A QR Code-based High-Precision Docking System for Mobile Robots Exhibiting Submillimeter Accuracy*

Georgios Bolanakis, Kostas Nanos, and Evangelos Papadopoulos, *Fellow, IEEE*

***Abstract*— This paper presents a high-precision docking system enabling controlled motion of mobile robots and automatic guided vehicles in micrometer range. The developed system employs a low-cost, vehicle installed camera and a docking base with Quick Response (QR) codes in the form of a two-dimensional matrix. The docking base is free of active components, eliminating the need for maintenance, and minimizing manufacturing or installation costs. A robust localization algorithm is developed that provides real-time position feedback at 60 Hz with submillimeter accuracy. Redundancy is established by calculating the vehicle's pose using multiple QR codes; thus, if one or more QR codes are damaged, the feedback stays unaffected. By design, the system is insusceptible to light variance. Using a prototype robotic platform, excellent mean absolute errors of 39 μm in position, and 8.8 millidegrees in orientation are achieved, providing experimental validation of the docking system.**

I. Introduction

Autonomous docking is a challenging, yet mandatory procedure for automatic guided vehicles and autonomous mobile robots (hereafter referred to as "*vehicles*"), that allows the execution of precise manipulation, inspection, and recharging operations. Docking can be defined as the motion from an initial position/ orientation (pose) to a desired docking pose, while following a safe trajectory [1]. In the literature, two-stage motion has been commonly proposed in order to carry out the docking procedure. Initially, the vehicle moves, at a relatively high speed, towards the docking area (*approximate docking*), see Fig. 1. By the time the vehicle has approached the docking base the second stage (*precision docking)* takes place. This stage refers to the execution of precise motions to perfectly align the vehicle with its final target. In most cases, this procedure is attained with special purpose hardware, installed at the vehicle itself and/ or the docking base. In precision docking, a close-range high-precision localization system is established providing position feedback to the vehicle's motion controller.

The ability of a vehicle to dock precisely may simplify significantly and accelerate common procedures in factories, warehouses, distribution centers and even clinical environments [2]-[5]. For most applications, millimeter accuracy is adequate. However, in recent years, docking with submillimeter accuracy has become essential for material and air-cargo handling applications [6]-[7]. Nevertheless, docking has attracted less interest by the research community as compared to other areas in mobile robotics [3]. Various approaches have been proposed, including but not limited to, the use of ultrasonic [8], laser [9], vision [5] and RFID [10] sensors. Multi-sensor fusion has, also, been suggested to reduce position uncertainty during autonomous mobile robot docking [11].

QR codes have been used as visual landmarks to guide vehicles in a passive docking base [12]; however, their use in the form of a two-dimensional matrix is limited to generic vehicle localization and path planning applications [13]-[14]. Zhang et al. showed experimentally that a positional error of 60 mm could be achieved using ceiling-placed QR codes [13]. The algorithm discards valuable information in the case in which multiple QR codes are detected, whereas a high cost, calibrated industrial camera is required to compensate for the large vehicle-ceiling distance.

Most docking systems achieve end accuracies up to a few millimeters and some hundreds of millidegrees. Only a few available solutions achieve submillimeter accuracy [6], [9], [15], which however, include complex designs relying on expensive hardware and/ or docking bases with active components that would require regular maintenance.

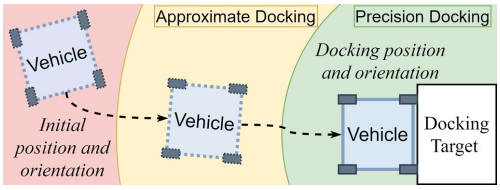


Fig. 1. Two-stage motion (approximate, precision) during docking.

In this work, a high-precision docking system is developed that uses QR codes installed at the docking base in the form of a two-dimensional matrix. The system is developed as part of an automated air-cargo handling procedure. No active components are included in the system's docking base, eliminating the need for regular maintenance and minimizing manufacturing and installation cost. Furthermore, the small vertical distance between the vehicle and the docking base allows the use of low-cost image sensors such as the ones in desktop web cameras. A robust localization algorithm has been developed that provides real-time position feedback to the vehicle's motion controller and adopts dynamically to variations in the distance between the camera and the docking base. The system exhibits redundancy since the vehicle's pose is calculated based on multiple QR codes at a time and thus, if one or more QR codes are damaged the feedback stays unaffected. This feature, combined with 1-D median filtering, results in feedback of exceptional accuracy. Experiments validate the overall efficacy of the developed system and demonstrate a mean absolute pose error of 39 μm and 8.8 millidegrees.

* The research work presented in this paper has received funding from the European Union's H2020-EU.3.4.5.1.-IADP Large Passenger Aircraft, Research and Innovation Programme, under grant agreement No 785472.

