Advanced Edge Detection of AprilTags for Precise Docking Maneuvers of Mobile Robots [⋆]

Jan Richter [*] David Bohlig [*] Andreas Nüchter [**] Klaus Schilling [**]

[*] *Zentrum für Telematik e.V. (ZfT), 97074 Würzburg, Germany (e-mail: info@telematik-zentrum.de).*
[**] *University of Würzburg, Chair of Computer Science, Robotics and Telematics, 97074 Würzburg, Germany (e-mail: schi@informatik.uni-wuerzburg.de)*

Abstract: The development of modules and routines into an existing server-based control station to handle approach and docking maneuvers of mobile systems is the central achievement of this work. Efficient material flow in a digitized factory environment requires that goods are automatically loaded and unloaded onto transport robots. For this purpose, accurate positioning of the robot to a loading station is necessary or to a charging station, to automatically charge the robot's energy storage. Apart from these applications, robots are furthermore increasingly used for assembly work in production facilities. The possibility of using a transport robot as a mobile montage platform at assembly stations with manipulators is investigated in this work. Crucial for this is the ability to repeatedly acquire a position of sub-centimeter accuracy relative to the station and additionally to transmit the precise final position to the station. The critical component here is the estimation of the mobile system's pose, which is estimated using camera images of artificial markers attached to the stations. Since the precision of a visual localization strongly depends on the detection accuracy of the markers, an optimization problem is formulated to approximate the edges of the markers using an edge model function and thus determine their corners more precisely. Finally, a planar continuous trajectory planning is implemented to determine the segments of a possible path between any two poses.

Keywords: Industry 4.0, Cyber Physical Systems (CPS), Cloud robotics

1. INTRODUCTION

At Zentrum für Telematik e.V. (ZfT), work has long been underway on a control station for autonomous mobile systems, which should enable the management of any number of robots via an intuitive browser-based user interface (see Fig. 1). However, there is still no interaction possibility between the autonomous systems and stations. The modules implemented in this work will provide the basis for this by solving the problem of close proximity approach and providing a first interface for information exchange.

Other projects at ZfT involve the in-house assembly of small satellites. The individual components and subsystems of satellites are assembled according to modular design principle and connected via standardized interfaces. This requires maximum precision in many manufacturing steps, such as the placement of solar panels on the satellite's frame. For this reason, among others, ZfT plans to use advanced, robot-assisted production techniques to have parts of this work performed by industrial robots and eventually to realize complex interrelated manufacturing steps up to fully automated production, Zeitler (2017). The results of this work will contribute to the realization of this project.

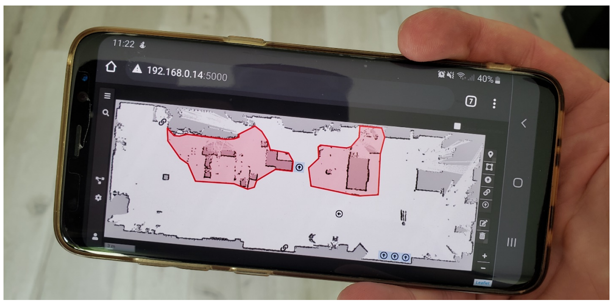


Fig. 1. Browser-based GUI of the control station at ZfT.

2. STATE OF THE ART

Relative positioning has been the subject of research for many years and has been investigated in numerous studies using different approaches. De Ponte Müller (2017) provides a comprehensive overview of different sensors for estimating the relative position of autonomous vehicles. These include egocentric or non-cooperative approaches

[⋆] Bayern Digital – a comprehensive investment program for Bavaria's digital future.

such as RADAR, LiDAR, and monocular, stereo, and time-of-flight camera systems.

A very simple but robust positioning system is used by the Kobuki robotic platform, Ju (2019). Three infrared emitters divide the area in front of a station into a left, right and center region. The robot, which may be located in one of the regions, is equipped with three infrared receivers – left, right and front. If the robot is in the central region, it must simply follow the signal from the central transmitter until it reaches the station. If the robot is in the left region, it must turn counterclockwise until the right sensor detects the signal from the left region. In this position, the robot looks at the central region, which it moves toward until the right sensor detects the signal from the central infrared emitter. After that, it only has to turn clockwise until the frontal IR sensor detects the central infrared emitter and follow it again. The procedure for the right region is analogous. An automatic charging system working in a similar way was presented by Doumbia et al. (2019) at ICCAR 2019. In Quilez et al. (2015), IR sensors are also used, but only for distance measurement and obstacle detection. For relative positioning, they use QR codes and then perform different approach strategies depending on the distance to the station. Thus, a visual approach using artificial markers is already being investigated here, but not to rely solely on visual pattern recognition to avoid potential errors due to camera calibration.

