DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 12 No. Especial 2 (2024) 28–33

Comparisson between generalized geometric triangulation and odometry
Comparativa entre triangulación geométrica generalizada y odometría

E. Mar-Castro [a,*], L.M. Aparicio-Lastiri [a], O.V. Pérez-Arista [a], R.S. Núñez-Cruz [a], E.D. Antonio-Yañez [a]

[a]*Laboratorio experimental de automatización y control, Universidad Politécnica de Tulancingo, 43629, Tulancingo, Hidalgo, México.*

Resumen

La localización en robótica móvil es fundamental para la realización de tareas autónomas. Por este motivo, se han desarrollado diferentes algoritmos para estimar la pose del robot, ya sea de forma relativa o absoluta. Una de las más conocidas es la odometría basada en ruedas, la cual es fácil de implementar pero con la desventaja de que el error en la estimación tiende a aumentar conforme pasa el tiempo produciendo un cálculo poco confiable. Por el contrario, los algoritmos de localización absoluta como la Triangulación Geométrica Generalizada (TGG) ofrecen una mayor precisión, aunque su implementación puede requerir sistemas de medición más avanzados y la estimación de la pose puede ser lenta. Este trabajo compara estos dos algoritmos y muestra lo que sucede con la estimación de pose después de un período de tiempo. Los resultados presentados se obtuvieron en una zona de pruebas real equipada con un robot Turtlebot3 modelo burger.

Palabras Clave: Pose, robot diferencial, odometría, triangulación geométrica generalizada, ArUco.

Abstract

The localization in mobile robotics is essential for carrying out autonomous tasks. For this reason, different algorithms have been developed to estimate the robot's pose, either relatively or absolutely. One of the best known is Wheel-based Odometry, which is easy to implement but the error tends to increase with respect to time producing an unreliable estimation. In contrast, absolute localization algorithms such as Generalized Geometric Triangulation (GGT) offer higher accuracy, although their implementation may require more advanced measurement systems, and pose estimation can be slow. This work compares these two algorithms and shows what happens with the pose estimation when used for period of time. The presented results were obtained in a real testing area equipped with a Turtlebot3 model burger robot.

Keywords: Pose, differential robot, odometry, generalize geometric triangulation, ArUco.

1. Introduction

Localization is an essential part of the navigation of mobile robots, which consists of estimating the position and orientation of the robot at each moment in time (Campbell *et al.*, 2020). To achieve this, many algorithms have been designed that use sensors such as encoders, inertial measurement units, cameras, LiDAR, ultrasonics, etc. Absolute measurement techniques are more accurate than those that depend on integrating changes in the robot's motion, i.e., relative or dependent on a previous pose state in space. Typically, absolute localization techniques use sensors that allow the robot to sense its environment and estimate its position based on this.

Some robots often use Global Positioning Systems (GPS), which is an absolute localization system that utilizes satellite information to estimate their position. The use of this type of technology is beneficial in environments where it is guaranteed that the GPS signal will be perceivable (Ohno *et al.*, 2004). However, in many cases, such systems are not practical because robotic systems require information with the highest possible accuracy. Moreover, the GPS signal may not be perceivable in the robot's environments. To address these challenges, one of the main techniques employed for this purpose is odometry, which relies on sensor information that measures the robot's motion. This information can estimate changes in its position over time, and a relative pose concerning the initial pose can be

*Autor para correspondencia: enrique.mar1292@upt.edu.mx
Correo electrónico: enrique.mar1292@upt.edu.mx (Enrique Mar-Castro), luis.aparicio1293@upt.edu.mx (Luis Mario Aparicio-Lastiri), omar.perez2115005@upt.edu.mx (Omar Vicente Pérez-Arista), rafael.nunez@upt.edu.mx (Rafael Stanley Núñez-Cruz), elba.antonio@upt.edu.mx (Elba Dolores Antonio-Yañez).

Historial del manuscrito: recibido el 12/12/2023, última versión-revisada recibida el 12/12/2023, aceptado el 10/03/2024, publicado el 22/04/2024. **DOI:** https://doi.org/10.29057/icbi.v12iEspecial2.12240

calculated. However, it is well known that the estimated pose drifts over long integration times in relative localization algorithms due to the accumulation of sensor measurement noise, whether from internal or external factors (Siegwart *et al.*, 2011).

