J Intell Robot Syst (2014) 76:539–562
DOI 10.1007/s10846-014-0041-x

A Practical Multirobot Localization System

Tomáš Krajník · Matías Nitsche · Jan Faigl · Petr Vaněk · Martin Saska · Libor Přeučil · Tom Duckett · Marta Mejail

Received: 6 August 2013 / Accepted: 24 February 2014 / Published online: 24 April 2014
© Springer Science+Business Media Dordrecht 2014

Abstract We present a fast and precise vision-based software intended for multiple robot localization. The core component of the software is a novel and efficient algorithm for black and white pattern detection. The method is robust to variable lighting conditions, achieves sub-pixel precision and its computational complexity is independent of the processed image size. With off-the-shelf computational equipment and low-cost cameras, the core algorithm is able to process hundreds of images per second while tracking hundreds of objects with millimeter precision. In addition, we present the method's mathematical model, which allows to estimate the expected localization precision, area of coverage, and processing speed from the camera's intrinsic parameters and hardware's processing capacity. The correctness of the presented model and performance of the algorithm in real-world conditions is verified in several experiments. Apart from the method description, we also make its source code public at http://purl.org/robotics/whycon; so, it can be used as an enabling technology for various mobile robotic problems.

Keywords Localization · Mobile robotics · Computer vision · Swarm robotics

T. Krajník (✉) · T. Duckett
Lincoln Centre for Autonomous Systems,
School of Computer Science, University of Lincoln,
Lincoln, UK
e-mail: tkrajnik@lincoln.ac.uk

T. Duckett
e-mail: tduckett@lincoln.ac.uk

T. Krajník · J. Faigl · P. Vaněk · M. Saska · L. Přeučil
Faculty of Electrical Engineering, Czech Technical
University in Prague, Prague, Czech Republic

J. Faigl
e-mail: faiglj@fel.cvut.cz

P. Vaněk
e-mail: vanekpe5@fel.cvut.cz

M. Saska
e-mail: saskam@fel.cvut.cz

M. Nitsche · M. Mejail
Laboratory of Robotics and Embedded Systems
Faculty of Exact and Natural Sciences,
University of Buenos Aires, Buenos Aires,
Argentina

M. Nitsche
e-mail: mnitsche@dc.uba.ar

M. Mejail
e-mail: marta@dc.uba.ar

1 Introduction

Precise and reliable position estimation remains one of the central problems of mobile robotics. While the problem can be tackled by Simultaneous Localization and Mapping approaches, external localization systems are still widely used in the field of mobile robotics both for closed-loop mobile robot control and for ground truth position measurements. These

external localization systems can be based on an augmented GPS, radio, ultrasound or infrared beacons, or (multi-) camera systems. Typically, these systems require special equipment, which might be prohibitively expensive, difficult to set up or too heavy to be used by small robots. Moreover, most of these systems are not scalable in terms of the number of robots, i.e., they do not allow to localize hundreds of robots in real time. This paper presents a fast vision-based localization system based on off-the-shelf components. The system is precise, computationally efficient, easy to use, and robust to variable illumination.

The core of the system is a detector of black-and-white circular planar ring patterns (roundels), similar to those used for camera calibration. A complete localization system based on this detector is presented. The system provides estimation of the roundel position with precision in the order of millimeters for distances in the order of meters.

The detection with tracking of a single roundel pattern is very quick and the system is able to process several thousands of images per second on a common desktop PC. This high efficiency enables not only tracking of several hundreds of targets at a camera frame-rate, but also implementation of the method on computationally restricted platforms. The fast update rate of the localization system allows to directly employ it in the feedback loop of mobile robots, which require precise and high-frequency localization information.

The system is composed of low-cost off-the-shelf components only – a low-end computer, standard webcam, and printable patterns are the only required elements. The expected coverage, precision, and image processing speed of the system can be estimated from the camera resolution, computational power, and pattern diameter. This allows the user to choose between high-end and low-end cameras, estimate if a particular hardware platform would be able to achieve the desired localization frequency, and calculate a suitable pattern size for the user's specific application.

Ease of the system setup and use are also driving factors of the proposed implementation, which does not require user-set parameters or an intricate set-up process. The implementation also contains an easy tool for camera calibration, which, unlike other calibration tools, does not require user interaction. At the same time, the implementation is proposed as a library, which can be integrated into commonly used computer vision frameworks, such as OpenCV.

The main intention of this paper is to present the system principle, its theoretical properties and real performance characteristics with respect to the intended application. Therefore, we present a model of the localization arising from theoretical analyses of the vision system and experimental evaluation of the system performance in real scenarios with regard to its practical deployment.

2 Related Work

External localization systems are widely used in the field of mobile robotics, either for obtaining ground truth pose data or for inclusion in the control loop of robots. In both scenarios, it is highly desirable to have good precision and high-frequency measurements. Here, both of these aspects are analysed in related works and are specifically addressed in the proposed system.

Localization systems for mobile robots comprise an area of active research; however, the focus is generally on internal localization methods. With these methods, the robot produces one or more estimates of its position by means of fusing internal sensors (either exteroceptive or proprioceptive). This estimation can also be generally applied when either a map of the environment exists *a priori* or when the map is being built simultaneously, which is the case of SLAM approaches [1]. When these internal localization systems are studied, an external positioning reference (i.e., the ground truth) without any cumulative error is fundamental for a proper result analysis. Thus, this research area makes use of external localization systems.

While the most well-known external localization reference is GPS, it is also known that it cannot be used indoors due to signal unavailability. This fundamental limitation has motivated the design of several localization principles, which can be broadly divided into two major groups by means of the type of sensors used: active or passive.

In the former group, several different technologies are used for the purpose of localization. One example [2] of active sensing is the case of a 6DoF localization system comprised of target modules, which include four LED emitters and a monocular camera.

Markers are detected in the image and tracked in 3D, making the system robust to partial occlusions and increasing performance by reducing the search area to the vicinity of the expected projection points. Experiments with this system were performed using both ground and aerial robots. The mean error of the position estimation is in the order of 1 cm, while the maximum error is around 10 cm. The authors note that for uncontrolled lighting scenarios passive localization systems appear to be more suitable.

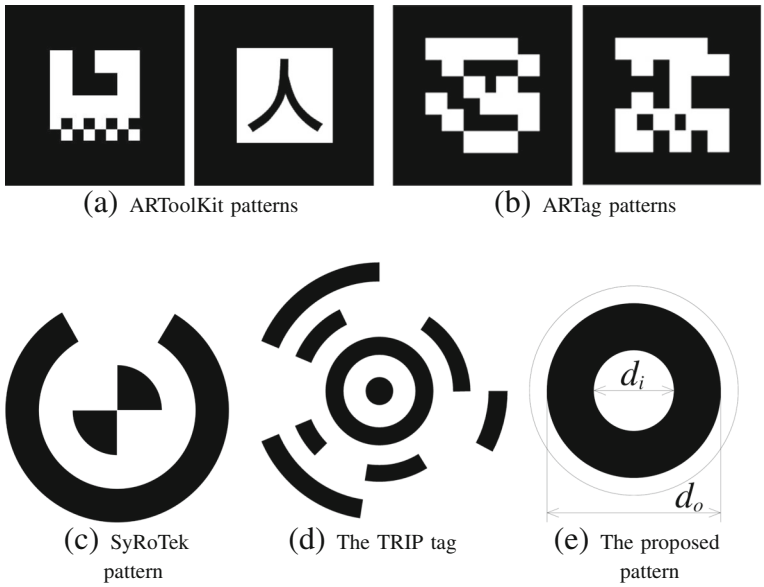
Another active sensor approach is the NorthStar [3] localization system, which uses ceiling projections as a non-permanent ambient marker. By projecting a known pattern, the camera position can be obtained by reprojection. The authors briefly report the precision of the system to be around 10 cm.

In recent works, the most widely used approach is the commercial motion capture system from ViCon [4]. This system is comprised of a series of high-resolution and high-speed cameras, which also have strong infra-red (IR) emitters. By placing IR reflective markers on mobile robots, sub-millimeter precision can be achieved with updates up to 250Hz. Due to these qualities, ViCon has become a solid ground-truth information source in many recent works and, furthermore, has allowed development of closed-loop aggressive maneuvers for UAVs inside lab environments [5]. However, this system is still a very costly solution, and therefore, it is not applicable to every research environment. This issue has motivated several works proposing alternative low-cost localization systems.

Several passive vision-based localization methods were also proposed in recent literature, using simple planar printable patterns, which reduce significantly the cost and difficulty of use and setup. Several of these works employ augmented-reality oriented markers, which not only permit obtaining the pose of the target but can also encode additional information like target ID. In this area, the software libraries most widely used for this purpose are ARTag [6] and ARToolKit+ [7], both based on its predecessor ARToolKit [8], see examples of patterns in Fig. 1. These target detectors were used in several works in order to obtain localization information about mobile robots, either explicitly as a part of a pose estimation system [9, 10] or as ground-truth data [11].

In [9], ARToolKit markers are used for obtaining the pose of several ground robots. The homography from 3D-to-2D space (ground floor) is computed by defining the work area by placing four ad-hoc markers, which are manually detected in the image. In more recent work, the authors proposed the ARTag [6] system that was later extensively analysed in [12]. However, the analysis is focused on detection and confusion rates, and it does not report the real accuracy in position estimation. Similar systems are explored in [13], but details of their precision are not reported.

One particular system, which is based on AR markers similar to ARTag and ARToolKit, is ArUco [14].



(a) ARToolKit patterns (b) ARTag patterns

(c) SyRoTek pattern (d) The TRIP tag (e) The proposed pattern

Fig. 1 Patterns used in passive vision-based global localization systems

The main aspects of this method are: easy integration into C++ projects, based exclusively on OpenCV and a robust binary ID system with error correction which can handle up to 1024 individual codes. The detection process of AR markers in ArUco consists of: an adaptive thresholding step, contour extraction and filtering, projection removal and code identification. When the intrinsic camera parameters are known, the extrinsic parameters of the target can be obtained. Due to the free availability of the implementation and lack of performance and precision reports, this system is analyzed in the presented work, see Section 6.5.

Since the previous pattern detectors were conceived for augmented-reality applications, other works propose alternative target shapes, which are specifically designed for vision-based localization systems with high precision and reliability. Due to several positive aspects, circular shaped patterns appear to be the best suited as fiducial markers in external localization systems. This type of pattern can be found (with slight variations) in several works [15–18].

The SyRoTek e-learning platform [19] uses a ring shaped pattern with a binary tag (see Figure) to localize up to fourteen robots in a planar arena. The pattern symmetry is exploited to perform the position and orientation estimation separately, which allows to base the pattern localization on a two-dimensional convolution. Although this convolution-based approach has proven to be reliable enough to achieve 24/7 operation, its computational complexity still remains high, which lead to its implementation on alternative platforms such as FPGA [20].

