












A Versatile Visual Navigation System for Autonomous Vehicles

Filip Majer¹, Lucie Halodová¹, Tomáš Vintr¹, Martin Dlouhý²,
Lukáš Merenda³, Jaime Pulido Fentanes⁴, David Portugal⁵,
Micael Couceiro⁵, and Tomáš Krajník¹

¹ Artificial Intelligence Center, Czech Technical University, Prague, Czech Republic
{majerfil,tomas.krajnik}@fel.cvut.cz

² Czech University of Life Sciences Prague, Prague, Czech Republic

³ VOP CZ, Šenov u Nového Jičína, Czech Republic

⁴ Lincoln Center for Autonomous Systems, University of Lincoln, Lincoln, UK

⁵ Ingeniarius, Ltd., Coimbra, Portugal

Abstract. We present a universal visual navigation method which allows a vehicle to autonomously repeat paths previously taught by a human operator. The method is computationally efficient and does not require camera calibration. It can learn and autonomously traverse arbitrarily shaped paths and is robust to appearance changes induced by varying outdoor illumination and naturally-occurring environment changes. The method does not perform explicit position estimation in the 2d/3d space, but it relies on a novel mathematical theorem, which allows fusing exteroceptive and interoceptive sensory data in a way that ensures navigation accuracy and reliability. The experiments performed indicate that the proposed navigation method can accurately guide different autonomous vehicles along the desired path. The presented system, which was already deployed in patrolling scenarios, is provided as open source at www.github.com/gestom/stroll_bearnav.

1 Introduction

Considerable progress in sensor development, machine perception and scene understanding along with increasing computational power of embedded devices had a significant impact on the development of unmanned vehicles. Autonomous cars, which are able to navigate in structured environments of highways and roads and react appropriately to standard traffic situations, are already on the market. While these cars are using active sensors like radars and lidars for obstacle detection and avoidance, their ability to understand the environment is based on their capability to accurately interpret imagery from their on-board cameras.

The work has been supported by the Czech Science Foundation project 17-27006Y and by the Segurancas roboTicos coOPERativos (STOP) research project (CENTRO-01-0247-FEDER-017562), co-funded by the Agencia Nacional de Inovacao within the Portugal2020 programme.

Thus, computer vision methods constitute the core of the autonomous navigation systems of these vehicles. One of the main advantages of the cameras is their small size and consumption, which makes them ideal for deployment on small robots with limited payload. Many of these robots cannot carry expensive and bulky sensors like 3d or 2d lidars, and they have to rely solely on vision systems.

The paper [11] divides the vision systems for mobile robotics into map-less, map-based and map-building based. Map-less systems recognise traversable structures in the environment (such as roads or highway lanes) and use this information to calculate motion commands for autonomous vehicles [10]. These systems have reached a considerable maturity, and they have already demonstrated their ability to be deployed in real cars quite some time ago [36,37]. However, not all environments have a clear, easy-to-recognise structure such as roads and highways, which clearly define the directions for driving. To be able to navigate to desired locations in environments without a straightforward structure, robots use environment representations (maps) which, with the help of on-board sensors, allow them to estimate their positions. According to [11], navigation systems which require that these maps are provided apriori are called “map-based” systems. An example of such system is a method that uses a CAD-like model of a building to allow navigation in indoor environment [19]. Since an accurate model of the operational environment is often not available, Behzadian et al. [3] propose a localisation scheme, which can deal with coarse, hand drawn maps. Another way to deal with the lack of prior maps is to endow the robots with an ability to build these maps themselves – these systems are referred to as “map-building” ones. Some of these systems employ methods commonly called visual SLAM (Simultaneous Localisation and Mapping) [12,18,29], which are able to build maps and localise the robots at the same time. The maps built by these systems represent the space in a metric way, and they can be passed to the localisation and planning submodules of the mobile robots, which then attain the ability to move across all of the mapped space. Another class of map-building methods do not represent the mapped space metrically or in a globally consistent way. These systems typically cannot move across all of the mapped space, but are constrained to the vicinity of the path they have been taught during the mapping. While somewhat limited in versatility, these techniques (called ‘map-and-replay’) have demonstrated their ability to support long-term operation of mobile robots [8,24,31,35]. Furthermore, they do not require skilled personnel to be set up – one has to simply guide a robot along a desired path by means of teleoperation.

Several researchers have proposed different schemes of visual teach-and-replay navigation [5,7,28,33,34]. Some of the systems build metric, local maps, which do not have to be globally consistent. For example [14] proposed to use traditional SLAM techniques to build a series of overlapping local maps and switch between them on-the fly. Similarly [33] build a map off-line from the camera feed gathered during a human-guided drive and use it later on for localisation. Other methods use even simpler schemes for teach-and-replay, which rely more on visual servoing rather than metric maps, e.g. [5,28] create a visual

memory path by simply storing the images the robot has encountered during a teleoperated drive. Similarly, [34] represents the taught path as a sequence of locations, associated with a set of salient visual features. To move between these locations, a robot can use a relatively simple feature association and visual servoing. Another, even simpler method is proposed by [7], who extract salient features from the onboard camera and associate these with different segments of the taught path. When navigating a given segment autonomously, their robot moves forward and steers according to the relative positions of the currently recognised and already mapped features. A segment end is detected by means of comparing its last mapped image with the current view. The paper [35] mathematically and experimentally proves that a robot navigating along a previously recorded polygonal route does not need a globally consistent map or explicit position estimation. Later work, described in [22], extends the mathematical proof provided in [35], which allows deployment of the navigation to a wider class of mobile robots.

Many of the teach-and-repeat systems are reported to be robust enough to deal with realistic outdoor conditions and environment changes, which makes them suitable for long-term deployment of autonomous robots. The robustness of these systems to environment variations is often improved by their ability to adapt to the environment changes during the course of robot deployment, i.e. when the robot repeatedly traverses the desired route. For example, [17, 21] propose to use trainable image feature descriptors and [15, 31] employ the dynamic maps [4, 8, 9] to adapt the environment models and perception methods to the nature of the changes observed. A paper by Halodová et al. demonstrates that the employment of adaptive methods throughout the visual teach-and-repeat navigation pipeline [16] significantly improves the accuracy and robustness of mobile robot navigation.