The authors are with the School of Mechanical Engineering, National Technical University of Athens, (e-mail: gbolanakis@hotmail.com, {knanos, egpapado}@central.ntua.gr, tel.: +30 210-772-1440).

II. System Overview

A holonomic robotic platform (RP) was developed to simplify and accelerate air-cargo handling procedures [7]. The RP's foremost goal is to load and lock the air-cargo containers, also called Unit Load Devices (ULDs), on predefined locations inside an aircraft's Cargo Compartment (CC). *Approximate* docking is performed by following magnetic lines installed on the floor of the CC. Once the RP/ ULD have approached the locking position, the *precision* docking system takes over, accurately aligning the RP and the carried ULD with its locking pose. Next, a ULD integrated locking mechanism is activated, and four locking pins located at the corners of the ULD base are inserted safely into the corresponding holes. An overview of the key components is shown in Fig. 2. End accuracy better than 500 μm must be satisfied for *each* of the four locking pins simultaneously. This condition, combined with the substantial distance of 0.67 m between the locking pins and the RP's geometrical center, increases tremendously the accuracy requirements of the particular docking procedure.

In the developed system, the docking base consists of QR codes in the form of a two-dimensional matrix fixed at a predefined floor position. Each QR code contains its own coordinates in a Cartesian coordinate system (CS) defined by the grid shown in Fig. 3. The camera is placed at the RP's center, directed downwards, and at a distance of 40 mm above the floor allowing the detection of approximately 6 QR codes in each captured image frame.

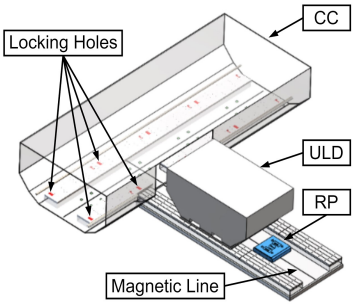


Fig. 2. Application overview. The RP is moving towards the ULD.

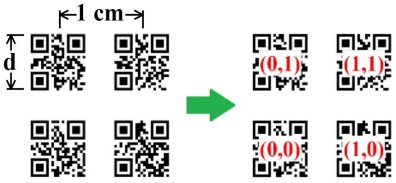


Fig. 3. Docking base QR codes, with side dimension d, fixed in the form of a two-dimensional matrix. Integrated QR coordinates are returned in cm.

III. Localization Algorithm

Two coordinate systems are defined (Fig. 4). The first (CS-0) is defined by the docking base QR codes, while the second one (CS-1) is a robotic platform-fixed CS that coincides with the RP camera-fixed CS, since the camera is installed at the geometrical center of the RP. Let $({}^0x_r, {}^0y_r, \theta_r)^T$ be the RP's pose with respect to CS-0.

The goal of the developed localization algorithm is to estimate the RP's pose using an image frame provided by its camera. The algorithm is summarized in six steps (Fig. 5), described next thoroughly.

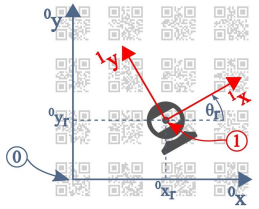


Fig. 4. Coordinate systems. The CS-0 is fixed on the docking base (defined by the QR codes) and the CS-1 on the RP (defined by its camera).

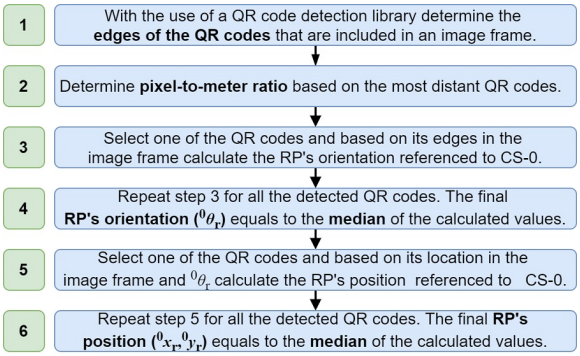


Fig. 5. Steps composing the developed localization algorithm.

Let n be the total number of the QR codes that are read by the RP camera and decoded by the QR code detection library, as described in Section V. For the i^{th} $i=1,...,n$ QR code read, its self-contained coordinates in CS-0, $({}^0x_i, {}^0y_i)^T$ are returned. Also, the detection library returns the location ${}^1\hat{\mathbf{c}}_{ij} = ({}^1x_{ij}, {}^1y_{ij})^T$, $j=1,...,4$ of its four corner points C_{ij}, $j=1,2,3,4$ expressed in pixel coordinates and referenced in CS-1, see Fig. 6. The i^{th} QR code center point, is denoted by M_i.

