments. In this article, a novel approach is presented that can provide compass information for localization, purely based on the visual appearance of a room. A robot using such a visual compass can quickly learn a cylindrical map of the environment, consisting of simple statistical features that can be computed very quickly. The visual compass algorithm is efficient, scalable and can therefore be used in real-time on almost any contemporary robotic platform. Extensive experiments on a Sony Aibo robot have validated that the approach works in a vast variety of environments.

© 2008 Elsevier B.V. All rights reserved.

## 1. Introduction

A central goal of robotics and AI is to be able to deploy a mobile robot that acts autonomously in the real world [1]. An important prerequisite for this goal is that the robot can localize itself in the world [2]. Much progress has been made in this field over the past years [3].

From the spectrum of possible sensors, laser range scanners have often been used for robot localization. However, these types of sensors have a number of drawbacks. Laser range scanners are relatively heavy and their measurements lack – especially in natural environments (like corridors or similar offices) – sufficient features of distinguishable quality. Vision has long been advertised as providing a solution to these problems. From a conceptual perspective, it is more desirable that a robot makes use of the same sensor modalities as humans do. The real world is full of visual indications that have especially been installed to facilitate (human) localization, such as public signage, arrows or other salient landmarks (like high buildings).

Given the continuous progress both in the hardware domain and computer vision algorithms, we expect to see more approaches on camera-based localization in the near future. In this paper, we will focus on approaches that have been designed to localize mobile robots in real-time in natural environments, both indoor and outdoor. The real world setting requires that the robot is able to adapt to new or unknown scenes, and is able to cope with potentially dynamic changes in the environment, like walking people or subtle changes in (daylight) illumination.

Many current publications address the problem of simultaneous localization and mapping (SLAM). Here, a robot is deployed into a previously unknown environment, in order to explore, map and localize itself within this environment. The SLAM problem is considered to be one of the central problems in mobile robotics, as SLAM is a prerequisite for most higher-level robot applications. Solving the SLAM problem, however, requires the successful integration of many underlying techniques (like feature extraction, mapping, exploration and localization).

Within visual SLAM, different types of camera setups currently co-exist. Although single and stereo vision approaches are clearly the most popular in the literature, trifocal systems and multi-camera rigs also have proven advantages of their own. In this paper, we want to focus on monocular (single viewpoint) systems with a limited field-of-view.

In this monocular setting, only the bearing towards objects can be estimated easily. The absence of depth information makes 3D reconstruction much more difficult as in the stereo setting. A related problem to SLAM is called visual odometry in robotics, or structure-from-motion (SFM) in computer vision literature. Here, the goal is to estimate the robot's trajectory from recent camera

\* Corresponding address: Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee, Geb. 079, Germany. Tel.: +49 761 203 8010; fax: +49 761 203 8007.

\*\* Corresponding address: Informatica Instituut, Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands.

E-mail addresses: [sturm@informatik.uni-freiburg.de](mailto:sturm@informatik.uni-freiburg.de) (J. Sturm), [a.visser@uva.nl](mailto:a.visser@uva.nl) (A. Visser).

URLs: <http://ais.informatik.uni-freiburg.de> (J. Sturm), <http://www.science.uva.nl/research/isla> (A. Visser).

images, without necessarily building (and storing) an explicit map. Closely related are approaches for visual compassing, where the aim is to recover the viewing (yaw) angle from the camera images.

In this paper, a novel approach to visual compassing will be described. The incoming images are split into vertical sectors, and a color class transition pattern is derived statistically by counting. In the training phase, the robot creates a computationally efficient cylindrical (1-dimensional) map of its surroundings – step-wise, by taking multiple snapshots with its camera in different directions. In the localization phase, the camera orientation can be estimated by matching the seen color transition pattern with the stored model using a maximum-likelihood estimator. In real-world experiments, it is verified that this visual compass is robust to small translations and thus can supply a mobile robot with valuable information.

**Outline.** This paper is organized as follows: first, related approaches will be described in Section 2. Our visual compass approach is subsequently introduced in Section 3. As all necessary operations are computationally extremely efficient, the visual compass has been implemented on a Sony Aibo robot as described in Section 4. The results of our experiments will then be presented in Section 5. Finally, the characteristics of the approach will be summarized in Section 6 and our results will be discussed in comparison to other approaches.

## 2. Related work

The focus of this paper is centered on camera-based approaches that can localize a mobile robot in real-time in natural environments. Existing approaches have been evaluated and the following characterizing properties have been identified:

**Geometric vs. appearance-based.** Geometric approaches assume the existence of landmarks. Their relative bearing, distance and (not necessarily unique) identity can be estimated and stored in a 3D feature map. Newly perceived features can then be matched with the stored map in order to derive the robot's pose. On the other hand, appearance-based approaches analyze incoming images directly by statistical means. The extracted data is stored in a topological map. By finding the nearest neighbor to a newly perceived data sample, the robot can estimate its current absolute pose.

**Real-time.** Although many approaches claim the attribute real-time, true video-rate processing (30 Hz) is only achieved by a small fraction of current implementations. Processing times in the magnitude of 1fps are common.

**Odometry.** Knowing the motor commands or odometry sensors may provide an initial guess about the robot's path. Approaches that take advantage of odometry information certainly have an advantage over approaches that do not, but in turn they lose part of their generality. Furthermore, in some applications, it is either not desirable or not possible to use information from odometry. Many visual odometry approaches try to derive the odometry information from camera images only, and therefore have, by definition, no access to other odometry sources (like wheel encoders or action commands).

**Long-term stability.** Differential approaches compare a small number of recent images to determine the relative movement of the robot. Such solutions intrinsically are prone to long-term drifting, and are opposed to the concept of repeatable localization. To avoid drifting, the robot has to create a long-term map, that allows absolute localization (and recovery from kidnaps) even for extended periods of time.

**Initialization.** In particular for single-camera systems, map initialization is, for most approaches, a distinguishable issue, in order to find the camera-to-world scale. Either the robot uses prior knowledge about the shape and size of some objects in the scene, or it uses its body movements (and odometry) to incrementally estimate the correct scale of the map.

**Environments.** While a few solutions exist that work in large environments of high complexity (such as buildings or outdoor scenes like a forest), others limit the application domain to a single room with no or little occlusion.

**Accuracy.** Each localization algorithm can be evaluated in one or more testing environments, in order to determine its mean localization error. The estimated pose is compared to the ground truth, that is typically measured by an external source (like a laser range scanner or GPS). Depending on the approach, the measured error is then expressed in the difference of recovered path length or the accumulated distance of the true pose.

**Robustness.** Unfortunately, only a few publications evaluate the robustness of their algorithms towards dynamic changes in the environment. These could be natural changes in illumination or the reaction of the localization system to walking people in populated environments.

In the following, a short overview of existing approaches on visual SLAM, visual odometry and visual compassing will be given, and their results will be compared with our results at the end of this article.

The patented vSLAM algorithm [4] extracts SIFT features from incoming images. The features of three images taken at 20 cm distance from each other are – after a mutual consistency check – stored as a single 3D landmark percept in a landmark database. As a single landmark then consists of the 3D points of around 100 SIFT features, each landmark percept is unique, which eases data association and improves recognition. For localization, a particle filter is used. Each of its particles additionally contains a Kalman filter bank in order to estimate the global positions of the stored landmarks relative to each other.

