ViCTOriA: Visual Compass To Orientate Accurately

Patrick M. de Kok Georgios Methenitis Sander Nugteren

University of Amsterdam, P.O. Box 94216, 1090 GE Amsterdam

Abstract

In this paper a generalisation of the model-based visual compass is presented, which uses a grid to store features in angle bins instead of using a cylindrical map. This results in an easily updatable model, which can account for a changing environment. Image colors are discretized to an automatically generated color profile, and transitions between these classes within vertical lines are used as feature. To improve the map's recency and speed up discovery, communication can be exploited to share the color profile and map updates are shared among a network of agents, such as in the RoboCup Soccer Standard Platform League setting.

Experiments show how quickly the coverage of the model increases with a randomly walking robot, and also how the robot can obtain a reliable measurement of its own orientation using the model.

1 Introduction

Self-localisation is an important part of robotics. This can be done by, for example, using an external sensor, such as GPS. However, GPS is often not fine-grained enough and does not work indoors or in other environments where reception is bad. Relying on motor odometry is unsuccessful in many cases as well. A robot may move over slippery or rough terrain or it can bump into undetected obstables, which both influence the distance traversed with similar motor odometry. One way to solve this problem is to use visual information, such as optical flow [2] or localizing based on predefined landmarks [3]. To counter the problem of symmetrical maps in the last method, a compass can be used [1, 4, 5]. Such a method estimates the robot's heading by comparing the relative position of distant, automatically generated feature points in its camera image over time.

Visual compass methods can be divided in model-free and model-based methods. Model-free methods [1, 4] compute the relative heading based on features occuring in recent images. The accumulation of frame-to-frame motion errors are reduced heuristically. In specific situations an absolute heading can be computed, for instance, when the robot's pose is aligned in a known way with respect to another object.

A model-based visual compass, as presented by Sturm and Visser [5], first builds a cylindrical map of visual features and stores it on the robot, which can then be used to find its absolute heading heading. This map is usually made during an initialisation period by making a full turn, but is not updated afterwards. The error in the computed orientation increases when the robots moves away from the point where the cylindrical map has been recorded, unless the found features are at a near infinite distance from the robot.

We present a generalisation of the heading-only localisation method presented by Sturm and Visser, which can also incorporate new measurements and measurements from other robots into its own map. Because the map is updated during runtime, our method can account for slight changes in the environment. As it is a visual compass to orientate accurately, even when moved from the original point of initialisation and in a slightly dynamic environment, the method is named ViCTOriA.

This method is developed and used in the RoboCup Soccer Standard Platform League (SPL) robot soccer competition. The implementation runs on an Aldebaran Nao v4 humanoid robot. Anderson and Hengst explain that the Nao can only estimate its heading odometry using visual, monocular methods. We have chosen to extend a model-based visual compass, as we expect it will aid the already existing augmented Monte Carlo localisation module [3] in converging its particles with the correct heading after the body has fallen.

The remainder of this document is structured as follows. Section 2 explains the differences between model-free and model-based visual compasses. In section 3 the deviations of our compass with the one it is based on are presented; Sturm and Visser's approach is summarized in section 3.1, section 3.2 presents the storage structure of the features, the generation of features is described in section 3.3, the online phase is explained in section 3.4, and section 3.5 describes how observations can be shared in a multi-agent setting. Experimental results and a short discussion can be found in section 4. Future work is discussed in section 5, while the article is concluded in section 6.

2 Background

Both model-free and model-based visual compasses have their uses. Because they do not keep track of the environment for more than just a few frames, model-free visual compasses can be used in dynamic environments and when the robot needs to explore areas of unknown size. In general there is no initialisation phase. However, when a big change in heading occurs such that the features from previous frames are not observed anymore, for instance, due to drifting or the robot falling, model-free methods cannot estimate the change in heading with respect to the previous estimate reliably. Another issue is the accumulation of frame-to-frame motion error over time, although error reduction heuristics are proposed [1].

Typical model-based visual compasses do not handle dynamic environments well, as they do not update the cylindrical mapping, which is typically generated in an initialisation phase. When the robot moves to an area of which the features have not been observed in the initialisation phase, the model cannot estimate the orientation reliably. This may occur for instance in cases where the environment is dynamic, or when the robot has moved out of sight from the point of initialisation [5]. The direction from the initialisation point to the feature's physical location differs greatly from the direction from the current observation point to the feature's physical location. However, it can keep track of its heading, even after a big change in heading occurs such as after drifting or falling. Because it does not compute the heading based on previous estimates, the error does not increase over time in a static environment.