On the other hand, absolute localization techniques also are used, which do not rely on previous pose states. Instead, implementing these techniques allows robots to calculate their pose based on the current measurement acquired by sensors. To achieve this, prior information about the environment is needed. This means that the absolute positions of specific objects must be known so that, based on them, the robot's position can be estimated with respect to the inertial reference frame of the workspace as presented in (Cavanini *et al.*, 2017). Accuracy can be even more challenging indoors due to signal attenuation and lack of direct visibility. Furthermore, absolute localization often requires the installation of additional infrastructure, such as visual markers, beacons, or GPS antennas, which can be costly and restrict the robot's mobility to specific areas.

The comparison of localization strategies is essential in the field of mobile robotics because the results they yield enable users to choose one method over others. In the literature, there are reported comparisons of localization strategies that demonstrate the accuracy of the tested algorithms, as shown in (Fauser *et al.*, 2017) (Almeida *et al.*, 2018). On the other hand, (Esan *et al.*, 2020) provides a review of navigation systems in mobile robots, and it also mentions some of the most commonly used localization methods in these systems, which are classified based on the type of localization they offer, i.e., relative or absolute. The variables analyzed in this review include implementation cost, accuracy, computational power, among others.

The main contribution of this work is to highlight the advantages and disadvantages of wheel-based odometry (WBO) and generalized geometric triangulation (GGT) through experimental tests that measure the magnitude of path following errors. Both algorithms were chosen because of their relatively easy implementation. Furthermore, these algorithms allow for estimating the robot's pose. However, generalized geometric triangulation is a method that does so absolutely. This means that the expected results, when compared to WBO, should be more robust and accurate. In Section II explains the Odometry algorithm, detailing its components. Section III provides a comprehensive description of the critical elements of the GGT algorithm. Section IV discusses the computational implementation of both algorithms. Section V outlines the conducted test. Section VI presents the test results, while in Section VII, the conclusion and future work are mentioned.

2. Pose estimation using Odometry

Currently, there are different types of Odometry due to the use of various sensors to estimate the robot's motion. For instance, Odometry that uses cameras is known as Visual Odometry. According to (Scaramuzza y Fraundorfer, 2011), this algorithm estimates the system's movement by detecting features in the environment. Therefore, to achieve good results, it is necessary for the scene to be well-illuminated and have sufficient textures for detection.

If lasers, such as LiDAR, are used instead of cameras, the algorithm is known as Laser Odometry. LiDARs function as environment scanners, generating point clouds that can be three-dimensional or two-dimensional. By comparing these point clouds, the robot's pose can be estimated. This type of Odometry is typically used in indoor environments. Still, algorithms have also been developed that combine LiDAR with inertial sensors and encoders to apply it in outdoor settings, as shown in (Zhou *et al.*, 2017). LiDAR odometry is used as an auxiliary positioning system in outdoor environments to reduce accumulated error when applying other types of Odometry based on wheel encoders and inertial measurements.

Inertial Odometry, as described in (Campbell *et al.*, 2020), utilizes accelerometers, gyroscopes, and magnetometers to estimate the robot's pose. However, these systems are prone to error accumulation, which means that over time, the estimated pose will deviate significantly from the actual pose.

Another common type of Odometry, widely used in ground mobile robots, is Wheel-based Odometry. This technique requires knowledge of the wheel speeds to estimate the robot's pose. This algorithm shares the same disadvantage as inertial Odometry since it relies on integrating the wheel speeds, which are often affected by environmental noise, such as wheel slippage. The accumulation of these errors leads to posing estimation drift.

In this work, WBO was used. This algorithm was developed for a differential robot, and the forward kinematic model was calculated for its implementation. The derivation of the differential robot model has been widely reported in the literature. According to (Siegwart *et al.*, 2011), the contribution of each wheel to the robot's motion can be expressed as follows:

$$\dot{\xi}_v = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix}. \tag{1}$$

Where $\dot{\xi}_v$ is the vehicle's velocity vector $[\dot{x}_v, \dot{y}_v, \dot{\theta}_v]^T$ with respect to its own reference frame O_v. $\dot{\varphi}_1$ and $\dot{\varphi}_2$ are the angular velocities of the wheels, r is the wheel radius, and l is the distance between the wheel and the center of the axis that connects them.