In the approach of Kaess and Dellaert [5], a robot is equipped with a 8-camera-rig to obtain high-resolution omni-directional images. Per image, fast Harris corner detection is applied to find salient feature points. Mutual consistency checks filter out outliers or bad matches, and the found features are then tracked over time. The feature tracks, around 70 per image, are then combined to produce a motion hypothesis using RANSAC that is subsequently integrated with odometry into the robot's estimated path (using Levenberg–Marquardt minimization). Due to the relatively short tracking length of the corner features (4 frames on average), the approach is, however, prone to long-term drifting. In [6], a very similar approach is described, that in contrast relies on only a single monocular camera. Here however, as neither vehicle odometry is used nor loop closure is applied, the motion hypotheses are accumulated over time, which, due to the dead reckoning of visual odometry, leads to positional drifting over time.

A second geometric approach on visual odometry is given in [7,8]. A Shi and Tomasi saliency operator is used to extract stable features, that are subsequently stored as landmarks in the map. For each new feature, a 1D particle filter is initialized in order to estimate the landmarks depth (typically within 5 frames). As the landmarks are added persistently to a map, no long-term drift occurs. By the use of an innovation matrix, the algorithm can also determine how much uncertainty could be eliminated when the camera would look in a certain direction (active vision).

In the work of Montiel and Davison [9], a feature-based visual compass is described: here, a hand-held camera tracks distant points, that are ideally located at infinity. The task is then to

recover the 3D rotation of the camera. Features are initialized and deleted on-the-fly, and their positions are tracked using an extended Kalman filter bank.

More closely related to our work, an appearance-based visual compass is developed in [10,11]. The robot compares each captured image with the previous one, and computes the Manhattan distance between the two images. The new image is now rotated pixel-wise in a horizontal direction, in order to find the rotation under which the distance becomes minimal. This rotation is then reported to the robot as differential compass information. However, such an approach is computationally expensive and, moreover, prone to long-term drifting.

In contrast to the work above, we designed our algorithm in such a way that it can operate in real-time on board an autonomous robot. The robot is also able to initialize its own map, and the final accuracy is measured over the full process of mapping and localization together. Our visual compass approach extracts simple statistical features from the images, that can be computed, learned and matched very quickly.

### 3. Approach

The basic idea behind the visual compass approach is as follows: as the robot should be able to quickly orient in any given environment, an intuitive approach would be to store a 360° panoramic image of the surroundings and use it subsequently as a cylindrical map. Then the robot could match every new incoming image with this cylindrical map to estimate its orientation.

Typical cameras produce – despite their limited field-of-view – an enormous amount of data that needs to be reduced significantly before it can be used for localization in real-time on board a mobile robot. In this work, we develop an appearance-based feature extractor that is based on counting the transition frequencies between color classes in the image. Such simple statistical features are fast to compute, and therefore can be used on robots even with limited processing power in real-world applications. For a quick overview of our processing pipeline, see Fig. 1.

We have structured this section as follows: first, processing starts with the raw images acquired from a monocular camera. We define how a color-class transition pattern  $\mathbf{z}$  is extracted from an incoming camera image  $F$ . Then a histogram filter is constructed that is able to learn a cylindrical map  $\hat{\mathbf{m}}$  of the color-class transition patterns in different directions. Finally, a localization filter is constructed with which the robot can estimate its pose  $\mathbf{x}$  from an incoming camera image  $F$ , given a cylindrical map  $\mathbf{m}$  of the environment.

#### 3.1. Camera model

We use a rotating camera model similar to [9], where the robot is standing at the origin of the world, with pitch (or elevation) angle  $\phi$ , roll angle  $\theta$  and yaw (or heading) angle  $\psi$ . This results in the camera projection matrix  $C$  that describes the transform from homogeneous world coordinates to homogeneous pixel coordinates [12] for a camera with focal length  $f_x, f_y$ , skew  $s$ , and principal point  $c_x, c_y$ , (i.e.,

$$C = \begin{pmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} R_x(\psi)R_y(\theta)R_z(\phi). \quad (1)$$

We assume that the robot stands on an even floor, such that the pitch and roll angles can be computed from the odometry of the robot's camera. It is now the task of the visual compass to estimate the yaw angle  $\psi$  from the images  $F_t$  taken by the camera  $C$  at time step  $t$ . As we are primarily interested in pixels originating from distant objects in the scene (ideally at infinity), we project

the ground plane into the camera image and only take pixels in account that lie above the horizon. We address a single pixel by  $F_t[x, y]$  (with  $x \in \{0, 1, \dots, w-1\}$  and  $y \in \{0, 1, \dots, h-1\}$ ), each having a color from a three-dimensional color space  $[0, 1]^3$ .

#### 3.2. Color class transition patterns

Our approach on appearance-based localization is based on the idea that the colors in a natural environment are not uniformly distributed. More specifically, we assume that the frequency pattern of color class transitions strongly depends on the yaw angle of the camera. Note that at this point, we treat the whole image as a single measurement of which we extract a single color class transition pattern along a vertical scanline passing through the optical center of the camera.

Processing starts with the incoming pixel image  $F_t$  taken by the robot's camera, see Fig. 1. Of each incoming camera image, simple statistical features  $\mathbf{z}$  are extracted, i.e., the transition frequencies between color classes, see Fig. 2. The frequency pattern corresponding to each image is sampled uniformly from a small number of neighboring pixel pairs along the scanline. Due to the probabilistic sampling, there is no need to smooth or de-noise the incoming image, as the probability of a transition between color classes to be sampled will converge in the limit to the true relative frequency.

#### 3.3. Image feature extraction

We decided to discretize the continuous color space into the  $n$  most significant color classes  $|\mathbb{C}| = n$ . By using such a color discretization function  $g$ , each color  $v \in [0, 1]^3$  can be mapped onto its corresponding color class  $c = g(v) \in \mathbb{C}$ . Such a mapping can be created by a robot autonomously by clustering the color space, for example by using the Expectation–Maximization algorithm with a Gaussian mixture model [13]. An efficient and working implementation of this approach has been used by the Dutch Aibo Team in [14].

The color class transition frequencies can be counted from an incoming image very quickly. For visual compassing, we want to count the color class transitions in a vertical direction in the world coordinate frame, as visualized in Fig. 2 (middle). This requires to project the vertical scanline  $l_w(s)$  (with  $0 \leq s \leq 1$ ) with heading  $\psi_{\text{scan}}$  in the world frame to a scanline  $l_c(s)$  in the camera frame, by computing

$$l_c(s) = C l_w(s). \quad (2)$$

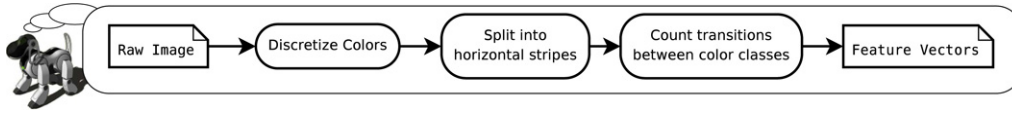
Of course, the valid range  $\psi_{\text{scan}}$  of visible scanlines is limited by the opening angle  $\alpha$  of the camera and the current camera yaw  $\psi$ , i.e.,

$$\psi - \alpha/2 \leq \psi_{\text{scan}} \leq \psi + \alpha/2. \quad (3)$$

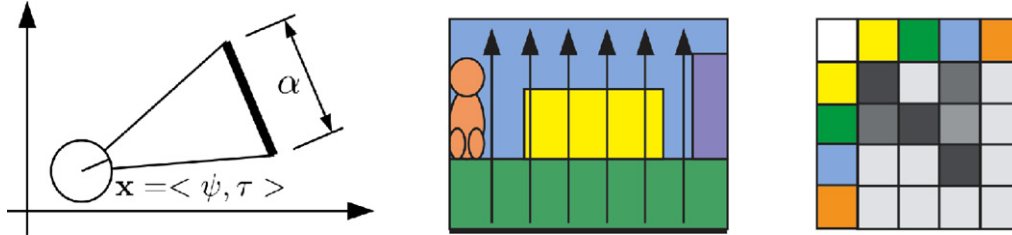
Along such a scanline  $l_c(s)$ , the robot can now simply count the number of transitions  $z'_{ij}$  between every pair of color classes  $i$  and  $j$ . These absolute frequencies can then be divided by the total sum of transitions on the line in order to obtain the relative frequency  $z_{ij}$  for each pair of color classes, i.e.,

$$z_t^{ij} = \frac{1}{\sum_{i',j' \in \mathbb{C}} z_t^{i'j'}} z_t^{ij}. \quad (4)$$