Inspired by the robustness of the systems described in [14, 31] and the simplicity of the method proposed in [35], we implemented a versatile visual navigation system with provable navigation stability. The main advantage of the system is that it does not require precisely calibrated, high-resolution cameras or accurate odometry, but it works with off-the-shelf equipment. This not only makes the system low-cost, but it also significantly reduces the time required for its integration into a given robotic platform. Furthermore, given certain conditions (described in [22, 35]) the system ensures that the navigated vehicles do not deviate from the taught trajectory even in low-visibility conditions. However, the system is based on a theoretical model [22], which is quite coarse and it neglects the type of the robot kinematics. The simplicity of the model and proof presented in [22] causes doubts regarding the system's ability to be deployed to various classes of autonomous vehicles in diverse environments and the soundness of the aforementioned proof. Moreover, the experiments presented in [22] were constrained in both environment size and number of platforms and the experimental evidence provided in [22] was not compelling for audience familiar with field robotics.

Thus, in this paper, we deploy the visual navigation system proposed in [22] to six different mobile vehicles with different type of kinematics and perform

experiments evaluating the correctness of the mathematical proof presented in [22]. We do not focus on the theoretical model and mathematical proof and aim to provide an engineer’s perspective of the system described. Apart from the experimental evaluation of the system navigation accuracy, we provide details regarding the time required to deploy the system on various platforms. We hope that the experiments, which demonstrate that the visual navigation can guide toy-like robots as well as military UGVs, will be convincing enough, so that the validity of the proof presented in [22] will be accepted with more confidence.

2 Navigation Method Description

As mentioned before, the method presented here combines two separate phases: teach and replay (or repeat). During the teach phase, a human operator manually drives the robot through the environment along the desired path. After that, the robot is able to repeat the taught path. To achieve the ability to repeat the path autonomously, the robot creates a map by processing its on-board camera image and storing the image features it saw along the path. Furthermore, the robot stores the commands issued by the operator during the teaching phase. Thus, the only inputs the system requires for both teach and replay are odometry and monocular camera image. The system can dynamically change between different feature extraction algorithms depending on the computational power available. A typical choice is the upright-SURF (Upright Speeded Up Robust Features) [2], Binary Robust Independent Features (BRIEF) [6] or Adaptive and Generic Accelerated Segment Test (AGAST) [26].

2.1 Environment Representation

The way the navigation system represents the environment determines its function, accuracy and robustness. In our case, the environment is represented from a robot-centric way as a linear topological structure, where the taught path is represented as a set of discrete, local maps that contain the images captured by the robot’s on-board camera along with the extracted features. Each local map is associated with the distance from the path start and with the commands issued by the robot operator during the teaching phase. The Fig. 1 shows an example of the environment representation: let us have a robot was driven through the environment along the blue line, while using odometry to measure the distance it travelled so far. Each time the robot operator issues a new command (i.e. he changes the forward or angular velocity), the robot stores this command along with the currently travelled distance. Moreover, at the beginning of the teaching and each time the distance exceeds a certain interval (typically 0.1–0.5 m) from the last local map, the robot saves the current image and its features into a separate local map. These local maps are represented as red dots in the Fig. 1.

2.2 Image Processing

The purpose of the feature extraction is to detect image components, which will help to re-identify a location when the robot captures an image of the

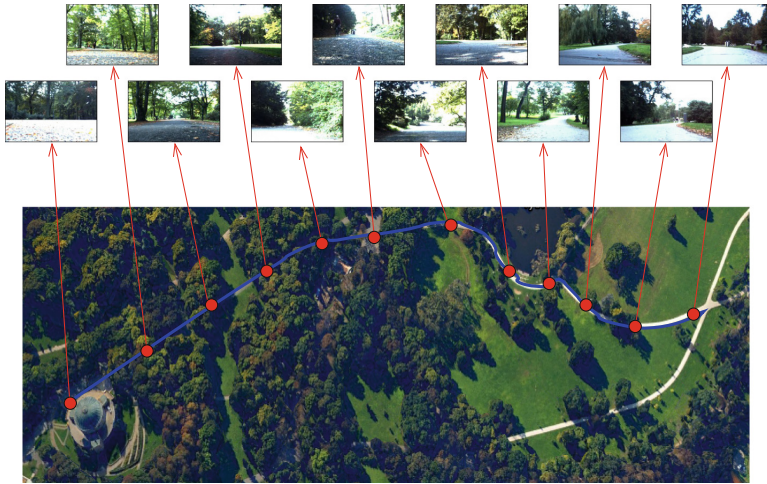


Fig. 1. The taught path is represented by odometric information (blue) and a series of local maps (red dots). The maps contain captured images and their features. (Color figure online)

same location next time. Extraction of image features is a crucial component of the navigation, because it determines what kind of information about the environment is stored in the local maps and what information is then used to steer the robot. Extracting the image features from the camera works in two steps, keypoint detection and feature description. The keypoint detector locates areas (or points) in the image, which are easy to recognise even if the given scene will be viewed from another viewpoint. Typically, the keypoints are high-contrast corners or blobs, which are localised by means of Hessian matrix analysis (in the case of SURF features [2]) or by a direct comparison of a pixel's brightness value with its neighbourhood [26]. The feature descriptor characterises the detected keypoint surroundings. In the case of the SURF algorithm [2], the 64-floating number descriptive vector comprises of intensity gradients around the detected keypoint. The BRIEF descriptor [6] is a 256-bit long binary string calculated by comparing the intensities of 256 pixel pairs around the keypoint. One of the main challenges of outdoor operation is unstable illumination which strongly influences both the detection and description phases of image processing. In order to cope with the illumination changes, our image acquisition and processing modules are actively adapting their parameters (such as camera exposure and Hessian threshold) to achieve high-contrast images and to extract the desired number of image features [16].

2.3 Teaching the Path

During the teaching phase, the robot is driven by means of a joystick or other device along a path, which it is supposed to autonomously traverse later on.

When the teaching starts, the robot sets its distance counter to zero and saves the current image and its features into a local map, which is indexed by distance of zero. As soon as the operator sets the forward or angular velocity, the robot associates these with the zero distance as well, saves the command to the map and starts to move while measuring the travelled distance by odometry. Whenever the operator issues new angular or forward velocity, the robot saves these to the map along with the current distance information. The robot also measures the distance travelled since the last local map recording and whenever this distance exceeds a certain threshold, the robot creates a new local map, associates it with the distance travelled since the teaching started and fills it with the last captured image and its features. When the operator indicates that the teaching phase is over, the robot saves the stop command with the current distance, creates the last local map and provides the operator with basic statistics of the maps created. At the end of the teaching phase, the map should have a similar structure as the one shown in Fig. 1.

The resulting map is neither metrically accurate, nor globally consistent, because for self-localisation, the robot is using only odometry which is subject to drift. However, this map creation process does not aim to create metric, globally-consistent environment representation, because that is not necessary to achieve autonomous operation [14]. Rather, the mapping aims to capture only the information necessary for subsequent autonomous traversals.