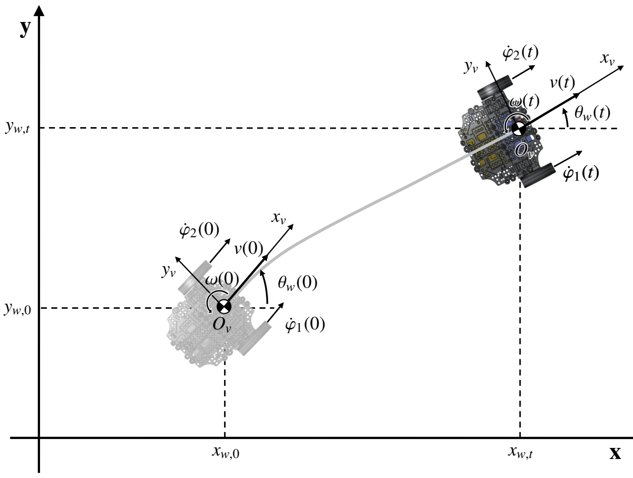

Figure 1: Odometry algorithm representation.

To obtain the robot's velocities in the world reference frame

O_w, it is necessary to rotate $\dot{\xi}_v$ from O_v to O_w, as follows:

$$\dot{\xi}_w = \begin{bmatrix} \dot{x}_w \\ \dot{y}_w \\ \dot{\theta}_w \end{bmatrix} = R(\theta)^{-1}\dot{\xi}_v. \quad (2)$$

Where $R(\theta)$ is the rotation matrix around z axis and is defined as follows:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

To estimate the robot's position in the world frame, vector $\dot{\xi}_w$ needs to be integrated with respect to time:

$$\xi_w(t) = \int_0^t \dot{\xi}_w(T)\,dT + \xi_w(0). \quad (4)$$

Figure 2 shows the blocks diagram of the WBO algorithm.

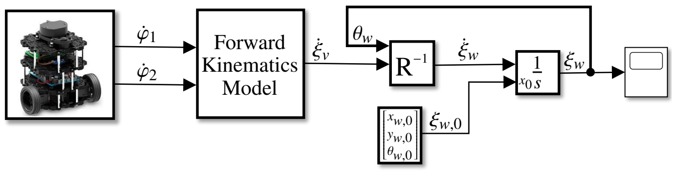

Figure 2: Robot pose transformation to inertial reference frame.

The estimation produced by the original WBO algorithm can be improved using sensor fusion techniques, such as Kalman filter (Housein *et al.*, 2022) with the estimation produced by other relative localization algorithms (Liu *et al.*, 2016) or even with absolute localization algorithms (Lobo *et al.*, 2014).

3. Pose Estimation using Generalized Geometric Triangulation Algorithm

The Geometric Triangulation algorithm measures the *angles* between the robot's x_v axis and each marker, whose positions are known in the world coordinate system O_w. This is what sets this method apart from others like Trilateration, which relies on measuring the *distances* between the robot and the markers.

To estimate the robot's position using this algorithm, the robot must perceive at least three markers, as shown in Figure 3. The markers must be uniquely identified in order to associate its corresponding position, after that the orientation of each marker with respect to the robot is calculated. The variable λ_i denotes the relative angle of each marker i with respect to x_v.

The angle λ_{ij} is the "viewed" oriented angle between markers i and j, i.e., $\lambda_{ij} = \lambda_j - \lambda_i$. If $\lambda_i > \lambda_j$, then $\lambda_{ij} = 2\pi + (\lambda_j - \lambda_i)$. This defines an arc between these two markers, which represents a set of possible robot positions.