Using this technique, the robot can extract  $z_t^{ij}$  by counting the number of color class transitions in the image.



**Fig. 1.** Overview of the feature extraction pipeline used by our visual compass. From the incoming images, simple statistical features are extracted, such that all processing is possible in real-time and on-board an autonomous system.



**Fig. 2.** Overview of the visual compass approach. **Left:** The robot stands at a certain pose  $\mathbf{x} = \langle \tau, \psi \rangle$  and captures an image  $F_t$  using its single monocular camera with opening angle  $\alpha$ . **Middle:** The image is scanned from bottom to top for transitions between color classes. **Right:** These transitions are counted and the transition frequency pattern  $\mathbf{z}$  is extracted. This feature vector is then used for mapping and localization.

3.4. Transition patterns

In total, there are  $n \times n$  transition frequencies between each of the  $n$  distinct color classes defined earlier. We denote the transition frequency pattern  $\mathbf{z}$  as follows:

$$\mathbf{z} = \begin{pmatrix} z^{11} & \dots & z^{1n} \\ \vdots & \ddots & \vdots \\ z^{n1} & \dots & z^{nn} \end{pmatrix}. \tag{5}$$

This transition frequency pattern is the feature vector that we use for mapping and localization.

Each time the robot captures an image, it will extract a slightly different frequency measurement due to camera noise, indicated by the series  $(\dots, \mathbf{z}_t, \mathbf{z}_{t+1}, \dots)$ .

We will now assume that this measurement series is generated by an underlying distribution  $\mathbf{Z}$ . For our approach, we have chosen to model each of the elements  $\mathbf{Z}^{ij}$  of the transition pattern  $\mathbf{Z}$  by means of an individual histogram distribution  $\mathcal{H}$ , defined by the absolute frequencies of  $k$  logarithmically scaled bins  $m_{(1)}^{ij}, \dots, m_{(k)}^{ij}$ .

$$\mathbf{Z}^{ij} \sim \mathcal{H} [m_{(1)}^{ij}, \dots, m_{(k)}^{ij}] = \mathcal{H} [m_{(1:k)}^{ij}]. \tag{6}$$

This histogram distribution is defined by the following probability distribution:

$$p(z^{ij} | m^{ij}) = \frac{1}{\sum_{k=1}^n m_{(k)}^{ij}} \begin{cases} m_{(1)}^{ij} & \text{if } 2^{-1} < z^{ij} \leq 2^0 \\ \vdots & \vdots \\ m_{(k)}^{ij} & \text{if } z^{ij} \leq 2^{-(k-1)}. \end{cases} \tag{7}$$

When the robot now starts to move (or to turn its head), the camera yaw angle  $\psi$  will change, and because the robot will then see a different part of the scene, we now add this angular dependency to all previously introduced variables like the observed feature vector  $\mathbf{z}(\psi)$ , the generating distribution  $\mathbf{Z}(\psi)$ , which is defined by the currently seen part of the map  $\mathbf{m}(\psi)$ . Note that this map is cylindrical and therefore has  $2\pi$  periodicity.

3.5. Mapping

During mapping, we need to estimate the generating distributions  $\mathbf{Z}(\psi)$  for all yaw angles  $\psi$  from the measurement series

$(\langle z_1, \psi_1 \rangle, \langle z_2, \psi_2 \rangle, \dots)$ . As initially no map is available, one solution to estimate  $\psi_t$  is to rely completely on camera odometry information from the robot. As soon as the first part of the map is initialized, this can already be used to support this potentially inaccurate camera yaw  $\psi$  (see Section 4). For the moment however, we can assume that the camera yaw  $\psi$  is known during learning.

The robot can learn the cylindrical map by estimating the parameters  $\{\hat{m}_{(1:k)}^{ij}(\psi)\}$  constituting the individual histograms, and finally of the complete estimate of the cylindrical map  $\hat{\mathbf{m}}$ .

Fortunately, histogram distributions are particularly computationally inexpensive to estimate. Their parameters  $\hat{m}_{(1:k)}^{ij}(\psi)$  are direct estimators, as they actually represent the absolute frequencies of the corresponding bin. Each time a measurement  $z^{ij}(\psi)$  falls into a certain bin  $k^*$ , its corresponding counter  $\hat{m}_{(k^*)}^{ij}$  of the cylindrical map can be increased by 1.

For a single direction, the evaluation complexity of this expression is linear in the number of histogram bins  $k$ , i.e.,  $p(z^{ij} | m^{ij}(\psi)) \in O(k)$ .

3.6. Localization

After the map has been filled in sufficiently, the robot can use it to estimate the yaw angle  $\psi$  of its camera. If the robot now extracts a feature vector  $\mathbf{z}$  with unknown  $\psi$ , it can compute the likelihood that this observation was made under a particular camera yaw angle  $\psi$  – given the cylindrical map  $\mathbf{m}$  – by computing the likelihood distribution over all  $\psi$ , i.e.,

$$p(\psi | \mathbf{z}, \mathbf{m}) = \frac{p(\mathbf{z} | \psi, \mathbf{m}) p(\psi)}{p(\mathbf{z})} \propto p(\mathbf{z} | \psi, \mathbf{m}), \tag{8}$$

using (7) and Bayes rule.

Given a feature vector  $\mathbf{z}$  extracted from a camera image, the robot can compute  $p(\psi | \mathbf{z}, \mathbf{m})$ —expressing the likelihood that the camera yaw angle was  $\psi$  while observing  $\mathbf{z}$ . Hereby, we assume that all feature vectors  $\mathbf{Z}^{ij}(\psi)$  are mutually independent. Clearly, this assumption is an approximation which is only valid as long as the selected color classes are sufficiently different. In this case, we obtain:

$$p(\mathbf{z} | \psi, \mathbf{m}) = \prod_{i,j \in \mathbb{C}} p(z^{ij} | \psi, m^{ij}). \tag{9}$$

For a one-shot localization, one could select the camera yaw  $\psi$  with the maximum likelihood by computing

$$\hat{\psi}_{ML} = \operatorname{argmax}_{\psi} p(\mathbf{z}_t | \psi, \mathbf{m}). \tag{10}$$

However, the likelihood distribution will potentially contain multiple (local) maxima, induced by ambiguity present in the environment. The same pattern could match with several parts of the map, like for example a series of windows on a building. Although a single transition pattern does not have to be unique, a sequence of transition patterns is likely to contain more information to resolve the ambiguity. Such a larger sequence of local transition patterns can already be extracted from a single image by sampling the field of view horizontally by using multiple scanlines at a certain scanning resolution  $\sigma$ , as is described in the next subsection.