A purely visual artificial marker recognition based method is presented by Alijani (2017), where in real time the relative position of the robot with respect to AR-Tags – artificial markers similar to QR codes – is computed and respective commands are passed to the actuators to reach a target region of 4 cm diameter. Mateos (2020) also solely relies on visually detected artificial markers to allow swimming robot swarms to dock with each other. The markers he uses are called AprilTags and are used in this work as well. Thanks to a funnel-shaped guidance a relative yaw angle accuracy of $\pm 27.5\,^\circ$ is sufficient for a successful connection, with a lateral deviation of ± 4 cm. Of particular interest is the use of the so-called “AprilTags3D” method, in which two angled LCD screens display AprilTags instead of purely planar markers, thereby increasing the detection rate and reducing the yaw angle error. In addition, the displays are able to respond dynamically to different scenarios.

As yet, visually detected artificial markers (fiducials) play a minor role in docking methods, as more robust and easier-to-implement alternative sensors for estimating relative position exist. Therefore, there are not yet many approaches in the field of docking to improve detection accuracy in general. However, the precise detection of fiducials is of interest in many areas such as localization, calibration, or as optical reference points for automated manufacturing processes, as it is a completely passive method, without IR, RF, WiFi, or the like. Abawi et al. (2004) have already examined the influence of the relative orientation of the artificial markers to the camera and identified in this the main cause for systematic misestimations and increased standard deviations. Abbas et al. (2019) thus propose three methods that reduce this influence in an AprilTag based state estimation. First, a custom-built yaw axis gimbal tracks the center of the tag in real time. In addition, a trigonometric correction of the yaw angle is performed to align the camera with the center of the tag, and ultimately they apply a probabilistic Monte Carlo sensor error model to AprilTags. Of course, the developers of the AprilTag library also contribute methods to improve detection. Wang and Olson (2016) present improved edge detection to obtain better vertex position approximation from them.

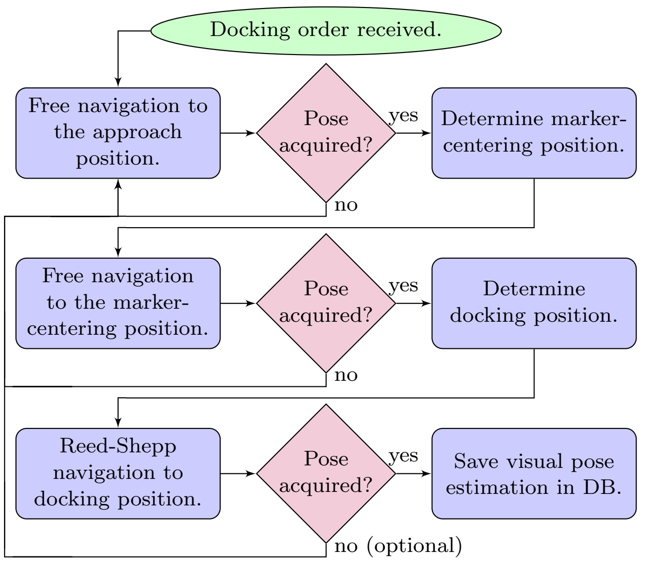

Docking order received.
Free navigation to the approach position.
Pose acquired?
yes
no
Determine marker-centering position.
Free navigation to the marker-centering position.
Pose acquired?
yes
no
Determine docking position.
Reed-Shepp navigation to docking position.
Pose acquired?
yes
no (optional)
Save visual pose estimation in DB.

Fig. 2. Flowchart of the docking routine.

3. CONCEPT

In a similar way, the accuracy of the detection is increased in this work and will be discussed in more detail in the next subsection.

The rough sequence of the docking process is shown in Fig. 2. The diagram shows that the process consists of repetitive routines, divided from left to right into approach, validation and acquisition of pose states.

The marker-centering position ensures that the AprilTags are in the center of the camera image, reducing systematic misestimations, Abawi et al. (2004). The approach position can be freely chosen, depending on the situation and is intended to help acquire the marker-centering position with repeatable precision. In this work, WFT’s robotic platform “HelMo” is used, whose robust chassis, due to the massive castor wheels – despite the robot’s weight of 710 kg – causes the robot to be pushed slightly away to the side of the direction of rotation, when the castor wheels are turned in the direction of travel. The wheels cannot be rotated by hand as not enough force can be applied, so the approach position is offset roughly 2 m behind the marker-centering position so that the wheels are aligned in the direction of travel at the start of the approach to the docking position.