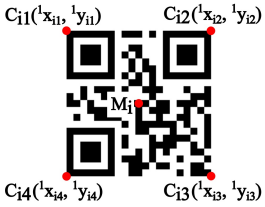


Fig. 6. The corner points returned by the QR code detection library. The center point M_i is shown also.

As a first step in the calculation of the RP's pose, the pixel-to-meter ratio R must be determined so that the pixel coordinates of a QR code (denoted by ^) can be accurately translated to cartesian coordinates referenced to CS-1. To this end, the vector ${}^1\hat{\mathbf{m}}_i$ of the i^{th} QR code center point, M_i, expressed in CS-1 pixel coordinates, is given by

$${}^1\hat{\mathbf{m}}_i = \left(\left({}^1x_{i1} + {}^1x_{i3}\right)/2,\ \left({}^1y_{i1} + {}^1y_{i3}\right)/2\right)^T,\ i=1,K,n \quad (1)$$

To minimize computation errors due to random image noise, the ratio is calculated based on the two most distant QR codes, i.e., $i=d_1,d_2$. Let M_{d1} and M_{d1} be their center points. Since their coordinates are referenced to CS-0 $({}^0\mathbf{m}_i, i=d_1,d_2)$ and are already given by the detection library, R can be calculated as follows:

$$R = \frac{\left\| {}^1\hat{\mathbf{m}}_{d_1} - {}^1\hat{\mathbf{m}}_{d_2} \right\|}{\left\| {}^0\mathbf{m}_{d_1} - {}^0\mathbf{m}_{d_2} \right\|} \quad (2)$$

In the case where only one QR code is detected ($n=1$), the ratio is calculated based on the coordinates of its corners ${}^{1}\hat{\mathbf{c}}_{11}$, ${}^{1}\hat{\mathbf{c}}_{12}$ (see Fig. 6), and its known side dimension d:

$$R = \left\| {}^{1}\hat{\mathbf{c}}_{11} - {}^{1}\hat{\mathbf{c}}_{12} \right\| d^{-1} \quad (3)$$

Now it is possible to transform coordinates expressed in pixel coordinates to length coordinates referenced to CS-1, by dividing their components with R.

Next, the RP's orientation in CS-0, $\theta_{r,i}$, as calculated using the corners of the i^{th} QR code (Fig. 7a) is

$$\theta_{r,i} = -\angle({}^{1}\mathbf{c}_{i2} - {}^{1}\mathbf{c}_{i1}) = -atan2({}^{1}y_{i2} - {}^{1}y_{i1}, {}^{1}x_{i2} - {}^{1}x_{i1}) \quad (4)$$

Eq. (4) is repeated for all n read QR codes, and the RP's orientation referenced to CS-0 is obtained as,

$$\theta_r = median\{\theta_{r,i}\},\ i = 1,\ldots,n \quad (5)$$

To increase the accuracy, 1-D median filtering is employed in order to discard outliers caused by the uncalibrated camera and the random image noise.

Assuming that the RP origin coordinates calculated based on the coordinates of the i^{th} QR code, ${}^{0}x_{r,i}$ and ${}^{0}y_{r,i}$, are known, and using the estimate of θ_r given by (5), one can set up the homogeneous transformation matrix, ${}^{0}T_{1,i}$, that translates position vectors from CS-1 to CS-0,

$${}^{0}T_{1,i} = \begin{pmatrix} \cos\theta_r & -\sin\theta_r & {}^{0}x_{r,i} \\ \sin\theta_r & \cos\theta_r & {}^{0}y_{r,i} \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

Considering that ${}^{0}\mathbf{m}_i = ({}^{0}x_i, {}^{0}y_i, 1)^T$ and ${}^{1}\mathbf{m}_i = ({}^{1}x_i, {}^{1}y_i, 1)^T$ are the homogeneous position vectors of the i^{th} observed QR code center point, M_i, referenced to CS-0 and CS-1 respectively (Fig. 7b), using (6) we can write

$$\begin{pmatrix} {}^{0}x_i \\ {}^{0}y_i \\ 1 \end{pmatrix} = {}^{0}T_{1,i} \begin{pmatrix} {}^{1}x_i \\ {}^{1}y_i \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_r & -\sin\theta_r & {}^{0}x_{r,i} \\ \sin\theta_r & \cos\theta_r & {}^{0}y_{r,i} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{1}x_i \\ {}^{1}y_i \\ 1 \end{pmatrix} \quad (7)$$

Solving (7) for ${}^{0}x_{r,i}$ and ${}^{0}y_{r,i}$, the following equations result

$${}^{0}x_{r,i} = {}^{0}x_i - \cos\theta_r\, {}^{1}x_i + \sin\theta_r\, {}^{1}y_i \quad (8)$$

$${}^{0}y_{r,i} = {}^{0}y_i - \cos\theta_r\, {}^{1}y_i - \sin\theta_r\, {}^{1}x_i \quad (9)$$