2.4 Autonomous Navigation

As soon as the teaching phase is over, the operator can initiate the replay phase, where the robot navigates autonomously. The operator indicates which map to load and places the robot close to the intended path start. The method loads all the saved local maps with image features and operator commands – both commands and maps are indexed by their distance from the path start. Then, it loads the commands and local map stored at zero distance. It applies the commands, and as it moves forwards, it eventually reaches a distance where new commands were issued during teaching, and applies them further. Essentially, the robot replays the commands from the teaching phase.

However, if it would only replay these commands, its trajectory would not be the same as during teaching for several reasons. Firstly the initial angle, where the robot starts cannot be exactly identical to the one during teaching and even slightly different angle would cause the robot to diverge from the desired path after some time. Secondly, the wheel- or track-surface interaction is a source of significant uncertainty due to slippage, variable surface and tire pressure etc. Thus, the robot needs to use the local maps to correct for the aforementioned sources of position inaccuracy. This is typically performed by associating the image features from the map to the image features of the current view and using these associations to determine the robot position relatively to the local map. The relative position is then fused with the odometric information by a Kalman or particle filter, obtaining a robot position estimate.

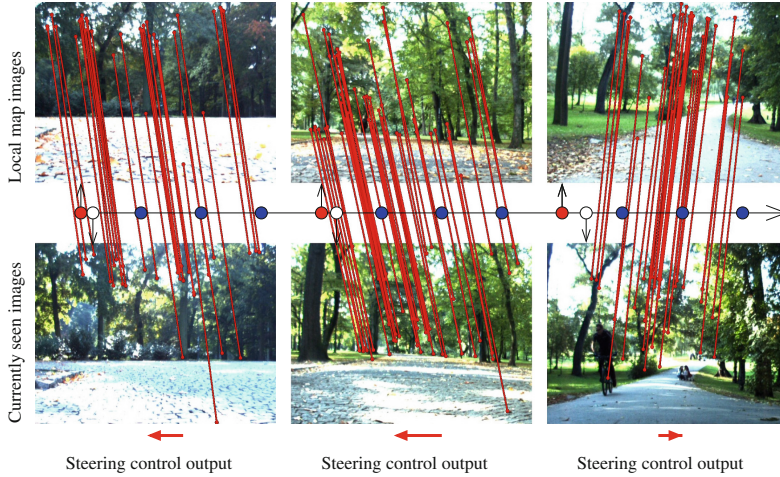


Fig. 2. Navigation principle: a robot at a given distance from the path start (white circle) select the closest map (red circle) and established correspondences between the visible and map features. Difference between the horizontal coordinates of the feature pairs allows to determine the robot steering velocity (shown as red arrows). (Color figure online)

However, accurate and reliable position estimation usually requires that the number of established associations is relatively high, most of correspondences are correct, the features are evenly distributed across the image, they do not lie on planar surfaces or deformable objects, features are not too far away and the images are not blurred or deformed due to rolling shutter effects and the camera is well calibrated. In the real world, many of these assumptions are violated, which causes standard visual-based position estimation to be inaccurate or to fail occasionally. On the other hand, if one needs to recover only the camera heading relatively to the local map, most of the aforementioned conditions do not have to be met, because the heading estimation techniques are simpler than in the case of full 6d position estimation. The situation is even simpler for the case of ground robots, which need to estimate the heading in one direction only, because they move on (locally) planar surfaces. Moreover, the papers [22,35] provide mathematical and experimental evidence, which indicates that unlike for other navigation types, heading estimation in teach-and-repeat navigation can keep the robot position error bound.

This allowed us to implement a simpler method of using the map information to keep the robot close to the taught path. Our system uses the travelled distance information to load an appropriate local map. Then, we extract the features from the current camera view and associate them with the features from the map using the ratio match method proposed in [25]. After that, we find the most frequent horizontal displacement of the mapped and currently visible features using a histogram voting scheme. This displacement, which corresponds to the

robot lateral divergence from the taught path, is then fed into the robot steering controller so that the robot turns to keep this displacement close to zero, see Fig. 2. In this way, the robot is always steered towards the taught path.

2.5 Navigation System Model

From the aforementioned description, it's clear that the robot is able to correct the lateral position error as it moves along the path due to the heading corrections. However, the correction of the longitudinal error component through the heading correction is not straightforward as it's somewhat counter-intuitive. We will show that if the robot moves along a path that is not just a straight line, the heading correction reduces both longitudinal and lateral error.

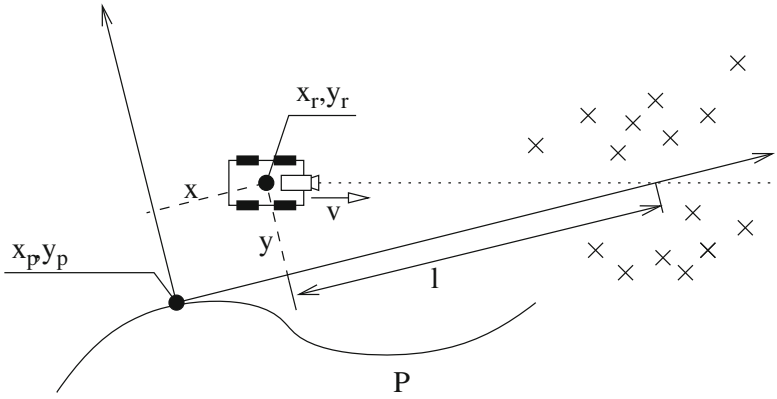


Fig. 3. Robot position error chart. The robot at a position x_r, y_r uses a local feature map of the taught path at the position x_p, y_p . Courtesy of [22].

The basic position evolution model is shown in Fig. 3. Let us assume that a robot navigated by our system is at position x_r, y_r and that it assumes that it's located on the taught path at position x_p, y_p . Thus, the robot position error x, y equals to position x_r, y_r in the coordinate system of the reference point x_p, y_p on the taught path. The evolution of the error over time \dot{x}, \dot{y} is affected by the movement of the local coordinate system along the taught path and the movement of the robot itself. As the robot moves forwards with velocity v , so does the reference point (the path is indexed by travelled distance). Furthermore, the reference system rotates with a velocity given by the local path curvature κ and reference point velocity v . Thus, the position error evolution model is

$$\begin{aligned} \dot{x} &= +\kappa v y - v + v \cos(\varphi) + s_x \\ \dot{y} &= -\kappa v x + v \sin(\varphi) + s_y, \end{aligned} \tag{1}$$

where $\kappa v y$, $\kappa v x$ and $-v$ are caused by the rotation and translation of the local coordinate system as the reference point x_p, y_p moves along the taught path,