RoboCup Soccer SPL takes place in an area with a defined size and shape. In most positions and orientations, the robot is able to recognize the features recorded in a cylindrical map during an initialisation phase. Because the robots bump into each other and fall often, the visual compass method should be robust against big changes of heading inbetween frames. The compass should also cope with a dynamic environment; the crowd watching the game will move during a match. The generation of a map should be quick; only 45 seconds are allowed for all robots in the game to move from the sideline to their starting positions. After that, game time starts. Initialisation should either happen in these 45 seconds or during game time, which increases risk of losing the early advantage.

Our approach should fulfill all requirements. The first three are satisfied by applying a model-based visual compass. Our approach is novel in that it can generate multiple cylindrical maps, of which only one is initialized. The existing and empty maps are updated whenever the system has enough certainty about its location and its orientation. When querying the model, the non-empty compass corresponding with the nearest grid cell is consulted. Besides the orientation, the method also returns a confidence measure of the angle, based on the distance from the observation point and time since the feature is updated. This satisfies the fourth constraint on the project, to cope with a dynamic environment. The fifth constraint, quick initialisation, is solved by having two defense players generate an initial map amd share it among team members.

Because of the limited processing power on the Nao, feature detection and feature matching should be leightweight operations. Anderson and Hengst consider only a band of grey-scale pixels around the horizon

and propose an 1-dimensional version of SURF, which can be heavily optimized. Sturm and Visser extract features from a limited number of vertical scan lines. The pixels are discretized into $\|\mathbb{C}\| = n$ clusters of the most significant colors occuring in the images taken during the initialisation phase. A bigram model of color classes, containing the information how often a pixel from class c_1 follows a pixel from class c_2 with $c_1, c_2 \in \mathbb{C}$, per vertical scan line is used as feature.

3 Approach

Our approach is an extension on the work done by Sturm and Visser.

ViCTOriA is by design split in two phases, an offline phase and an online phase. The offline phase corresponds to the initialisation phase of Sturm and Visser where an initial map is built by having the robot make a turn of 360 degrees. A good point to build such a map is at the middle of the field, as the maximum distance from a robot on the field is minimized. Besides, most colors from the area outside the field will be observed, which will result in an appropriate discretized color model.

The online phase includes the query and update mechanisms. The robot will query the system about its orientation with features, and its location and previous orientation, and when the system is certain enough about the new orientation estimate, it will update the appropriate map with the observed features.

When the online phase starts, the robot will initially rely heavily on the map at the center of the field. While it is moving, the robot will add new features to its model and update existing features.

It is possible to start navigating without going through the offline phase, but a bigger error in orientation is expected. In the case without an offline phase, a constant number of frames should be reserved for the color clustering process or a predefined set of pixel values should be hard-coded, and new pixels should be classified by means of finding the nearest cluster centroid in color space.

In the remainder of this section, a summary of the original approach is presented first. Following that, the deviations of our approach are presented.

3.1 Original approach

The processing pipeline of Sturm and Visser's visual compass method can be described as follows. First, the robot gathers a set number of images while making a full turn. With a clustering method, such as the Expectation-Maximisation algorithm with a Gaussian mixture model [6], the robot autonomously clusters the colors into the $\|\mathbb{C}\| = n$ most significant color classes. Every ψ_{scan} degrees of the field of view, a vertical line is selected. The algorithm computes the frequencies of pixels of one color class followed by any other color class. This process has is depicted by figure 3. A discretized image Together with the orientation metadata it is stored in the cylindrical map.

When querying the model, the authors use a Bayesian method where they optimize the probability $p(F_t|\psi, \mathbf{m})$ of an ordered sequence of features F_t given the camera's yaw ψ and the cylindrical map $\mathbf{m}$, so as to optimize the data likelihood by choosing the correct angle.

They introduce a localisation filter, orientational belief filter, and a translational belief filter. Studying this part is outside of the scope of this project. The university's RoboCup SPL team's current framework already uses its own localisation method, based on an augmented Monte Carlo localisation module [3].

