A low cost indoor localization system for mobile robot experimental setup

S Adinandra, A Syarif
Department of Electrical Engineering, Faculty of Industrial Technology, Universitas Islam Indonesia, Indonesia

E-mail: *s.adinandra@ui.ac.id

Abstract. Indoor localization becomes one of the most important part in mobile robot system. One fundamental requirement is to provide an easy-to-use and practical localization system for real-time experiments. In this paper we propose a combination of a recent open source virtual reality (VR) tools, a simple MATLAB code and a low cost USB webcam as an indoor mobile robot localization system. Using the VR tools as a server and MATLAB as a client, the proposed solution can cover up to 1.6 [m] x 3.2 [m] with the measurement position accuracy up to 1.2 [cm]. The system is insensitive to light, easy to move and can be quickly set up. A series of successful real-time experiments with three different mobile robot types has been conducted.

1. Introduction
Locomotion and positioning are two important parameters in mobile robot systems. Typical locomotion for mobile robot are differential drive, synchronous drive, tricycle drive, car type, tank type and walking type, see Figure 1 for examples. Each robot has its own potential and uniqueness. For example, the differential drive and car type are well known for their simplicities and functionalities. Both are widely used in application despite the lack of sideways movements. On the other hand, the synchronous drive walking type and the tank type are more flexible to environment changes. However, these types are typically designed to move slow to cover the overall working space and different terrains. (1)

In order to locate themselves in the working area, robots are equipped with localization systems. In mobile robot systems, localization is divided into two methods: indoor and outdoor localizations. For outdoor localization, global positioning system (GPS) is the main option that currently takes over almost the entire outdoor system. This phenomenon occurs due to the increasing accuracy of GPS combined with the easiness to find one as well as the low cost of the system. The DARPA challenge is one famous example of where mobile robots (car) autonomously move using GPS as their main navigation system, see. http://archive.darpa.mil/grandchallenge/.

However, GPS cannot be used indoor. Positioning of mobile robots is done by means of sensors attached to the robots or environment. The first method is sometimes called as relative indoor positioning system. A set of sensor is attached the robot or group of robots so that position and orientation of a robot relative to other robots or environment are known. Example are the use of ultrasound sensor, infrared, or on board camera. The advantage of relative method is robustness to
environment changes. Since sensors are embedded in robot system, changing environment can be easily compensated by the robot. However, it requires a non-trivial algorithm to process the environment information so that the position of the robot is obtained. Example of this method can be found in (2), (3), (4), (5), (6), (7), (8), and (9).

The second method, where sensors are attached to the environment, are sometimes called absolute localization. The “absolute” can be interpreted as the absolute position of the robot obtained within the environment. Since the sensors are attached to the environment, the position of the robot is always in the correct dimension of the environment. Typical solution is to attached camera system that covers the environment. The absolute method is not robust to environment changes but does not require a non-trivial algorithm since regular image processing method can be used. Examples can be found in (10), (11), (12), and (13).

![Differential drive](image1.png) ![Car type](image2.png) ![Tank type](image3.png)

**Figure 1.** Example of mobile robot locomotion.

The relative indoor localization is suitable for a mature mobile robot system, when a highly accurate localization with non-trivial algorithm. However, price can be another issue. For testing a simple test and experiment of a new locomotion algorithm, the absolute localization is preferable due to simplicity in installation. A moderate accuracy can be easily obtained, less robust to environment changes. Yet, the overhead camera used to capture the robot position typically is a camera that has a wide opening angle and insensitive to light. Thus, it is usually an expensive camera.

This paper proposes an easy-to-use and low cost solution to the absolute indoor localization system. Easy-to-use is interpreted as the quickness to set up the camera system. The low cost means the camera must be a webcam with cost less than 150 USD. The main contributions of this paper are as follow: i) an easy to set-open source and low cost indoor localization system, ii) a customizable client-server for easy scaling up the system in terms of number of robots. This paper provides the low cost version to the solution used in (11), (14), (15), and (16).

