



## UvA-DARE (Digital Academic Repository)

### Orientation finding using a grid based visual compass

Methenitis, G.; de Kok, P.M.; Nugteren, S.; Visser, A.

**Publication date**

2013

**Document Version**

Author accepted manuscript

**Published in**

BNAIC

[Link to publication](#)

**Citation for published version (APA):**

Methenitis, G., de Kok, P. M., Nugteren, S., & Visser, A. (2013). Orientation finding using a grid based visual compass. *BNAIC*, 25, 128-135.  
[http://bnaic2013.tudelft.nl/proceedings/papers/paper\\_87.pdf](http://bnaic2013.tudelft.nl/proceedings/papers/paper_87.pdf)

**General rights**

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

**Disclaimer/Complaints regulations**

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

# Orientation finding using a grid based visual compass

Georgios Methenitis   Patrick M. de Kok   Sander Nugteren   Arnoud Visser

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

## Abstract

In this paper an extension of the model-based visual compass is presented, which can be updated continuously, allowing a robot to orient itself in a changing environment. To build a model, colors in the image are discretized to an automatically generated color profile, and transitions between these classes within vertical lines are used as feature.

Experiments show how well a model can be learned at the center of field and how this model can be extended to other location with a randomly walking robot. Finally, the strength of the approach is demonstrated in a dynamic environment, where a good estimate of the orientation is maintained while the surroundings are changed in a controlled way.

## 1 Introduction

Self-localization 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 obstacles, 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 visual 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 occurring 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. This map is usually made during an initialization 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 an extension of the heading-only localization 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 initialization and in a slightly dynamic environment, the method is named ViCTORiA.

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 and the online phase is explained in section 3.4. Experimental results and a short discussion can be found in section 4. Future work is discussed in section 5. Finally, section 6 serves as an epilogue to this article.

## 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 initialization 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 build up a global map, in contrast with the local map of the model-free approach. This global map can be used to estimate an absolute orientation, in contrast with a relative orientation of the model-free approach. In general model-based visual compasses do not handle dynamic environments well. The approach described in this study is an exception, because the map is constantly updated.

A global map is typically learned during an initialization phase at a certain central point in the environment. When the robot moves away from that initialization point, gradually the confidence in the orientation diminishes [5].

The RoboCup Soccer Standard Platform League (SPL) is an ideal testing environment (see figure 2) for such visual compass, because the soccer game 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 initialization phase. Because the robots bump into each other and fall quite often, the visual compass method should be robust against big changes in orientation, which requires a method which delivers an absolute estimate. 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. Initialization should either happen



Figure 1: A game of the Dutch Nao Team at the 2013 RoboCup competition. Note the colorful surroundings.

in these 45 seconds or during game time, which increases risk of losing the early advantage.

Because of the limited processing power on the NAO, feature detection and feature matching should be lightweight operations. Anderson et al. consider only a band of gray-scale pixels around the horizon and propose a 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 occurring in the images taken during the initialization 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 The ViCTORiA Approach

ViCTORiA is by design split in two phases, an offline phase and an online phase. The offline phase corresponds to the initialization 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. The robot ignores the field and only observes color patterns above the horizon. The colors outside the field (such as advertisement boards and audience) 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.

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

#### 3.1 Original approach

The ViCTORiA approach is an extension on the work done by Sturm and Visser. Note that that this study was performed in a comparable environment (a soccer field), but with another robot (the four legged AIBO robot). The processing pipeline of the original approach 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-Maximization algorithm with a Gaussian mixture model [7], the robot autonomously clusters the colors into the  $\|\mathbb{C}\| = n$  most significant color classes. Every  $\psi_{scan}$  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 is depicted by figure 3. A discretized image together with the orientation metadata 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  $\psi$  and the cylindrical map  $\mathbf{m}$ , so as to optimize the data likelihood by choosing the correct angle.