The work is composed of 3 areas:

(1) The pose estimation and implicitly the calibration of the camera.
(2) The optimization of the robot’s internal localization.
(3) And the trajectory planning.

Each of these sub-areas is implemented modularly and has been tested extensively in isolation. In the following, the individual areas are addressed in this order, a few basics are explained, the most important experiments are summarized and the results are shown. Finally the result of the whole system is shown in the form of the docking maneuver.

3.1 Advanced Edge Detection

In Hagara and Ondrácek (2014), four methods for edge detection with subpixel accuracy are compared, with the result that the method AEF (Approximation with Erf Function) of Hagara and Kulla (2011) achieves the highest precision in sharp as well as blurred images, but at the same time is also the slowest. However, since the running time is negligible for the application in this work, the Edge Refinement of Wang and Olson (2016) will be replaced by an edge function fitted by a non-linear least square optimization.

Edges obtained from natural images are usually not ideal step/ramp edges by any means. Instead, they are usually affected by one or more of the following effects:

- Focal blur caused by a finite depth of field and point spread function.
- Penumbral blur caused by shadows of an extended light source.
- Shadow cast by a smooth object.

In a number of papers Lee et al. (2018); Hagara and Kulla (2011); Zhang and Bergholm (1997), used an S-function, shown in Fig. 3, as the simplest extension of the ideal step-edge model to approximate the effects of edge blur in practical applications. Thus, a one-dimensional image containing exactly one edge placed at l is modeled as:

$$f(x) = \frac{k}{2}\left[\operatorname{erf}\left(\frac{x-l}{\sqrt{2}\sigma}\right)+1\right]+h \tag{1}$$

with the parameter h indicating the lowest pixel intensity in the evaluation area of the edge, the contrast value k, the edge blur factor σ and the error function (Weisstein (2002)):

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt \tag{2}$$

In this work an S-function is used for modeling as well, but (1) and (2) are modified so that the position l of the edge is set to zero, resulting in it having its highest slope exactly at $x = 0$, i.e. exactly on the edge, and (2) is replaced by the hyperbolic tangent function, since it is easily implemented and quickly computed (Astanin (2013)).
This results in the new model function:

$$f(x) = \frac{k}{2}\left[\tanh\left(\frac{x}{\sqrt{2}\sigma}\right)+1\right]+h \tag{3}$$

It is known that the edges of an AprilTag are rectilinear in an undistorted image. Thus, the positions of the edges in the individual rows and columns of an image are not independent. To find the edge model parameters – which best represent the edge along the line that best passes through all contiguous edges of the one-dimensional images – a non-linear optimization is applied.

The parameters θ sought are those of the S-function (3) and those of the straight line passing through the edge:

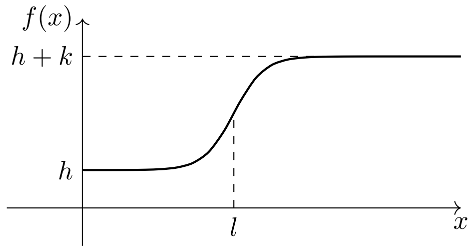


Fig. 3. Representation of the S-function (3).

$$\theta = \{\boldsymbol{n}_0, d, h, k, \sigma\} \tag{4}$$

The normal vector $\boldsymbol{n}_0$ and the distance d, together with any location vector $\boldsymbol{p}$, describe the Hessian normal form:

$$d(P,g) = \boldsymbol{p}\cdot\boldsymbol{n}_0 - d \tag{5}$$

It is often used to calculate the distance $d(P,g)$ of any point P to a straight line g.

By means of well-chosen initial values, an optimization problem is formulated from (3) and (5), whose rough procedure includes the following points:

(1) Isolate the detected marker as a "Region of Interest" (ROI) from the rest of the image.
(2) Create a binary mask between the vertices already found by AprilTag3 to use only relevant pixels for the residual calculation.
(3) Determine the initial values $\boldsymbol{n}_0$ and d for each edge.
(4) Pass the ROI, the mask and the optimization parameters θ to the cost function (Ceres solver).
(5) Determine the intersection points from the optimized line parameters $\boldsymbol{n}_0$ and d.