When using three markers the point of intersection between the arcs generated by λ_{12} and λ_{31} corresponds to the robot's position. According to (Esteves *et al.*, 2004), the robot's pose is calculated using the following equations:

$$x_w = x_1 - L_1 \cdot \cos(\phi + \tau), \quad (5)$$

$$y_w = y_1 - L_1 \cdot \sin(\phi + \tau), \quad (6)$$

$$\theta_w = \phi + \tau - \lambda_1. \quad (7)$$

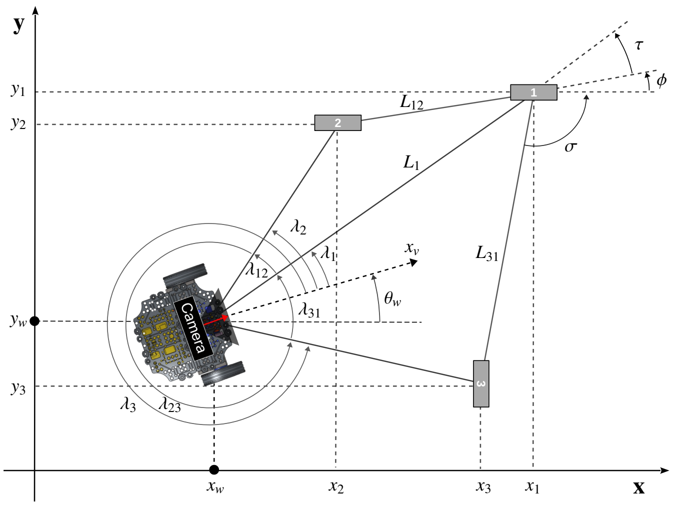

Figure 3: Calculations performed in the GGT algorithm.

Where (x_1, y_1) represents the coordinates of marker 1 in the Cartesian plane of the workspace, L_1 is the distance between the robot's reference frame and marker 1. ϕ is the angle resulting from the intersection between the x axis of the inertial reference frame of the world and the line formed between markers 1 and 2. The angle τ is the angle of the arc formed between marker 2 and the robot, measured from marker 1.

To calculate L_1, the following equation is used:

$$L_1 = \begin{cases} \frac{L_{12}\cdot\sin(\tau+\lambda_{12})}{\sin(\lambda_{12})} & \text{If } |\sin(\lambda_{12})| > |\sin(\lambda_{31})| \\ \frac{L_{31}\cdot\sin(\tau+\sigma-\lambda_{31})}{\sin(\lambda_{31})} & \text{otherwise} \end{cases}. \quad (8)$$

L_{12} and L_{31} are the distances between the same markers. To calculate them, we use the formula for the distance between two points:

$$L_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \quad (9)$$

Angle σ is defined as the angle formed between x_w and the line formed by markers 1 and 3, plus the angle ϕ with the intersection point at the coordinates of marker 1.

The calculation of τ is performed as follows:

$$\tau = \tan^{-1}\left[\frac{\sin\lambda_{12}\cdot(L_{12}\cdot\sin\lambda_{31} - L_{31}\cdot\sin\gamma)}{L_{31}\cdot\sin\lambda_{12}\cdot\cos\gamma - L_{12}\cdot\cos\lambda_{12}\cdot\sin\lambda_{31}}\right]. \quad (10)$$

The algorithm implemented in this work is the generalized geometric triangulation, which differs from traditional geometric triangulation in that the markers do not need to be arranged consecutively counterclockwise. The angles λ_{12} and λ_{31} can be equal to or greater than 180°, and the markers can be placed anywhere in the working plane because the algorithm can estimate the pose even if the robot is outside the triangle formed by the three markers.

However, there are two critical restrictions to consider for this algorithm. Firstly, it is well known in the literature that at least three markers must be "visible" to estimate the robot's position. If fewer than three markers are visible, the robot's pose cannot be calculated using this method because there would be multiple possible poses.

Secondly, even if three markers are visible, it does not guarantee that the position estimation can be performed correctly. This is because the algorithm is limited when the circles generated by the arcs of the angles λ_{12} and λ_{31} lie on the same circumference. In this case, the intersection of these two arcs will be another arc rather than a single point, similar to the first restriction.

The authors in (Pierlot y Van Droogenbroeck, 2014), mention several triangulation algorithms with 3 objects in a general context. Additionally, it is noted that several authors have implemented multi-markers triangulation algorithms. These algorithms propose methods to improve accuracy in pose estimation when more than 3 markers are available. Among these methods, computing the average of the estimated poses of *n* markers is mentioned.