Sturm and Visser combined the orientational belief filter with a translational belief filter into a localization filter. This estimate could be combined with the location estimate based on landmarks, provided by a particle filter which was part of the team's framework [6]. The current framework has a localization method based on an augmented Monte Carlo localization module [3].

#### 3.2 Collection of compasses

To reduce the error that arises by moving away from the initialization point, the proposed model constantly builds up a model for several visual compasses, distributed over the field. 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 is closest 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 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.

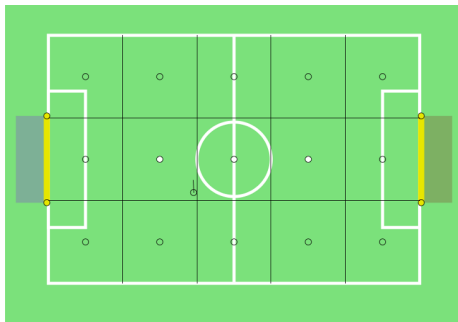


Figure 2: The field is divided up into a grid. Initially, each grid cell represents an empty visual compass. When the robot starts observing its environment, it will add features to the compass map of the grid cell it is in. When a feature is stored, a small dash in the orientation of the corresponding 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 (left from the center in this image).

### 3.3 Features

Not every scan line qualifies as a feature in the ViCTORiA 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 these observed 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% of the pixels in a frame are above the horizon, the frame is considered not informative enough, and that frame is skipped. Otherwise, when the frame contains enough information, 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-Maximization algorithm with a Gaussian mixture model, as an implementation of this is already provided in the SPL framework. Figure 3 shows an example of a discretized image after  $k$ -means clustering with  $k = 8$  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:

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

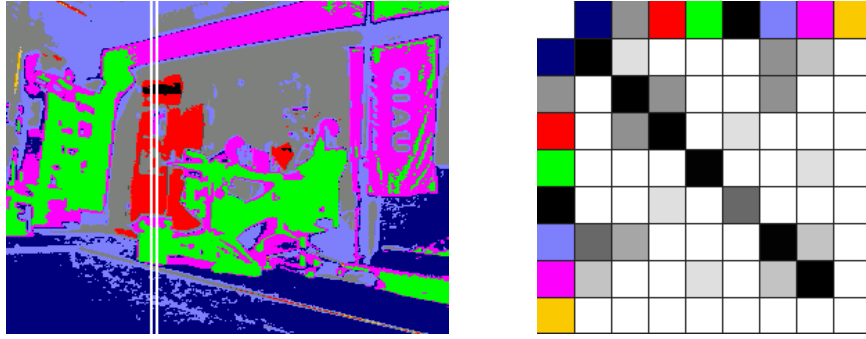


Figure 3: Color transition matrix extraction process. *Left*: A discretized sample image from the Intelligent Robotics Lab of the University of Amsterdam, discretized to 8 colors is shown, with one scan line illustrated between the white lines. *Right*: A visualization of a color transition matrix. The 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, gamma 0,25 is used to make differences visible enough.

The similarity measure between two features is computed using the distance between the two features  $diff(p_{f_1}, p_{f_2})$ , the difference in angle  $diff(\alpha_{f_1}, \alpha_{f_2})$ , the difference of the features themselves  $diff(f_1, f_2)$  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 subtracted 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.

As the used framework’s localization module is multimodal, certainty about the robot’s position can be expressed as the number of hypotheses, in a grid cell. Orientations are computed for grid cells containing at least 20% of the hypotheses. This threshold is chosen, so that when the robot is near a crossing of four cells, readings are still meaningful. This also allows for investigating the orientation when the localization module is unsure about the field half the robot is on.

When similar enough matches have been found, the new features will be compared with the old features already present in the cells, by computing  $m(f_{new})$  and  $m(f_{old})$ . If  $m(f_{new})$  is greater than  $m(f_{old})$ , the cell’s old feature is replaced by the new feature.

## 4 Experiments

The first experiment tests the accuracy of the initialization phase, so the update mechanism of ViCTOriA was disabled. The robot was placed exactly in the center of a field of  $6 \times 4$  meters 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 learned model, the robot was placed in the same position and performed the same turn. However, now the information provided from the augmented Monte Carlo localization module was not used, computing the orientation only by the output of the visual compass. The information provided by this other localization module is used as the ground truth, which is accurate especially in the turn motion. The average error that the visual compass achieved was less than 0.10 radians. 357 of 390 frames have been used. 33 frames have been skipped, as less than 60% of the observed pixels were above the horizon. In this experiment the training set was very similar to the test set. The experiment could be further extended by changing the circumstances.

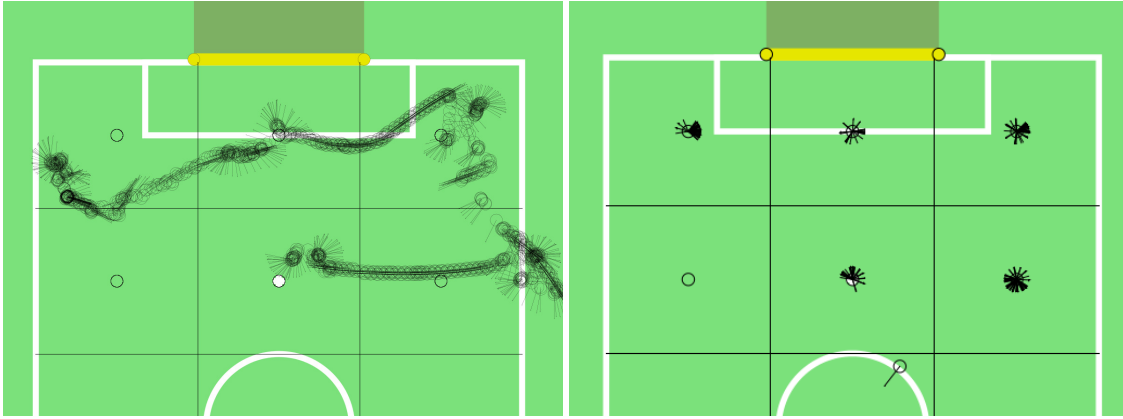


Figure 4: *Left:* The robot walks randomly over the soccer field. Its path according to the already implemented augmented Monte Carlo localization module [3] is depicted as a series of circles and dashes, representing its position and orientation, respectively. *Right:* After walking the random path, the robot has learned 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.

However, this experiment gives an insight on an upper boundary of the accuracy of the orientation estimate.

The second experiment consists of an initialization phase and an application phase. The experiment tests if during a soccer game enough information is collected to fill the visual compasses distributed over the field. Figure 4 illustrates 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. On the left, the trace of the robot poses during the walking are shown as they came from the augmented Monte Carlo localization 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 on the right of figure 4.

In a third experiment, ViCTORiA's robustness to changes in the environment is tested. Ten full turns at the center of the field have been recorded, where in each consecutive dataset an extra advertisement board is visible in the environment. These boards have been randomly positioned around the field to represent the random positioning of viewers during real games. No updates of the model have taken place. The model has been trained on the center spot with either two, six, nine or ten boards on the field side. The boards are placed within 50 cm of the field side. The average error in radians have been visualized in figure 5.

As can be expected, the average error is the lowest when tested on the same dataset as the model has been trained on. In general, datasets with a similar amount of boards have lower errors than datasets with a larger difference in the number of boards. So the method is quite insensitive for small changes in the environment. Yet, the worst average error occurred when the environment was changed quite drastically; the difference between 2 or 6 advertisement boards is a change of the appearance of half the horizon. The resulting worst average error of 0.41 radians is still a useful estimate within RoboCup Soccer; robots mainly need to distinguish their own goal from the opponent's (which has a threshold of 1.57 radians).

## 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 et al. [1], or texture based features. Performance could be compared in lab environments and at RoboCup events.

It is also possible to extend this framework in a multi-agent setting. Robots can benefit from each other's observations (features), just by broadcasting messages with features among them. In this way, communication can be used to achieve a faster adaptation to the environment (update the map in multiple locations at

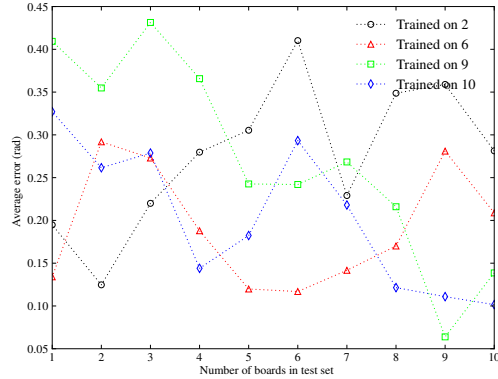


Figure 5: Average error in radians. ViCTORiA has been trained only on its center grid cell with either two, six, nine or ten advertisement boards on the field side, and tested with one to ten advertisement boards on the field side.

the same time). For this to work color transition independence between different cameras is a requirement. As it is illustrated in figure 6, the summed histograms for each of the three color channels of YUV422, show that the same color clustering method can be used with independently calibrated cameras with the same results.

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 obstructions in the field of view.

Currently, the horizon is used to determine when an image frame is appropriate and contains enough information above the field to perform the extraction of scan lines. For soccer and other environments where the floor contains too few features for orientational localization, 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 localization methods to improve the certainty of not only orientation but also location.

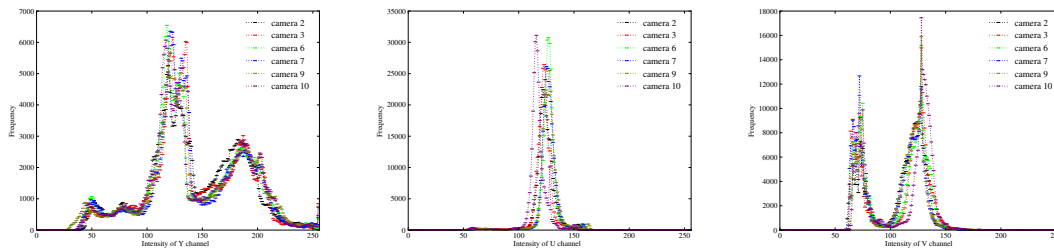


Figure 6: Frequencies of pixel values for the Y (left), U (center) and V (right) channels for four images taken with the lower camera on six different NAO heads. Preliminary experiments seem to indicate that the frequency patterns are shifted. This shift might be due to different the difference in automatically set white balance.

## 6 Conclusion

In this paper, a novel approach of the model-based visual compass is presented. It maintains several visual compasses in a grid. This map can be built up in a relatively short time, using only one robot. The method is robust against drastic changes in the environment, in the worst case resulting in an error of 0.41 radians, which is enough to discriminate in a soccer match between the own goal and the goal of the opponent.

## References

- [1] Peter Anderson, Yongki Yusmanthia, Bernhard Hengst, and Arcot Sowmya. Robot localisation using natural landmarks. In *RoboCup 2010: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes on Artificial Intelligence series*, pages 118–129. Springer, June 2013.
- [2] Andrea Giachetti, Marco Campani, and Vincent Torre. The use of optical flow for road navigation. *IEEE Transactions on Robotics and Automation*, 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.
- [4] Frédéric Labrosse. Visual compass. In *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] Jürgen Sturm, Arnoud Visser, and Niek Wijngaards. Dutch aibo team: Technical report robocup 2005. Technical Report, Dutch Aibo Team, October 2005.
- [7] Jakob Verbeek. *Mixture models for clustering and dimension reduction*. PhD thesis, Universiteit van Amsterdam, 2004.