The evaluation complexity of the likelihood distribution grows quadratically with the number of color classes  $n$  and linearly with the number of histogram bins  $k$ , i.e.,  $p(\mathbf{z}|\mathbf{m}) \in O(n^2 \cdot k)$ .

### 3.7. Multiple scanlines

Much of the ambiguity in the localization process can be removed when the robot scans along multiple scanlines through the image at yaw angles  $\psi_{\text{scan}} \in \Psi$ .

A horizontal scanning resolution of  $\sigma$  induces a set of vertical scanlines at yaw angles

$$\Psi_{\text{scan}} = [\psi - \alpha/2, \psi + \alpha/2] \cap (\sigma\mathbf{N}). \quad (11)$$

This implies that now – within the camera image  $F_t(\psi)$  – a sequence of  $\alpha/\sigma$  feature vectors  $\{\mathbf{z}_t(\psi_{\text{scan}})|\psi_{\text{scan}} \in \Psi_{\text{scan}}\}$  to be extracted. Evaluating a sequence rather than a single transition pattern greatly reduces the ambiguity during localization. If the measurements lie substantially far away from each other, these measurements can be assumed to be independent, leading to the following equation:

$$\begin{aligned} p(F_t|\psi, \mathbf{m}) &= p(\{\mathbf{z}_t(\psi_{\text{scan}})|\psi_{\text{scan}} \in \Psi_{\text{scan}}(\psi)\}|\psi_{\text{scan}}, \mathbf{m}) \quad (12) \\ &= \prod_{\psi_{\text{scan}} \in \Psi_{\text{scan}}(\psi)} p(\mathbf{z}_t(\psi_{\text{scan}})|\mathbf{m}). \quad (13) \end{aligned}$$

Given a limited field-of-view camera, the robot can only perceive a limited part of its surroundings from a single image. Active vision can be used to increase its perceptive field: if the robot has a turnable camera, it can scan its surroundings to remove ambiguities very quickly. Additionally, mobile robots can turn their whole body to see other parts of the scene, allowing them both to learn the full cylindrical map  $\mathbf{m}$  and to improve localization.

So far, from an algorithmical point of view, the evaluation complexity of (13) is reciprocal in the angular resolution  $\sigma$ , or in total, the computational cost is now in the order of magnitude of  $p(F_t|\psi, \mathbf{m}) \in O(n^2 \cdot \frac{1}{\sigma})$ .

### 3.8. Localization filter

Robot localization [15] aims at improving the estimation of the pose  $\mathbf{x}_t$  at time  $t$ , taking into account the movements of the robot ( $\mathbf{u}_1, \dots, \mathbf{u}_t$ ) and the observations of the environment taken by the robot ( $\mathbf{z}_1, \dots, \mathbf{z}_t$ ). This is typically modelled by a Markov process that goes through the sequence  $\mathbf{x}_0 \xrightarrow{\mathbf{u}_1} (\mathbf{x}_1, \mathbf{z}_1) \xrightarrow{\mathbf{u}_2} \dots \xrightarrow{\mathbf{u}_t} (\mathbf{x}_t, \mathbf{z}_t)$ . The Markov assumption states that the current robot's pose only depends on the previous state  $\mathbf{x}_{t-1}$ , the current action  $\mathbf{u}_t$ , and the current observation  $\mathbf{z}_t$ .

We model the robot's current belief about its pose by the belief distribution  $\text{bel}(\mathbf{x}_t)$ . Using the Bayes Filter [3], this belief distribution is updated in each time step according to

$$\overline{\text{bel}}(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}) \text{bel}(\mathbf{x}_t) d\mathbf{x}_{t-1} \quad (14)$$

$$\text{bel}(\mathbf{x}_t) = \eta p(\mathbf{z}_t|\mathbf{x}_t) \overline{\text{bel}}(\mathbf{x}_t), \quad (15)$$

where  $p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$  is called the *motion model* and  $p(\mathbf{z}_t|\mathbf{x}_t)$  is called the *sensor model*.

(14) and (15) define a recursive system for estimating the position of the robot. Both the continuous representation and the integral are difficult to implement and are therefore typically approximated, for example using a probability grid, a Kalman filter or Monte-Carlo sampling methods like particle filters.

In the following two subsections, the required motion and sensor models will be constructed. Subsequently, a localization filter that estimates the robot's heading using these models will be presented.

#### 3.8.1. Motion model

We assume that odometry feedback  $\mathbf{u}_t$  is available in each time step. This odometry can either be measured directly (from the wheels or legs), or can be derived from the issued action commands. We assume an additive noise component  $\xi \sim \mathcal{N}(\mathbf{0}, \Sigma)$  arising from slippage and other uncontrollable effects of the environment that is normally distributed.

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{u}_t + \xi_t \quad (16)$$

or equivalently, by splitting the pose  $\mathbf{x} = \langle \psi, \tau \rangle$  in its rotational  $\psi$  and translational  $\tau$  components

$$\langle \psi_t, \tau_t \rangle = \langle \psi_{t-1}, \tau_{t-1} \rangle + \langle \mathbf{u}_t^\psi, \mathbf{u}_t^\tau \rangle + \langle \xi_t^\psi, \xi_t^\tau \rangle \quad (17)$$

$$= \langle \psi_{t-1} + \mathbf{u}_t^\psi + \xi_t^\psi, \tau_{t-1} + \mathbf{u}_t^\tau + \xi_t^\tau \rangle. \quad (18)$$

#### 3.8.2. Sensor model

The posterior probability distribution that a compass sensor (given a cylindrical map  $\mathbf{m}$ ) estimates when supplied with a measurement  $\mathbf{z}_t$  is given in (19). This distribution can directly be used as the sensor model:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) = p(\mathbf{z}_t|\langle \psi, \tau \rangle, \mathbf{m}) \simeq p(\mathbf{z}_t|\psi, \mathbf{m}). \quad (19)$$

Note, here, that we make the assumption that the posterior is only depending on the camera yaw  $\psi$ , and not on the translational component  $\tau$ . In Section 3.10, this sensitivity on the translation is again incorporated. Here, we assume basically that the perceived color transition patterns originate from very distant objects, ideally at infinity, such that translational component of the camera can safely be neglected.

The posterior stated in (19) forms the sensor model of the visual compass, by relating the measure feature vectors  $\mathbf{z}_t$  to the world state  $\mathbf{x}_t$ , given a cylindrical map  $\mathbf{m}$ .

### 3.9. Orientational belief filter

Note that for robot localization, we use two different techniques: we will first explain how the robot can estimate its heading  $\text{bel}^\psi(\mathbf{x})$  over time, and then extend the method to allow us to estimate the translational component  $\text{bel}^\tau(\mathbf{x})$ .

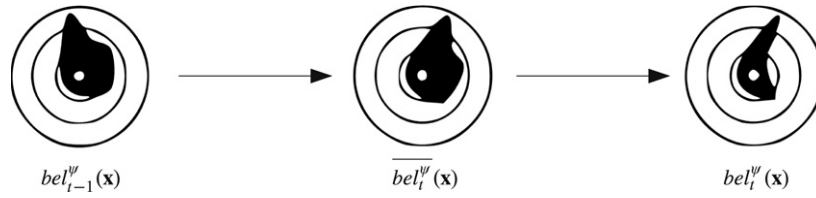
The orientational filter has to track the orientational belief  $\text{bel}^\psi(\mathbf{x})$  of the robot. As our visual compass only measures the 1D rotational yaw component of  $\mathbf{x}$ , it is sufficient to filter it on a 1D circular grid, at a resolution comparable to the resolution  $\sigma$  of the sensor measurements  $\mathbf{z}$ .