Using (8) and (9), ${}^{0}x_{r,i}$ and ${}^{0}y_{r,i}$ are calculated for all $i = 1,\ldots,n$. Then, using the median values, the RP's position coordinates referenced to CS-0 are calculated as

$${}^{0}x_r = median\{{}^{0}x_{r,i}\},\ i = 1,\ldots,n \quad (10)$$

$${}^{0}y_r = median\{{}^{0}y_{r,i}\},\ i = 1,\ldots,n \quad (11)$$

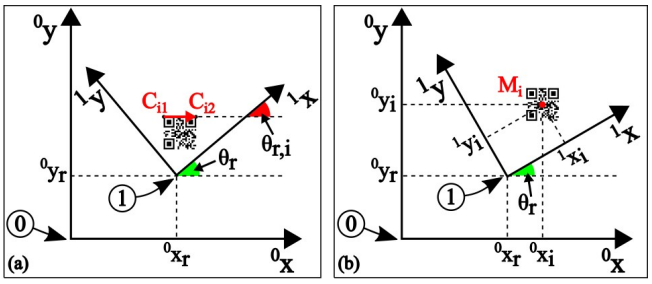


Fig. 7. (a) Calculation of RP's orientation, and (b) position components.

IV. Motion Control

A prototype omnidirectional robotic platform using four mecanum wheels has been developed to validate the developed algorithm and test the precision docking system throughout its development (Fig. 8). During the docking procedure it is assumed that the RP moves with geometric center linear velocity $v = (v_x, v_y)^T$, angular velocity ω, while the wheels rotate with speeds ω_1, ω_2, ω_3 and ω_4.

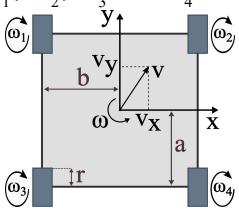


Fig. 8. Schematic representation of the prototype RP.

Considering the kinematic constraint involving point contact and rolling without slipping [16], the inverse kinematics sets the desired motor speeds according to

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} 1 & 1 & -(a+b) \\ -1 & 1 & (a+b) \\ -1 & 1 & -(a+b) \\ 1 & 1 & (a+b) \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} \quad (12)$$

where r is the wheel radius, and a, b are defined in Fig. 8.

Position and orientation control of the RP is realized with the use of a nested control structure composed of two loops; the outer loop is used to control the RP's pose while the inner loop maintains the desired wheel speeds (Fig. 9). The outer loop employs three independent PD controllers, running at 60 Hz and aiming at minimizing the RP's position e_x, e_y and orientation e_θ errors with respect to a targeted RP's pose (x_t, y_t, θ_t). Using position feedback provided by the localization system, see Fig. 9, the outer loop controller calculates the RP input linear $v^* = (v_x^*, v_y^*)^T$ and angular ω^* velocities, according to

$$v_x^* = K_{px} e_x + K_{dx} \dot{e}_x \quad (13)$$

$$v_y^* = K_{py} e_y + K_{dy} \dot{e}_y \quad (14)$$

$$\omega^* = K_{p\theta} e_\theta + K_{d\theta} \dot{e}_\theta \quad (15)$$

where K_{pi}, K_{di}, $i = x, y, \theta$, are proportional and derivative gains respectively. The gains were selected with a Ziegler–Nichols-type method. Due to the RP's symmetry, identical PD gains have been selected for v_x^*, v_y^*.

The corresponding wheel speeds $(\omega_1^*, \omega_2^*, \omega_3^*, \omega_4^*)$ are derived using (12) and fed into the fast inner PID control loop which is running at 1 kHz on the motor drivers, with feedback provided by individual quadrature motor encoders.

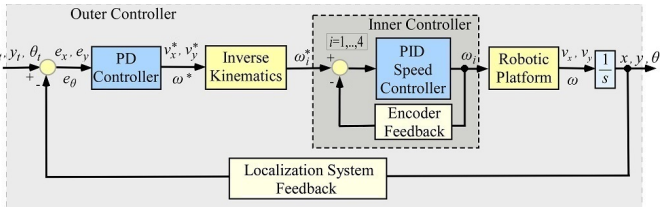


Fig. 9. Control of the prototype RP using a nested control structure.