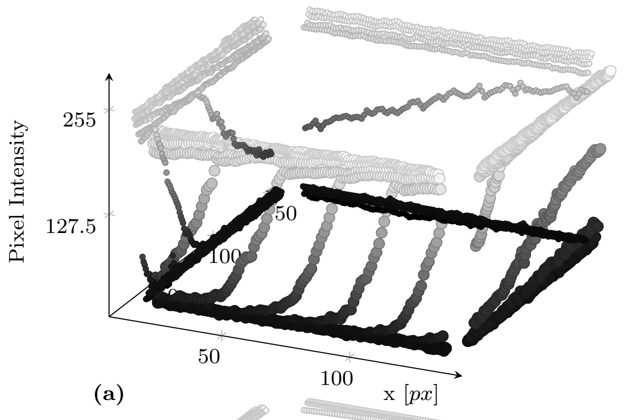
The residuals are the differences of the intensity between each pixel in the image and the estimated intensity of the edge model with respect to the pixel distance to the edge.

If one of the termination conditions of the Ceres solver is met and a minimum has been found, the result of a marker looks like Fig. 4. The intensities of the pixels in the mask **(a)** are almost identical to the estimated intensities of the model function **(b)**.

One of the results of the experiments is that artificially induced blur and adaptive blurring of the ROI further improves corner point detection – especially for synthetically generated images where the edges more closely resemble a step function. Depending on the degree of blur of the original image and the orientation of the markers, a specifically implemented filter method selects OpenCV methods Bilateral and GaussianBlur filters from different kernel sizes, which is applied once to the ROI before edge approximation, thus significantly reducing the position error and standard deviation.

3.2 Robot internal localization

The robot's operating system has two internal localization methods. One is adaptive monte carlo localization, which does not drift over time, but jumps to adjust the position on the map when new sensor information is received. Moreover, it is impossible to predict when this jump will happen and how well the current position estimate matches the map. This is especially a problem for slow movements.



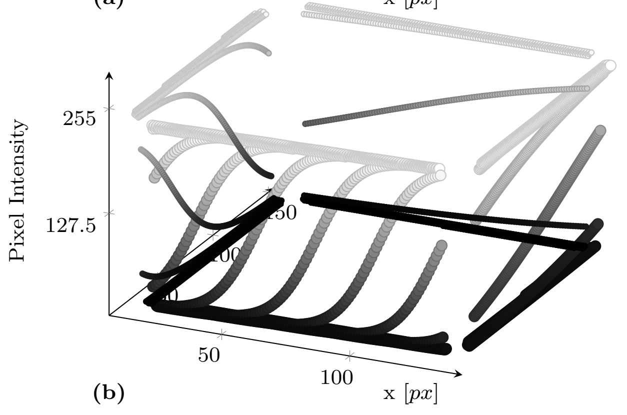


Fig. 4. Plot of pixel intensity within the mask in the ROI **(a)** and edge model **(b)**.

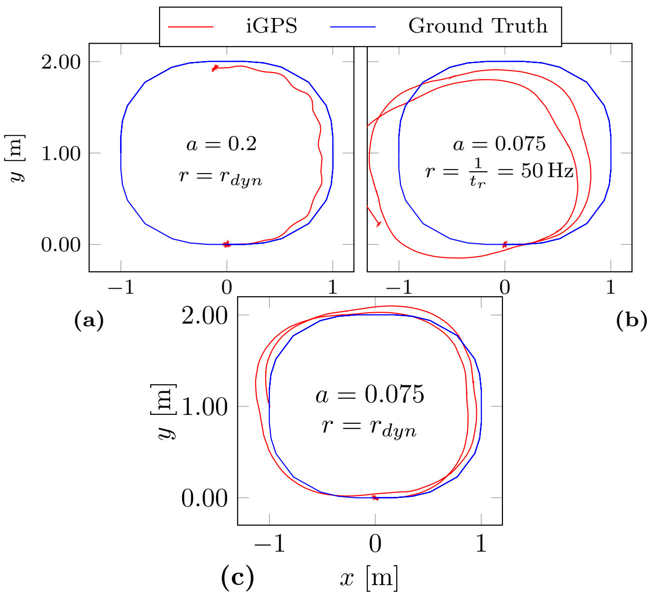


Fig. 5. iGPS of robot position using different controller parameters for rate r and damping factor a.

The 2nd method is odometry, which does not have the problem of discrete jumps, but is affected by unbounded drift that results from adding up deviations. Unlike jumps, however, it is possible to estimate the extent of drift and reduce it by adjusting the controller.

For this reason, odometry was chosen, in part because the heavy robot with its robust suspension is known for good odometry localization.

Two parameters were identified as decisive optimizations. Firstly, the damping a of the controller, which applies the deviations from the nominal path directly proportional to the controller inputs. The higher, the faster deviations are compensated, but there is also the risk that the system starts to oscillate. Too low and the deviations aren't compensated fast enough, which also leads to a divergence from the nominal path. Finally, a damping value of 0.075 has been found to be optimal.