At each time step, the belief  $\text{bel}^\psi(\mathbf{x}_t|\mathbf{u}_t, \mathbf{z}_t, \mathbf{m})$  is updated according to the motion and sensor model, using the update rules from the Bayes filter and a mixing constant  $\lambda$ :

$$\overline{\text{bel}}_t^\psi(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}) \text{bel}^\psi(\mathbf{x}_t) d\mathbf{x}_{t-1} \quad (20)$$

$$\text{bel}_t^\psi(\mathbf{x}_t) = \lambda p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) + (1 - \lambda) \overline{\text{bel}}_t^\psi(\mathbf{x}_t|\mathbf{u}_t^\psi, \mathbf{z}_t, \mathbf{m}). \quad (21)$$

In Fig. 3, this filter process on a circular grid is sketched graphically.



**Fig. 3.** Visualization of the Bayesian filtering process. In the first step, the belief distribution  $\text{bel}_{t-1}^\psi(\mathbf{x})$  is updated according to the motion model (odometry correction). In the second step, the sensor model is applied and a new belief distribution  $\text{bel}_t^\psi(\mathbf{x})$  becomes available.

This belief distribution may in principle have any arbitrary shape. In order to extract an expedient single yaw estimate that the robot can use, for example, for planning and navigation, we approximate the belief distribution  $\text{bel}^\psi(\mathbf{x})$  with a Gaussian distribution centered at  $\hat{\mu}_t^\psi$  and with variance  $\hat{\sigma}_t^\psi$ , i.e.,

$$\text{bel}_t^\psi(\mathbf{x}) = \mathcal{N}(\hat{\mu}_t^\psi, \hat{\sigma}_t^\psi). \quad (22)$$

The evaluation complexity of both belief updates is reciprocally linear in the angular resolution  $\sigma$ , i.e.,  $\text{bel}_t^\psi(\mathbf{x}_t \parallel \mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{z}_t) \in O(1/\sigma)$ .

### 3.10. Four-spot sensor model

With the sensor model as constructed in (19), it is possible to estimate the robot's yaw angle  $\psi$  by learning the cylindrical map  $\hat{\mathbf{m}}$  only at a single training spot. Under the assumption that the transition frequency patterns are produced by objects at infinite distance, it is clearly not possible to estimate the translational component. However, if we weaken this assumption and assume that the perceived panorama has a reasonable distance to the robot, then the resulting projective distortions can be used to estimate additionally the robot's translation on the map. Imagine a room with some posters on the wall. Such a room would still allow a robot to robustly estimate of the robot's heading, while the matching quality expressed by the measurement likelihood will decrease substantially, the farther the robot moves away from its original training spot.

In order to additionally estimate the translational component of the robot's pose in two dimensions, more than one training spot is required: from now on, the global map is extended to consist of  $q$  spots  $m^{(1:q)} = (m^{(1)}, \dots, m^{(q)})^T$  learned at the training spots  $T^{(1)}, \dots, T^{(q)}$ .

Then, for each spot, the likelihood response of the sensor model (i.e., the maximal likelihood) can be used as an indicator for the success of the match between the current image with the training spot. We found in several experiments, that the likelihood response is strongly related to the distance of the robot from the training spot. In most indoor environments, the likelihood function even decreases strongly and monotonously. For an example see Fig. 8 (left) of measurements obtained in our robot lab.

A single goodness-of-fit indicator can thus be computed by extracting the maximal likelihood  $L_t^{(i)}$  from corresponding posterior distribution  $p(\mathbf{z}_t | \psi, \mathbf{m}^{(i)})$ :

$$L_t^{(i)} = \max_{\psi} p(\mathbf{z}_t | \psi, \mathbf{m}^{(i)}). \quad (23)$$

From the combined sensor readings  $\mathbf{z}_t^{(1:q)}$  consisting of the indicators of all  $q$  compass sensors

$$\mathbf{z}_t^{(1:q)} = \langle L_t^{(1)}, \dots, L_t^{(q)} \rangle, \quad (24)$$

we propose to compute the translation directly by computing the weighted sum of the training spots  $T^{(1)}, \dots, T^{(q)}$  according to the

corresponding likelihoods  $\mathbf{z}_t^{(1:q)}$ ,

$$\hat{\tau}(\mathbf{z}_t^{(1:q)}, m^{(1:q)}) = \frac{1}{\sum_{i=1}^q \hat{L}_t^i} \sum_{i=1}^q \hat{L}_t^i T^i. \quad (25)$$

For this translational interpolation technique to work, the likelihood of each individual training spot needs to drop linearly with increasing distance. For arbitrary environments, this approximation will not hold in general. From our indoor experiments however, we see that this assumption is justified in typical office and home environments. In such rooms, the reported likelihood will be high in the vicinity of the training spot, and drop the more the robot moves away, as projective distortions reduce the likelihood of the match increasingly. For a visualization of the likelihood surface, consult Fig. 8 in Section 5.

The evaluation complexity of the multi-spot sensor, additionally, is linear in the number of training spots  $q$  used, i.e.,  $\hat{\tau}(\mathbf{z}_t^{(1:q)}, m^{(1:q)}) \in O(q \cdot c^2 \cdot n \cdot \frac{1}{\Delta\phi_{\text{filter}}})$ .

### 3.11. Translational belief filter

The sensor model described in (25) can directly be used by a Bayesian filter. In each time step, the translational belief  $\text{bel}^\tau(\mathbf{x}_t)$  is updated first by the motion model as defined in (16) and subsequently by the sensor model as constructed in (25), using a mixing constant  $\lambda$ :

$$\overline{\text{bel}}_t^\tau(\mathbf{x}_t) = \int p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1}) \text{bel}^\tau(\mathbf{x}_t) d\mathbf{x}_{t-1} \quad (26)$$

$$\text{bel}_t^\tau(\mathbf{x}_t) = \lambda p(\mathbf{z}_t^{(1:q)} \parallel m^{(1:q)}) + (1 - \lambda) \overline{\text{bel}}_t^\tau(\mathbf{x}_t | \mathbf{u}_t^\tau, \mathbf{z}_t, \mathbf{m}). \quad (27)$$

The algorithmic complexity of the translational filter is again linear in the number  $q$  of acquired training spots, so  $\text{bel}_t^\tau(\mathbf{x}_t) \in O(q)$ .