3.2 Collection of compasses

The proposed model fuses readings from several visual compasses, to reduce the error that arises by moving away from the initialisation point. The field is divided into a grid, and each grid has a number of angle bins. Each grid cell contains a feature buffer represented by the circle in the middle of each cell. For positions outside of the field, the cell that it is closer to the position is used. In this 3-dimensional object measurements are stored whenever it is applicable. The number of grid cells and the number of angle bins in each buffer are configurable. For the purposes of our experiment and considering the poor computational power of the

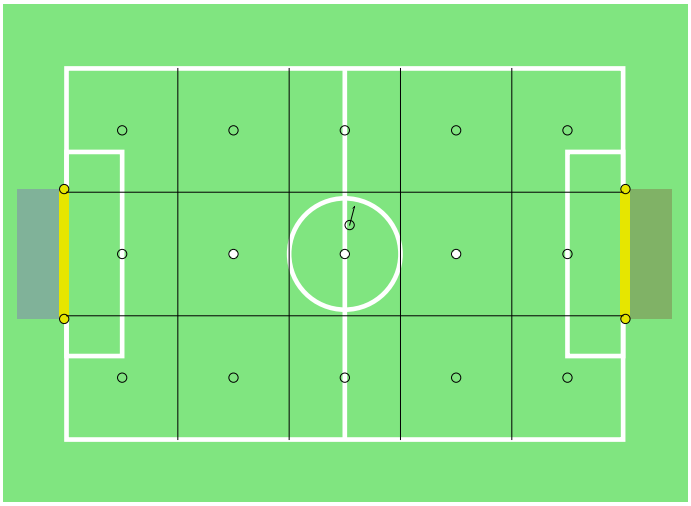

Figure 1: The field is divided up into a grid. Initially, each grid cell represents a model-based visual compass with an empty map. When the robot starts observing its environment, it will add features to the compass map of the grid cell it is in. When a feature is stored, a small dash in the orientation of the corresponding angle bin will be attached to the central circle in the robot's current cell. The robot is represented by a small circle and dash, which moves across the field.

Nao, ViCTOriA is configured with a 15-cell grid with 180 angle bins in each cell. Figure 3.2 illustrates this layout, together with a Nao's position and orientation.

3.3 Features

Not every scan line qualifies as a feature in our approach. The field, its lines and other robots do not contain interesting features to include in the map. Other robots might even harm the map's quality, as robots are extremely dynamic and thus their features will not be observed at the same position. Most of this noise occurs below the horizon, represented by a skew line in the image. If less than 60% pixels in a frame are above the horizon, the frame is considered not informative enough, and that frame is skipped. When the frame is informative enough, the scan lines will be drawn perpendicular to the horizon towards the top of the image. By doing so the orientation of scan lines is normalized. From every informative frame 10 scan lines are selected. As the camera has a horizontal field of view of 60.97 degrees, this corresponds with taking a scan line every 6.097 degrees.

Images are provided in YUV422 color space. Our implementation uses k-means clustering instead of the Expectation-Maximisation algorithm with a Gaussian mixture model, as an implementation of this is already provided in the SPL framework. Figure 2 is an example of a discretized image after k-means clustering with $k = 10$ has been performed on multiple images.

The feature extraction proceeds unchanged. However, the color transition matrices are normalized. A scan line of n pixels contains $n - 1$ transitions, and thus each value in the matrix is divided by $n - 1$.

3.4 Querying and updating the model

The model can be queried by using a collection of features obtained from one image. As they are also labeled with an angle and location, they can be looked up in the map. The equations used to compute the certainty of a match are as follows:

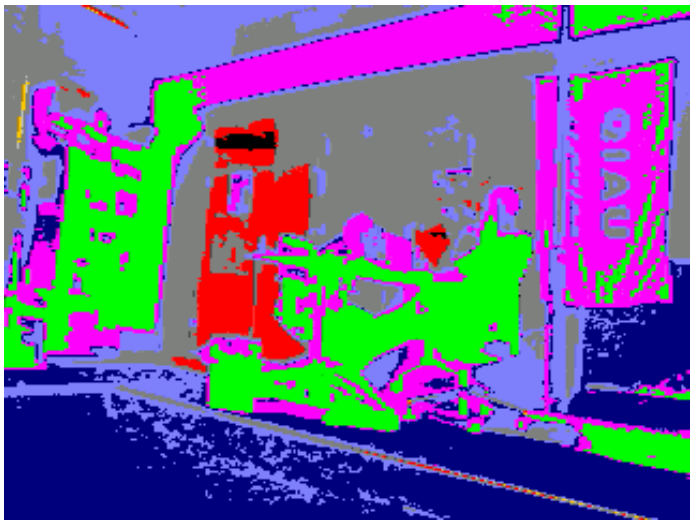

Figure 2: An image from the Intelligent Robotics Lab of the University of Amsterdam, discretized to 8 colors. The 8 most significant clusters in YUV422-space have been selected from multiple images by k-means clustering. Colors in the image do not represent the centroids, but are chosen for their distinguisability.