The 2nd parameter is the rate of the control loop, which is set to 50 Hz by default. It has been found that execution times of path segments that are only slightly longer than a multiple of 50 Hz (or 20 ms) lead to noticeable overshoots. For this reason, the dynamic rate was implemented, which divides the segment execution time t_s so that it is always a multiple of the rate and the rate is always approximately 50 Hz.

$$i = \frac{t_s}{t_r}$$
$$r_{dyn} = \begin{cases} \frac{\lfloor i \rfloor}{i t_r} - \Delta t & \text{, if } i \geq 1 \\ \frac{1}{t_s} & \text{, otherwise} \end{cases} \tag{6}$$

From the segment execution time t_s and the standard execution time t_r in seconds, the number of loop iterations i is obtained. If $i \geq 1$, the exact rate in hertz is determined, which is a multiple of t_s. In addition, a very short Δt is subtracted to prevent unintentional exceeding of t_s.

Fig. 5 **(a)** shows the plot with the default damping of 0.2 and dynamic rate, **(b)** shows the plot with adjusted damping and default rate of 50 Hz and **(c)** shows the plot with optimized parameters.

3.3 Trajectory planning

The last area concerns trajectory planning based on the Reed-Shepp method, Reeds and Shepp (1990). The method calculates planar paths consisting of line segments and tangential circular arcs with minimum radius. In total, the set of paths includes 48 different paths, which in turn consist of a maximum of five segments. However, the curvature of this path type is discontinuous. Discontinuities occur at the transitions of circular arcs, where a real car, if it wanted to follow such a path exactly, would have to stop at each break in curvature to realign its front wheels. Curvature continuity is therefore a desirable property and this has been achieved by approximating the Reeds-Shepp segments by clothoids. The curvature progression of klothoids increases linearly, resulting in a gradual acceleration transition from straight ahead to circular instead of an abrupt jerk. Clothoids are used, for example, as transition curves in road construction as well as in railroad construction and is calculated quite efficiently.

Fig. 6 shows all possible paths between the two black poses and in red the approximated clothoid path. Approximated, because the robot's control system only processes curve segments with constant curvature, therefore each clothoid is approximated with circular segments of different curvature. Tests showed that the deviation at the target point is in the range of the floating point precision of numbers of

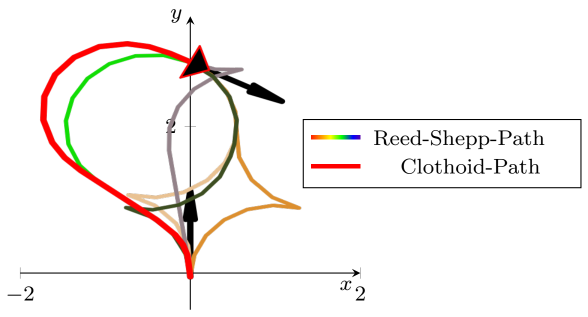


Fig. 6. Visualizes all generated Reed-Shepp paths of a simple trajectory and in red the approximated path with continuous curvature.

the type double. Thus, the trajectory planning is guaranteed not to induce any notable error in the docking routine.

In addition, to keep the number of segments low, paths with fewer segments are preferred and, due to the robot's castor rolls, travel direction changes are sorted out.

This restriction and the approach complicate the user input of a docking maneuver, due to which the creation is supported visually in the graphical user interface.

4. RESULTS

4.1 Advanced Edge Detection

Simulation In order to check whether the extended edge detection works and achieves better results than the AprilTag3 library, a simulation was carried out with artificially generated AprilTags. This way, any influences due to hardware and calibration could be excluded.

Since these are generated tags, the precise position of the vertices on the image plane is known and thus is usable as ground truth.

Rotation around the y-axis of the simulation.

Method	$\sigma_r[10^{-3}]$	$\mu_r[10^{-3}]$	$\sigma_p[10^{-3}]$	$\mu_p[10^{-3}]$
AprilTag3	34.3947	41.3505	61.6882	175.3330
Adv. Det.	12.8442	8.0157	50.2945	40.9599
Filter-Method	7.9674	3.3684	28.7507	25.1204

Rotation around the z-axis of the simulation.

Method	$\sigma_r[10^{-3}]$	$\mu_r[10^{-3}]$	$\sigma_p[10^{-3}]$	$\mu_p[10^{-3}]$
AprilTag3	8.7379	11.7599	37.1436	53.6931
Adv. Det.	0.2610	0.3851	1.3879	3.9553
Filter-Method	0.1545	0.1760	1.0208	2.9557

Table 1. Standard deviation and mean of the reprojection and position errors of AprilTag3, the advanced edge detection without preprocessing (Adv. Det.) and with preprocessing by selected blur filters (Filter-Method).