In [16], a planar pattern consisting of a ring surrounding the letter "H" is used to obtain the relative 6DoF robot pose with an on-board camera and IMU (Inertial Measurement Unit) to resolve angular ambiguity. The pattern is initially detected by binarization using adaptive thresholding and later processing for connected component labeling. For classifying each component as belonging to the target or not, a neural network (multilayer perceptron) is used. The input to the neural network is a resized 14×14 pixel image. After testing for certain geometric properties, false matches are discarded. Positive matches corresponding to the outer ring are processed by applying the Canny edge detector and ellipse fitting, which allows computation of the 5DoF pose. Recognition of the "H" letter allows to obtain the missing yaw angle. The precision in 3D position is in the order of 1 cm to 7 cm depending on the target viewing angle and distance, which was at the maximum around 1.5 m.

Probably the most similar approach to the proposed system in this work is the TRIP localization system [17]. In TRIP, the pattern comprises of a set of several concentric rings, broken into several angular regions, each of which can be either black or white. The encoding scheme, which includes parity checking, allows the TRIP method to distinguish between 3^9 patterns. For detecting the tags, adaptive thresholding is performed and edges are extracted. TRIP only involves processing edges corresponding to projections of circular borders of the ring pattern, which are detected using a simple heuristic. These edges are used as input to an ellipse fitting method and then the concentricity of the ellipses is checked. TRIP achieves a precision similar to [16] in position estimation (the relative error is between 1 % and 3 %), but only a moderate performance (around 16 FPS at the resolution 640×480) is achieved using an 1.6 GHz machine. The authors report that the adaptive thresholding step is the most demanding portion of the computation. To the best of our knowledge, there is no publicly available implementation.

Finally, a widely used, simple and freely available circular target detector can be found in the OpenCV library. This "SimpleBlobDetector" class is based on traditional blob detection methods and includes several optional post-detection filtering steps, based on characteristics such as area, circularity, inertia ratios, convexity and center color. While this implementation is originally aimed for circular target detection, by tuning the parameters it is possible to find elliptical shapes similar to the ones proposed in the present work and thus it is compared to the proposed implementation.

In this work, a vision-based external localization system based on a circular ring (roundel) pattern is proposed. An example of the pattern is depicted in Fig. 1. The algorithm allows to initiate the pattern search anywhere in the image without any performance penalty. Therefore, the search is started from the point of the last known pattern position. Since the algorithm does not contain any phase that processes the entire image, successful tracking causes the method to process only the area occupied by the pattern. Therefore, the algorithm's computational complexity is independent of the image size. This provides a significant performance boost, which allows

to track thousands of patterns in real-time using a standard PC. By performing an initial unattended calibrating step, where the reference frame is defined, pose computation of ground robots moving on a plane is performed with millimeter precision using an off-the-shelf camera.

The real world performance of the proposed method makes it highly competitive with the aforementioned state-of-the-art methods. Moreover, its computational complexity is significantly lower, which makes the method superior for scenarios with embedded computational resources and real-time constraints. These findings are supported by the experimental results and a comparison with the selected localization methods presented in Section 6.

3 Pattern Detection

The core of the proposed computationally efficient localization system is based on pattern detection. Fast and precise detection is achieved by exploiting properties of the considered pattern that is a black and white roundel consisting of two concentric annuli with a white central disc, see Fig. 1.

The low computational requirements are met by the pattern detection procedure based on on-demand thresholding and flood fill techniques, and gathering statistical information of the pattern on the fly. The statistical data are used in consecutive tests with increasing complexity, which determine if a candidate area represents the desired circular pattern.

The pattern detection starts by searching for a black segment. Once such a segment is detected and passes the initial tests, the segment detection for a white inner disc is initiated at the expected pattern center.

Notice, that at the beginning, there is no prior information about the pattern position in the image; hence, the search for the black segment is started at a random position. Later, in the subsequent detections, when a prior pattern position is available, the algorithm starts detection over this area. For a successfully re-detected (tracked) pattern, the detection processes only pixels belonging to the pattern itself, which significantly reduces the computation burden. Since the method is robust (see following sections for detection limits), tracking is generally successful and thus the method provides very high computational performance.

After the roundel is detected, its image dimensions and coordinates are identified. Then, its three-dimensional position with respect to the camera is computed from its known dimensions and camera re-projection techniques, and its coordinates are transformed to a coordinate frame defined by the user, see Section 4.

In this section, a detailed description of the pattern detection based on an efficient thresholding is presented together with an estimation of the pattern center and dimensions and a compensation of the incorrect diameter estimation, which has a positive influence to the localization precision. Moreover, a multiple pattern detection capability is described in Section 3.6.

3.1 Segmentation

Algorithm 1: Flood-fill segmentation

Input: $(p, \rho_{exp}, class)$: p – starting pixel position; ρ_{exp} – expected area to bounding box dimensions ratio; $class$ – searched segment type (white or black)

Output: $(u, v, b_u, b_v, \mu, \text{valid})$: (u, v) – segment center; (b_u, b_v) – bounding box; μ – average brightness; valid – validity

$s_{id} \leftarrow s_{id} + 1$ // increment segment ID
$q_{old} \leftarrow q_{end}$ // store previous queue end
pixel_class$[p] \leftarrow s_{id}$ // mark pixel as processed and
queue$[q_{end}++] \leftarrow p$ // push its position to the queue
// #1 perform the flood fill search
while $q_{end} > q_{start}$ **do**
  $q \leftarrow$ queue$[q_{start}++]$ // pull pixel from the queue
  // and check its neighbours
  foreach $offset \in \{+1, -1, +w, -w\}$ **do**
    $r \leftarrow q + offset$
    if pixel_class$[r] = unknown$ **then**
      pixel_class$[r] \leftarrow$ classify(Image$[r], \tau$)
    if pixel_class$[r] = class$ **then**
      queue$[q_{end}++] \leftarrow r$
      pixel_class$[r] \leftarrow s_{id}$
      update $u_{min}, u_{max}, v_{min}, v_{max}$ from r_u, r_v
valid $\leftarrow false$
// # 2 test for the pattern size and roundness
$s \leftarrow q_{end} - q_{old}$
if $s > min\_size$ **then**
  $u \leftarrow (u_{max} + u_{min})/2$ // segment center x-axis
  $v \leftarrow (v_{max} + v_{min})/2$ // segment center y-axis
  $b_u \leftarrow (u_{max} - u_{min}) + 1$ // estimate segment width
  $b_v \leftarrow (v_{max} - v_{min}) + 1$ // estimate segment height
  $\rho \leftarrow \rho_{exp}\pi b_u b_v / 4s - 1$ // calculate roundness
  if $-\rho_{tol} < \rho < \rho_{tol}$ **then**
    $\mu \leftarrow \frac{1}{s}\sum_{j=q_{old}}^{q_{end}-1}$ Image$[j]$ // mean brightness
    valid $\leftarrow true$ // mark segment as valid

The pattern detection is based on an image segmentation complemented with on-demand thresholding that searches for a contiguous set of black or white pixels using a flood-fill algorithm depicted in

Algorithm 1. First, a black circular ring is searched for in the input image starting at an initial pixel position p_0. The adaptive thresholding classifies the processed pixel using an adaptively set value τ as either black or white. If a black pixel is detected, the queue-based flood-fill algorithm procedure is initiated to determine the black segment. The queue represents the pixels of the segment and is simply implemented as a buffer with two pointers q_{start} and q_{end}.

Once the flood fill is complete, the segment is tested for a possible match of the outer (or inner) circle of the pattern. At this point, these tests consist of a minimum size (in terms of the number of pixels belonging to the segment) and a roundness measure within acceptable bounds. Notice, that during the flood-fill search, extremal pixel positions can be stored. This allows to establish the bounding box of the segment (b_u and b_v) at any time. Besides, after finding a segment, the queue contains positions of all the segment's pixels. Hence, initial simple constraints can be validated quickly for a fast rejection of false positives.

In the case where either test fails, the detection for further segments continues by starting from the next pixel position (i.e., a pixel at the position p_0+1). However, no redundant computation is performed since the previous segment is labeled with a unique identifier.

The first roundness test is based on the pattern's bounding box dimensions and number of pixels. Theoretically, the number of pixels s of an elliptic ring with outer and inner diameters d_o, d_i and dimensions b_u, b_v should be

$$s = \pi/4 b_u b_v \frac{d_o^2 - d_i^2}{d_o^2}. \tag{1}$$

Therefore, the tested segment dimensions and area should satisfy the inequality

$$\rho_{tol} > \left| b_u b_v \pi/4 \frac{\rho_{exp}}{s} - 1 \right|, \tag{2}$$

where ρ_{exp} equals 1 for white and $1 - d_i^2/d_o^2$ for black segments. The value of ρ_{tol} represents a tolerance range, which depends on the camera radial distortion and possible pattern deformation and spatial orientation.

If a black segment passes the roundness test, the second flood-fill search for the inner white segment is initiated from the position corresponding to the segment centroid. If the inner segment passes the minimum size and roundness tests, further validation tests are performed. These involve the concentricity of both segments, their area ratio, and a more sensitive circularity measure (discussed in the following sections). If the segments pass all these complex tests, the pattern is considered to be found and its centroid position will be used as a starting point p_0 for the next detection run. The overall pattern detection algorithm is depicted in Algorithm 2.

Algorithm 2: Pattern detection

Input: $(p_0, \tau, \texttt{Image})$: p_0 – position to start search; τ – threshold; `Image` being processed
Output: (c, p_0, τ): c – the pattern data; p_0 – position to start the next search; τ – an updated threshold

$s_{id} \leftarrow 0$; $i \leftarrow p_0$ // initialize
// #1 search throughout the image
repeat
  if `pixel_class`$[i] = unknown$ **then**
    if classify(`Image`$[i], \tau) =$ black **then**
      `pixel_class`$[i] \leftarrow$ black
  // initiate pattern search
  if `pixel_class`$[i] =$ black **then**
    // search for outer ring
    $q_{end} \leftarrow q_{start} \leftarrow 0$
    $c_{\text{outer}} \leftarrow$ flood-fill_seg$(i, \rho_{outer}, \text{black})$
    if valid(c_{outer}) **then**
      // search for inner ellipse
      $j \leftarrow$ center(c_{outer})
      $c_{\text{inner}} \leftarrow$ flood-fill_seg$(j, \rho_{inner}, \text{white})$
      if valid(c_{inner}) **then**
        // test area ratio (no. of pixels):
        $s_{outer} = |c_{outer}|$, $s_{inner} = |c_{outer}|$
        if $\frac{s_{outer}}{s_{inner}} \approx \frac{d_o^2 - d_i^2}{d_i^2}$ **then**
          check segments for concentricity
          compute ellipse semiaxes $\mathbf{e_0}, \mathbf{e_1}$
          if $q_{end} \approx \pi |e_0 e_1|$ **then**
            assign segment ID or
            compensate illumination
            mark segment as valid
            break
  $i \leftarrow (i+1) \bmod$ sizeof(`Image`) // go to next pixel
until $i \neq p_0$;
// #2 set the thresholding value
if valid(c_{inner}) **then**
  $\tau \leftarrow (\mu_{outer} + \mu_{inner})/2$
  // hide pattern in multiple pattern detection
  paint over all inner ellipse pixels as black;
else
  $\tau \leftarrow$ binary search sequence
// #3 perform the cleanup
if *only two segments examined* **then**
  reset `pixel_class`[] inside bounding box of c_{outer}
else
  reset entire `pixel_class`[]

3.2 Efficient Thresholding

Since the segmentation looks only for black or white segments, the success rate of the roundel

detection depends on the threshold parameter τ, especially under various lighting conditions. Therefore, we proposed to adaptively update τ whenever the detection fails according to a binary search scheme over the range of possible values. This technique sets the threshold τ consecutively to values $\{1/2, 1/4, 3/4, 1/8, 3/8, 5/8 \ldots\}$ up to a pre-defined granularity level, when τ is reset to the initial value.