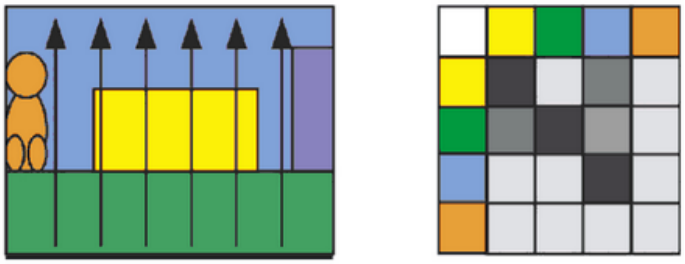

Figure 3: Color transition matrix extraction process. A discretized sample image is shown on the left, with six scan lines illustrated. On the right is a color transition matrix visualized. The left column and top row represent the color classes. The other cells represent how often the row color is followed by the column color in a scan line. Frequencies are visualized by shading; higher frequencies are dark colored. Image courtesy of Sturm and Visser [5].

$$sim(f_1, f_2) = \frac{1}{\mathit{diff}(l_{f_1}, l_{f_2})} \times \frac{1}{\mathit{diff}(\alpha_{f_1}, \alpha_{f_2})} \times \frac{1}{\mathit{diff}(f_1, f_2)} \times m(f_1) \times m(f_2)$$
$$m(f) = e^{(t-t_f)} \times p(l_f) \times p(\alpha_f)$$

The similarity measure between two features is computed using the distance between the two features $\mathit{diff}(l_{f_1}, l_{f_2})$, the difference in angle $\mathit{diff}(\alpha_{f_1}, \alpha_{f_2})$, the difference of the features themselves $\mathit{diff}(f_1, f_2)$ and the measurement certainties of both features $m(f_1)$ and $m(f_2)$. These measurement certainties are computed using the certainty of location $p(l_f)$ and angle $p(\alpha_f)$ at the time of measurement, and $e^{(t-t_f)}$, where t is the current time and t_f is the time of measurement. This ensures that the reliability of features over time will go down. The difference between each feature vector is computed, taking into consideration that most frequent transitions are not so informative. For example, assume a white wall with a red horizontal stripe of 1 pixel wide on it. The red stripe is more informative than the white wall. So, the color transition frequencies are

substracted from one. By doing so, the pixels of the red stripe are weighed more than the frequent ones. In this way, informative features play a more important role in the difference measure.

The search is started in the grid position corresponding to the location of the robot. If the obtained similarity is not high enough, the search will spread to neighbouring squares until a sufficiently good match can be found.

When a similar enough match has been found, the new features will be compared with the old features already present in the map, by computing $m(f_{new})$ and $m(f_{old})$. If $m(f_{new})$ is greater than $m(f_{old})$, the old feature is replaced by the new feature.

3.5 Multi-agent observations

The official RoboCup Soccer SPL field is 9×6 meters. One can imagine that it is not feasible for a single robot to play the game, train all its compasses, and afterward the training to have benefit from the method. To speed up the discovery of the field and keeping all visual compasses updated to the possibly dynamic environment, robots can communicate their features with corresponding metadata among each other through Wi-Fi. For this, the color models of the robots must be standardized. While all Naos use the same camera, there are differences in the registered colors. Ideally, all robots can perform an offline phase, where they send all images to a single unit to cluster the colors. This single unit then could return the centroids of the clusters to all robots. Because the color models are still closely related, the robots would have a similar color model in this discretized space.

However, the processing power of the Nao is limited and clustering so many pictures will take quite some time, which is not available during official games. Only images of a single robot will be used to determine the discretized color model. Color models from the different cameras seem similar enough in preliminary tests.