V. Mechatronic System Implementation

A. *Hardware*

The RP's body frame (Fig. 10) is constructed of aluminum profiles. Four 4" AndyMark HD mecanum wheels, have been mounted symmetrically at the RP's corners; the wheels are specially designed to ensure that the distance between the wheel's axis and the contact point stays constant during rotation. Four Maxon DC motors, paired with GT2 timing belts, provide up to 8 Nm of continuous torque per wheel. A bench power supply provides electrical power for the individual electrical and electronic components. Two SDC2160 dual output motor controllers and two MGS1600GY magnetic guide sensors are attached to the RoboCAN network, a RoboteQ's proprietary meshed networking scheme allowing multiple devices to operate together as a single system. Control commands are calculated at a desktop computer and sent to the RoboCAN network through a USB interface.

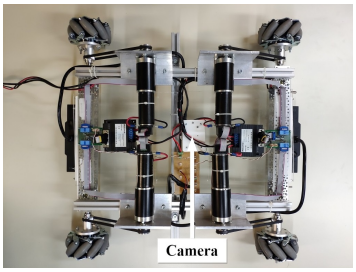
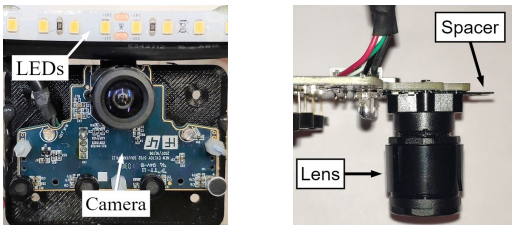


Fig. 10. The Robotic Platform (RP), with 4 mecanum wheels and camera.

The small height of the RP over the docking base allows the use of low-cost image sensors. Here, the Sony Eye Camera, integrating a low-cost high FPS image sensor, has been selected for the precision docking system. A mounting adapter has been designed and printed in-house to attach the camera at the center of the RP. The secure fixing of the camera, combined with the specially designed mecanum wheels, guarantee that the camera axis stays perpendicular to the floor throughout docking. To ensure stable video feedback rate of 60 Hz (1280x680 pixels) and minimize blur effect during motion, general-purpose LED lights are placed at its perimeter, (Fig. 11a). Moreover, a 3D-printed spacer with thickness of 0.4 mm has been placed between the sensor and the lens module (Fig. 11b) to adjust the focal point accordingly. A laser printer is used to print the docking base. The printer is fed with typical A4 plain paper and the printing resolution is set to 1200x1200 dpi.



(a) (b)

Fig. 11. (a) Camera installation and general-purpose LEDs on the RP, (b) Camera focal point adjustment using 3D-printed spacer.

B. *Software*

The desktop computer runs Ubuntu 16.04.6 LTS (Xenial Xerus), a free and open-source Linux distribution based on Debian. Individual ROS nodes [17] have been developed to implement the motion controller (*/control*), read position feedback from the localization process (*/rp_pose*), and send velocity commands to the motor drivers (*/robocan*). More specifically, the "/control" node constantly calculates the angular velocities of the wheels at 60 Hz rate (limited by the camera's feedback rate) and sends the control commands to the "/robocan" node. This is a dedicated ROS node that establishes data exchange between the "/control" node and the hardware components attached to the RoboCAN network. The "/robocan" node, in turn, is responsible for distributing the calculated angular velocities to the motor drivers. Communication between the "/control" and the "/robocan" nodes is based on request/ reply ROS services.

The implementation of the localization algorithm is made in the JAVA programming language with the use of the BoofCV library [18], an open-source computer-vision library. BoofCV provides powerful and robust tools dedicated for QR code detection; the corners of a QR code are recognized and sent to the localization algorithm with sub-pixel resolution. ROS does not support the JAVA programming language and for this reason a separate ROS node "/rp_pose" has been developed to establish real-time communication between the ROS ecosystem and the Java Virtual Machine (JVM) that is constantly running the localization algorithm. This function is achieved through network sockets using the client–server programming model. An overview of the software's architecture is shown in Fig. 12.

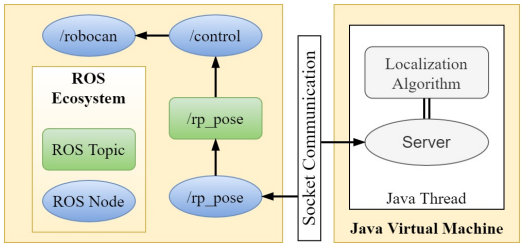


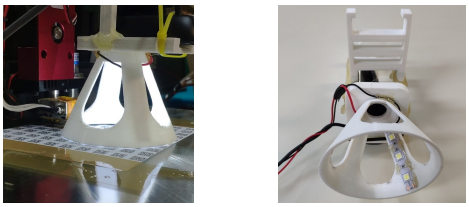
Fig. 12. Architecture of the software running on the Desktop PC.