When the pattern is successfully detected, the threshold is updated using the information obtained during detection in order to iteratively improve the precision of segmentation:

$$\tau = \frac{\mu_{outer} + \mu_{inner}}{2}, \tag{3}$$

where μ_{outer}, μ_{inner} correspond to the mean brightness value of the outer and inner segments, respectively.

The computationally intensive full image thresholding is addressed by on-demand processing over each pixel analyzed during the detection. At the very first access, the RGB values of the image are read and a pixel is classified as either black or white and the classification result is stored for further re-use in the subsequent steps. Moreover, whenever the tracking is successful, only the relevant pixels are thresholded and processed by the two-step flood fill segmentation. Clearing the per-pixel classification memory area is also efficiently performed by only resetting the values inside the pattern's bounding box. As a result, the detection step is not directly dependent on the input image resolution, which provides a significant performance gain. If the tracking is not successful, extra memory accesses resulting from this on-demand strategy are negligible compared to a full-image thresholding approach.

3.3 Pattern Center and Dimensions

After the black and white segments pass all the initial tests, a more sophisticated roundel validation is performed. The validation is based on a more precise roundness test using estimation of the ellipse (pattern) semiaxes. All the information to calculate the ellipse center u, v and the semiaxes $\mathbf{e_0}, \mathbf{e_1}$ is at the hand, because all the pattern pixels are stored in the flood-fill queue. Hence, the center is calculated as the mean of the pixel positions. After that, the covariance matrix $\mathbf{C}$, eigenvalues λ_0, λ_1, and eigenvectors $\mathbf{v_0}, \mathbf{v_1}$ are established. Since the matrix $\mathbf{C}$ is two-dimensional, its eigen decomposition is a matter of solving a quadratic equation. The ellipse semiaxes e_0, e_1 are calculated simply by

$$\mathbf{e_i} = 2\lambda^{1/2}{}_i \mathbf{v_i}. \tag{4}$$

The final test verifying the pattern roundness is performed by checking if the inequality

$$\rho_{prec} > \left| \pi \frac{|\mathbf{e_0}||\mathbf{e_1}|}{s} - 1 \right| \tag{5}$$

holds, where s is the pattern size in the number of pixels. Unlike in the previous roundness test (2), the tolerance value of ρ_{prec} can be much lower because (4) establishes the ellipse dimensions with subpixel precision.

Here, it is worth mentioning that if the system runs on embedded hardware, it might be desirable to calculate $\mathbf{C}$ using integer arithmetic only. However, the integer arithmetic might result in a loss of precision, therefore $\mathbf{C}$ should be calculated as

$$\mathbf{C} = \frac{1}{s} \sum_{i=0}^{s-1} \begin{pmatrix} u_i u_i & u_i v_i \\ u_i v_i & v_i v_i \end{pmatrix} - \begin{pmatrix} uu & uv \\ uv & vv \end{pmatrix}, \tag{6}$$

where u_i and v_i are the pattern's pixel coordinates stored in the queue and u, v denote the determined pattern center.

3.4 Pattern Identification

The ratio of the patterns' inner and outer diameters does not have to be a fixed value, but can vary between the individual patterns. Therefore, the variable diameter ratio can be used to distinguish between individual circular patterns. If this functionality is required, the system user can print patterns with various diameter ratios and use these ratios as ID's.

However, this functionality requires to relax the tolerance ranges for the tests of inner/outer segment area ratio, which might (in an extreme case) cause false positive detections. Variable inner circle dimensions also might mean a smaller inner circle or a thinner outer ring, which might decrease the maximal distance at which the pattern is detected reliably. Moreover, missing a priori knowledge of the pattern's diameter ratio means that compensation for incorrect diameter estimation is not possible, which might slightly decrease the method's precision.

3.5 Compensation of Incorrect Diameter Estimation

The threshold separating black and white pixels has a significant impact on the estimation of the pattern dimensions. Moreover, the pixels on the black/white border are affected by chromatic aberration, nonlinear camera sensitivity, quantization noise, and image compression, see Fig. 2. As a result, the borderline between the black ring and its white background contains a significant number of misclassified pixels.

The effect of pixel misclassification is observed as an increase of the ratio of white to black pixels with increasing pattern distance. The effect causes the black ring to appear thinner (and smaller), which has a negative impact on the distance estimation. However, the inner and outer diameters of the pattern are known, and therefore, the knowledge of the true d_o and d_i can be used to compensate for the aforementioned effect. First, we can establish the dimensions of the inner white ellipse e_{0i} and e_{1i} in the same way as in Section 3.3. We assume the pixel misclassification enlarges the inner ellipse semiaxes $\mathbf{e_{0i}}$, $\mathbf{e_{1i}}$ and shrinks the outer semiaxes $\mathbf{e_{0o}}$, $\mathbf{e_{1o}}$ by a value of t. Since the real inner d_i and outer d_o pattern diameters are known, the true ratio of the areas can be expressed as

$$\frac{d_i^2}{d_o^2} = r = \frac{(e_{0i} - t)(e_{1i} - t)}{(e_{0o} + t)(e_{1o} + t)}, \tag{7}$$

where t can be calculated as a solution of the quadratic equation

$$(1-r)t^2 - t(e_{0i}e_{1i} + re_{0o}e_{1o}) + e_{0i}e_{1i} - re_{0o}e_{1o} = 0. \tag{8}$$

The ambiguity of the solution can be resolved simply by taking into account that the corrected semiaxes lengths $e_{0i} - t$, $e_{1i} - t$ must be positive. The compensation of the pattern diameter reduces the average localization error by approximately 15 %.

3.6 Multiple Target Detection

The described roundel detection method can also be used to detect and track several targets in the scene. However, a single threshold τ is not well suited to detecting more patterns because of illumination variances. Besides, other differences presented across the working area may affect the reflectance of the pattern and thus result in different gray levels for different patterns, which in turn requires a different τ value for each pattern. Individual thresholding values not only provide detection robustness but also increase precision by optimizing pixel classification for each target individually.

Multiple targets can be simply detected in a sequence one by one, and the only requirement is to avoid detection of the already detected pattern. This can be easily avoided by modifying the input image after a successful detection by painting over the corresponding pixels, i.e., effectively masking out the pattern for subsequent detection runs.

Detection of multiple targets can also be considered in parallel, e.g., for obtaining additional performance gain, using a multi core processor. In this case, it is necessary to avoid a possible race condition and mutual exclusion has to be used for accessing the classification result storage.

An initial implementation of the parallel approach using OpenMP and multi-processor system did not yield a significant speedup. Furthermore, due to the

Fig. 2 Undesired effects affecting the pattern edge

high performance of detection of a single pattern, the serial implementation provides better performance than the parallel approach. Therefore, all the presented computational results in this paper are for the serial implementation.

4 Pattern Localization

The relative pattern position to the camera module is calculated from the parameters established in the previous step. We assume that the radial distortion of the camera is not extreme and the camera intrinsic parameters can be established by the method [21] or similar. With this assumption, the pattern's position is computed as follows:

1. The ellipse center and semiaxes are calculated from the covariance matrix eigenvectors and transformed to a canonical camera coordinate system.
2. The transformed parameters are then used to establish coefficients of the ellipse characteristic equation, which is a bilinear form matrix (also called a cubic).
3. The pattern's spatial orientation and position within the camera coordinate frame is then obtained by means of eigen analysis of the cubic.
4. The relative coordinates are transformed to a two- or three-dimensional coordinate frame defined by the user.

A detailed description of the pattern position estimation is presented in the following sections.

4.1 Ellipse Vertices in the Canonical Camera System

The ellipse center u'_c, v'_c and semiaxes $\mathbf{e'_0}$, $\mathbf{e'_1}$ are established in a canonical camera form. The used canonical form is a pinhole camera model with unit focal lengths and no radial distortion. The transformation to a canonical camera system is basically a transform inverse to the model of the actual camera.

First, we calculate the image coordinates of the ellipse vertices $\mathbf{a_{0,1}}$ and co-vertices $\mathbf{b_{0,1}}$, and transform them to the canonical camera coordinates $\mathbf{a'_{0,1}}$, $\mathbf{b'_{0,1}}$. The canonical coordinates of the (co)vertices are then used to establish the canonical center and canonical semiaxes. This rather complicated step is performed to compensate for the radial distortion of the image at the position of the detected ellipse.

Since the ellipse center $\mathbf{u}$ and semiaxes $\mathbf{e_0}$, $\mathbf{e_1}$ are known, calculation of the canonical vertices $\mathbf{a'_{0,1}}$ and co-vertices $\mathbf{b'_{0,1}}$ is done simply by adding the semiaxes to the ellipse center and transforming them:

$$\begin{aligned}\mathbf{a'_{0,1}} &= g'\left((u \pm e_{0x} - c_x)/f_x, (v \pm e_{0y} - c_y)/f_y\right)\\ \mathbf{b'_{0,1}} &= g'\left((u \pm e_{1x} - c_x)/f_x, (v \pm e_{1y} - c_y)/f_y\right)\end{aligned},$$

where g' is the radial undistortion function and $f_{x,y}$, $c_{x,y}$ are the camera focal lengths and optical center, respectively. Using the canonical position of the ellipse vertices, the ellipse center u', v' and axes $\mathbf{e'_0}$, $\mathbf{e'_1}$ are then calculated as

$$\begin{aligned}\mathbf{e'_0} &= (\mathbf{a'_0} - \mathbf{a'_1})/2\\ \mathbf{e'_1} &= (\mathbf{b'_0} - \mathbf{b'_1})/2\\ \mathbf{u'_c} &= (\mathbf{a'_0} + \mathbf{a'_1} + \mathbf{b'_0} + \mathbf{b'_1})/4\end{aligned}.$$

After this step, we have all essential variables to calculate the ellipse characteristic equation.