The rest of this paper is organized as follow. In preliminaries section, the concept of ARToolkit and how the open source software can be customized are presented. In system design section, the detailed of proposed solution is explained. In experimental results and analysis, the performance of the proposed solution is given. Finally, at the end of this paper conclusions and recommendation are given.

### 2. Preliminaries

The overhead camera localization system typically consists of two main components: a camera and an image processing system. The image processing system is a system that can read a specific marker. In this work, we modify an open source virtual reality kit that emerges in popularity. The software is chosen because of its low computing power needs. In this section a quick overview of the software architecture and detailed explanation of the overhead camera indoor localization system is given.

#### 2.1. AR Toolkit: an open source virtual reality software
The emerging popularity of Virtual Reality (VR) in gaming, supported by many smartphone, opens up a wide variety of open source VE programming tools. One of them is the ARToolkit (www.artoolkit.org). AR Toolkit is an open source platform that enable the user to create different type of virtual reality application. The website state the following:

ARToolKit 6 is a fast and modern open source tracking and recognition SDK which enables computers to see and understand more in the environment around them. It is being built from the ground up using modern computer vision techniques, up to the minute coding standards and new technologies developed in-house at DAQRI. ARToolKit 6 is being released under a Free and Open Source License that will allow the AR community to use the software in commercial products as well as for research, education and hobbyist development.

The AR Toolkit concept and architecture are shown in Figure 2. ARToolkit read custom marker so that the position, orientation (poses) and property of the marker can be read and processed by the software. At the user side, depending on the customized code, a specific virtual view will pop up. The openness of ARToolkit enable user to make any virtual environment, e.g. boxes, dolls, animals, vehicles, and others. As depicted in Figure 2, the “Unity native plugin” tackle the computer vision task and provide two important data from the marker reading, i.e. the poses and unique property of individual marker. Poses data is important to determine marker’s poses, i.e. virtual object poses. The property of marker is then visualized as different object according user’s design. This concept is typical for an image processing system. However, ARToolkit is able to display a 3D virtual object while keeping the computation needs low. This feature is very important since authors have done test using other software that result in higher computation needs.

![Figure 2. AR Toolkit concept (left) and software architecture (right).](image)

2.2. An overhead-camera indoor localization

Indoor localization for mobile robots can be divided into two categories: relative and absolute localization. Relative localization means that each robot is equipped with a distance/position sensor so that the position of robots can be determined by combination of measurement of each sensor. The advantage of this methodology is that robot can easily adapt to a new environment. As long as the workspace size is within the sensor range, robot position can be detected. The drawback of this method is the need to have a good tracking algorithm that sometimes can be too complex. Example of this method can be found in (3), (17), and (18).

The absolute localization is typically built using a camera system. One or more cameras are attached on top of a certain working space. Using a specific set of markers, the position and orientation of the robot can be tracked. Examples of this setup can be found in (14) and (19), see Figure 3. The advantage of this method is the simplicity in tracking the marker. Many easy-to-use and open source vision system are available. The main drawback of this system is less flexibility when the
size of the area changes. However, for testing and fast prototyping of a new algorithm, this method is preferable due to its easiness in installation.

Figure 3. An overhead camera indoor localization system for mobile robot experimental setup (14).

3. System design

3.1. Proposed solution

In this work we modify the ARToolkit so that the reading of the marker can be accessed using other software. The proposed solution is depicted in Figure 4.

Figure 4. The proposed indoor localization system.

Although the proposed solution looks trivial, the reality says otherwise. ARToolkit is an open source software that can be customized using only the ARToolkit SDK. This situation is not suitable to our purposes since we practically only need the position and orientation of the markers that will be put on top of the robot. Thus, we create a client-server method using socket programming to access the marker position and orientation, where the ARToolkit act as server and MATLAB as the client. The choice of using MATLAB is simply for fast prototyping and does not count for the “low cost” term. C or Java can also be used as a client. Figure 4 also shows that once the localization is set, a closed loop control mechanism can be easily implemented. Thus, testing, experimenting and evaluating with new algorithms become quick and easy.
3.2. AR toolkit modification: client-server

Zoom in to the detail, the proposed solution works like the scheme shown in Figure 5. From the “Unity scene” block, the resulting scene root is ported into UDP communication. In this way the position and orientation of the robot can be read in MATLAB (or other client). The relative position of the marker is calibrated to the real size of the working space inside the server. The UDP communication gives an easy way to customize data frame segmentation. The segmentation is needed to identify different marker ID’s position and orientation.