By leveraging the principles of geometry and the relationships between the markers, this method allows for robust and precise localization. However, it is essential to consider potential challenges, such as occlusions or excessive noise in marker detection, which may affect the reliability of the localization results. Overall, geometric triangulation provides a valuable technique for enhancing the pose estimation capabilities of mobile robots in various applications, including navigation, mapping, and human-robot interaction.

4. Computational implementation of the pose estimators

The presented two algorithms were implemented on a commercial differential robot, named TurtleBot3 Burger model. This robot has 2 Dynamixel XL430-W250 motors that incorporate absolute position encoders. Also has a Raspberry Pi 4B with 4 GB of RAM and a Raspberry Pi Camera V2.0. The operating system used is Ubuntu Mate 20.04 for Raspberry pi usage. The image processing, pose estimation using the Generalized Triangulation algorithm and trajectory generation were performed on a separate external computer with an Intel Core i5 processor and 16 GB of RAM. To compare the perform of both localization algorithms, a system vision was place in the top of the test area. The images from this system are processed in a third computer with an Intel core i7 processor and 8 GB of RAM.

There are several projects in the field that uses markers to estimate pose using artificial vision, for this project ArUco markers were used. While it is true that with a single marker and knowledge of its position in the Cartesian plane, the robot's pose could be estimated (as demonstrated in the validation method employed and explained in Section V), ArUco markers were chosen for their ease of generation and the extensive documentation available for their implementation. Also, other technology could be use, for example, in (Kristalina *et al.*, 2016) presents an application using Wireless Sensor Networks (WSNs) with Zigbee radio-frequency modules.

The ROS package used for image processing is called `aruco_detect`, which calculates the transformation between the camera and the detected markers. This package receives information from another package called `usb_cam`, which is the interface between the camera and the software. It publishes a message with the captured camera information and applies image corrections using calibration parameters.

Once the transformation of the markers with respect to the camera is estimated in the `aruco_detect` node, the orientation of each marker is estimated using the position information (x_i, y_i) obtained from marker *i*. This is done using the trigonometric function $arctan(y/x)$.

The process described above is carried out in a separate node called `aruco_getOrientation`, part of the package developed in this work named `triangulation`. This node subscribes to the topic `/fiducial_transforms` and publishes another topic named `/arucos_detected_info`. The information published is a custom data type that includes the marker IDs and their orientations with respect to the camera.

Now, the data vector provided by the topic `/aruco_detected_info` is read in another node called `GGT_algorithm`, where the entire GGT algorithm is implemented.

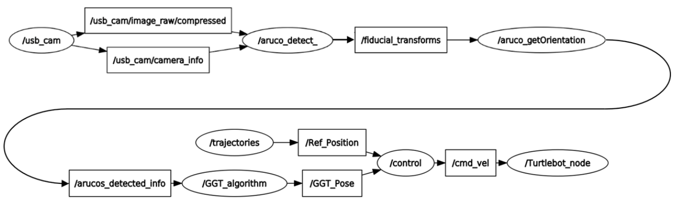

Figure 4: ROS nodes graph.

In the proposed approach the robot's pose can be estimated using all the markers detected by the robot. To achieve this, we implemented an algorithm for generating unique combinations. The pose will be estimated once if the robot detects only three markers. However, ten poses will be estimated if the robot detects five markers. In other words, z

Where C_n is the number of possible combinations without repetition, and x represents the number of markers used for the combinations, which in our case is 3. The number of markers n can be determined by the size of the data vector received by the `GGT_algorithm` node. The implementation of this algorithm allows us to reduce the variation in the estimation of the robot's pose when more than three markers are detected, thereby improving accuracy. Each calculated pose is passed through an Exponential Moving Average (EMA) filter, which is defined by the following formula:

$$\xi_{I,s} = \alpha M + (1 - \alpha)\xi_{I,s-1}. \tag{11}$$

In this case, $\xi_{I,s}$ represents the absolute position vector of the robot, M is the unfiltered estimated position vector, $\xi_{I,s-1}$ is the previous absolute position vector of the robot, and α is the gain applied to the filter. The value of α ranges between 0 and 1, where a value close to 1 gives more weight to the current estimation, reducing the filtering effect, while a lower value gives more importance to the previous pose, resulting in a slower response from the filter.