4.2 Ellipse characteristic equation

Notice that each point u, v lying on an ellipse satisfies the characteristic equation of an ellipse:

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}^T \begin{pmatrix} q_a & q_b & q_d \\ q_b & q_c & q_e \\ q_d & q_e & q_f \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \mathbf{X^T Q X} = 0, \quad (9)$$

where $\mathbf{Q}$ is called a conic. Thus, the parameters of the matrix $\mathbf{Q}$ are calculated from the ellipse center and axes as follows:

$$\begin{aligned}q_a &= +e'_{0u}e'_{0u}/|\mathbf{e'_0}|^2 + e'_{0v}e'_{0v}/|\mathbf{e'_1}|^2\\ q_b &= +e'_{0u}e'_{0v}/|\mathbf{e'_0}|^2 - e'_{0u}e'_{0v}/|\mathbf{e'_1}|^2\\ q_c &= +e'_{0u}e'_{0u}/|\mathbf{e'_1}|^2 + e'_{0v}e'_{0v}/|\mathbf{e'_0}|^2\\ q_d &= -u'_c q_a - v'_c q_b\\ q_e &= -u'_c q_b - v'_c q_c\\ q_f &= +q_a u'^2_c + q_c v'^2_c + 2q_b u'_c v'_c - 1\end{aligned}. \quad (10)$$

4.3 Pattern Position

Once the conic parameters $\mathbf{Q}$ are known, the position and orientation of the pattern can be obtained

by means of eigenvalue analysis [22]. Let the $\mathbf{Q}$ matrix eigenvalues and eigenvectors be $\lambda_0, \lambda_1, \lambda_2$ and $\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}$, respectively. Since $\mathbf{Q}$ represents an ellipse, its signature is (2, 1) and we assume that $\lambda_0 \geq \lambda_1 > 0 > \lambda_2$. According to [16], the position of the circle can be calculated as:

$$\mathbf{x_c} = \pm \frac{d_o}{\sqrt{-\lambda_0 \lambda_2}} \left(\mathbf{q_0} \lambda_2 \sqrt{\frac{\lambda_0 - \lambda_1}{\lambda_0 - \lambda_2}} + \mathbf{q_2} \lambda_0 \sqrt{\frac{\lambda_1 - \lambda_2}{\lambda_0 - \lambda_2}} \right),$$

where d_o is the circular pattern diameter. The ambiguity of the sign can be resolved by taking into account that the pattern is located within the camera field of view. Thus, if the first component of the $\mathbf{x_c}$ vector is negative, the vector $\mathbf{x}$ is simply inverted.

4.4 Transformation to the Global Coordinates

The position $\mathbf{x_c}$ of the circular pattern established in the previous step is in a camera centered coordinate frame. Depending on the particular application scenario, our system allows to transform the pattern coordinates to a 3D or 2D coordinate frame defined by the user. The user just places four circular patterns in the space covered by the camera and provides the system with their real positions.

4.4.1 Global Coordinate Frame – 3D Case

In the case of the 3D localization, the three patterns at positions $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}$ define the coordinate origin and x and y axes, respectively. The transformation between the global $\mathbf{x} = (x, y, z)^T$ and camera centered $\mathbf{x_c} = (x_c, y_c, z_c)^T$ coordinate systems can be represented as

$$\mathbf{x} = \mathbf{T} (\mathbf{x_c} - \mathbf{t_0}),$$

where $\mathbf{t_0}$ equals $\mathbf{x_0}$ and $\mathbf{T}$ is a similarity transformation matrix.

The user can define the coordinate system simply by putting three "calibration" patterns in the camera field of view and designating the pattern that defines the coordinate system origin $\mathbf{t_0}$ and the x and y axes. Using the pattern positions (let us define them as $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}$, respectively), the system calculates the transformation between the camera and global coordinate systems, i.e., the vector $\mathbf{t_0}$ and matrix $\mathbf{T}$. Establishing the translation vector $\mathbf{t}$ is straightforward – it corresponds to the camera coordinates of the pattern at the global coordinate origin, i.e., $\mathbf{t} = \mathbf{x_0}$.

The x and y axes of the coordinate frame are defined by vectors $\mathbf{t_1} = \mathbf{x_1} - \mathbf{x_0}$ and $\mathbf{t_2} = \mathbf{x_2} - \mathbf{x_0}$, respectively. Since we assume an orthonormal coordinate system, the z axis vector can be simply calculated as a cross product $\mathbf{t_3} = \mathbf{t_1} \times \mathbf{t_2}$. From an algebraic point of view, the matrix $\mathbf{T}$ represents a transformation of the vector $\mathbf{x}' = \mathbf{x} - \mathbf{t}$ to a coordinate system defined by the basis $\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}$. Therefore, the matrix $\mathbf{T}$ can be calculated simply as

$$\mathbf{T} = \begin{pmatrix} t_{1x} & t_{2x} & t_{3x} \\ t_{1y} & t_{2y} & t_{3y} \\ t_{1z} & t_{2z} & t_{3z} \end{pmatrix}^{-1}. \tag{11}$$

Having established the vector $\mathbf{t}$ and matrix $\mathbf{T}$, any point in the camera coordinate frame can be transformed to the coordinate frame defined by the user.

When the user places four patterns in the camera field of view, four independent coordinate transformations are calculated using each pattern triplet. The pattern position x' is then calculated as their mean, which results in increased system accuracy.

4.4.2 Global Coordinate Frame – 2D Case

Two-dimensional localization can be generally more precise than full three-dimensional localization. This is because the estimation of the pattern position depends mainly on the pattern distance, especially in cases when the pattern image is small. Estimation of the pattern distance can be simply avoided if all the patterns are located only in a plane, e.g., ground robots operating on a floor.

In this case, the transformation from the image coordinates to an arbitrary world plane is a homography, and (homogeneous) spatial coordinates $\mathbf{x}$ of the patterns can be calculated directly from their canonical coordinates $\mathbf{u}'$ simply by $\mathbf{x} = \mathbf{H}\mathbf{u}'$, where $\mathbf{H}$ is a 3×3 homography matrix. Similarly to the case of three-dimensional localization, the user can define $\mathbf{H}$ just by placing four patterns in the camera field of view and providing the system with their positions in the desired coordinate frame.

5 Sensor Model

In this section, we present three mathematical models that can be used to estimate the expected performance of the system. The main purpose of these models is

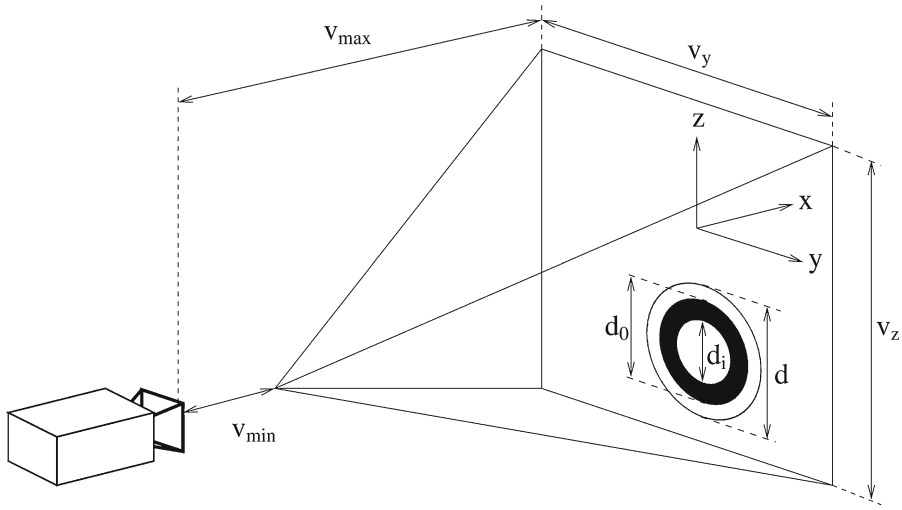


Fig. 3 Geometry of the operational space

to support the selection of the most suitable camera, processing hardware, and pattern size according to the particular application scenario. The first model calculates the localization system coverage from the pattern dimensions, camera resolution, and field of view. The second set of equations provides estimation of the localization precision based on the camera parameters, pattern dimensions, and required coverage. Finally, the third model estimates the necessary computational power to track the given number of patterns at the desired frame rate.

5.1 Localization System Coverage

Regarding the practical deployment of the localization system, its most critical property is its coverage or "operational space", i.e., the space where the pattern is reliably detected and localized. The dimensions of the operational space are affected by the camera focal length and radial distortion, image resolution, pattern diameter, and pattern spatial orientation.

For the sake of simplicity, the effect of radial distortion on the shape of the operational space is neglected and an ideal pinhole camera is assumed. Considering this ideal model, the operational space has a pyramidal shape with its apex close to the camera, see Fig. 3. The parameters of the operational space are the minimal and maximal detectable distances v_{min}, v_{max} and base dimensions v_y, v_z.

A pattern can be detected if it "fits" in the image and its central part and black ring are recognizable. Therefore, the pattern image dimensions must be lower than the camera resolution, but higher than a certain value. To estimate the dimensions, we assume the camera focal lengths f_x, f_y and radial distortion parameters have been established by a calibration tool[1], e.g., MATLAB calibration toolbox or similar software based on [21]. Then, the width and height w_p, h_p of the pattern in pixels can be calculated by

$$w_p = f_x \frac{d_o}{x} \cos(\varphi), \qquad h_p = f_y \frac{d_o}{x} \cos(\psi), \quad (12)$$

where x is the pattern distance from the image plane, d_o is the pattern diameter, and φ and ψ represent the pattern tilt.

5.1.1 Minimal Localization Distance

The minimal distance v_{min}, at which the pattern can be detected regardless of its orientation, is given as

$$v_{min} = d_o \max\left(\frac{f_x}{w}, \frac{f_y}{h}\right), \quad (13)$$

where w and h is the image horizontal and vertical resolution in pixels, respectively. One has to realize that the fractions f_x/w and f_y/h correspond to the camera field of view. Hence, the camera field of view remains the same regardless of the current resolution settings and the distance v_{min} can be considered as independent of the camera resolution.

[1]Such a tool is also a part of the proposed system available online at [23].

5.1.2 Maximal Localization Distance

The pattern has to be formed from a sufficient number of pixels to be detected reliably. Therefore, the pattern pixel dimensions have to exceed a certain value that we define as D. The value of D has been experimentally established as 12. We also found that D might be lower than this threshold for exceptionally good lighting conditions; however, $D = 12$ represents a conservative value. Having D, the maximal detectable distance v'_{max} of the pattern can be calculated as

$$v'_{max} = \frac{d_o}{D} \min(f_x \cos(\varphi),\ f_y \cos(\psi)). \tag{14}$$

Notice a higher camera resolution increases the focal lengths f_x and f_y; so, setting the camera resolution as high as possible maximizes the area covered by the localization system.

On the other hand, (14) does not take into account the camera radial distortion and it is applicable only when the pattern is located near the optical axis. The radial distortion causes the objects to appear smaller as they get far away from the optical axis. Thus, the distance v'_{max} at which the pattern is detected along the optical axis is higher than the maximal detectable distance v_{max} of the pattern located at the image corners. Therefore, the dimension v_{max} of the operational space is smaller than v'_{max} by a certain factor and v_{max} can be calculated as

$$v_{max} = \frac{d_o}{D} \min(k_x f_x \cos(\varphi), k_y f_y \cos(\psi)), \tag{15}$$

where k_x and k_y represent the effect of the radial distortion. The values of k_x and k_y can be estimated from the differential of the radial distortion function close to an image corner:

$$\begin{aligned} k_x &= 1 + \frac{dg(r_x, r_y)}{dx} \\ &= 1 + 2k_1 r_x + 4k_2(r_x^3 + r_x r_y^2) + \ldots \\ k_y &= 1 + \frac{dg(r_x, r_y)}{dy} \\ &= 1 + 2k_1 r_y + 4k_2(r_y^3 + r_y r_x^2) + \ldots \end{aligned},$$

where r_x and r_y can be obtained from the camera optical axis and focal lengths as c_x/f_x and c_y/f_y. For a consumer grade camera, one can assume that the radial distortion would not shrink the pattern more than by 10 %; so, a typical value of $k_{x,y}$ would be between 0.9 and 1.0.