VI. Experimental Results

A. *Preliminary x-y Position Experiments*

Preliminary experiments have been conducted to support the development of the localization algorithm and provide valuable information regarding the technical specifications and limitations of the docking system overall. To run these experiments, the motion system of a desktop 3D printer was employed, see Fig. 13a, and a mounting adapter was designed and printed in-house to secure the camera on the printer's end effector, see Fig. 13b.

An overview of the 3D printer experimental setup is presented in Fig. 14. Real-time position feedback (x_p, y_p) of the 3D printer is established using a modified version of the Marlin firmware [19]. The desired target trajectories are precomputed and saved in an external memory card. Two separate JAVA threads are running. The first receives position feedback from the 3D printer through a USB to RS232 interface. The second one implements the localization algorithm and constantly calculates the camera's pose with respect to the CS defined by the grid of QR codes. At the end of an experiment the position feedback and corresponding timestamps are saved.



(a) (b)

Fig. 13. (a) Camera installed on desktop 3D printer. (b) The mounting adapter developed in-house.

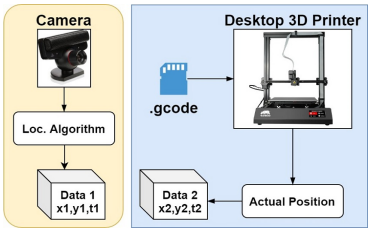


Fig. 14. Experimental 3D printer setup for the preliminary experiments.

Initially, a localization algorithm based on [13] was tried, where the camera's pose was calculated using a single, and randomly selected QR code. However, trajectory tracking experiments employing the motion system of the 3D printer, revealed positional errors far exceeding the maximum allowable value. To this end, the localization algorithm was enhanced by calculating the camera's pose based on every detected QR code and then applying 1-D median filtering to the estimated set of pose values. Therefore, steps 4 and 6 were introduced in the developed algorithm, see Fig. 5. These additions dramatically increased the resulting accuracy.

To evaluate the performance of the developed algorithm, the 3D printer was commanded to perform a circle trajectory of 1 mm radius. Position feedback (x_p, y_p) and (x, y) derived from the 3D printer and the localization algorithm, respectively, are shown in Fig. 15. Mean absolute errors of 49 μm on the x axis and 51 μm on the y axis, were observed. Furthermore, this experiment revealed that a total delay of 38 ms was expected to be introduced in the docking system, consisting of hardware (camera and USB2.0 interface) and software induced delays (8 ms for the localization algorithm).

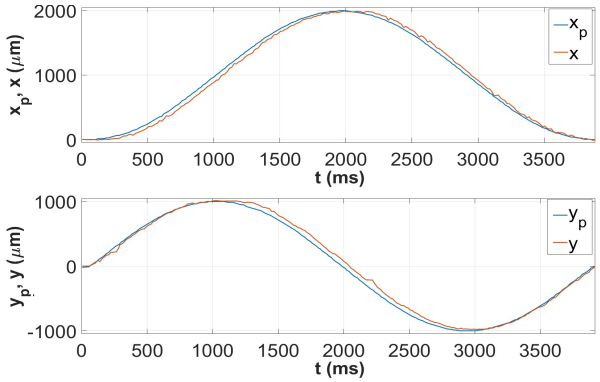


Fig. 15. Position feedbacks while following a circle trajectory.

B. *Precision Docking Experiments*

The end accuracy achieved by the developed docking system was investigated through comprehensive precision docking experiments. To determine the RP's position and orientation errors, the RP's frame was modified appropriately to hold two needles, $N_1(-c,c)$ and $N_2(c,-c)$, Fig. 16. The needles were placed in diametrically opposing locations to correspond to the actual locations of a ULD's locking pins. Two digital microscopes (DMs) embedding calibration slides of 100 μm pitch were employed to record position displacements of the needles. When both needle positions are within 500 μm of their target positions, precision docking is achieved.

At first, the RP was manually placed approximately at the center of the docking base. Using feedback provided by the localization system, final adjustments were made to perfectly align the coordinate systems of the RP and the docking base. Then, fine adjustments were made to the placement of the DMs to ensure that the angle of each needle was 45 degrees with respect to the calibration slide grid lines. At this point, screenshots were captured from the DMs to save the initial location of the needles.

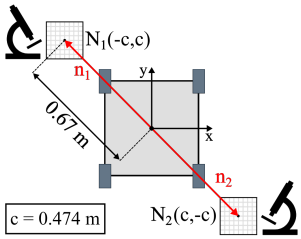


Fig. 16. Location of the two needles attached to the RP's frame.

Next, the RP was manually transferred to a random position and orientation on the docking base and then commanded to move back to the initial position, attain this position for two seconds and stop, see Fig. 17. This last phase of the experiment was repeated for fifteen times and all error measurements were collected. At the end of each iteration, screenshots from the DMs were captured and compared with the initial screenshots to determine the error displacements of the needles; (e_{x1}, e_{y1}) for N_1 and (e_{x2}, e_{y2}) for N_2, Fig. 18.