φ is the robot heading in the local coordinate frame, the terms $+v \cos(\varphi)$ and $+v \sin(\varphi)$ reflect the robot movement, and s_x and s_y are random variables, which represent perturbations caused by odometric errors, wheel slippage etc. Since the histogram voting method turns the robot to keep the local map features at the same positions as seen from the reference point during the teaching, the robot's orientation φ in the path reference frame is determined by the average distance of these features l and the robot lateral displacement from the path y , see Fig. 3. In particular, the robot steers so that $\varphi = -\arctan(y/l)$. Considering that the distance of the landmarks l is much higher than the absolute value of the robot lateral displacement y , the robot orientation φ can be approximated as $-yl^{-1}$. Thus, one can approximate $v \sin(\varphi)$ by $-vy l^{-1}$ and $v \cos \varphi$ by v , and rewrite Eq. (1) as

$$\begin{aligned} \dot{x} &= +\kappa v y - v + v & + s_x \\ \dot{y} &= -\kappa v x & - v y l^{-1} + s_y. \end{aligned} \quad (2)$$

Rewriting Eq. (1) in a matrix form results in

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & +\kappa v \\ -\kappa v & -v l^{-1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} s_x \\ s_y \end{pmatrix}, \quad (3)$$

which is a linear continuous system in the form of $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{s}$. Such system is stable, i.e. x, y do not diverge, if the real component of eigenvalues of its matrix \mathbf{A} are lower than 0. Factoring out the velocity v from \mathbf{A} , the eigenvalues $\lambda_{0,1}$ can be calculated as:

$$\begin{pmatrix} \lambda_0 \\ \lambda_1 \end{pmatrix} = v \operatorname{eig} \begin{pmatrix} 0 & +\kappa \\ -\kappa & -l^{-1} \end{pmatrix} = \frac{1}{2}v \left(-l^{-1} \pm \sqrt{l^{-2} - 4\kappa^2} \right). \quad (4)$$

Assuming that the robot moves forward ($v > 0$) along a path with non-zero curvature ($\kappa \neq 0$) and the features it uses for navigation are in front of the robot at a finite distance ($0 < l < \infty$), the real parts of eigenvalues $\lambda_{0,1}$ are always smaller than 0. *Thus, if the robot moves forwards along a curved line, both longitudinal (x) and lateral (y) components of its position error do not diverge.* \square

The navigation model is rather crude and it reflects the real robot behaviour only in cases, where the landmark distance l is much larger than the absolute value of the lateral position error y . However, assuming $l \gg |y|$ allows to create a linear model and determine its stability through eigenanalysis. Another important assumption is that the x component of the robot position error is not large enough to cause the robot to use a wrong map for its heading correction – this corresponds to small values of the random variable s_x . Finally, we assume that the robot can control its heading so that it sees the features almost at the same spots as during the mapping phase, i.e. the absolute values of s_y are small as well.

Due to the odometry, vision and other errors, encompassed in the variables s_x, s_y , the robot position error will not be zero. Rather, it will stabilise at some value, which will be proportional to the size of perturbances s_x, s_y and inversely proportional to the landmark distance l . For a detailed proof and discussion of the model, see [22, 35].

2.6 System Implementation

To allow its easy portability and use by other research teams, we integrated our method into the Robotic Operating System (ROS). The core scheme is shown in Fig. 4. The *odometry_monitor* node reads the data from the odometry and computes travelled distance, which is transferred to the *map_preprocessor*, *navigator* and *mapper* nodes. The *mapper* node saves the current image with extracted features, which are provided by the *feature_extraction* node into the *map_storage*. It also records the speeds set by the operator. The *feature_extraction* node is grabbing the images from the on-board camera, extracts the features, and sends them to *mapper* and *navigator* nodes. The *map_preprocessor* node is used during the navigation phase, see Sect. 2.4. At the start, the *map_preprocessor* node preloads all relevant local maps and operator commands. Then, based on the information provided by the *odometry_monitor*, it forwards the corresponding commands and local map to the *navigator*. The *navigator* receives the currently visible features from the *feature_extraction* node, obtains the local map from the *map_preprocessor* and performs the histogram voting described in Sect. 2.4. The result of the histogram voting is then converted into a steering correction, which is added to the angular speed provided by the *map_preprocessor*. The resulting speeds are sent to the *robot*, which repeats the original commands from the teaching phase, while correcting its heading so that it stays on the path.

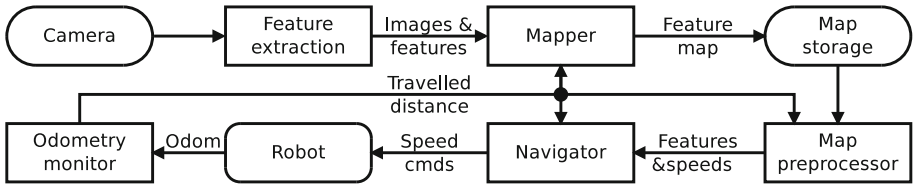


Fig. 4. Navigation system structure: ROS nodes and important topics.

All the aforementioned modules were implemented as ROS action servers with dynamically reconfigurable parameters, so that 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 which shows the current state of the system. Thus, the operator can see the currently used map, path profile data, number of visible image features, results of the histogram voting method etc. The described system is available as an open source code at [27].

3 Experimental Evaluation

The purpose of the experiments is to verify the ability of the navigation system to keep the autonomous vehicles on the taught path. The system is based on the

model introduced in Sect. 2.5, which predicts that a robot driven by our navigation system is able to gradually reduce its position error and thus, keep it within acceptable bounds. The aim of the experimental setup for each autonomous vehicle, which we were working with, was to verify the aforementioned hypothesis. To do so, we first teach the given robot a closed path. Then, we introduce a large position error by displacing the robot from the path start. After that, we let the robot drive along the path several times, and after each path traversal, we measure (typically by hand) its distance to the path start. This distance corresponds to the robot position error after traversing one loop of the path. If the model introduced in Sect. 2.5 is correct, and the values of s_x, s_y are small, then the initial, artificially introduced position error should gradually diminish, and the robot position error should stabilise. This stabilised value indicates the overall navigation error of our system for a given platform and environment. To thoroughly evaluate the correctness of the mathematical model and robustness of the navigation method, we performed the aforementioned experiments using several mobile vehicles with a different type of kinematics. Videos of some of the experiments are provided on a special webpage reachable from the website of the navigation system [27].

The second part of our evaluation is aimed at a more practical issue: integration. We simply measured the time it took us to deploy our system onto a given platform and to perform the deployment. Since the deployment is divided into two steps: integration and debugging, we measure these times separately. Although these times depend on the experience of the system integrators and are inherently subjective, they still indicate how difficult was the deployment across the individual platforms.