**Figure 5.** The detailed client-server proposed solution.

MATLAB, as the client, once receive information from the ARToolkit via UDP communication then computed the necessary control algorithm to minimize the difference between target and real-time robot movement. The ARToolkit continuously read the marker on top of the robot so that real-time movements of the robots can be tracked. In parallel, the nice 3D virtual reality still can be viewed. Furthermore, the 3D look can also be easily customized according to user specification.

3.3. System integration

The first step in integration, is to calibrate the relative frame position with respect to the real size of the working space. Since the goal is to have a quick and easy implementation, the calibration is done using the simplest way possible. Once the server and client is set, we need to do the following.

Suppose that the working space of the robot is as depicted in Figure 6. Put a marker to a position that later will be referred as original position, i.e. (0,0), in Figure 6 is shown by the “o” marker. Then from that point try to move the marker in two directions that later will be referred as “x” dan “y” direction/coordinates. While moving find the furthest distances/point from the “o” point that still can be tracked by ARToolkit, in Figure is denoted by “xx” and “yy”. At each “o”, “xx”, and “yy” point, record the value read by the software. Once recorded, use the simple equation

$$y = mx + c$$

(1)

to get the correct position of the marker with respect to real working space. Equation (1) need to be computed twice, each for x and y direction. The resulting calibration can be embedded either at server or client side. Once the calibration is completed, the client is ready for testing, experimenting or evaluating any control algorithm or mobile robot movements.
It is to be noted that in this work, for performance evaluation, the proposed solution is tested with omnidirectional mobile robot equipped with control algorithm presented in (11).

4. **Experimental results and analysis**

4.1. **Calibration results**

We test the proposed solution in a working space with the size of $1.6 \, [\text{m}] \times 3.2 \, [\text{m}]$, see Figure 6 (right). We use a webcam that cost less than 100 USD, positioned at the middle of the working space 1.5 [m] above the ground. Using the procedure described in Section 3.4, the resulting calibration for the test area is as follow:

$$x_{\text{real}} = 0.49x_{\text{read}} - 6.223$$

$$y_{\text{real}} = 0.501y_{\text{read}} - 7.314$$

where $x_{\text{real}}$ and $y_{\text{real}}$ represent the real position of the robot in the working space and $x_{\text{read}}$ and $y_{\text{read}}$ represent the reading (in pixels) by the overhead camera and ARToolkit. To get an easy measurement verification, the working space is covered with a layer of banner marked with boxes that indicates a certain Cartesian coordinate position, see Figure 7. It is to be noted that the system can be directly used at other place with similar size as long as the camera is positioned 1.5 [m] above the ground and at the center of gravity of the area.

![Figure 6. Example of working area and how to calibrate the resulting measurement from ARToolkit.](image)

4.2. **Static experiments: test the accuracy of the measurement**

The static experiments are mainly to check the consistency of the tracking of robot position with static objects. The experiment scenario is simple. Three markers/robots are placed on the area. We modify the position and orientation of the robots. For each modification we record the position captured by the server-client as well as the manual measurement of center of the robots on the area. Some of the results are shown in Figure 7. The standard deviation of the difference between the client reading and manual measurements of each markers are in the range of 0.8 [cm] to 1.2 [cm] both for $x$ and $y$ direction, which indicate the accuracy of the proposed system. The accuracy cannot go lower because of the error in measuring the position of the robots, i.e. actual robot position is measured by manually computed the center of the robot at the area, see Figure 7.
4.3. Dynamic experiments: test of trajectory tracking control problem

For the dynamic experiments, two scenarios are considered. For the first scenario, similar scenarios to the static experiments are considered. However, in the dynamic case, the robots are moved manually and automatically (without controller) so that the real-time measurement are recording data of moving robots/markers. All robots are of holonomic type, i.e. three omni wheels and four omni wheels type.