Kuka-Experiment The experiment was then repeated with real images and to better compare the results, the conditions were recreated as best as possible using a kuka robot. However, after the promising results of the simulation, the increase in accuracy has been tremendously reduced. Consequently, the Filter-Method improves the mean by about 20 % in the y-rotation and about 45 % in the z-rotation, making the distribution of the reprojection errors in y and z at least somewhat more equalized, and the standard deviation is reduced by 30 %, resulting in less significant outliers.

Method	Axis	$\sigma_r[10^{-2}]$	$\mu_r[10^{-2}]$
AprilTag3	y	2.5950	4.1752
	z	3.2529	6.2448
Adv. Det.	y	3.1498	4.6555
	z	2.6678	5.5096
Filter-Method	y	2.1715	2.4423
	z	2.2467	3.9345

Table 2. Standard deviation and mean of the reprojection errors when rotating around the y- and z-axis of the Kuka experiment.

4.2 Pose estimation

After testing the detection accuracy, the accuracy of the pose estimation was investigated. For this purpose, the external localization system iGPS was used, which allows position determinations in the micrometer range, Depenthal (2009). The iGPS sensor was placed near the camera center and both systems were transferred to a common reference system to allow ad hoc analysis. In addition, a hand-eye calibration was performed.

In order to determine the pose of the camera, it must be pointed at the AprilTags, which initially recorded the 13 poses in Fig. 7. The AprilTag pattern is at the origin, to which all poses point.

Pose determinations using planar markers such as AprilTags are subject to inaccuracies and ambiguities when viewed directly from above, which is clearly seen here. While the angled poses are slightly positively skewed with respect to the iGPS, the poses in the center are skewed in the opposite direction.

This problem is studied in detail in Abawi et al. (2004) and in Abbas et al. (2019) additionally the influence of the position of the tag in the image itself, with the result that the pose estimation has the highest accuracy in the angular range from 25 ° to 75 ° and with the camera orientation centered. For these reasons, the four pose pairs in the center of Fig. 7 are not included in the hand-eye transformation calculation and the ±25 ° cone was avoided for further testing.

Thus, before hand-eye calibration, the deviation between the camera and the iGPS sensor averaged 4 cm in the direction of motion, i.e., the sensor is 4 cm behind the center of the camera and is rotated about 1.8°. After calibration, this decreased to about 3 mm and 0.15° in orientation measured at distances from 1.4 m to 6.7 m.

However, it is difficult to provide a clear range of error, since in addition the reprojection error leads to small misestimates of the distance. Fig. 8 shows the progression of the reprojection error relative to the distance of the camera to the pattern and the maximum distance error to be assumed, which is calculated from the difference of the true iGPS distance and the intercept theorem:

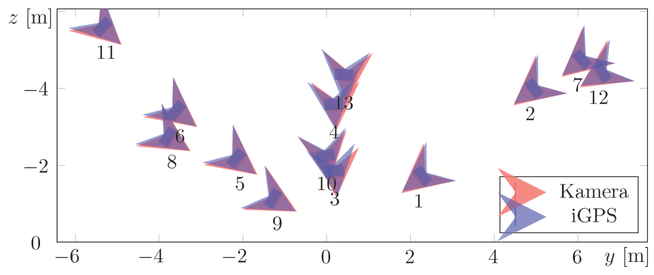


Fig. 7. The captured poses for hand-eye calibration.

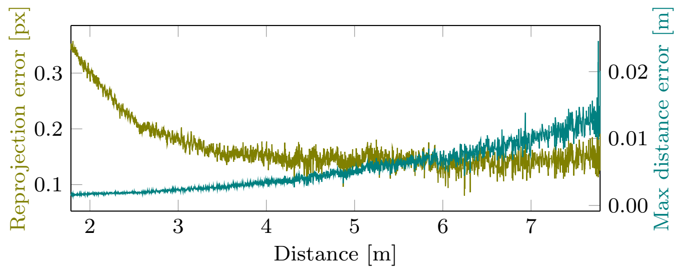


Fig. 8. Reprojection error relative to the distance of the camera to the pattern and the resulting maximum distance error to be assumed according to the ray theorem.

$$Z = \frac{fT_x}{d_x} \tag{7}$$

With the focal length f in pixels, the real width T_x of the object in meters referring to the horizontal dimensions, and the disparity, i.e. the pixel width, d_x. With a double reprojection error Δd_x as the distance-dependent error term of the pixel width $d_x = d_x + \Delta d_x$ results in the distance error. The increased reprojection error at the beginning is compensated by the large pixel width, resulting in a smaller distance error than at larger distances with smaller reprojection error but significantly smaller pixel width.