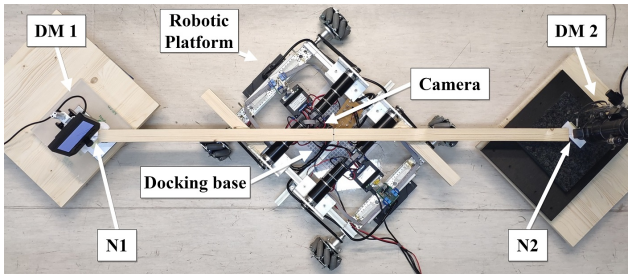


Fig. 17. Overview of the experimental setup. Precision docking is realized.

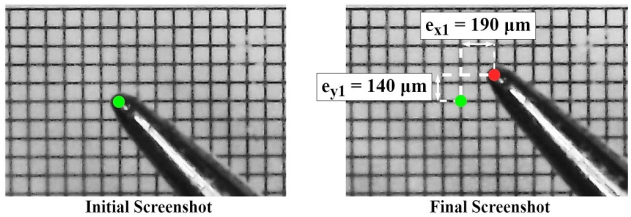


Fig. 18. Initial and final digital microscope screenshots to determine errors (e_{x1}, e_{y1}) for N1. The needle angle with respect to the grid is 45 degrees.

The RP's geometrical center coincides with the midpoint of the line segment N_1N_2. Therefore, the final position errors of the RP geometric center (e_x, e_y) and the angle error e_θ are calculated using the following equations:

$$e_x = \frac{(-c+e_{x1})+(c+e_{x2})}{2} = \frac{e_{x1}+e_{x2}}{2} \quad (16)$$

$$e_y = \frac{(c+e_{y1})+(-c+e_{y2})}{2} = \frac{e_{y1}+e_{y2}}{2} \quad (17)$$

$$e_\theta = 3\pi/4 - \angle({}^0\boldsymbol{n}_1 - {}^0\boldsymbol{n}_2) \quad (18)$$

The position errors that were derived from the screenshots are shown in Fig. 19. Mean position errors (Euclidian distance from the initial position) of 124 μm and 99 μm were observed for N_1 and N_2 respectively. It is noted that a maximum displacement of 250 μm is demonstrated, i.e., *half* of the maximum value set (500 μm). The calculated position and orientation errors of the prototype RP are presented in Fig. 20. Mean absolute errors of 27 μm (x axis), 25 μm (y axis) and 39 μm (Euclidian distance) were observed. Finally, regarding the RP's orientation, the results demonstrate a mean absolute error of 8.8 millidegrees and a maximum absolute error of 17.9 millidegrees.

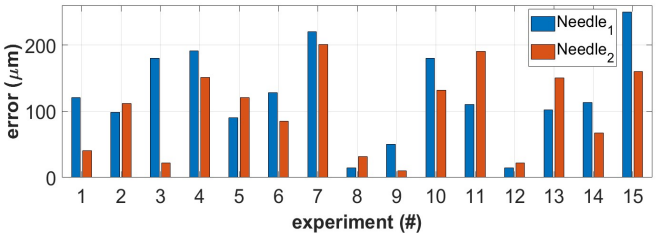


Fig. 19. Position errors of the two needles N_1 and N_2.

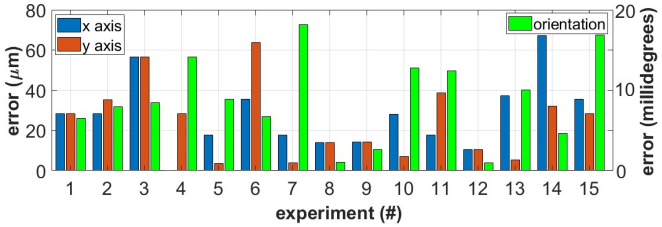


Fig. 20. Absolute position and orientation errors of the RP at its geometrical center.

VII. Acknowledgment

The authors wish to thank Mr. Kostas Machairas for his contribution in the development of the robotic platform.

VIII. Conclusion

The design and the implementation of a QR code-based high-precision docking system was presented that can be used by mobile robots and automatic guided vehicles in factories, warehouses and distribution centers. A low-cost camera was employed combined with a passive docking base with QR codes in the form of a two-dimensional matrix. A robust algorithm which provides positional feedback to a mobile vehicle's motion controller at 60 Hz while introducing 38 ms of delay was developed. Preliminary experiments based on a 3D printer's motion system supported the development of the localization algorithm. Comprehensive docking experiments using a prototype omnidirectional robotic platform validated the system and demonstrated excellent submillimeter end accuracy, exceeding position and orientation requirements. The developed system can be used in any type of holonomic mobile vehicle incorporating a single-board computer. Adaptation in non-holonomic vehicles is feasible, provided that a larger docking base is employed to allow maneuvers. The proposed system can be integrated easily in current material and cargo handling applications, as it uses low-cost elements and requires minimal maintenance due to the absence of active components in the docking base.