3.1 Cameleon Robot

The first platform used in the experiments is the tracked robot CAMELEON ECA, which is a 0.7×0.6 m vehicle with a payload over 25 kg. The platform has two main independently controllable tracks and two auxiliary flippers for movement in difficult terrain. It is also equipped with two cameras, one in the front and one in the back of the body of the robot. As these cameras are located too low over the ground, they are not suitable for visual navigation in grassy terrains. Thus, we installed eCon's TARA stereocam, and we use the images from its left camera. This camera is attached to the robot superstructure, which also contains a stand for the control laptop and the Fenix 4000 lumen torch used for night experiments. For the fully equipped Cameleon platform, see Fig. 5.

The experiments took place at Hostibejk Hill in Kralupy nad Vltavou, Czechia, where the robot repeatedly travelled a 70 m long, complex path over concrete, grass and asphalt surfaces. Approximately one-third of the path, a small building was in the robots field of view. Otherwise, it perceived mostly trees and natural structures. This experiment was unique because of the tracks the platform consisted of, and these cause its odometry to be highly inaccurate. Despite that, the robot could quickly suppress the position error we introduced, stabilising it at values below 0.1 m, see Fig. 5.

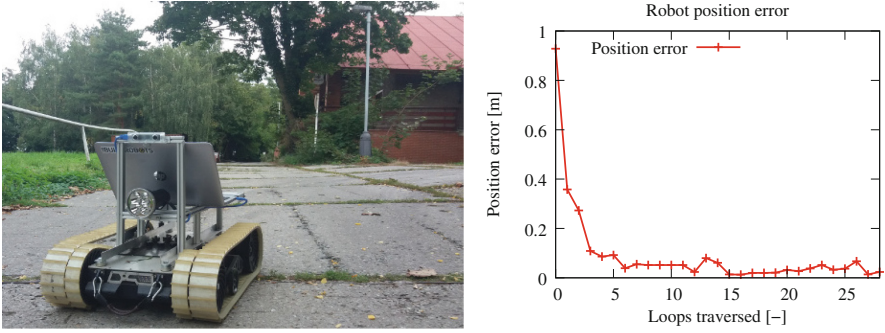


Fig. 5. Cameleon robot and its position error during the autonomous path traversal.

Integration of the system onto the Cameleon platform required to implement a ROS bridge to command the robot and to obtain odometry, which took 5 h of implementation and 1 h of testing time. The experiment itself took 1 h. The main problems encountered were caused by occasional ‘freezing’ of the PC, which took 1–3 s. During these periods, the robot was driving incorrectly and often deviated from the intended path. However, as soon as the system started to react, the robot started to follow the taught path and reduced the position error rapidly.

3.2 MMP-5 Robot

To demonstrate the system’s ability to work indoors on a small robot, we deployed it on the MMP-5 platform, which is made by TheMachineLab. It is a small robot with dimensions 0.3×0.3 m, payload over 3 kg and four-wheel differential drive along with a control unit, which can control each wheel independently via a simple serial protocol. This platform, however, does not provide odometry, so we estimated the travelled distance by time and by the motors’ PWM duty. We equipped this robot a low-resolution USB camera and a computer based on an AT3IONT-I miniATX board with Intel Atom 330 CPU.

The indoor experiments took place in the entrance building of the Czech technical university in Prague at Karlovo Náměstí campus. We taught the robot a 17 m lemniscate-shaped path, displaced it from the path start by 1.4 m and let it traverse the path 20 times. The progress of the robot was monitored by an external localisation system [20], which was tailored specifically for small platforms [1]. The use of the external localisation system, based on a high-resolution camera, allowed to monitor the evolution of the robot position error over time with centimetre accuracy, see [22]. Furthermore, the system was used to determine the robot position error after each path traversal, so we did not have to measure the displacement by hand. The experimental results, shown in Fig. 6, indicate that even a robot without wheel encoders and only a low resolution camera, which suffered from motion blur, was able to navigate through the taught path while reducing the initial error. The overall achieved position

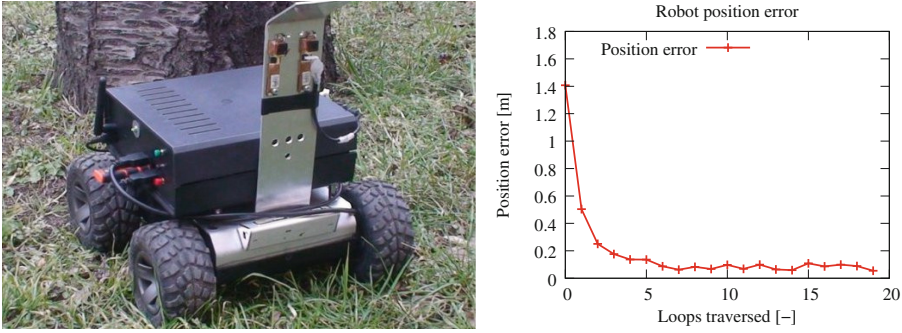


Fig. 6. MMP-5 robot and its position error during the autonomous path traversal.

error was about 0.1 m, and the integration and testing of the system took around 4 and 2 h respectively.

The main problem faced was related to the low computational power of the onboard PC, which caused issues with image processing and real-time control.

3.3 John Deere Tractor

The John Deere X300R is a small tractor, see Fig. 7, with gasoline 13.8 kW motor and automatic K46 hydrostatic transmission. It weights around 300 kg. Its wheelbase is 1.25 m and overall dimensions are $2.5 \times 1.0 \times 1.1$ m. This machine, which has a car-like drive, was modified and equipped with hydraulic steering and linear motor with position feedback for pedal control. Due to the mechanics of the control system, both steering and velocity can be controlled only coarsely and with a significant time delay – this corresponds to large values of s_x and s_y in the navigation model in Eq. (3).

The tractor is also equipped with a set of sensors to support autonomy: 4 emergency stop buttons, active front bumper, rotary encoders, Hall sensor for measuring steering angle, set of indicators and switches (including manual/automatic digital input), 2D Lidar SICK LMS100 and a AV3135 Dual Sensor H.264 Day/Night camera. The low level electronic components (pedal position, encoders, steering, I/O) are controlled via CAN bus modules, connected to an APU2 computer. The navigation algorithm was running on the same PC as in the case of Cameleon ECA, i.e. Intel i3 laptop, which was connected to APU2 computer via ethernet. Apart from the laptop, which was running the navigation system, we installed our own camera to the tractor front.

This experiment took place in an experimental agricultural field in the campus of the Czech University of Life Sciences in Prague. During the experiments, the system had to deal with images blurred due to the engine vibrations, as well as rain, which affected both the environment appearance and wheel-surface interaction. Eventually, the rain became too heavy and we had to terminate the experiment prematurely. Despite of that, the robot was able to suppress the initial position error, stabilising it around 0.5 m, see Fig. 7.

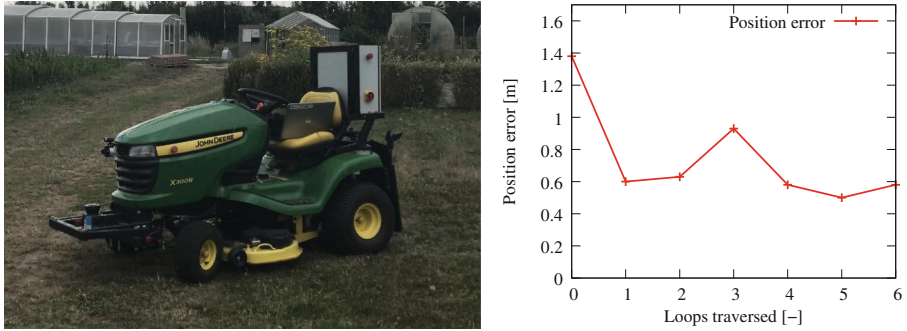


Fig. 7. John Deere platform and its position error during the autonomous drive.

Integration of the system took 2 h, initial testing 5 h and the experiment took 1 h. The main encountered problems were caused by rather slow and coarse hydromechanical control of the steering and velocity. Furthermore, we experienced a significant lag when retrieving images from the robot’s IP camera, and thus, we had to install our own USB cam.

3.4 TAROS 6 × 6 Combat Support UGV

We tested the system on a large, 6 × 6 drive unmanned ground vehicle called ‘TAROS’, developed by the VOP company. The primary purpose of TAROS is direct support of mechanised, reconnaissance and special forces in their operations. Currently, the TAROS platform offers long-range teleoperation, GPS-based waypoint navigation and autonomous people following. However, its sensory suite, consisting of a Velodyne 3d lidar, three 2d SICK lidars and pan-tilt cameras, is sufficient for fully autonomous operation in adverse terrain.

One of the main features of TAROS is its modularity. Its core module, which contains all systems required for autonomous operation, is propelled by four electrically-driven wheels with independent suspensions. The wheels are equipped with odometric sensors and can be steered and driven individually. Each TAROS extension module adds two individually steered wheels, which allows configuring TAROS as 6 × 6 or 8 × 8 wheel drive UGV. For our experiments, we used the TAROS in 6 × 6 configuration with a hybrid power extension module.

The TAROS control system offers a convenient serial protocol which allows an ordinary PC to retrieve low-level (such as specific wheel angles) as well as high-level (such as the distance of the closest obstacle) information. Furthermore, the protocol enables low-level control of individual wheel actuators as well as high-level driving modes, which allow controlling the angular and forward speed in four different wheel configurations. The chosen protocol allowed a quick implementation of a basic ROS driver for the TAROS platform. The driver retrieves odometric information from the TAROS middle wheels and utilises a

driving mode, which controls the TAROS heading by steering the forward and rear wheels while keeping the middle wheels straight.

Similarly to the previous case, the platform's IP camera images were provided with a significant delay, and thus, we used the USB camera of the i3 laptop, which was used in the previous experiments.

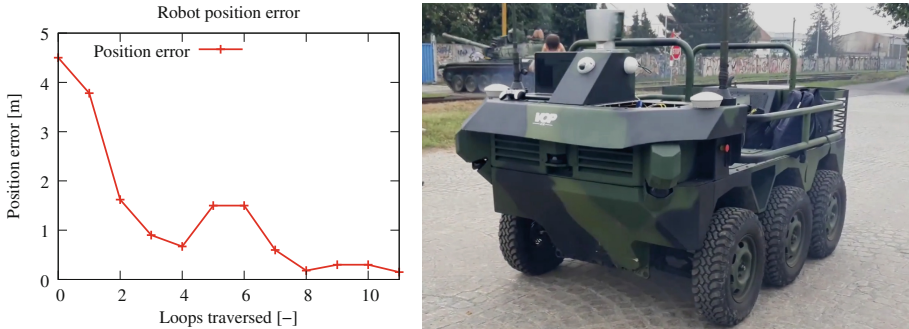


Fig. 8. TAROS platform and its position error during the autonomous drive.

The experiments took place at a small testing polygon in VOP CZ, which produces the TAROS platform. During the test, the robot was taught a 100 m long, closed path, and then it was displaced by 4.5 m and let to traverse the path autonomously 11 times. The error was gradually diminished, and despite of the platforms large size, the error was finally reduced below 0.2 m, see Fig. 8. The sudden increase of the position error in the 5th loop traversal was caused by the control computer 3 s 'freeze', leading to an overshoot at the path end. The Fig. 8 shows that as the robot continued to traverse the path, this error was gradually diminished as well.

Integration of the system took 2 h, its preliminary testing 3 h and the experiment itself took 1 h. Our software expected that the odometry is provided with at least 10 Hz update rate while the TAROS robot provides odometric information at 5 Hz. A significant part of the testing time was dedicated to identifying and correcting this issue in software.

3.5 Thorvald Agricultural Robot

Thorvald is a platform intended for various agricultural scenarios. This platform can be equipped with several different modules, which extends the variety. It has four wheels, which can be steered and actuated individually and the width between the wheels can be changed depending on the desired agricultural application. We have used the standard wheel configuration with the width of the wheelbase of 1.5 m. All the wheels are equipped with suspension modules, which allows the platform to drive in rough terrain. Low-level robot control is performed by an onboard PC, which was connected to our Intel i3 laptop over ethernet. Again, we used a standard USB camera as the main sensor.



Fig. 9. Thorvald robot and the path traversed during its test.

In the case of Thorvald, we took another approach to the testing. Instead of traversing a single loop several times, we taught it a 600 m long route around the Isaac Newton building of the Lincoln University, displaced the robot by 2 m and traversed this path once. Furthermore, during autonomous traversal, we had to manually steer the robot about 2 m from the path to avoid a large van, which stopped in the way. Despite that, the robot could reach the path end location with less than 0.3 m error.

Integration of the control system took approximately 1 h, and testing took 1 h. The experiment itself took 2 h. The main issue encountered was related to interfacing the robot control PC with the laptop running the navigation system.