4.3 Docking maneuver

Unlike in the experiments before, the docking maneuver requires not only a repeatable execution of an isolated module, but a collaboration of a complex software structure consisting of three programming languages and dozens of libraries. From the job creation in the user interface to the last action of the program flowchart in Fig. 2, user input is received, databases are updated, job scheduling is performed, data exchange with the robot is maintained, photogrammetric methods are applied, trajectories are planned, and the status of the job is dynamically acted upon.

The upper three Histograms in Fig. 10 show the distribution of camera and iGPS poses at the marker-centering position. From the isolated experiments of the individual components before, it is known that a relative yaw angle smaller than 25° between the camera and the pattern should be avoided, since this leads to inaccuracies and ambiguities in the pose determination, and that the trajectory should be chosen as straight as possible to minimize the drift of the odometry. For these reasons, the marker-centering position is positioned at the far edge of the

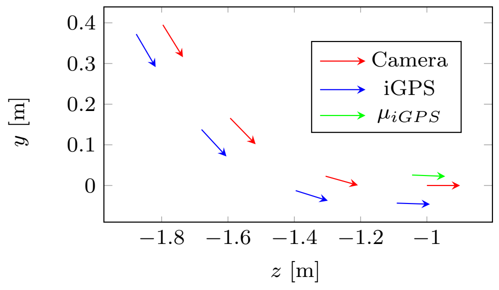


Fig. 9. Representation of the average calculated path starting at the mean camera pose in red, respectively at the mean iGPS pose in blue and the mean iGPS docking position in green. For better illustration, only the last four trajectory poses are shown.

interval at 25 ° to 27 ° at a distance of approximately 3.45 m.

As in the hand-eye calibration, the known systematic difference of the camera orientation of approx. 1.5° reads from the mean values of the measured orientation, as well as the offset along the line of origin of approx. 4 cm. Thus, the distribution of the measured poses of the iGPS and the camera in the marker-centering position is surprisingly similar. However, to prevent error propagation, the pose estimates were not hand-eye calibrated to the sensor frame of the iGPS.

The lower three histograms in Fig. 10 show the distribution of the docking position with the aim to dock one meter in front of the pattern with alignment to the origin.

The poses in Fig. 9 show briefly the path if the iGPS sensor (blue) or the camera (red) would correspond exactly to the robot pose – since it is difficult to say exactly which of the two is better aligned. Knowing that a yaw angle of 0° implies an orientation parallel to the z-axis and the target docking pose is given at $y = 0$ m and $\psi = 0°$, this systematically leads to a smaller deviation from the marker-centered orientation to the docking orientation. This deviation directly affects the trajectory planning, resulting in approx. 1.5° less turning in, which eventually leads to a shift of the mean value of the docking orientation ψ (in the lower right histogram in Figure 10) from 0° to −1.8°. This thus leads to similar values for z and ψ, but a shift of the y values to negative. The green arrow in 9 shows the mean iGPS pose, the last red arrow shows the target docking pose, and the last blue arrow shows the expected pose when the iGPS sensor matches the robot base in the marker-centering position.

The isolated tests had shown that the trajectory planning does not induce any errors and that the pose estimation of the camera causes distance measurement errors in the range of 3 mm and 0.15°, which is within the range of the expected errors due to the distance and the reprojection error.

It is therefore reasonable to assume that the deviations from the expected pose, i.e. the last blue pose, are to a large extent due to the influences of the controller module, which has not yet been included and which tends to turn in a little further and to swing out slightly during the transition to segments of lower curvature.

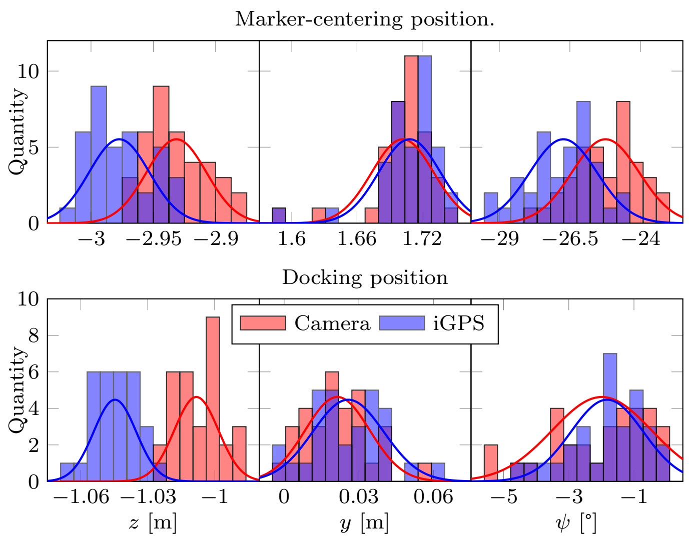


Fig. 10. Evaluation of camera and iGPS data at the marker-centering and docking positions.