Regarding implementing the Odometry algorithm, we utilized information from the `turtlebot3_bringup` package, which publishes the `/odom` topic containing the robot's odometry information. This package was also used to control the angular and linear velocities of the robot using both the Odometry and GGT algorithms. This is achieved by publishing the velocity commands to the `/cmd_vel` topic.

5. Experimental Testing

To conduct the comparison, a test area was built (figure 5) based on the requirements of the GGT algorithm in such a way that it could ensure that the robot could see at least 3 markers while following the given path. For this reason, a total of forty-eight markers of size 7x7 bits, measuring 10x10 cm, were placed around the walls that delimit the area. This were spaced 15 cm apart from center to center between each marker. Also, other factors that impact the selection of the size are the camera lens aperture and the distance at which the robot perceives the markers.

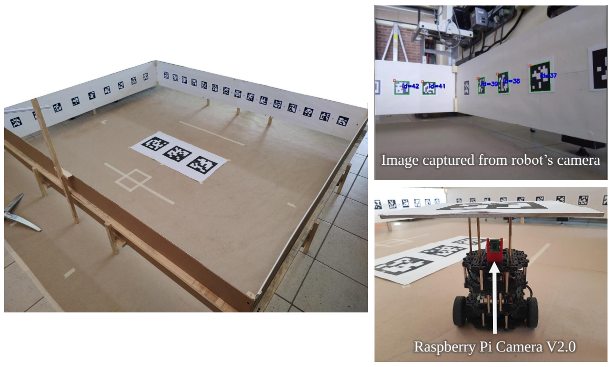

Figure 5: Test area built and robot adaptations.

Four other markers were placed, with three fixed to the floor and one mounted on the robot. These four markers are used to estimate the robot's pose in an absolute manner by calculating the transformation between the marker on the robot and each fixed floor marker. As mentioned in Section IV, another position estimation system was designed using a camera placed above the test zone. This system was implemented on a separate computer to avoid affecting the performance of image processing and pose estimation when using the GGT algorithm.

The test to compare the pose estimation between the algorithms mentioned in this work involves the robot following a predefined trajectory consisting of straight segments and semicircular arcs for a defined duration. The shape of the trajectory can also be seen in figure 6, indicated in blue. In the test, the robot completed a total of 15 laps, with each lap taking 1 minute. The trajectory was generated using parametric equations. The following two equations were used to create the semi-circular arcs:

$$p_x = r\cos(\psi(t)) + x_c, \quad (12)$$

$$p_y = r\sin(\psi(t)) + y_c. \quad (13)$$

Where r is the radius of the semicircle, $\psi(t)$ is the angle as a function of time, and x_c and y_c are the offsets from which the semicircle will start. The parametric equations for the straight line segments are given by:

$$p_x = (t - t_i)\frac{x_f - x_i}{t_f - t_i} + x_i, \quad (14)$$

$$p_y = (t - t_i)\frac{y_f - y_i}{t_f - t_i} + y_i. \quad (15)$$

Where t is the current time in seconds, t_i is the time at which the straight segment should start, t_f is the time at which it should end, x_i and y_i are the coordinates from which the segment will start, and x_f and y_f are the coordinates at which it should end. The advantage of generating the trajectory in this way is that the references for the controllers are smooth.

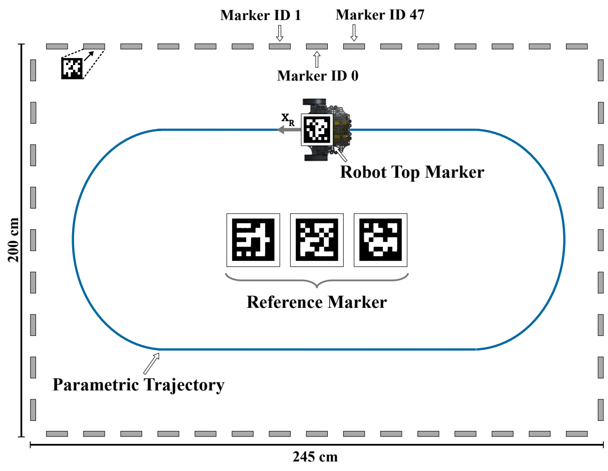

Figure 6: Layout diagram of markers and trajectory in the test zone.