At each time step  $t$ , the translational belief distribution  $\text{bel}_t^\tau(\tau_t)$  is assumed to be normally distributed with mean  $\mu_t^\tau$  and variance  $\sigma_t^\tau$ , i.e.,

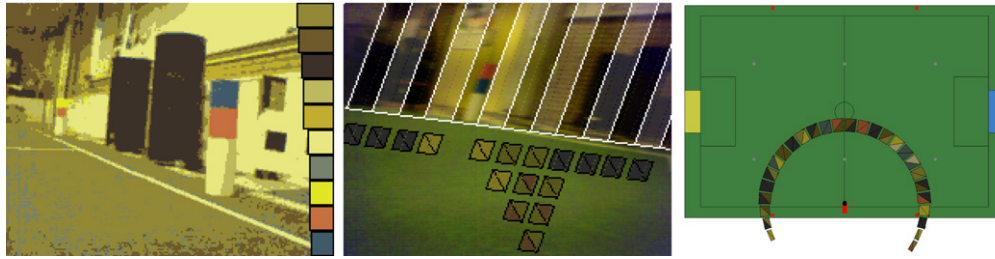
$$\text{bel}_t^\tau(\mathbf{x}^\tau) = \mathcal{N}(\hat{\mu}_t^\tau, \hat{\sigma}_t^\tau). \quad (28)$$

From both the rotational and the translational filter, a full pose estimate is now available for higher-level functions such as planning and navigation:

$$\hat{\mathbf{x}} = \langle \hat{\mathbf{x}}^\psi, \hat{\mathbf{x}}^\tau \rangle. \quad (29)$$

## 4. Implementation

The Sony Aibo is an entertainment robot used by many institutes as a robust 3-DOF legs, and an internal 64 bit MIPS processor running at 567 MHz. A particular problem of the CMOS camera in the robot's head is that it cannot be assumed that the scanning grid is orthogonal. The angle between the horizontal and the vertical depends on the velocity of the head movements because of the distortion introduced by the rolling shutter of the built-in CMOS camera, see Fig. 4.



**Fig. 4.** Visualization of the individual processing steps of our visual compass. **Left:** The RGB camera images from the robot's camera are discretized into  $m = 10$  color classes. **Middle:** The robot divides the image in vertical stripes of  $\sigma = 4.5^\circ$ , and extracts the transition frequency pattern of each stripe. The dominant color class transitions are displayed below each stripe. **Right:** Visualization of a cylindrical map learned at the UVA Robolab. Per stripe, only the most frequent color class transition is displayed.

#### 4.1. Sensor model

An angular resolution of  $\Delta\phi_{\text{sensor}} = 4.5^\circ$  was chosen for the cylindrical map and the sensor model. As the robot's head is attached to 3-DOF motorized neck, there is some variance in the control of the corresponding motors, such that smaller (optical) angular resolutions are not reasonable. The number of  $m = 10$  color classes was chosen, the equivalent to the number of (artificial) color classes on a RoboCup soccer field. Parameter studies revealed that this number can be reduced significantly while still yielding comparable results. Further, a scanning grid evaluates by default 25% of the pixels of each incoming camera image [16].

#### 4.2. Localization filters

The circular belief grid of the orientational filter operates at an angular resolution of  $\Delta\phi_{\text{filter}} = 1^\circ$ , and the mixing constant  $\lambda$  was chosen so that half of the belief mass is replaced after 0.3 s.

The translational filter uses a pattern of 4 training spots to estimate the robot's position. Due to the limited processing power of the Aibo, it was decided to evaluate per frame only a single sensor model  $p(f_i | \phi, \hat{m}^{(i)})$ , and to leave the estimates of the others as they were. Note that this still results in a update rate of 7.5 fps. The filter updates its belief distribution with the same mixing rate  $\lambda$  as the orientational filter.

#### 4.3. Behaviors

Several behaviors have been implemented as finite state machines that allow the robot to calibrate and learn natural environments autonomously.

When learning a new map, the robot trusts, initially, completely on its odometry. As, walking on four legs, it is extremely prone to slip, the Aibo learns the first part of the map only by turning its head. Thereby, the Aibo can already initialize more than half of the map. Subsequently, the robot has to turn its body on the spot. To increase stability, the robot already uses the partial map for estimating its orientation during turning. The learning behavior allows the robot to learn a full circle in 8 steps of  $45^\circ$ . By default, the learning time in each direction is 5 s. Note, that after the Aibo has turned for approximately  $180^\circ$ , one end of the partially learned map becomes again visible on the other side. This implicitly leads to a loop closure of the cylindrical map.

Additionally, two localization behaviors were created that can be used to measure and demonstrate the capabilities of the visual compass approach. The measurement behavior makes the Aibo look around for 5 s and stores its heading estimate for later evaluation. The homing behavior uses the localization estimate to guide the robot back to the center, while heading forward.

A simple button interface was implemented that allows the user to activate the most important functions. All computations were executed in real-time directly on-board the robot.

## 5. Results

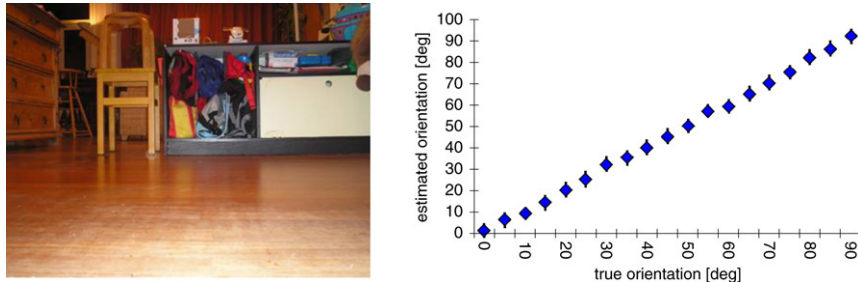
We conducted a large number of indoor and outdoor experiments to verify the validity of our approach. In Section 5.1, we will first analyze the performance of the compass filter in a large number of different environments, such as a living room, an outdoor soccer field and a lab room. It will be shown that the accuracy of the visual compass is very high when the robot stands at the initial training spot, and diminishes gracefully with increasing distance. In Section 5.2, we will then show that the likelihoods reported by the compass sensor can be used to estimate qualitatively, the distance from the training spots. This was experimentally verified by a visual homing experiment in our robot lab. Further experiments have been conducted to evaluate the robustness, validate its applicability in several natural environments, measure the actual computation times and evaluate the initial choice of parameters.

### 5.1. Orientational filter

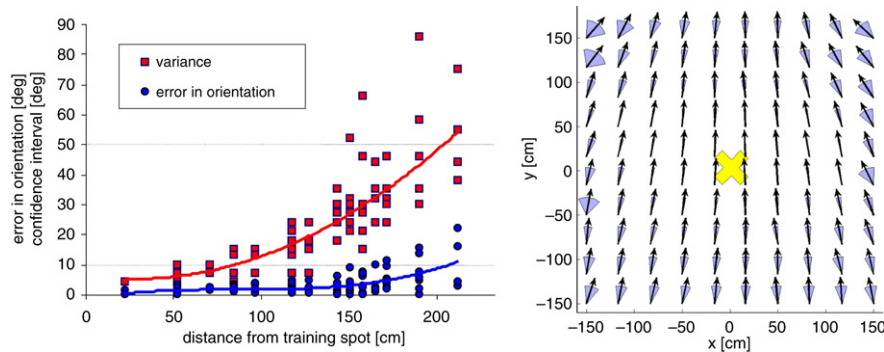
In a first experiment, a robot was trained in the center of an ordinary, but brightly lit living room as shown in Fig. 5 (left). After training was complete, the robot was turned manually in steps of  $5^\circ$ , and the robot recorded its estimated yaw angle. As it can be seen from the plot in Fig. 5 (right), the estimates match the real orientation of the robot very closely.

Then we conducted a more detailed experiment in our robot lab, a room of approximately  $7 \times 7$  m. On one side of the room, there are tables, chairs and computers. At the other side of the room there is an open space, with a RoboCup soccer field. We trained the robot in the middle of the field, and then moved it across a grid composed of  $10 \times 10$  cells with a side length of 33 cm per cell, in order to study the degradation in terms of localization accuracy. In Fig. 6 (left), the estimated angular error and variance are plotted as a function of the distance from the training spot. Close to the training spot, the error is less than  $5^\circ$ , but increases quickly with the growing distance from the training spot. After 2 m, the orientational error is already above  $45^\circ$ . Interestingly, there is a systematic drift in the heading estimates reported by our visual compass, as displayed in Fig. 6 (right), depending on the heading of the robot, which seems to be induced from the projective distortions in the room. Note that no explicit use is made of the colored landmarks that surround our RoboCup soccer field.