4 Experiments

The first test performed in order to evaluate our approach, the update mechanism of ViCTOriA was disabled. The robot was placed exactly in the center of a field of 6×4 meters[1] for learning the map and the significant colors for clustering. The grid was set to 15 cells with 180 angle bins per cell. To test the learnt model, the robot was placed in the same position and performed the same turn. However, now the information provided from the augmented Monte Carlo localisation module was not used, computing the orientation only by the output of the visual compass. The information provided by this other localisation module is used as as ground truth, which is accurate especially in the turn motion. The average error that the visual compass achieved was less than 0.2 degrees. 357 of 390 frames have been used. 33 frames have been unselected, as less than 60% of the observed pixels were above the horizon. Of course, this experiment has little value since the training set was very similar to the test set. However, this experiment gives an insight on how color transition are a very informative feature to orientate accurately.

Figures 4 and 5 illustrate the update process of a randomly walking robot. The walking lasted for three minutes, almost one third of a RoboCup Soccer SPL game period, which lasts ten minutes. Figure 4 demonstrates the trace of the robot poses during the walking as they came from the augmented Monte Carlo localisation module. The robot started with an empty feature map, but the update process of ViCTOriA updated almost 70% of the map in these 5 cells where the robot walked. This is shown in figure 5. It is expected that with communication between multiple agents and observation broadcasting, a big part of the map will be learned by the robots in only a few minutes of game play.

[1] This is the official RoboCup Soccer SPL 2012 field size. The current official field of 9×5 meters does not fit in the Intelligent Robotics Lab.

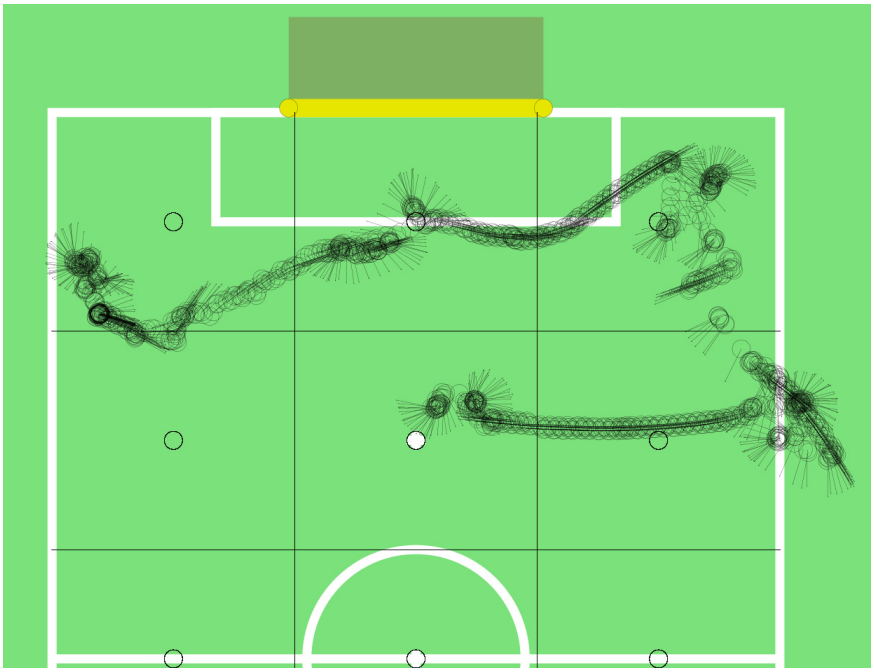

Figure 4: The robot walks randomly over the soccer field. Its path according to the already implemented augmented Monte Carlo localisation module [3] is depicted as a series of circles and dashes, representing its position and orientation, respectively. The orientations of the learnt features after walking this path are represented in figure 5.