To follow this trajectory, two PID controllers were implemented to control the robot's linear and angular velocity. The equation for the distance between two points mentioned in (9) was used to calculate the error in linear velocity. In this case, the point (x_a, y_a) represents the reference point, and (x_b, y_b) represents the current estimated position of the robot. Thus, the control signal for the linear velocity depends on the distance between the robot and the current point generated by the trajectory.

The error signal for the second controller is generated by considering the heading orientation of the robot with respect to the reference point given by the trajectory. As there are two points in the Cartesian plane, the angle between them is calculated using the arctangent. There are two ways to consider this error signal, one in a counterclockwise direction and the other in a clockwise direction.

6. Comparison of results

In the proposed comparison the robot autonomously navigate through 15 lap during 15 minutes with an average speed of $0.085\frac{m}{s}$. The trajectory obtained during each test was recorded using the vision system installed at the top of the testing area for further comparison. This process was carried out for both algorithms.

In Figure 7 at the top, the path following performed by the robot with WBO is graphically depicted. In this graph, it can be observed that during the initial minutes (approximately 4 minutes), the path followed by the robot closely matches the desired trajectory. However, after this time, the described path starts to deviate somewhat. This is because errors arising from finite encoder resolution, imperfections in the floor, wheel slippage, among other factors, begin to accumulate, leading to a divergence in the pose estimation.

In contrast, figure 7 at the bottom displays the results obtained in the same test but this time using the GGT algorithm.

From this graph, it is evident that the robot maintains a very similar path to the reference throughout the entire duration of the test. This is because the robot's estimated pose is not affected by previous estimation errors, as is the case with WBO. However, this algorithm also experiences moments where the estimation is not entirely accurate, especially when the robot encounters semi-circular sections of the trajectory. These errors are attributed to changes in lighting conditions and vibrations caused by floor irregularities. Nonetheless, the robot's pose does not diverge.

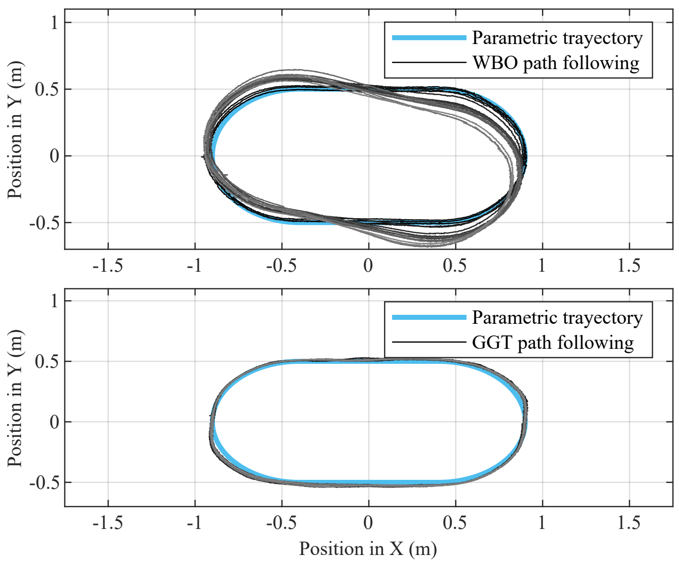

Figure 7: Path following test results from both algorithms. The robot's trajectory is shown in grayscale.

In figure 8 illustrates the magnitude of the following error. It can be observed that the GGT algorithm has a maximum pose estimation error of 0.28 m, while the odometry error grows over time. Furthermore, it can be observed that the following error in the GGT is bounded, which means corrective measures can be applied to improve pose estimation. In the case of WBO, this is not possible because the error grows over time.