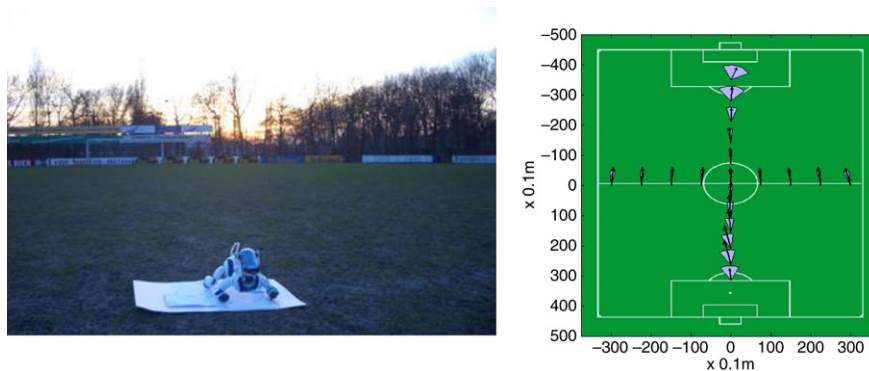
Finally, we repeated this experiment on a outdoor human soccer field of  $60 \times 100$  m, where we obtained similar results, see Fig. 7 (left). On the field, the robot could mainly see the border advertisements around the field, the podium for the audience and several larger trees. Note that also the sky (including a few clouds and the sun) was visible to the robot. As it can be seen from the evaluation in Fig. 7 (right), the robot shows (up to scale) a similar accuracy of compass estimates, compared to the indoor experiments.



**Fig. 5.** Experiment conducted in a ordinary, but brightly lit living room at a private home. The robot was rotated manually in steps of  $5^\circ$ . The robot's heading estimates have been plotted against the true orientation.



**Fig. 6.** Exhaustive compass experiment conducted at the UvA Robolab. **Left:** Quantitative results of orientation filter experiments. True error and variance plotted versus the distance from the training spot. **Right:** Detailed view on the reported heading estimates. Close to the training spot, the variance is relatively small, but increases steadily the further the robot is moved away. The projective distortions in the room induce a systematic error of the compass towards the center of the robot's orientation during training.



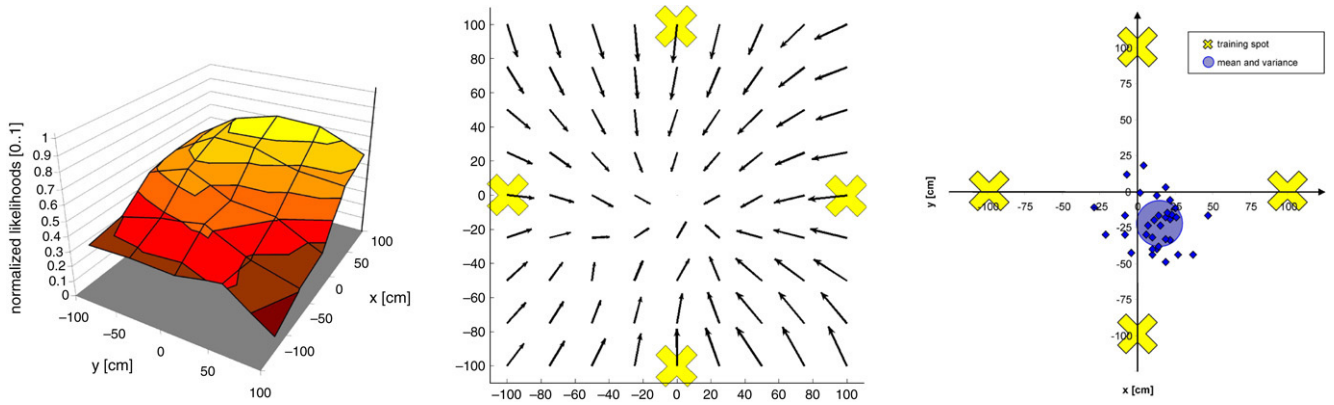
**Fig. 7.** Experiment conducted at the Zeeburgia outdoor soccer field in Watergraafsmeer, Amsterdam. The robot was moved across the whole field on its primary axes. The arrows display the reported heading, while the blue arcs around the arrow visualize the reported variance. The heading estimates point nicely in direction of the opponent's goal.

## 5.2. Translational filter

If the robot stands exactly at the training spot, the measured color class transition frequencies will perfectly match the stored ones in the map, thus yielding a high likelihood  $p(\mathbf{z}|\psi, \mathbf{m})$ . If, however, the robot is moved away, projective distortions and variations in illumination will lead to a different panoramic view and, thus, to reduced likelihoods. To verify this assumption, we trained the robot on a single spot, and measured the reported likelihoods when the robot was moved across a grid of  $2 \times 2$  m around the training spot. Fig. 8 (left) shows the results. It can be seen that the normalized log-likelihoods drop monotonously with increasing distance. This observation initially gave rise to the idea to use the 4 linearly independent training spots to estimate the robot's position in between, as described before in Section 3.10.

We then carried out the visual homing experiment<sup>1</sup>: The kickoff circle of the RoboCup field was chosen as the center spot, and the robot learned the panorama of four adjacent spots at the distance of 1 m from the center, autonomously. Although the robot walked to the training spots autonomously, using its learned cylindrical map as in its orientational filter to support odometry, the robot sometimes went a bit astray and then had to be aligned manually, before the cylindrical map learning on the next spot was started. Finally, the robot walked back to the target spot in order to perform a normalization of the individual likelihoods (so that unity likelihoods  $L_t^{(i)} = 1$  were reported from all compass sensors at the target spot).

<sup>1</sup> The video is available online: <http://www.informatik.uni-freiburg.de/~sturm/sturm08ras.html>.



**Fig. 8.** Experiments of translational filtering. **Left:** Likelihoods measured by the visual compass, while the robot is moved across a  $2 \times 2$  m grid. The robot was trained on spot (100, 0). The likelihood values drop almost monotonously with the distance from the training spot. **Middle:** The robot can use the reported likelihoods as a navigation function. The four training spots are marked with a yellow cross. **Right:** Measured end positions in a visual homing experiment. The blue dots correspond to the positions where the robot stopped.

After learning was complete, the robot was kidnapped and placed somewhere randomly on the field (with random orientation). From there, the robot had to walk back to the center using both the translational filter as a navigation function, as visualized in Fig. 8 (middle). We then recorded the position  $\tau_t$  where the robot stopped (for longer than 5 s). In total, 33 kidnappings have been executed and the reached positions have been measured, see Fig. 8 (right).

In most cases the robot stopped close to the target spot. However, it can also be seen that the robot walked (on average) to a spot slightly behind and to the right of the center spot. Finally, because it is not visible in the graph, it has to be stated that the robot never left the field or stopped on a point outside the displayed area.

We found, that the average position where the robot stopped was located at mean  $\tau_t = (-22 \text{ cm} + 12 \text{ cm})^T$  with a variance of  $\text{var } \tau_t = (17 \text{ cm } 15 \text{ cm})^T$ .

The systematic deviation to the bottom right can be interpreted in such a way that the likelihoods of the individual sensor models (or cylindrical maps) do not decrease equally fast; the likelihood normalization on only a single spot does not seem to be sufficient.

A standard deviation in this magnitude is acceptable when compared to other techniques. The Dutch Aibo Team [17] for example reports a standard deviation of 13.5 cm, making use of an explicit model of a RoboCup soccer field. It should be noted however, that the Dutch Aibo team could localize any arbitrary position on the field. With the translational filter, visual homing is only possible to a single learned spot.

In order to be able to get more than a qualitative estimate on the current position, even for spots other than the center, it would be necessary to gain more insight in the kind of shape of the likelihood decay when the robot is moved away from the training spot. Considering these results, it can be concluded that the visual compass approach can, in principle, be used for translational localization. Several aspects, however, remain subject to future research.

### 5.3. Further experiments

Several more experiments have been conducted that validated other important properties [16]:

- By modifying the illumination, it could be shown that the algorithm still performs well with only half of the illumination as during the learning, which is quite remarkable.

- Parameter studies revealed that three color classes are sufficient for accurate compassing. Moreover, the pixel coverage of the scanning grid could be reduced to less than 1% of the camera image without substantial impact on the prediction accuracy.
- With runtime experiments, the approach could be proven to be particularly efficient: with the original set of parameters, a single camera frame can be processed in less than 4 ms for learning and 13 ms for localization. This is fast enough to allow processing at the full frame rate of the Aibo, i.e., 30 fps.

## 6. Discussion and conclusions

The most important results of this work can be summarized as follows:

- The visual compass approach can supply a mobile robot with accurate heading information.
- Robots using visual compassing can learn quickly to orientate in new environments.
- Multiple learned spots together, can be used for navigation for visual homing.
- The visual compass algorithm is efficient and economic; it can be executed in real-time on board a mobile robot, and works well even with low-quality cameras.

Further experiments [18] have demonstrated that our approach is applicable in a vast variety of different environments and is robust to environmental changes.

The characteristics of the visual compass can be compared to several existing approaches. A natural comparison is the localization module as used in the RoboCup Soccer competition. Of course, these methods are based on completely different principles, relying on the recognition of a number of artificial landmarks that were placed around the field to facilitate localization. However, the resulting performance turns out to be comparable: UT Austin [19] did an extensive study on optimizing the localization on the field and reported final accuracies of comparable magnitude.

As the visual compass described in [10,11] only compares the latest two images, and therefore small errors in estimation add up over time, this makes their results difficult to compare.

Other approaches (such as [4]) estimate maps of landmarks. The resulting localization is more stable over time, but the necessity for unique feature extraction (like SIFT) is computationally expensive, resulting in frame rates of around 1 fps.

The approach of visual odometry from [6] is one of the fastest in this comparison; features are tracked as long as they are visible, and a global motion vector is estimated from all feature tracks that

enjoy the largest support. Although their implementation is quite accurate, it again suffers from long term drift, as no global map is constructed.

We conclude that our Visual Compass is a light-weight alternative to classical geometric localization. The Visual Compass can be used on almost any camera-based mobile robot due to its good scalability and extreme efficiency, and therefore has a wide field of potential application.

## References

- [1] R.A. Brooks, Elephants don't play chess, *Robotics and Autonomous Systems* 6 (1-2) (1990) 3-15.
- [2] I.J. Cox, Blanche—An experiment in guidance and navigation of an autonomous robot vehicle, *IEEE Transactions on Robotics and Automation* 7 (1991) 193-204.
- [3] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.
- [4] N. Karlsson, L. Goncalves, M.E. Munich, P. Pirjanian, The vSLAM algorithm for navigation in natural environments, *Korean Robotics Society Review* 2 (2005) 51-67.
- [5] M. Kaess, F. Dellaert, Visual SLAM with a multi-camera rig, Tech. Rep. GIT-GVU-06-06, Georgia Institute of Technology, Feb 2006.
- [6] D. Nistér, O. Naroditsky, J. Bergen, Visual odometry for ground vehicle applications, *Journal of Field Robotics* 3 (2006) 3-20.
- [7] A. Davison, Real-time simultaneous localisation and mapping with a single camera, in: *Proc. of the IEEE International Conference on Computer Vision*, vol. 2, 2003, pp. 1403-1410.
- [8] A. Davison, W. Mayol, D. Murray, Real-time localisation and mapping with wearable active vision, in: *Proc. IEEE International Symposium on Mixed and Augmented Reality*, IEEE Computer Society Press, 2003, pp. 18-27.
- [9] J.M.M. Montiel, A.J. Davison, A visual compass based on SLAM, in: *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, Orlando, FL, USA, 2006*, pp. 1917-1922.
- [10] F. Labrosse, Visual compass, in: *Proceedings of Towards Autonomous Robotic Systems*, University of Essex, Colchester, UK, 2004, pp. 85-92.
- [11] F. Labrosse, The visual compass: Performance and limitations of an appearance-based method, *Journal of Field Robotics* 23 (10) (2006) 913-941.
- [12] R.I. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, ISBN: 0521540518, 2004.
- [13] J.J. Verbeek, Mixture models for clustering and dimension reduction, Ph.D. thesis, Universiteit van Amsterdam, December 2004.
- [14] D. van Soest, M. de Greef, J. Sturm, A. Visser, Autonomous color learning in an artificial environment, in: *Proc. 18th Dutch-Belgian Artificial Intelligence Conference, BNAIC'06, Namur, Belgium, 2006*, pp. 299-306.
- [15] J.M. Porta, B.J.A. Kröse, Appearance-based concurrent map building and localization, *Robotics and Autonomous Systems* 54 (2) (2006) 159-164.
- [16] J. Sturm, An appearance-based visual compass for mobile robots, Master's thesis, Universiteit van Amsterdam, December 2006.
- [17] J. Sturm, A. Visser, N. Wijngaards, Dutch Aibo Team: Technical report RoboCup 2005, Tech. rep., Dutch Aibo Team, October 2005.
- [18] A. Visser, J. Sturm, F. Groen, Robot companion localization at home and in the office, in: *Proc. 18th Dutch-Belgian Artificial Intelligence Conference, BNAIC'06, Namur, Belgium, 2006*, pp. 347-354.
- [19] M. Sridharan, G. Kuhlmann, P. Stone, Practical vision-based Monte Carlo localization on a legged robot, in: *IEEE International Conference on Robotics and Automation, 2005*, pp. 3366-3371.



**J. Sturm** obtained his M.Sc. degree in Artificial Intelligence at the Universiteit van Amsterdam, the Netherlands. Since 2007, he is a Ph.D. student in the Lab for Autonomous Intelligent Systems at the University of Freiburg, Germany. His research interests include probabilistic techniques for localization and navigation, machine learning approaches in robotics including self-perception, model learning, and learning by imitation.



**A. Visser** has been a researcher at the Universiteit van Amsterdam since 1991. He participated in several national and international projects for the Informatics Institute, concerning developments in robotics and sensor systems. He is an active member of the RoboCup community, a member of the Dutch Committee and one of the Technical Committees. His current research interest is focused on exploration with multi-robot systems. To tackle the multi-agent decision problem in realistic settings, approximation models and algorithms have to be developed to keep the search for the solution tractable. The representation of the world and the observations of changes in the world play a key role in finding a robust solution. The methodology to share and fuse the observations coming from multiple sources over time is the challenge to be taken.