3.6 STOP Robot

Finally, to demonstrate that the system can be deployed on indoor platforms as well, we tested it using the STOP robot, see Fig. 10. The STOP robot is a low-cost mobile robot platform for monitoring and surveillance of buildings and facilities, aiming at keeping intruders away, preventing the misuse of spaces, and more importantly, to act in a timely manner in case someone infiltrates the premises or violates rules of use of the spaces, e.g. by alerting security operators [32]. Several STOP robots, which are intended to cooperate, are under development by Ingeniarius, Ltd. from Coimbra, Portugal within the scope of the STOP R&D Project initiative¹. The robot features a 360-degree RPLIDAR A2 laser range finder, an Orbbec Astra RGBD camera, a touchscreen interface, and a powerful Intel NUC mini PC. Low-level robot control is performed by a Particle Photon board, which is connected to the main NUC mini PC over serial. Being fully ROS-compliant, the robot benefits from navigation software, and a specifically tailored localisation system, which fuses laser scan matching odometry, global orientation from a digital compass, encoder odometry and an ICP correction system for global localisation. In our experiments, we kept the aforementioned localisation system on, so that we could use it as a source of ground truth position information. As in the previous experiments, the robot

¹ <http://stop.ingeniarius.pt>.

was controlled only by the navigation method proposed, which is solely based on odometric data and monocular camera image stream.

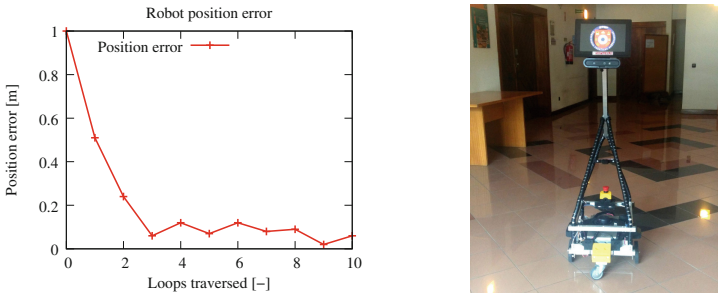


Fig. 10. STOP robot platform and its position error during the autonomous drive.

The experimental evaluation took place in an entrance hall of the Ingeniarius Ltd., which produces the STOP robot. The robot was taught a ~ 10 m long path, then it was displaced by 1 m, and taught to traverse the path autonomously 10 times. The error gradually diminished and stabilised at ~ 0.1 m, see Fig. 10.

Before integration, we needed to perform the system update, which took more than 3 h. Since the robot supports ROS natively, integration took less than 1 h preliminary testing 1 h, and the experiment itself lasted 40 min. The main issue encountered was caused by different versions of OpenCV software libraries for image processing.

3.7 Experiments Summary

The experiments, summarised in Table 1 indicated that the ‘convergence’ theorem, introduced in Sect. 2.5 and papers [22,35], is valid for different types of vehicles, including tracked, car-like, with independently steered wheels, electrically or hydraulically controlled, or with combustion engines. Furthermore, the accuracy of the navigation exceeded the typical accuracy of (non-RTK) GPS service, the robots were able to correct initial position errors up to 4.5 m (see Fig. 8), and integration of the system into various platforms was relatively quick and did not require special customisation or extensive parameter tuning. Thus, the experimental results were not only in accordance of the ‘convergence’ theorem predictions, but they also demonstrated the maturity of the navigation system, which is provided as open-source C++ code at www.github.com/gestom/stroll_bearnav. The summary of the experiments is provided in Table 1, which shows the achieved navigation accuracy, deployment, testing and experimental times.

The main lesson learned from the experiments is that while the navigation system is mature enough to be easily integrated into various platforms, its reliability is still affected by software issues originating from the fact, that the Robotic

Table 1. Navigation accuracy and integration times on different platforms

| Platform | Navigation | | Time [h] required for | | |
|------------|--------------|---------------|-----------------------|---------|------------|
| | Accuracy [m] | Travelled [m] | Integration | Testing | Experiment |
| Cameleon | 0.05 | 2000 | 5 | 1 | 1 |
| MMP5 | 0.07 | 340 | 4 | 2 | 2 |
| John Deere | 0.50 | 180 | 2 | 5 | 1 |
| TAROS | 0.15 | 1100 | 2 | 3 | 1 |
| Thorvald | 0.30 | 600 | 1 | 1 | 2 |
| STOP | 0.09 | 100 | 1 | 1 | 1 |

Operating System is running on Linux distributions, which are not meant to provide real-time response.

3.8 Additional Simulations

Since the aforementioned experiments were aimed at proving the system accuracy in real scenarios, the number of experimental trials and scenarios was rather limited. To evaluate the method in a wider set of trajectories and environmental configurations, we simulated the system behaviour in 1000000 randomly generated scenarios with trajectory curvatures ranging from 0.001 to 1 m^{-1} , landmark distances between 0.1 and 2 m, initial robot position errors between 0 and 1 m in both x and y directions, odometric errors up to 5% and heading estimation errors up to 5° . The simulations indicated that for every of the 1000000 configurations, the robot gradually converged to the intended trajectory.

4 Conclusion

We presented a versatile teach-and-repeat vision-based navigation system, which is easy to deploy on a variety of ground robot platforms. The system is capable to guide mobile vehicles along a path previously taught by a human operator. Instead of creating a globally-consistent metric map, our method creates a sensorimotor record of a teleoperated drive, consisting of odometric and visual information. The autonomous navigation system is based on the retrieval and replay of the recorded experience instead of estimating the global position of the robot in the 2d/3d space. We presented a mathematical model of the navigation, which allows to calculate the evolution of the robot position error over time. Then, through eigenanalysis, we show that the robot position error does not diverge, which ensures navigation accuracy and reliability.

While being computationally efficient, the system does not require camera calibration, and it can deal with realistic outdoor lighting conditions. Furthermore, the system is easy to operate, because it only requires that the robot is manually guided along a desired path, which it can repeat autonomously later on.

To verify the correctness of the ‘convergence’ theorem the system is based on, we tested it in simulation and on a variety of ground robot platforms with different types of kinematics and propulsion. The experiments indicated not only that the navigation accuracy exceeds the typical non-RTK GPS precision, but also that the integration of the system into a new robot is straightforward and it takes only a few hours. The presented system, which was already deployed in several scenarios including aerial inspection [13, 23, 30], is provided as open source at www.github.com/gestom/stroll_bearnav. Datasets and videos of the experiments are available at www.github.com/gestom/stroll_bearnav/wiki.

Acknowledgments. We thank the VOP.cz for sharing their the data and the TAROS vehicle. We would like to thank also Milan Kroulík and Jakub Lev from the Czech University of Life Sciences Prague for their positive attitude and their help to perform experiments with the John Deere tractor.

References

1. Arvin, F., Krajník, T., Turgut, A.E., Yue, S.: COS Φ : artificial pheromone system for robotic swarms research. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015)
2. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
3. Behzadian, B., Agarwal, P., Burgard, W., Tipaldi, G.D.: Monte carlo localization in hand-drawn maps. In: IROS, pp. 4291–4296. IEEE (2015)
4. Biber, P., Duckett, T.: Dynamic maps for long-term operation of mobile service robots. In: RSS (2005)
5. Blanc, G., Mezouar, Y., Martinet, P.: Indoor navigation of a wheeled mobile robot along visual routes. In: IEEE International Conference on Robotics and Automation (ICRA) (2005)
6. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_56
7. Chen, Z., Birchfield, S.T.: Qualitative vision-based path following. IEEE Trans. Robot. Autom. **25**, 749–754 (2009). <https://doi.org/10.1109/TRO.2009.2017140>
8. Churchill, W.S., Newman, P.: Experience-based navigation for long-term localisation. IJRR **32**, 1645–1661 (2013). <https://doi.org/10.1177/0278364913499193>
9. Dayoub, F., Cielniak, G., Duckett, T.: Long-term experiments with an adaptive spherical view representation for navigation in changing environments. Robot. Auton. Syst. **59**, 285–295 (2011)
10. De Cristóforis, P., Nitsche, M., Krajník, T.: Real-time monocular image-based path detection. J. Real Time Image Process. **11**, 335–348 (2013)
11. DeSouza, G.N., Kak, A.C.: Vision for mobile robot navigation: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 237–267 (2002). <https://doi.org/10.1109/34.982903>

12. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_54
13. Faigl, J., Krajník, T., Vonásek, V., Přeučil, L.: Surveillance planning with localization uncertainty for UAVs. In: 3rd Israeli Conference on Robotics (2010)
14. Furgale, P., Barfoot, T.D.: Visual teach and repeat for long-range rover autonomy. *J. Field Robot.* **27**(5), 534–560 (2010). <https://doi.org/10.1002/rob.20342>
15. Halodová, L.: Map Management for Long-term Navigation of Mobile Robots. Bachelor thesis, Czech Technical University, May 2018
16. Halodová, L., Dvořáková, E., Majer, F., Ulrich, J., Vintr, T., Krajník, T.: Adaptive image processing methods for outdoor autonomous vehicles. In: Mazal, J. (ed.) MESAS 2018. LNCS, vol. 11472, pp. 456–476 (2018)
17. Zhang, N., Warren, M., Barfoot, T.: Learning place-and-time-dependent binary descriptors for long-term visual localization. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE (2016)
18. Holmes, S., Klein, G., Murray, D.W.: A square root unscented kalman filter for visual monoSLAM. In: International Conference on Robotics and Automation (ICRA), pp. 3710–3716 (2008)
19. Kosaka, A., Kak, A.C.: Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP: Image Underst.* **56**(3), 271–329 (1992)
20. Krajník, T., et al.: A practical multirobot localization system. *J. Intell. Robot. Syst.* **76**, 539–562 (2014). <https://doi.org/10.1007/s10846-014-0041-x>
21. Krajník, T., Cristóforis, P., Kusumam, K., Neubert, P., Duckett, T.: Image features for visual teach-and-repeat navigation in changing environments. *Robot. Auton. Syst.* **88**, 127–141 (2017)
22. Krajník, T., Majer, F., Halodová, L., Vintr, T.: Navigation without localisation: reliable teach and repeat based on the convergence theorem. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)
23. Krajník, T., Nitsche, M., Pedre, S., Přeučil, L., Mejail, M.E.: A simple visual navigation system for an UAV. In: 9th International Multi-Conference on Systems, Signals and Devices (SSD), pp. 1–6. IEEE (2012)
24. Kunze, L., Hawes, N., Duckett, T., Hanheide, M., Krajník, T.: Artificial intelligence for long-term robot autonomy: a survey. *IEEE Robot. Autom. Lett.* **3**, 4023–4030 (2018). <https://doi.org/10.1109/LRA.2018.2860628>
25. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**(2), 91–110 (2004)
26. Mair, E., Hager, G.D., Burschka, D., Suppa, M., Hirzinger, G.: Adaptive and generic corner detection based on the accelerated segment test. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6312, pp. 183–196. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15552-9_14
27. Majer, F., Halodová, L., Krajník, T.: Source codes: bearing-only navigation. https://github.com/gestom/stroll_bearnav
28. Matsumoto, Y., Inaba, M., Inoue, H.: Visual navigation using view-sequenced route representation. In: IEEE International Conference on Robotics and Automation (ICRA), Minneapolis, USA, pp. 83–88 (1996)
29. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **31**, 1147–1163 (2015)

30. Nitsche, M., Pire, T., Krajník, T., Kulich, M., Mejail, M.: Monte Carlo localization for teach-and-repeat feature-based navigation. In: Mistry, M., Leonardis, A., Witkowski, M., Melhuish, C. (eds.) TAROS 2014. LNCS (LNAI), vol. 8717, pp. 13–24. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10401-0_2
31. Paton, M., MacTavish, K., Berczi, L.-P., van Es, S.K., Barfoot, T.D.: I can see for miles and miles: an extended field test of visual teach and repeat 2.0. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics. SPAR, vol. 5, pp. 415–431. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67361-5_27
32. Portugal, D., Pereira, S., Couceiro, M.S.: The role of security in human-robot shared environments: a case study in ROS-based surveillance robots. In: 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 981–986. IEEE (2017)
33. Royer, E., Lhuillier, M., Dhome, M., Lavest, J.M.: Monocular vision for mobile robot localization and autonomous navigation. *Int. J. Comput. Vis.* **74**(3), 237–260 (2007). <https://doi.org/10.1007/s11263-006-0023-y>
34. Segvic, S., Remazeilles, A., Diosi, A., Chaumette, F.: Large scale vision based navigation without an accurate global reconstruction. *IEEE International Conference on Computer Vision and Pattern Recognition. CVPR 2007*, Minneapolis, Minnesota, pp. 1–8 (2007)
35. Krajník, T., Faigl, J., Vonásek, V., et al.: Simple, yet Stable bearing-only Navigation. *J. Field Robot.* **27**, 511–533 (2010)
36. Thorpe, C., Hebert, M.H., Kanade, T., Shafer, S.A.: Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Trans. Pattern Anal. Mach. Intell.* **10**(3), 362–373 (1988)
37. Wallace, R.S., Stentz, A., Thorpe, C.E., Moravec, H.P., Whittaker, W., Kanade, T.: First results in robot road-following. In: *IJCAI*, pp. 1089–1095. Citeseer (1985)