5.1.3 Base Dimensions

Knowing the maximal detectable distance v_{max}, the dimensions of the localization area "base" v_y and v_z can be calculated as

$$v_y = w\frac{v_{max}}{f_x} - 2d_o, \qquad v_z = h\frac{v_{max}}{f_y} - 2d_o, \tag{16}$$

where w and h are the horizontal and vertical resolutions of the camera used, respectively. Considering a typical pattern, the value of d_o is much smaller than the localization area and can be omitted.

With Equations (13), (15), and (16) the user can calculate the diameter of the pattern and camera parameters from the desired coverage of the system. It should be noted that the presented model considers a static configuration of the module and the detected pattern. Rapid changes of the pattern's relative position may cause image blur, which might affect v_{max} and restrict the operational space.

5.2 Localization System Precision

Another important property of the localization system is the precision with which the system provides estimation of the pattern position. The precision of the localization is directly influenced by the amount of noise in the image and uncertainty in the camera parameters. The position estimation error also depends on the system operational mode, i.e., it is different for the three-dimensional and two-dimensional position estimations. The expected localization precision is discussed in the following sections for both the 2D and 3D cases.

5.2.1 Two-dimensional Localization by Homography

For the 2D localization, the pattern position is estimated simply from its center image coordinates. In the case of a ideal pinhole camera, the calibration procedure described in Section 4.4 should establish the relation between the image and world planes. Therefore, the precision of the position estimation is affected mainly by the image radial distortion. Since the uncertainties of the radial distortion parameters are

known from the camera calibration step, the error of radial distortion for x and y can be estimated from the differential of the radial distortion function

$$\begin{aligned}\eta_x &= x(\epsilon_1 r+\epsilon_2 r^2+\epsilon_5 r^3+2\epsilon_3 y)+\epsilon_4(r+2x^2)\\ \eta_y &= y(\epsilon_1 r+\epsilon_2 r^2+\epsilon_5 r^3+2\epsilon_4 x)+\epsilon_3(r+2y^2)\end{aligned}, \quad (17)$$

where η_x and η_y are the position relative errors, k_i are camera distortion parameters, ϵ_i are their uncertainties, and $r = x^2 + y^2$. The overall relative error of the two-dimensional localization can be expressed as

$$\eta_{hom} = \eta_{rad} = \sqrt{\eta_x^2 + \eta_y^2}. \quad (18)$$

Note that (17) does not take into account the camera resolution. Therefore, the model suggests that higher resolution cameras will not necessarily achieve better localization precision. This is further investigated in Section 6.2.2, where experimental results are presented. Also, note that in the standard camera calibration implementations, values of ϵ_i are meant as 99.7 % confidence intervals. To calculate the average error, i.e., the standard deviation, one has to divide η_{hom} by three.

5.2.2 Full Three-dimensional Localization

In the full 3D localization, the main source of the localization imprecision is incorrect estimation of the pattern distance. Since the pattern distance is inversely proportional to its diameter in pixels, smaller patterns will be localized with a higher error. The error in the diameter measurement is caused by quantization noise and by the uncertainty in the identification of the camera's intrinsic parameters, especially in the parameters of the image radial distortion. One can roughly estimate the expected error in the pattern distance estimation as

$$\eta_{3D} = \frac{\Delta_f}{f_x} + \Delta e_0 \frac{x f_x}{d_0} + \eta_{rad}, \quad (19)$$

where Δf is the error of the focal length estimation, Δe represents the error of the ellipse axis estimation due to image noise, and η_{rad} is the relative error of the radial distortion model. While Δ_f and η_{rad} can be calculated from the camera calibration parameters, Δe_0 is influenced by a number of factors that include camera thermal noise, lighting, motion blur etc. However, its current value can be estimated on the fly from the variance of the calibration (see Section 4.4.1) patterns' diameters.

In our experiments, the typical value of Δe_0 was around 0.15 pixels. This means that for a well-calibrated camera, the major source of distance estimation error is the ratio of image noise to the pattern projection size. Since the pattern image size (in the number of pixels) grows with the camera resolution, the precision of localization can be increased simply by using a high resolution camera or a larger pattern.

5.3 Computational Requirements

From a practical point of view, it is also desirable to estimate the necessary computational hardware needed to achieve a desired frame rate, especially for an embedded solution. The time needed to process one image can be roughly estimated from the number of patterns, their expected size, image dimensions, tracking failure rate, and the computer speed. For the sake of simplicity, we can assume that the time to process one frame is a linear function of the amount of processed pixels:

$$t = \big(k_0 + k_1(s_p(1-\alpha) + s_i\alpha)\big)\, no, \quad (20)$$

where k_0 represents the number of operations needed per pattern regardless of its size (e.g., a coordinate transformation), k_1 is a constant corresponding to the number of operations per pixel per pattern, s_p is the average size of the pattern in pixels, α is the expected failure rate of the tracking, s_i is the image size in pixels, n is the number of tracked patterns, and o is the number of operations per second per processor core given as a ratio $o = c/m$ of the entire processor MIPS (Million Instructions Per Second) m and the number of processor cores c. The constant k_0 has been experimentally estimated as 5.10^5 and k_1 as 900. The average size s_p of a pattern can be calculated from the camera parameters, pattern diameter, and average distance from the camera by (12). Thus, if the user wants to track 50 patterns with 30 pixel diameter using a machine with two cores and 53 GIPS (Giga Instructions Per Second), the expected processing time per image would be 1.2 ms, which would allow to process about 800 images per second.

The speed of the localization algorithm depends on the failure rate of the tracking α. Typically, if the pattern displacement between two frames is smaller than the pattern radius, the tracking mechanism causes the method to process only the pixels belonging to the pattern. This situation corresponds to α being equal to

zero. Thus, assuming that the pattern is not moving erratically, the method's computational complexity is independent of the processed image size. Moreover, the smaller the pattern image dimensions, the faster the processing rate is. Of course, equation (20) gives only a coarse estimate, but it might give the user a basic idea of the system processing speed. Equation (20) has been experimentally verified and the results are presented in Section 6.3.

6 Experiments

This section is dedicated to presentation of the experimental results verifying the mathematical models established in Section 5. First, the model of the operational space defined by (15) and (16) is tested to see if it corresponds to a real situation. After that, the real achievable precision of the localization is evaluated according to the model (17) and (19). Then, the real computational requirements of the algorithm are measured using different computational platforms and the model in (20) is validated. Finally, the performance of the proposed localization system is also evaluated according to the precise motion capture system and compared with the AR tag based approach ArUco [14] and the simple OpenCV circle detector.

6.1 Operational Space for a Reliable Pattern Detection

The purpose of this verification is to validate the model describing the area covered by the localization system. In Section 5.1, the covered space is described as a pyramid with base dimensions v_y, v_z and a height denoting the maximal detectable distance v_{max}.

6.1.1 Maximal Detection Distance

A key parameter of the operational space is the maximal distance for reliable pattern detection v_{max} that is described by (14). The following experimental setup has been used to verify the correctness of this model. Two different cameras have been placed on a mobile platform SCITOS-5 with precisely calibrated odometry. The proposed localization system was set up to track three circles, each with a different diameter. The platform has been set to move away from the circles at a constant speed and its distance from the patterns was recorded whenever a particular pattern was not detected. The recorded distances are considered as the limit v'_{max} of the system operational space. The same procedure was repeated with the patterns being slanted by forty degrees (Table 1).

During this experiment, the patterns were located approximately at the image center. As previously noted in Section 5.1.2, additional correction constants k_x, k_y have been introduced in Section 5.1.2 to take into account radial distortion effects, which cause the detected pattern to appear smaller when located at the image edges. The augmented model considering the radial distortion was verified in an additional experiment using a pattern with diameter 2.5 cm positioned at the image corner. In this case, the maximal detected distance was reduced by 7 % for a Logitech QuickCam Pro camera and by 7 % for an Olympus VR-340. These results are in a good accordance with the model introduced in Section 5.1.2, where the values of k_x and k_y were estimated to be between 0.9 and 1.0.

6.1.2 Base Dimensions

The dimensions of the coverage base are modeled by (16), which provides the dimensions of the expected coverage v_y and v_z. This model was verified using a similar setup to the previous experiment. The camera was placed to face a wall at a distance established in the previous experiment and four patterns were placed at the very corners of the image. This procedure was repeated for three different sizes of the pattern. The operation space dimensions, both measured and calculated by (16), are summarized in Table 2.

Table 1 Maximal distance for a reliable pattern detection

Camera type	Pattern $d_o[cm]$	Distance [m]			
		Measured		Predicted	
		0°	40°	0°	40°
Logitech QC Pro	2.5	1.4	1.3	1.6	1.3
	5.0	3.3	2.8	3.2	2.5
	7.5	4.4	3.9	4.9	3.7
Olympus VR-340	2.5	6.8	6.2	6.7	5.1
	5.0	13.2	11.4	13.4	10.3
	7.5	19.8	16.8	20.1	15.4

Table 2 Dimensions of the operational space

d_o [cm]	Dimensions [m]					
	Measured			Predicted		
	v_{max}	v_y	v_z	v_{max}	v_y	v_z
2.5	1.6	2.1	1.6	1.5	1.9	1.4
5.0	3.0	3.9	3.0	3.0	4.0	3.0
7.5	4.5	5.9	4.5	4.4	5.8	4.4

6.2 Localization Precision

The real localization precision, which is probably the most critical parameter of the localization system, was established experimentally using a dataset collected in the main entrance hall of the Faculty of Mechanical Engineering at the Charles square campus of the Czech Technical University. The entrance hall offers enough space and its floor tiles form a regular rectangular grid with dimensions 0.625 × 1.250 m. The regularity of the grid was verified by manual measurements and the established precision of the tile placement is around 0.6 mm.

We placed several patterns on the tile intersections and took five pictures with three different cameras from two different viewpoints (Figs. 4 and 5). The cameras used were a Creative Live! webcam, Olympus VR-340, and Canon 550D set to 1280×720, 4608×3456, and 5184×3456 pixel resolutions, respectively. The viewpoints were chosen at two different heights; so, the images of the scene were taken from a "side" and a "top" view.