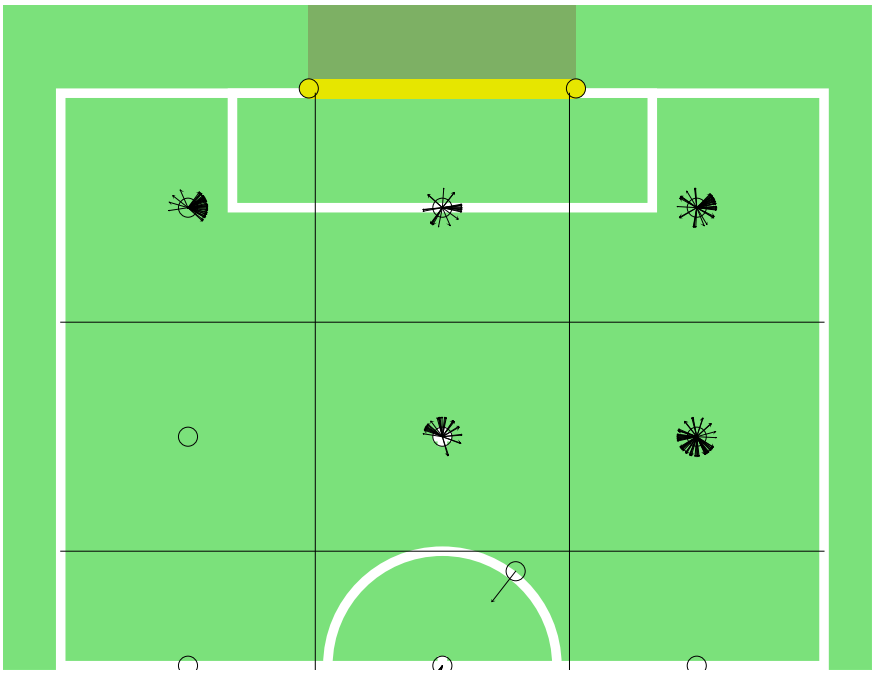

Figure 5: After walking the random path visualized in figure 4, the robot has learnt to associate certain features with orientations and locations. This is shown as a dash in the corresponding orientation at the center of the location's corresponding grid cell.

5 Future work

The current implementation uses color transition matrices as its features, but there are other alternatives, such as the 1D SURF used by Anderson and Hengst [1], or texture based features. Performance could be compared in lab environments and at RoboCup events.

Another point of improvement for an exploration scenario would be to perhaps change the grid cell size dynamically, or find a procedure to fuse grid cells that are similar. This is useful when the size of the environment is unknown, or when certain areas of the map might be less interesting than others. For example, a large open room with few different features could do with less grid cells than a room with many view obstructions.

Another extension of the grid-based approach, is to use a mixture of experts to weigh extimates from multiple compasses and come to a single believe.

Currently, we use the horizon to determine when an image frame is appropriate and contains enough information above the field to perform the extraction of scanlines. For soccer and other environments where the floor contains too few features for orientational localisation, a floor detector can be implemented, and all pixels above the floor polygon should be considered.

And finally, as the grid map that is constructed is essentially a storage of observations, this method could also be used with other localisation methods to improve the certainty of not only orientation but also location.

6 Conclusion

In this paper, a novel approach of the model-based visual compass is presented. It keeps track of several visual compasses in a grid. The method can be used in many applications, exploiting also the Wi-Fi com-

munication ability in multi-agent environment where all agents carry similar cameras.

Furthermore, the first experiment demonstrates that color transition matrices are good features to be used for localisation, as the robot could clearly distinguish between known orientations.

With the second experiment of the walking robot shows that the map can be built up in a relatively short time, using only one robot. When a team of multiple robots would be used, this would go even faster, and especially the important parts of the map, where the robots will frequently be located will get better updates. This also shows that the online updating of the model works, eliminating the need for an offline phase such as the ones needed by other model-based visual compasses.

References

[1] Peter Anderson and Bernhard Hengst. Fast monocular visual compass for a computationally limited robot. In *RoboCup Symposium*, 2012.

[2] Andrea Giachetti, Marco Campani, and Vincent Torre. The use of optical flow for road navigation. *Robotics and Automation, IEEE Transactions on*, 14(1):34–48, 1998.

[3] Amogh Gudi, Patrick de Kok, Georgios K Methenitis, and Nikolaas Steenbergen. Feature detection and localization for the RoboCup Soccer SPL. *Project report, Universiteit van Amsterdam (February 2013)*, 2013.

[4] Frédéric Labrosse. Visual compass. *Proceedings of Towards Autonomous Robotic Systems, University of Essex, Colchester, UK*, pages 85–92, 2004.

[5] Jürgen Sturm and Arnoud Visser. An appearance-based visual compass for mobile robots. *Robotics and Autonomous Systems*, 57(5):536–545, 2009.

[6] Jakob Verbeek. *Mixture models for clustering and dimension reduction*. PhD thesis, Universiteit van Amsterdam, 2004.