In addition, the influence of the robot's massive chassis should not be underestimated, whose wide hard rubber wheels of the differential drive unit, inevitably lead to a certain amount of slippage of the areas of the wheels that are not the pivot point of the curve, which could be perceived from a clearly perceptible squeaking in the curves. Presumably, this pivot point wanders depending on the degree of dirt on the floor and its condition.

Due to the age and regular use of the robot, a general decalibration of the drive unit is also not to be excluded. The exact weighting of the influencing factors on the error is difficult or impossible to research.

5. CONCLUSION

In this work, a fully functional visual fiducial detection based docking routine was integrated into a complex software architecture consisting of user interface, control station and robot operation software. For this purpose, an advanced corner detection was developed that fits a step edge model to each of the four AprilTag sides using cost minimization to estimate corner points with up to 40 % lower deviation from the standard AprilTag3 method while increasing robustness against outliers. Under simulated conditions, even a 10- to 100-fold reduction in reprojection error was achieved. This leads to very precise visual position determinations of less than 3 mm and yaw angle determinations of less than 0.15 ° deviation on average from the high-precision iGPS measuring system. The referred measurements were made from equally spaced distances to the pattern between 1.4 m to 6.2 m and under constant driving speed of 0.1 m/s.

To convert the precise relative localization into a path leading to the docking position, a trajectory planner was developed to generate a rough path from Reed-Shepp segments and continuous curvature of the segment transitions was ensured by clothoid approximation. Finally, to execute the generated paths, the robot controller were optimized with dynamic hyper-parameters for dead reckoning navigation and the robot operation software was adapted to ensure continuous localization.

Despite all this, the evaluations revealed that dead reckoning still induces large errors in the overall docking maneuver system, resulting in an average deviation of about 4.5 cm and 0.4 ° from the expected docking position due to an excessive turning in curve segments. Over numerous docking maneuvers, a repeatability of

$$\sigma_z = 0.96\,\text{cm},\ \sigma_y = 2.57\,\text{cm},\ \sigma_\psi = 1.11\,°$$

was achieved.

REFERENCES

Abawi, D., Bienwald, J., and Dorner, R. (2004). Accuracy in optical tracking with fiducial markers: an accuracy function for artoolkit.

Abbas, S.M., Aslam, S., Berns, K., and Muhammad, A. (2019). Analysis and improvements in apriltag based state estimation.

Alijani, F. (2017). Autonomous vision-based docking of a mobile robot with four omnidirectional wheels.

Astanin, S. (2013). Benchmark various sigmoid functions. `https://gist.github.com/astanin/5270668`.

De Ponte Müller, F. (2017). Survey on ranging sensors and cooperative techniques for relative positioning of vehicles.

Depenthal, C. (2009). igps - a new system for static and kinematic measurements.

Doumbia, M., Cheng, X., and Havyarimana, V. (2019). An auto-recharging system design and implementation based on infrared signal for autonomous robots.

Hagara, M. and Kulla, P. (2011). Edge detection with subpixel accuracy based on approximation of edge with erf function.

Hagara, M. and Ondrácek, O. (2014). Comparison of methods for edge detection with sub-pixel accuracy in 1-d images. 124–127.

Ju, Y. (2019). Automatic docking for kobuki.

Lee, S., Lee, S., and Pahk, H. (2018). Precise edge detection method using sigmoid function in blurry and noisy image for tft-lcd 2d critical dimension measurement. 69–78.

Mateos, L.A. (2020). Apriltags 3d: Dynamic fiducial markers for robust pose estimation in highly reflective environments and indirect communication in swarm robotics.

Quilez, R., Zeeman, A.S., Mitton, N., and Vandaele, J. (2015). Docking autonomous robots in passive docks with infrared sensors and qr codes.

Reeds, J. and Shepp, L. (1990). Optimal paths for a car that goes both forwards and bachwards. 367–393.

Wang, J. and Olson, E. (2016). AprilTag 2: Efficient and robust fiducial detection.

Weisstein, E.W. (2002). “erf.” from mathworld–a wolfram web resource.

Zeitler, F. (2017). Innospace masters - spin-in und spin-off ideen.

Zhang, W. and Bergholm, F. (1997). Multi-scale blur estimation and edge type classification for scene analysis. 219–250.