References

[1] Arkin, R., and Murphy, R., "Autonomous Navigation in a Manufacturing Environment", *IEEE Transactions on Robotics and Automation*, Vol. 6, Issue 4, Aug 1990, pp. 445 - 454.

[2] Arkin, R., Murphy, R., Pearson, M. and Vaughn, D., "Mobile Robot Docking Operations in a Manufacturing Environment: Progress in Visual Perceptual Strategies", *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, The Autonomous Mobile Robots and Its Applications*, 4-6 September 1989, Tsukuba, Japan.

[3] Bostelman, R., and Hong, T., "Review of Research for Docking Automatic Guided Vehicles and Mobile Robots", *National Institute of Standards and Technology*, 2016.

[4] Azadeh, K., De Koster, R. and Roy, D., "Robotized Warehouse Systems: Developments and Research Opportunities", *ERIM Report Series Research in Management ERS-2017-009-LIS, Erasmus Research Institute of Management,* 2017.

[5] Andersson, D., and Mansson, E., "Positioning and docking of an AGV in a clinical environment". *Master's Thesis,* Chalmers University of Technology, Department of Signals and Systems, 2012.

[6] Roeq.dk, "ROEQ – Precision Docking Stations". [Online]. Available: https://roeq.dk/products/docking-stations/.

[7] Bolanakis, G., Machairas, K., Koutsoukis, K., Mastrogeorgiou, A., Loupis, M., and Papadopoulos, E., "Automating Loading and Locking of New Generation Air-cargo Containers", *9th EASN Int. Conference on "Innovation in Aviation & Space",* 2019, Athens, Greece.

[8] Tong, F., Tso, S.K., and Xu, T.Z., "A high precision ultrasonic docking system used for automatic guided vehicle", *Sensors and Actuators A: Physical*, Vol. 118, Issue 2, February 2005, pp. 183-189.

[9] Wong, J., Nejat, G., Fenton, R., and Benhabib, B., "A neural-network approach to high-precision docking of autonomous vehicles/ platforms," *Robotica*, 25(4), 2007, pp. 479-492.

[10] Kim, M., Kim, H. and Chong, N., "Automated Robot Docking Using Direction Sensing RFID", *IEEE International Conference on Robotics and Automation*, 10-14 April 2007, Roma, Italy.

[11] Luo, R.C., Liao, C. and Lin, S., "Multi-Sensor Fusion for Reduced Uncertainty in Autonomous Mobile Robot Docking and Recharging", *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 10-15 October 2009, St. Louis, MO, USA, pp. 2203 - 2208.

[12] Quilez, R., Zeeman, A., Mitton, N., and Vadaele, J., "Docking autonomous robots in passive docks with Infrared sensors and QR codes", *TRIDENTCOM 2015*, June 24-25, Vancouver, Canada.

[13] Zhang, H., Zhang, C., Yang, W., and Chen, C., "Localization and navigation using QR code for mobile robot in indoor environment", *IEEE International Conference on Robotics and Biomimetics*, 6-9 Dec. 2015, Zhuhai, China.

[14] Atali, G., Garip, Z., Ozkan, S.S., and Karayel, D., "Path Planning of Mobile Robots Based on QR Code", *6th Int. Symposium on Innovative Technologies in Engineering and Science*, 9-11 November 2018, Alanya-Antalya, Turkey, pp. 31-38.

[15] Liu, Y., Xi, N., and Shen, Y., "High-Accuracy Positioning of an Industrial Robot Using Image/PSD-Based Hybrid Servo Control", *Int. Journal of Optomechatronics*, Vol. 5, Issue 2, 2011, pp. 170-187.

[16] Muir, P. and Neuman, C., "Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot", *IEEE International Conference on Robotics and Automation*, 1987, Raleigh, NC, USA.

[17] Quigley, M., et al., "ROS: an open-source Robot Operating System.", *IEEE International Conference on Robotics and Automation,* Workshop on Open Source Software, May, 2009, Kobe, Japan.

[18] Abeles, P., "BoofCV - An open source library for real-time computer vision", 2016. [Online]. Available: https://boofcv.org/.

[19] Marlinfw.org, "An open source firmware for 3D Printers", August, 2011. [Online]. Available: https://marlinfw.org/.