The resulting experiments are shown in Figure 8 and Figure 9. As illustrated in Figure 8, the recording for multiple object is done perfectly with similar accuracy to the static measurement. The left hand side of Figure 8 shown the direction of the movement of the robots, while the right hand side shows recorded data. In addition, in Figure 9 the robots are given orders to make a circle movement via a certain set of individual wheel speed. Since it is not controlled, individually robot executes the circle movement differently that results in random movement. Similar to the result in Figure 8, the random movement can be captured real-time with the sampling time for each measurement given in bottom-right of Figure 9. These two results show that the proposed solution can handle different type of movements and speeds of the robots.

Figure 7. Static experimental results. The area is covered with a banner marked by some Cartesian coordinates. The measurement is at the real robot is computed at the center of individual robot/marker. The “0”, “1”, and “3” indicates the marker IDs.

The second dynamic experiments also the last experiment is to test a trajectory tracking controller using the proposed solution. This is similar to the circular test, only now the robot’s movements are controlled. Without losing generality, primarily based on the first dynamic test, the test is done only with one robots. The performance of a group robots should follow the single robot results.

For the sake of clarity, similar reference trajectory as in (7) is considered, i.e. the robot must follow an-eight shape like geometry. The reference is timed for each sampling time. Due the limited working
space only movement in quadrant one is considered. Similar control algorithm as in (7) is implemented for a three wheels omnidirectional robot. The result is depicted in Figure 5.

As can be observed in Figure 9, the recorded movement are smooth and keep the accuracy as the previous two experiments. The robot can track the given reference (indicated by the solid black line). Thanks to the control algorithm proposed in (7), the robot movement is coinciding with the reference. The proposed indoor localization is able to record the mobile robot movement well.

**Figure 8.** The resulting of dynamic experiments (1): manually move the robots.
Figure 9. The resulting of dynamic experiments (2): uncontrolled circular task; top-left: initial position, top-right: end position, bottom-left": recorded movement, bottom-right: sampling time.

Although the real-time measurement is influenced by the performance of the control algorithm, the results suggest that the proposed solution fulfil the purpose to have an easy to implement indoor localization system. The effect of different control gain can be observed easily, as well the performance of other control algorithm with restriction on integer reading at the client side.

Compare to other method, e.g. (11), (14) or (20), the system provide less accurate position measurement. However, the result in Figure 10 suggests that for fast prototyping and testing a control algorithm, the proposed solution serves the purpose of having an easy-to-set and low cost indoor localization well.

5. Conclusions
In this work we proposed an easy to use and low cost indoor localization system using a combination of a cheap webcam and a customized ARToolkit-MATLAB client-server image processing system. The system can track several mobile robots/marker in real time with the accuracy of 0.8 [cm] to 1.2 [cm] both for x and y direction respectively. The system has been validated using three mobile robot system as well as a closed loop trajectory tracking control system. The proposed solution is suitable for rapid prototyping a control algorithm or analysing mobile robot characteristic.

For future work the calibration process will be done automatically. It means that once the system start it first will calibrate the relative camera measurement with respect to the real working space size without the help of any operator. Furthermore, another communication protocol can be revisited so that a better sampling time can be achieved.
Figure 10. The resulting movement of the trajectory tracking control problem test.

References
[1] Siciliano B and Khatib O 2008 Handbook of Robotics (Springer-Verlag Berlin)
[2] Di Giampaolo E, Martinelli F 2014 Mobile Robot Localization Using the Phase of Passive UHF RFID Signals IEEE Transaction on Industrial Electronics vol 61(1) pp 365–76.
[3] Borenstein J, Evererr H, Feng L, and Wehe D 1997 Mobile Robot Positioning: Sensors and Techniques. J Robot Syst. vol 14(4) pp 231–49.
[4] Kim S and Kim B 2013 Dynamic Ultrasonic Hybrid Localization System for Indoor Mobile Robots IEEE Transaction on Industrial Electronics. vol 60(10) pp 4562–73.
[5] Mi J and Takahashi Y 2016 Design of an HF-Band RFID System with Multiple Readers and Passive Tags for Indoor Mobile Robot Self-Localization Sensors vol 16
[6] Cheng H, Chen H, and Liu Y 2015 Topological Indoor Localization and Navigation for Autonomous Mobile Robot. IEEE Trans Autom. Sci. Eng. vol 12(2) pp 729–38.
[7] Chen K, Mi Y, Shen Y, Hong Y, Chen A, and Lu M. 2017 SparseLoc: Indoor Localization Using Sparse Representation IEEE Access vol 5 pp 20171–82.
[8] Kang W and Han Y. 2016 SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization IEEE Sens. J. vol.15(5) pp 2906–16.
[9] Zayets A and Steinbach E 2017 Robust WiFi-based indoor localization using multipath component analysis Proceedings of International Conference on Indoor Positioning and Indoor Navigation (IPIN) pp 1–8
[10] Kostic D, Adinandra S, Caarls J, Van De Wouw N, and Nijmeijer H 2009 Collision-free tracking control of unicycle mobile robots Proceedings of the IEEE Conference on Decision and Control
[11] Adinandra S and Ratnawati DA 2016 A Practical Coordinated Trajectory Tracking for A Group of Mixed Wheeled Mobile Robots with Communication Delays TELKOMNIKA (Telecommunication Comput Electron Control) vo. 14(4) pp 1397–407.
[12] Winterhalter W, Fleckenstein F, Steder B, Spinello L, and Bugard W 2015 Accurate Indoor Localization for RGB-D Smartphones and Tablets given 2D Floor Plans ProceedingIEEE/RSJ International Conference on Intelligent Robots and System pp 3138–43.
[13] BniLam N, Ergeerts G, Subotic D, Steckel J, and Weyn M 2017 Adaptive probabilistic model
using angle of arrival estimation for IoT indoor localization Proceedings International Conference on Indoor Positioning and Indoor Navigation (IPIN) pp 1–7

[14] Kostic D, Adinandra S, Caarls J, Van De Wouw N, and Nijmeijer H 2010 Saturated control of time-varying formations and trajectory tracking for unicycle multi-agent systems Proceedings of the IEEE Conference on Decision and Control

[15] Garcia De Marina H, Cao M, and ayawardhana B 2015 Controlling rigid formations of mobile agents under inconsistent measurements IEEE Trans Robotics and Autonomous Systems vol 31(1) pp 31–9.

[16] Yang Q, Fang H, Cao M, and Chen J 2016 Distributed trajectory tracking control for multiple nonholonomic mobile robots. IFAC-PapersOnLine vol 49(4)pp 31–6

[17] Yi S-Y and Choi B-W 2004 Autonomous Navigation of Indoor Mobile Robot Using Global Ultrasonic System. Mobile Robots: Perception and Navigation pp 383–93

[18] Li I-H, M.C C, Wang W-Y, Su S-F, and Lai T-W 2014 Mobile Robot Self-Localization System Using Single Webcam Distance Measurement Technology in Indoor Environments. Sensors vol 14 pp 2089–109.

[19] Adinandra S, Caarls J, Kostic D, Verriet J, and Nijmeijer H 2014 Flexible transportation in warehouses Automation in Warehouse Development (Springer-Verlag Berlin)

[20] Sadowska A, van den Broek T, Huijbers H, Kostic D, Van De Wouw N, and Nijmeijer H 2011 A virtual structure approach to formation control of unicycle mobile robots using mutual coupling International Journal of Control vol 84(11) pp 1886–902

Acknowledgments

The authors would like to thank Department of Electrical Engineering, Faculty of Industrial Technology, Universitas Islam Indonesia for providing financial supports of this work.