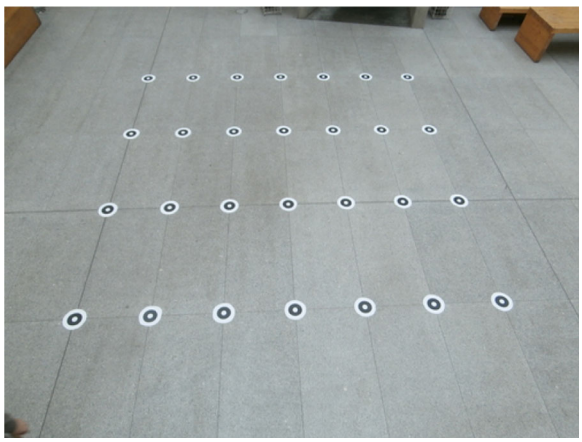


Fig. 4 Side view of the experiment

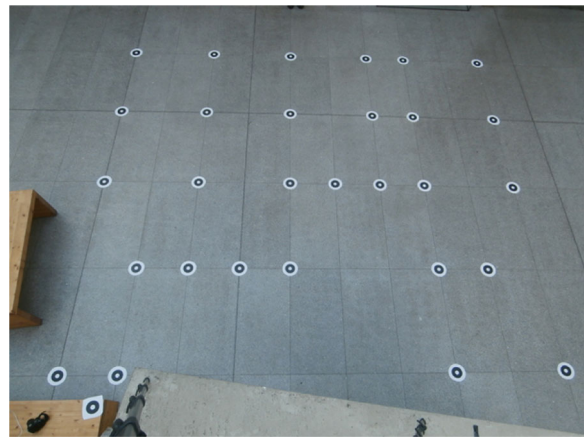


Fig. 5 Top view of the experiment

First, three or four of the patterns in each image were used to define the coordinate system. Then, the resulting transformation was utilized to establish the circle global positions. Since the circles were placed on the tile corners, their real positions were known precisely. The Euclidean distances of these known positions to the ones estimated by the system were considered as the measure of the localization error.

6.2.1 Three-dimensional Localization Precision

In this test, the system was set to perform full three-dimensional localization. In this model, the most significant cause of the localization error is the wrong distance estimation of the pattern (as noted in Section 5.2.2). The distance measurement is caused by an imperfect estimation of the pattern semiaxes lengths, see (19). The equation indicates that a camera with a higher resolution would provide a better precision.

The measured and predicted average and maximal localization errors for the individual pictures are shown in Table 3. The table also contains the predicted average localization error η_{pred} calculated by (3) for a comparison of the model and the real achieved precision.

6.2.2 Two-dimensional Localization Precision

In the case of indoor ground robot localization, we can assume that the robots move in a plane. The plane where the robots move and the image plane form a homography, which was previously defined by

Table 3 Precision of 3D position estimation

Image		Abs. [cm]		Rel. [%]		
camera	view	ϵ_{avg}	ϵ_{max}	η_{pred}	η_{avg}	η_{max}
Webcam	side	5.7	19.5	1.04	0.90	2.96
Webcam	top	3.7	12.1	0.68	0.61	1.83
VR-340	side	1.9	6.5	0.47	0.35	1.02
VR-340	top	3.2	11.0	0.54	0.50	1.39
C-550D	top	2.5	7.4	0.30	0.43	1.46

four reference patterns during the system setup. The real achievable precision of two-dimensional localization was measured within the same experimental scenario as the previous full 3D case. The average and maximal measured localization errors are depicted in Table 4. Similar to the previous case, the table contains the predicted mean error η_{pred} calculated by (17).

The results indicate that the assumption of ground plane movement increases the precision by an order of magnitude. Moreover, the results also confirm that increasing the image resolution does not necessarily increase the localization precision. Rather, the precision of localization is influenced mostly by the camera calibration imperfections. This fact confirms the assumptions presented in Section 5.2.1.

6.3 Computational Requirements

The purpose of this experiment was to evaluate the estimation of the computational requirements provided by the model proposed in Section 5.3. Thus, the hypothesis is to test if the algorithm processing speed estimation (20) conforms to the proposed assumptions. Moreover, in this experiment, we also verify if the algorithm complexity depends only on the pattern size rather than on the image resolution.

Table 4 Precision of 2D position estimation

Image		Abs. [cm]		Rel. [%]		
camera	view	ϵ_{avg}	ϵ_{max}	η_{pred}	η_{avg}	η_{max}
Webcam	side	0.23	0.62	0.03	0.04	0.08
Webcam	top	0.18	0.68	0.04	0.03	0.09
VR-340	side	0.64	1.40	0.11	0.12	0.22
VR-340	top	0.68	2.08	0.19	0.11	0.32
C-550D	top	0.15	0.33	0.03	0.03	0.07

6.3.1 Processing Time vs. Image and Pattern Dimensions

The model of computational requirements assumes that once the circles are reliably tracked, the system processing time is independent of the image size. In such a case, the image processing time is a linear function of the overall number of pixels belonging to all the patterns. Three synthetic datasets were created to verify this assumption. The first dataset consists of images with variable resolution and one circular pattern with a fixed size. The image resolutions of the second dataset are fixed, but the pattern diameter varies. Both pattern and image dimensions of the third dataset images are fixed; however, the number of patterns in each image ranges from one to four hundred. Each image of each dataset was processed one thousand times and the average time to track all the roundels in the image was calculated. The average processing time is shown in Fig. 6.

The presented results clearly show that the image processing time is proportional to the number of pixels occupied by the tracked circular patterns and does not depend on the processed image dimensions. Moreover, the results demonstrate the scalability of the algorithm, which can track four hundred robots more than one hundred times per second. The aforementioned tests were performed on a single core of the Intel iCore5 CPU running at 2.5 GHz and accompanied with 8 GB of RAM.

6.3.2 Processing Time Using Different Platforms

From a practical point of view, processing images at a speed exceeding the camera frame rate is not necessary. Rather, the algorithm might be deployed on systems with slower processing units. Thus, one should be able to establish what kind of computational hardware is needed for a particular setup. This can be roughly estimated using the time to process one image by means of (20). Three real world datasets and five different platforms, including two credit-card sized computers, were used to verify the model in a realistic setup.

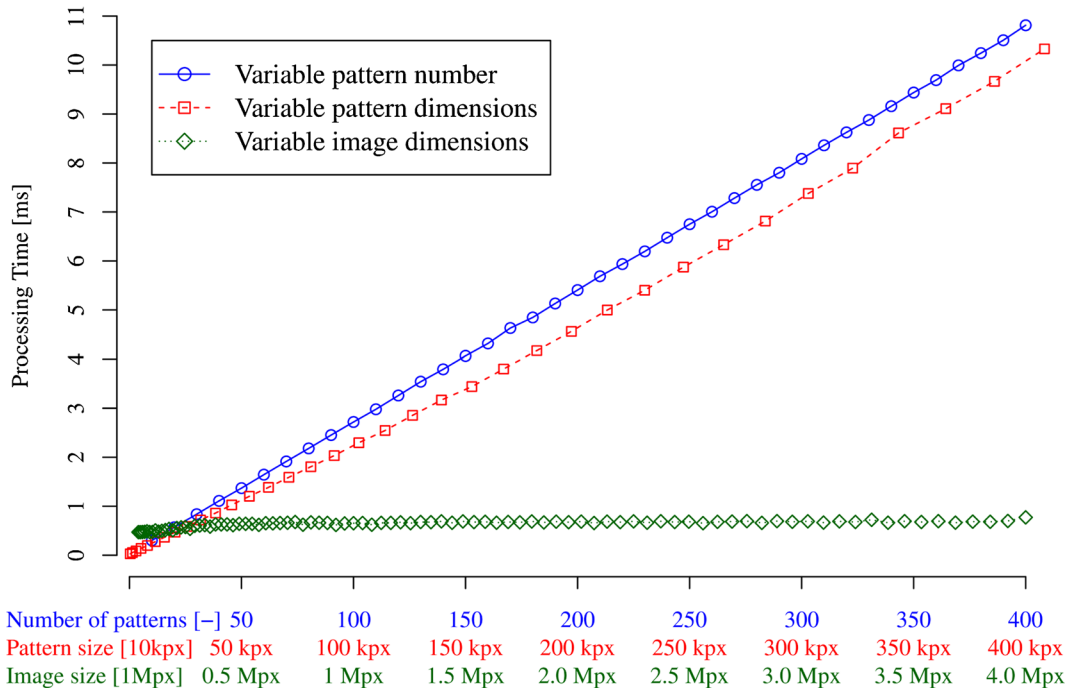


Fig. 6 Influence of the number of tracked patterns, pattern and image sizes on the method's speed

- The *"small"* dataset consists of one thousand images of a static pattern, which occupies approximately seven hundred pixels, i.e., 0.1 % of the image's total area.
- The *"large"* dataset is similar, but with a larger, sixty-pixel diameter pattern, occupying approximately 0.3 % of image pixels.
- The algorithm performance with these two datasets (*"small"* and *"large"*) is relevant in scenarios where the tracked objects are moving slowly and the camera is in a static position.
- The *"fast"* dataset contains 130 images of a fast moving pattern with a variable size. The dataset was tailored to cause failure of the tracking mechanism in one case. Thus, the performance of the algorithm with this dataset is similar to cases when the camera is not stationary or the tracked objects are moving quickly.

The average processing time per image for each dataset was measured and calculated by (20). The results summarized in Table 5 indicate the correctness of the model described in Section 5.3.

6.4 Comparison with a Precise Localization System

The real achievable precision of the localization system has been reported in Section 6.2; however, only for experiments with static targets, where the patterns were placed at the predefined positions. Such a setup provides verification of the precision for scenarios where the system tracks slowly moving robots. On the other hand, rapid movement of the tracked targets introduces additional effects, which might have a considerable impact on the system precision. First, the

Table 5 Required image processing time

	Processing time [ms]					
CPU	Measured			Predicted		
Dataset:	small	large	fast	small	large	fast
i-5 2450M	0.04	0.10	0.37	0.04	0.12	0.35
Atom N270	0.30	0.72	3.25	0.33	0.89	2.68
Pentium M	0.20	0.45	1.44	0.17	0.48	1.45
Odroid U2	0.27	0.89	2.76	0.29	0.79	2.86
Raspb. Pi	1.10	4.00	15.8	1.34	3.66	11.0

Table 6 Localization accuracy of a moving target

Mode	Abs. [cm]		Rel. [%]	
	ϵ_{avg}	ϵ_{max}	η_{avg}	η_{max}
2D	1.2	4.2	0.4	1.5
3D	3.1	11.2	1.2	4.4

captured images can be affected by motion blur and deformation caused by the camera's rolling shutter. Besides, there might be a delay in position estimation because standard USB cameras deliver the images with a delay caused by the interface's limited bandwidth. Therefore, we consider an additional experiment to evaluate the impact of these factors on the real performance of the presented global localization system. We consider a precise reference system and set up our localization system in an area where a high-precision motion capture system is installed and which is able to track multiple targets[2]. The motion capture system provides positions of the tracked targets 250 times per second with a precision up to 0.1 mm; so, it can be considered as a ground truth for our position measurements.

Four reference targets were placed in the area and a common coordinate system was calculated for both systems. After that, four sequences of targets moving at speeds up to 1.2 m/s were recorded by a Logitech QuickCamPro and the commercial motion capture system. Euclidean distances of target positions provided by both systems were taken as a measure of our system accuracy. The mean precisions of two- and three-dimensional localization were established as 1.2 cm and 3.1 cm, respectively (Table 6). Although the system's relative accuracy is lower that in the static tests presented in Section 6.2, centimeter precision is still satisfactory for many scenarios. The error is caused mostly by the image blur because of a long exposure rate set by the camera internal control. Careful setting of the camera exposure and gain parameters might suppress this effect. In fact, such a tuning has been made for localization of flying quadrotors, see Section 7.1.

It is also worth to mention that even though the commercial system is able to localize rapidly moving targets with a higher precision, its setup took more than thirty minutes while the presented system is prepared in a couple of minutes (just placing four patterns to establish the coordinate system).

[2]Human Performance Centre at the University of Lincoln

6.5 Comparison with Other Visual Localization Systems

The advantages and drawbacks of the presented localization system are demonstrated by a comparison of its performance with the well-established localization approaches based on AR markers and OpenCV. The performance of AR-based markers has been measured using the ArUco [14] library for detection and localization of multiple AR markers (similar to the ones used in ARTag and ARToolKit systems). A comparison with the OpenCV circular pattern detection is based on the OpenCV's "SimpleBlobDetector" class. The precision, speed, and coverage of all three systems were established in a similar way as described in the previous sections. For the sake of simplicity, we will refer to the presented system as *WhyCon*.

6.5.1 Precision Comparison

The localization precision of the ArUco-, OpenCV-, and WhyCon-based localization methods was obtained experimentally by the method described in Section 6.2. The comparison was performed on 4608×3456 pixel pictures taken by an Olympus VR-340 Camera from two different (*side* and *top*) viewpoints.

The achieved results are presented in Table 7. The WhyCon position estimation error is significantly lower than the error of ArUco and OpenCV in both the two- and three-dimensional localization scenarios.

Table 7 Localization precision comparison

mode		Relative error [%]					
		WhyCon		ArUco		OpenCV	
	view	*avg*	*max*	*avg*	*max*	*avg*	*max*
2D	side	0.12	0.19	0.20	0.41	0.52	1.03
	top	0.12	0.32	0.22	0.37	0.77	1.62
3D	side	0.31	1.10	0.63	2.52	–	–
	top	0.33	1.04	1.08	2.90	–	–

Moreover, we found that the OpenCV's blob radius calculation was too imprecise to reliably estimate the pattern distance and could not be used for the full 3D localization.

6.5.2 Performance Comparison on Different Platforms

The computational performance of the three evaluated systems was compared for three different platforms. The methods' performance was compared using two datasets similarly to the evaluation scenario described in Section 6.3.2. The *slow* dataset contains an easy-to-track pattern while for the *fast* datasets about 1 % of the images are tailored to cause a tracking failure.

The results presented in Table 8 indicate that the proposed algorithm is capable of finding the patterns approximately one thousand times faster than the traditional methods. Even in the unfavorable case where the patterns cannot be reliably tracked, the method outperforms ArUco and OpenCV hundred times. The performance ratio is even better for small embedded platforms with limited computational power. This property is favorable for deployment in the intended applications, especially under real-time requirements.

6.5.3 Range and Coverage Comparison

The AR fiducial markers are primarily intended for augmented reality applications and in a typical scenario, the localized marker is situated close to the camera. Therefore, the AR marker-based systems are not tuned for a reliable detection of distant patterns with small image dimensions. Thus, the range and coverage of the AR marker-based systems would be lower compared to WhyCon. On the other hand, OpenCV's circular blob detector can detect small circular patterns.

To estimate the ArUco and OpenCV detectors maximal range, we have established the minimal size (in pixels) that the tags need to have in order to be detected reliably. The sizes that correspond to the minimal pattern diameter D in the Equation (15) were established in a similar way as described in Section 6.1.1. While the OpenCV detector can find blobs larger than 12 pixels, the ArUco detector requires the AR marker side to be longer than 25 pixels. Therefore, ArUco's maximal detection range is less than a half of WhyCon's or OpenCV's range.

Table 8 Image processing time comparison

CPU	Processing time [ms]					
	ArUco		OpenCV		WhyCon	
Dataset:	fast	slow	fast	slow	fast	slow
i5 2450M	19	19	63	62	0.35	0.04
Pentium M	121	119	329	329	1.00	0.18
Odroid U2	148	149	371	366	0.93	0.28
Raspb. Pi	875	875	1795	1759	6.59	1.21

7 Practical Deployment

In this section, we present an overview of several research projects where the proposed circle detection algorithm has been successfully employed. This practical deployment demonstrates the versatility of the presented localization system. A short description of each project and comment about the localization performance is presented in the following sub-sections.

7.1 UAV Formation Stabilization

In this setup, the circle detection algorithm was considered for a relative localization and stabilization of UAV formations operating in both indoor and outdoor environments. A group of quadrotors are supposed to maintain a predefined formation by means of their relative localization. Each quadrotor UAV carries a circular pattern and an embedded module [24] running the localization method, see Fig. 7.

Thus, each UAV is able to detect other quadrotors in its vicinity and maintain a predefined relative position. Although the UAV's movements are relatively fast, we did not observe significant problems caused by image blur and the system detected the patterns reliably. This scenario demonstrates the ability to reliably detect circular patterns despite their rapid movements and variable lighting conditions. Moreover, it proved its ability to satisfy real-time constraints when running

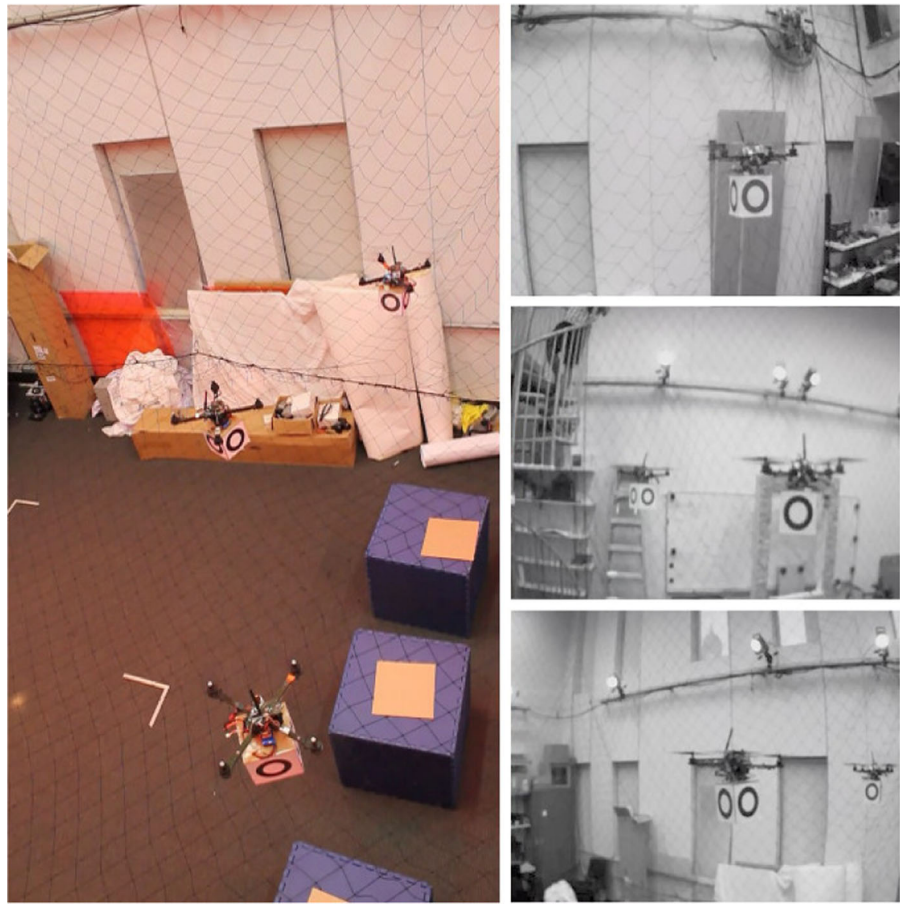


Fig. 7 Decentralized localization of quadrotor formation performed by the presented method. Courtesy of the GRASP laboratory, PENN

on computationally constrained hardware. The precision of the relative localization was in the order of centimeters [24].

7.2 Birds-eye UAV-based Localization System

The algorithm has also been used for relative localization of ground robots, which were supposed to maintain a predefined formation shape even if they lack direct visibility among each other. In this setup, one robot of the formation carried a heliport with the Parrot AR.Drone [25] quadrotor, which can take off and observe the formation from above using a downward-pointing camera. Each ground robot had a roundel pattern, which is elliptical rather than circular to provide also an estimate of the robot orientation. Using the roundel detection algorithm, the position and heading of the ground robots are provided by the flying quadrotor while it maintains its position above the formation.

Moreover, the heliport was designated by a circular pattern, which makes it possible to autonomously land the quadrotor after the mission end, see Fig. 8. Despite the relatively low resolution (168×144) of the UAV's downward-looking camera and its rapid movements,

Fig. 8 Mixed UAV-UGV robot formation

the overall localization precision was approximately 5 cm.

7.3 Autonomous Docking of Modular Robots

The Symbrion and Replicator projects [26] investigate and develop novel principles of adaptation and evolution of symbiotic multi-robot organisms based on bio-inspired approaches and modern computing paradigms. The robot organisms consist of large-scale swarms of robots, which can dock with each other and symbiotically share energy and computational resources within a single artificial life form. When it is advantageous to do so, these swarm robots can dynamically aggregate into one or many symbiotic organisms and collectively interact with the physical world via a variety of sensors and actuators. The bio-inspired evolutionary paradigms combined with robot embodiment and swarm-emergent phenomena enable the organisms to autonomously manage their own hardware and software organization.

In these projects, the proposed localization algorithm has been used as one of the methods for detecting power sources and other robots, see Fig. 9. The method demonstrated its ability to position the robot with a sub-millimeter precision, which is essential for a successful docking. The method's deployment in this scenario demonstrated not only its precision, but also its ability to run on computationally constrained hardware.

7.4 Educational Robotics

SyRoTek [19] is a remotely accessible robotic laboratory, where users can perform experiments with robots using their Internet connectivity. The robots operate within a flat arena with reconfigurable obstacles and the system provides an overview of the arena from an overhead camera. The project has been used for education and research by several institutions in Europe and Americas. An important component of SyRoTek is the localization system providing estimation of the real robots' positions.