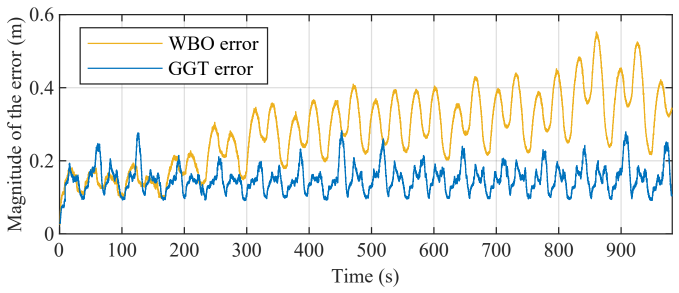

Figure 8: Magnitude of path following error of WBO and GGT.

7. Conclusions and future work

The results show that the GGT algorithm had a better pose estimation. This is because the estimation was based on the position and orientation of known markers in the testing area. On the other hand, in the WBO algorithm, the error due to the factors mentioned before, provokes the robot's pose estimation to diverge. However, the WBO algorithm required fewer components, as it only uses the encoder sensor in each motor and the microcontroller that receives the trajectory. In contrast, the GGT algorithm required a camera and a computer to process images.

As a future work, the fusion of the two algorithms is proposed in order to obtain the best of both cases. On one hand, a fast pose estimation (odometry) can be achieved, while on the other hand, an absolute estimation (triangulation) can be obtained.

References

Almeida, A. C., Neto, S. R., y Bianchi, R. A. (2018). Comparing vision-based monte-carlo localization methods. En *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pp. 437–442.

Campbell, S., O'Mahony, N., Carvalho, A., Krpalkova, L., Riordan, D., y Walsh, J. (2020). Where am i? localization techniques for mobile robots a review. En *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE.

Cavanini, L., Cimini, G., Ferracuti, F., Freddi, A., Ippoliti, G., Monteriù, A., y Verdini, F. (2017). A qr-code localization system for mobile robots: Application to smart wheelchairs. En *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–6.

Esan, O., Du, S., y Lodewyk, B. (2020). Review on autonomous indoor wheel mobile robot navigation systems. En *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6.

Esteves, J. S., Carvalho, A., y Couto, C. (2004). Generalized geometric triangulation algorithm for mobile robot absolute self-localization. En *2003 IEEE International Symposium on Industrial Electronics (Cat. No.03TH8692)*. IEEE.

Fauser, T., Bruder, S., y El-Osery, A. (2017). A comparison of inertial-based navigation algorithms for a low-cost indoor mobile robot. En *2017 12th International Conference on Computer Science and Education (ICCSE)*, pp. 101–106.

Housein, A. A., Xingyu, G., Li, W., y Huang, Y. (2022). Extended kalman filter sensor fusion in practice for mobile robot localization. *International Journal of Advanced Computer Science and Applications*, 13(2).

Kristalina, P., Pratiarso, A., Badriyah, T., y Putro, E. D. (2016). A wireless sensor networks localization using geometric triangulation scheme for object tracking in urban search and rescue application. En *2016 2nd International Conference on Science in Information Technology (ICSITech)*, pp. 254–259.

Liu, Y., Xiong, R., Wang, Y., Huang, H., Xie, X., Liu, X., y Zhang, G. (2016). Stereo visual-inertial odometry with multiple kalman filters ensemble. *IEEE Transactions on Industrial Electronics*, 63(10):6205–6216.

Lobo, A., Kadam, R., Shajahan, S., Malegam, K., Wagle, K., y Surve, S. (2014). Localization and tracking of indoor mobile robot with beacons and dead reckoning sensors. En *2014 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pp. 1–4.

Ohno, K., Tsubouchi, T., Shigematsu, B., y Yuta, S. (2004). Differential GPS and odometry-based outdoor navigation of a mobile robot. *Advanced Robotics*, 18(6):611–635.

Pierlot, V. y Van Droogenbroeck, M. (2014). A new three object triangulation algorithm for mobile robot positioning. *IEEE Transactions on Robotics*, 30(3):566–577.

Scaramuzza, D. y Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robotics &amp Automation Magazine*, 18(4):80–92.

Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents series. MIT Press, London, England, 2 edición.

Zhou, B., Tang, Z., Qian, K., Fang, F., y Ma, X. (2017). A LiDAR odometry for outdoor mobile robots using NDT based scan matching in GPS-denied environments. En *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE.