A precise teach and repeat visual navigation system based on the convergence theorem

Filip Majer, Lucie Halodová, Tomáš Krajník

Abstract— **We present a simple teach-and-repeat visual navigation method robust to appearance changes induced by varying illumination and naturally-occurring environment changes. The method is computationally efficient, it does not require camera calibration and it can learn and autonomously traverse arbitrarily-shaped paths. During the teaching phase, where the robot is driven by a human operator, the robot stores its velocities and image features visible from its on-board camera. During autonomous navigation, the method does not perform explicit robot localisation in the 2d/3d space but it simply replays the velocities that it learned during a teaching phase, while correcting its heading relatively to the path based on its camera data. The experiments performed indicate that the proposed navigation system corrects position errors of the robot as it moves along the path. Therefore, the robot can repeatedly drive along the desired path, which was previously taught by the human operator. The presented system, which is based on a method that won the RoboTour challenge in 2009 and 2008, is provided as open source at `www.github.com/gestom/stroll_bearnav`.**

I. Introduction

A considerable progress in visual-based systems capable of autonomous navigation of long routes can be observed during the last decades.

According to [1], these systems can be divided in: map-less, map-based and map-building based. Map-less navigation systems such as [2] aim to recognise traversable structures in the environment (such as roads, pathways, highway lanes) and these to directly calculate motion commands for the robot or autonomous vehicle. Map-based navigation systems rely on environment models that are known a priori [3]. Map-building-based systems rely on maps as for localisation and navigation as well, but they are able to build these maps themselves. Some of these vision-based methods can build maps and localise the robot at the same time and are referred to as visual SLAM (Simultaneous Localisation and Mapping) [4], [5], [6]. Other map-building approaches perform mapping during a teleoperated drive and use the map for navigation later for autonomous navigation [7]. This technique (called 'map-and-replay') is analogous to a popular practice in industrial robotics, where an operator teaches a robot to perform some task simply by guiding its arm along the desired path.

Teach and repeat methods are described in several papers [8], [9], [10], [11], [12], which approach this problem In [10], a camera is moved through an environment while recording a video. The video is then processed (which takes several minutes) and the resulting map can be subsequently used to localise the robot and guide it along the same trajectory. Authors of paper [11] extract salient features from the video feed on-the-fly and associate these with different segments of the teleoperated path. When navigating a given segment, the robot moves forward and steers left or right based on the positions of the currently recognised and already mapped features. The segment end is detected by means of comparing the mapped segment's last image with the current view. The paper [13] mathematically proves that a robot navigating along a previously recorded polygonal route does not need explicit localisation. The mathematical proof is supported by several experiments, where a robot repeatedly traverses long paths in natural environments.

While the method [13] showed good performance in outdoor environments that were subject to illumination variations and naturally-occurring changes, it had a few drawbacks. First, it could be taught only polygonal paths in a turn-move manner, so even a slight change of the movement direction during the mapping phase required to stop the robot, which made the training tedious. While navigating, the robot moved in a constant velocity, which did not respect environmental constraints (such as doors, narrow passages and people). Moreover, in cases of adverse lighting conditions, when the images did not contain enough features, the heading estimation could cause the robot to go off the taught path. Finally, the method was not available as an open-source package.

In this paper, we present a simple monocular 'map-and-replay' navigation method for an autonomous vehicle based on the *convergence theorem* proposed in [13]. The core of the system is an efficient vision-based method that does not explicitly estimate the robot position, but it only corrects the robot heading. During navigation, these corrections are superimposed over forward and angular velocity values that define the robot trajectory – these are stored during the mapping step. This makes the method robust to poor visibility conditions that commonly occur in outdoor environments.

Our experiments demonstrate that the robot's navigation fulfils the 'Closed path stability property' [13], causing the robot to converge to the taught trajectory even when it starts with a large position error. The experiments confirm that extending [13] to smooth path requires the robot to remember and reuse its velocity profile along the taught trajectory – without this information, the robot diverges off its path. Furthermore, the experiments show that the proposed system does not fail even the robot temporarily loses the ability to

Artificial Intelligence Center, Czech Technical University `majerfil@fel.cvut.cz`

The work has been supported by the Czech Science Foundation projects 17-27006Y and 15-09600Y

correct its heading from the visual information due to poor illumination conditions.

II. Navigation method description

The navigation system works in two steps: learn and repeat. During the learning phase, a robot is guided by an operator along a path, which is the robot supposed to autonomously navigate in the repeat phase. When learning, the robot extracts salient features from its onboard camera image and stores its current travelled distance and velocity. During autonomous navigation, the robot sets its velocity according to the travelled distance and compares the currently detected and previously mapped features to correct its heading.

The feature extraction method which detects salient objects in the robot's onboard camera image is a critical component of the navigation system because it is the only mechanism which the robot employs to reduce its position uncertainty. We have decided to use Speeded Up Robust Features (SURF) [14], [15] to identify features in the image. However, the system has been tested with other feature extraction algorithms as well [16].

A. Image processing

The SURF feature extractor is composed of two steps, detection of keypoints and description of their vicinity. The keypoint detection is based on Hessian matrix calculation, which indicates points in the image, which have sufficient contrast that makes them easy to localise and track. The description is based on image intensity gradient near the detected keypoints. Once the keypoints are detected and described, they can be matched to the keypoints stored in a map and the associations can be used to correct the robot heading. The quality of the features depends on the quality of the input image stream, which, in outdoor environments, suffers from varying illumination. To compensate the illumination instability, we select the exposure and brightness of the robot camera based on the results from [17]

B. Learning (mapping) phase

.

During this phase, the robot is driven through the environment by a human operator. The robot continuously measures the distance it travelled and whenever the operator changes the forward or angular velocity, the robot saves the current distance and the updated velocity values – we refer to this sequence as to a 'path profile'. Additionally, the robot continuously extracts image features from its onboard camera image and every 0.2 m, it saves the currently detected image features in a local map, which is indexed by the current distance the robot travelled.

C. Navigation phase

At the start of this phase, the robot loads the path profile and it creates a distance-indexed list of the local maps containing the image features. Then, it sets its forward and angular velocity according to the first entry of the path profile and it loads the first local map containing data about image features visible at the start of the path. As the robot moves forwards, it extracts image features from its onboard camera image and matches them to the ones loaded from the local map. The differences of the horizontal image coordinates of the matched features (i.e. the positions of the features in the camera image relative to the positions of the features in the pre-loaded map) are processed by a histogram voting method. The maximum of the histogram indicates the most frequent difference in the horizontal positions of the features, which corresponds to the shift of the image that was acquired during the mapping phase relative to the image that is currently visible from the onboard camera. This difference is then used to calculate a corrective steering speed, which is added to the speed from the velocity profile. If the histogram voting results in inconclusive results due to the low number of features extracted, e.g. when the robot faces a featureless wall, the camera image is over- or under-exposed, etc., the corrective angular speed is not added to the one from the velocity profile. Thus, in case the visual information is not sufficient to determine the heading, the robot simply steers according to the path profile data. As the robot proceeds forwards along the taught path, it continuously monitors its travelled distance, loads local maps and path profile data that correspond to this distance and repeats the steps described above. Thus, the path profile allows the robot to steer approximately in the same way as during the teaching phase, and the image matching corrects the robot heading whenever it deviates from the intended path.

D. System implementation

The navigation system was implemented in Robotic Operating System (Ros), version Kinetic. The system structure is shown in Figure 1. The *feature extraction* node extracts image features from the robot camera and passes them to the *mapping* and *navigator* nodes. The *distance monitor* node receives data from robot odometry and measures travelled distance, also sends special messages every time the robot passes a given distance, which is used by the *mapping node*, see Section II-B. The *mapper* node receives features from the *feature extraction* node and saves them into local map when it receives the aforementioned message from the *distance monitor* node. It also saves the path profile. The *map preprocessor* node loads all local maps and path profile, and then sends them to the *navigator* node based on the travelled distance received from the *distance monitor*. The *navigator* node receives the velocity profile and local feature maps and it matches the features from the maps to the currently visible features from the *feature extraction* node. It performs the histogram voting described in Section II-C, calculates the robot velocity and steers the robot along the path.

All the aforementioned modules were implemented as ROS action servers with dynamically reconfigurable parameters, so the robot operator can change their parameters during runtime or activate and deactivate the modules in case they are not necessary to run during the teaching or replay phase. Action servers also provide the operator with feedback, that shows the current state of the system. Thus, the operator

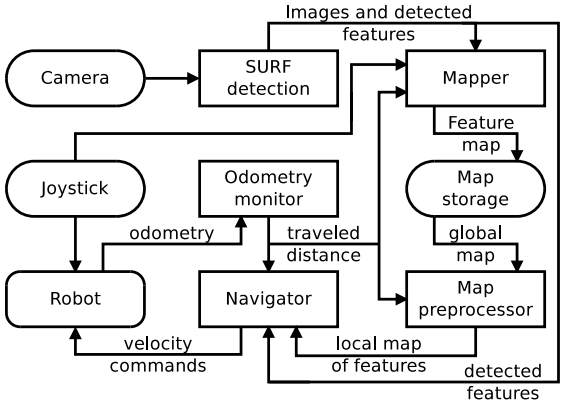

Fig. 1. Software structure of the presented system

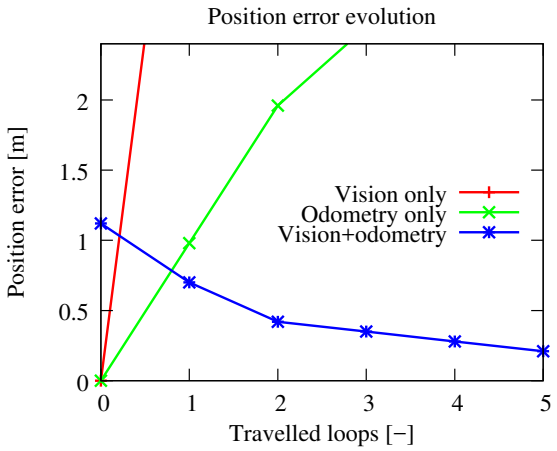

Fig. 2. Robot position error evolution as it travels

can see the currently used map, path profile data, number of visible image features, results of the histogram voting method etc. For future work, we would like to extend the system to work as an executor.

All the aforementioned modules are available as C++ open source code at [18].

III. Experimental evaluation

To evaluate the system's ability to repeat the taught path and to correct position errors that might arise during the navigation, we have taught it a closed, approximately 25 m long path in outdoor environment. Then, we let the robot to drive along the taught path repeatedly. Every time the robot completed a path loop, we measured its distance relative to the path end/start. In this way, we could quantitatively assess the robot's ability to adhere to the path it has been taught. The experiments were performed during the evening, so the lighting conditions changed from high to low illumination, which made the image-based navigation particularly difficult.

During the first experiment, we have let the robot run with the method described in [13]. Thus, the robot did not use the path profile data, but it moved forward with a constant speed and steered its heading according to the results of the image feature matching and histogram voting. The results of this experiment, shown in Figure 2 (as 'Vision-only') indicate, that it quickly diverged from the taught path as soon as it was supposed to perform a sharper turn.

In the second experiment, we have deactivated the vision-based heading corrections and we let the robot move according to the path profile only. The Figure 2 shows (in the 'Odometry-only' plot) that while it could complete the path with some success, it slowly diverged from the taught path. This means that without the visual feedback, the robot cannot traverse the path repeatedly.

The final experiment used both path profile and visual feedback as described in Section II. To verify if the robot can correct position errors that arise during navigation, we have started the autonomous navigation, not at the path start, but 1.2 m away. The Figure 2 (Vision+odometry) shows, that each time the robot completed the taught path, its position error decreased, which means that it could traverse the taught path repeatedly.

IV. Conclusion

A teach-and-repeat system based on image processing and odometry data was presented in this paper. During the learning phase, the robot stores its forward and angular speeds along with the local features, it extracted from its onboard camera image. Both velocity and image feature data are indexed by the distance the robot travelled from the path start. In the repeat phase, the robot navigates by replaying the stored velocities while correcting its heading from the visual data. The heading correction is performed by a histogram voting scheme over a set of horizontal position differences of previously mapped and currently detected image features. The experimental results show that the method is able to guide a mobile robot along a taught path in difficult illumination conditions. The presented system, which is based on an algorithm that won the RoboTour challenges in 2009 and 2008, is provided as open source at [18].

References

[1] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

[2] P. De Cristóforis, M. Nitsche, and T. Krajník, "Real-time image-based autonomous robot navigation method for unstructured outdoor roads," *Journal of Real Time Image Processing*, 2013.

[3] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *CVGIP: Image understanding*, vol. 56, no. 3, pp. 271–329, 1992.

[4] S. Holmes, G. Klein, and D. W. Murray, "A Square Root Unscented Kalman Filter for visual monoSLAM," in *International Conference on Robotics and Automation (ICRA)*, 2008, pp. 3710–3716.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[7] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human-guided experience," in *Proceedings of 6th International Conference on Intelligent Autonomous Systems*, 2000.

[8] G. Blanc, Y. Mezouar, and P. Martinet, "Indoor navigation of a wheeled mobile robot along visual routes," in *International Conference on Robotics and Automation*. IEEE, 2005, pp. 3354–3359.

[9] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, USA, 1996, pp. 83–88.

[10] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, Sep 2007.

[11] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics and Automation*, 2009.

[12] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "Large scale vision based navigation without an accurate global reconstruction," in *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR'07*, Minneapolis, Minnesota, 2007, pp. 1–8.

[13] T. Krajník, J. Faigl, V. Vonásek et al., "Simple, yet Stable Bearing-only Navigation," *Journal of Field Robotics*, 2010.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, 2008.

[15] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.

[16] P. De Cristóforis, "Vision-based mobile robot system for monocular navigation in indoor/outdoor environments," Ph.D. dissertation, University of Buenos Aires, Buenos Aires, 2012.

[17] L. Halodová and T. Krajník, "Exposure setting for visual navigation of mobile robots," in *Student Conference on Planning in Artificial Intelligence and Robotics (PAIR)*, 2017, [In review].

[18] F. Majer, L. Halodová, and T. Krajník, "Implementation: Bearing-only navigation." [Online]. Available: https://github.com/gestom/stroll_bearnav