Originally the localization was based on a convolution algorithm. Even though it is computationally demanding and rather imprecise, it demonstrated suitability for 24/7 operation. After replacement of this original localization system by the presented roundel-based system, the precision of the localization was improved. Moreover, the computational requirements were decreased as well [27]. In this deployment, the roundel pattern is formed from ellipses where the inner ellipse has slightly different dimensions, see Fig. 10, which allows to distinguish between individual robots. This use case demonstrates the ability of the system to operate in 24/7 mode. In addition, using different dimensions of

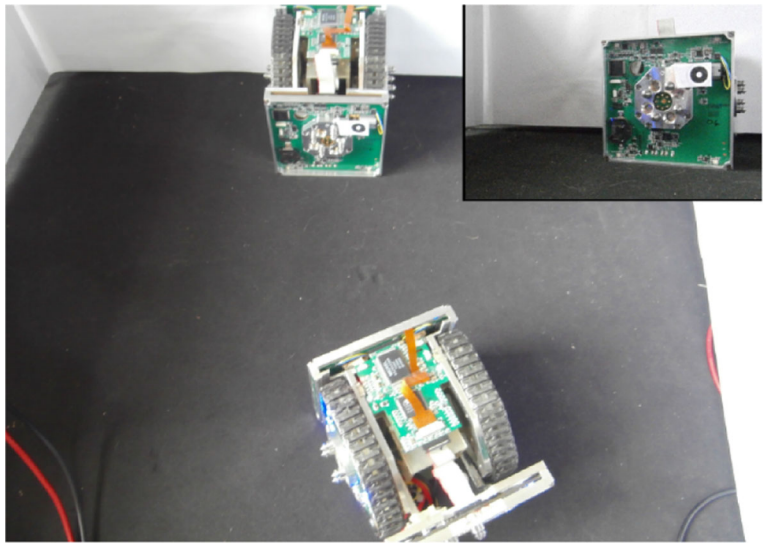


Fig. 9 Symbrion/Replicator robots during docking

Fig. 10 A top-down view to the SyRoTek arena

the inner ellipse allows to distinguish between 14 SyRoTek robots.

7.5 Ground Truth Assessment in Mobile Robot Navigation

BearNav (originally SURFNav) is a visual based navigation system for both ground [28] and aerial mobile [29] robots. The method is based on convergence theorem [30], which states that map-based monocular navigation does not need full localization, because if the robot heading is continuously adjusted to turn the robot towards the desired path, its position error does not grow above certain limits even if the position estimation is based only on proprioceptive sensing affected by drift. The aforementioned principle allows to design reliable and computationally inexpensive camera-based navigation methods.

The presented roundel based localization system was used to provide a continuous and independent measurement of the robot position error, which allowed to verify the convergence theorem and benchmark the individual navigation algorithms in terms of their precision, see Fig. 11. The system proved to be useful especially for aerial robots [29], which, unlike the ground robots, cannot be simply stopped for a manual position measurement.

7.6 Autonomous Charging in Long-term Scenarios

The STRANDS project [31] aims to achieve intelligent robot behaviour in human environments through adaptation to, and the exploitation of, long-term

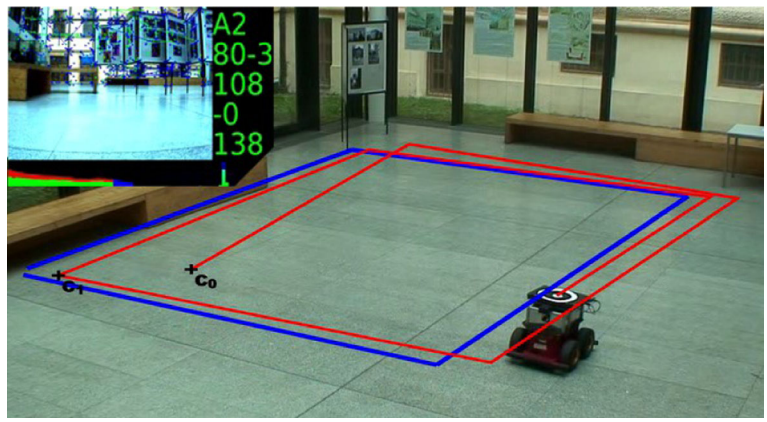


Fig. 11 Reconstructed trajectory of a mobile robot

Fig. 12 SCITOS-5 platform near its charging station. Notice the three o's of the label

experience. The project approach is based on a deeper understanding of ongoing processes affecting the appearance and structure of the robot's environment. This will be achieved by extracting qualitative spatio-temporal knowledge from sensor data gathered during months of autonomous operation. Control mechanisms that will exploit these structures to yield adaptive behaviour in highly demanding scenarios will be developed.

The circle detection method is used in the project as an initial solution of localization-related problems before more sophisticated implementations take its place. One of such deployments is the localization of the robot during its approach to a charging station, which has been solved by placing three patterns in the charging area, see Fig. 12.

8 Conclusion

We present a fast and precise vision-based system intended for multiple robot localization. The system's core component is based on a novel principle of circular roundel detection with computational complexity independent of the processed image size. The resulting system allows to localize swarms composed of several hundreds of robots with millimeter (2D) or centimeter (3D) precision, while keeping up with standard camera frame rates. In addition, we provide a model to calculate the sufficient camera and computer parameters to achieve the desired localization precision, coverage and update rate, which support potential users to decide which kind of equipment is needed for their particular setup.

The most notable features of the system are its low computational requirements, ease of use, and the fact that it works with cheap, off-the-shelf equipment. The system has been deployed already in a number of international mobile robotic projects concerning distributed quad rotor localization [24], visual based autonomous navigation [30], decentralized formation control [25], long-term scenarios [31], evolutionary swarm [26], and educational [19] robotics. Since the system has already proved to be useful in a variety of applications, we publish its source code [23]; so, other roboteers can use it for their projects. The experiments indicate that the presented system is three orders of magnitude faster than traditional methods based on OpenCV or AR markers while being more precise and capable of detecting the markers at a greater distance.

In the future, we plan to increase the precision and coverage of the system by using multiple cameras. We will plan to improve the tracking success rate by predicting the position of the target by considering the dynamics of the tracked object.

Acknowledgments The European Union supported this work within its Seventh Framework Programme projects ICT-600623 "STRANDS" and 216240 "Replicator". The Ministry of Education of the Czech Republic has given support by 7AMB12AR022 and 7E08006. The Ministry of Science of Argentina supported this work by project ARC/11/11. The work of J. Faigl was supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P. Christian Dondrup and David Mullineaux are acknowledged for help with experiments.

References

1. Thrun, S., Burgard, W., Fox, D., et al.: Probabilistic robotics, vol. 1. MIT press Cambridge (2005)
2. Breitenmoser, A., Kneip, L., Siegwart, R.: A monocular vision-based system for 6D relative robot localization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 79–85 (2011)

3. Yamamoto, Y., et al.: Optical sensing for robot perception and localization. In: IEEE Workshop on Advanced Robotics and its Social Impacts, pp. 14–17. IEEE (2005)
4. Vicon: Vicon MX Systems. http://www.vicon.com. [cited 8 Jan 2014]
5. Mellinger, D., Michael, N., Kumar, V.: Trajectory generation and control for precise aggressive maneuvers with quadrotors. Int. J. Robot. Res. **31**(5), 664–674 (2012)
6. Fiala, M.: 'ARTag', an improved marker system based on artoolkit (2004)
7. Wagner, D., Schmalstieg, D.: ARToolKitPlus for pose tracking on mobile devices. In: Proceedings of 12th Computer Vision Winter Workshop, pp. 139–146 (2007)
8. Kato, D.H.: ARToolKit. http://www.hitl.washington.edu/artoolkit/, [cited 8 Jan 2014]
9. Fiala, M.: Vision guided control of multiple robots. In: First Canadian Conference on Computer and Robot Vision, pp. 241–246 (2004)
10. Rekleitis, I., Meger, D., Dudek, G.: Simultaneous planning, localization, and mapping in a camera sensor network. Robot. Auton. Syst. **54**(11) (2006)
11. Stump, E., Kumar, V., Grocholsky, B., Shiroma, P.M.: Control for localization of targets using rangeonly sensors. Int. J. Robot. Res. (2009)
12. Fiala, M.: Comparing ARTag and ARtoolkit plus fiducial marker systems. In: Haptic Audio Visual Environments and their Applications, pp. 6–pp. IEEE (2005)
13. Bošnak, M., Matko, D., Blažič, S.: Quadrocopter hovering using position-estimation information from inertial sensors and a high-delay video system. J. Intell. Robot. Syst. **67**(1), 43–60 (2012)
14. ArUco: a minimal library for augmented reality applications based on opencv. http://www.uco.es/investiga/grupos/ava/node/26. [cited 8 Jan 2014]
15. Ahn, S.J., Rauh, W., Recknagel, M.: Circular coded landmark for optical 3d-measurement and robot vision. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1128–1133. IEEE (1999)
16. Yang, S., Scherer, S., Zell, A.: An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. J. Intell. Robot. Syst. **69**(1–4), 499–515 (2012)
17. Lo, D., Mendonča, P.R., Hopper, A., et al.: TRIP: A low-cost vision-based location system for ubiquitous computing. Pers. Ubiquit. Comput. **6**(3) (2002)
18. Pedre, S., Krajník, T., Todorovich, E., Borensztejn, P.: Hardware/software co-design for real time embedded image processing: A case study. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 599–606. Springer (2012)
19. Kulich, M., et al.: Syrotek - distance teaching of mobile robotics. IEEE Trans. Educ. **56**(1), 18–23 (2013)
20. Pedre, S., Krajník, T., Todorovich, E., Borensztejn, P.: Accelerating embedded image processing for real time: a case study. J. Real-Time Image Process. (2013)
21. Heikkila, J., Silven, O.: A four-step camera calibration procedure with implicit image correction. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1106–1112 (1997)
22. Yang, S., Scherer, S.A., Zell, A.: An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. J. Intell. Robot. Syst. **69**(1–4), 499–515 (2013)
23. Krajník, T., Nitsche, M., Faigl, J.: The WhyCon system. http://purl.org/robotics/whycon, [cited 8 Jan 2014]
24. Faigl, J., Krajník, T., Chudoba, J., Přeučil, L., Saska, M.: Low-cost embedded system for relative localization in robotic swarms. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 985–990, IEEE, Piscataway (2013)
25. Saska, M., Krajník, T., Přeučil, L.: Cooperative Micro UAV-UGV Autonomous Indoor Surveillance. In: International Multi-Conference on Systems, Signals and Devices, p. 36, IEEE, Piscataway (2012)
26. Kernbach, S., et al.: Symbiotic robot organisms: Replicator and symbrion projects. In: Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems, pp. 62–69. ACM (2008)
27. Cajtler, V.: Syrotek localization system. Bachelor thesis, Dept. of Cybernetics, CTU (2013). In Czech
28. Krajník, T., Přeučil, L.: A simple visual navigation system with convergence property. In: Proceedings European Robotics Symposium (EUROS), pp. 283–292 (2008)
29. Krajník, T., Nitsche, M., Pedre, S., Přeučil, L., Mejail, M.: A Simple Visual Navigation System for an UAV. In: International Multi-Conference on Systems, Signals and Devices, p. 34, IEEE, Piscataway (2012)
30. Krajník, T., et al.: Simple, yet stable bearing-only navigation. J. Field Robot. **27**(5), 511–533 (2010)
31. Hawes, N.: STRANDS - Spatial-Temporal Representations and Activities for Cognitive Control in Long-Term Scenarios. http://www.strands-project.eu, [cited 8 Jan 2014]