

Navigation Robot Simulator

Yousef Abulahbas¹, Abdulfattah Al ghariani², Fatma Dren³, Ayman Daw⁴ and Abdulkarim Alsaedawi⁵

^{1,2,3,4,5} Electrical & Electronics Engineering, College of Engineering Technology-Janzour.

Received 28 May 2024; revised 30 May 2024; accepted 01 June 2024

Abstract

This research aims to design a navigation system for a robot capable of determining its location, navigating towards a target, and studying the proportional differential integral controller. The robot simulates a vehicle with four wheels and a DC motor, controlled by LabVIEW software and a Lidar sensor. Feature locations measured visually, and a module for processing and analysis is developed. The project also outlines localization and SLAM basics, resulting in a concept for implementing Pose Graph SLAM in LabVIEW.

Keywords: Navigation, Simulation, SLAM, Lidar, LabVIEW, PID controller.

1. Introduction

Robotics research and development are focusing on mobile robots for transport, cleaning, and service applications, particularly in manufacturing and logistics. In 2013, around 21,000 service robots sold worldwide, with 1,300 being AGVs in logistics and manufacturing. Strong growth expected for service robots in both commercial and private sectors. Research demands for service robotics are more frequently in perception, navigation, and manipulation, often related to unsolved software problems. Autonomous navigation systems aim to guide mobile robots to preselected targets, while human-style map-based navigation requires constant pose and environment map creation.

2. Objective.

This paper investigates navigation components for mobile robots in LabVIEW, focusing on pose estimation and SLAM. It uses artificial landmarks with IDs for localization and a webcam for relative position measurement. The paper also examines image acquisition, processing, and analysis methods. The feasibility and implementation of tasks, landmark detection, self-localization, and SLAM will be examined using graphical programming in LabVIEW. The research is part of the World Skills Competition, allowing for direct implementation of mobile robots.

3. Delimitations, Limitations, Assumptions

The robot localization module focuses on landmark detection and pose estimation, considering environmental conditions according to the World Skills Competition. The module assumes the mobile robot moves in a planar plane, with all landmarks mounted at the same height. No quantitative requirements defined, such as real-time capability. The focus is on achieving high precision with low

process time, especially in computer-aided image processing, where effort and resources increase with precision.

4. Navigation.

Navigation is the art of steering a vehicle or airplane safely to its destination. It consists of three components: localization, mapping, and path planning. Classical navigation principles, dead reckoning, and the use of landmarks for localization are the foundations of modern robotic navigation. Mobile robotics has two types of navigation: reactive navigation and map-based planning. Reactive navigation used for simple applications like robotic vacuum cleaners, while map-based planning used for more robots that are sophisticated. Path planning requires the robot's position and the surrounding environment to identify by a map. Dead reckoning is a localization method without landmarks that estimates the robot's pose based on its speed, direction, and time of travel. It requires a vehicle model for accurate estimation of pose, which calculated using three variables: 2D coordinates, heading angle, odometer measurement, and random measurement noise.

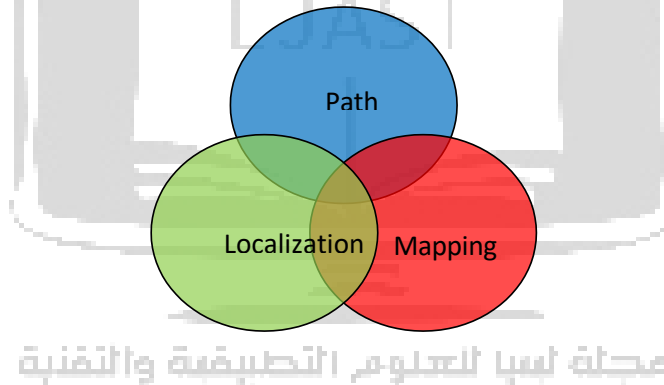


Fig. (1): Components of Navigation. SLAM equals
The intersection of Localization and Mapping

5. Pose Graph SLAM

In 1997, Lu and Milios proposed the first graph-based SLAM system, which refines the graph by optimizing equations and introducing a compound operator for mathematical modeling. The system consists of a front-end and a back-end connected by a graph. The front-end maps the robot's trajectory as a directed graph, with nodes representing individual poses and edges representing spatial constraints. The back-end adjusts node positions to minimize total error. As the robot moves, distance and orientation changes are measured using wheeled or visual odometer. The back-end balances deviation by pulling nodes closer to the true trajectory. The system illustrated in Figures 2–3, where circles and landmarks represent poses by stars.

This research discusses graph-based SLAM, also known as Pose Graph SLAM or full SLAM, a state-of-the-art technique for computing maps with speed and accuracy. It estimates all previously taken robot poses, making it more intensive in computation. Despite this, it is a smoothing approach.

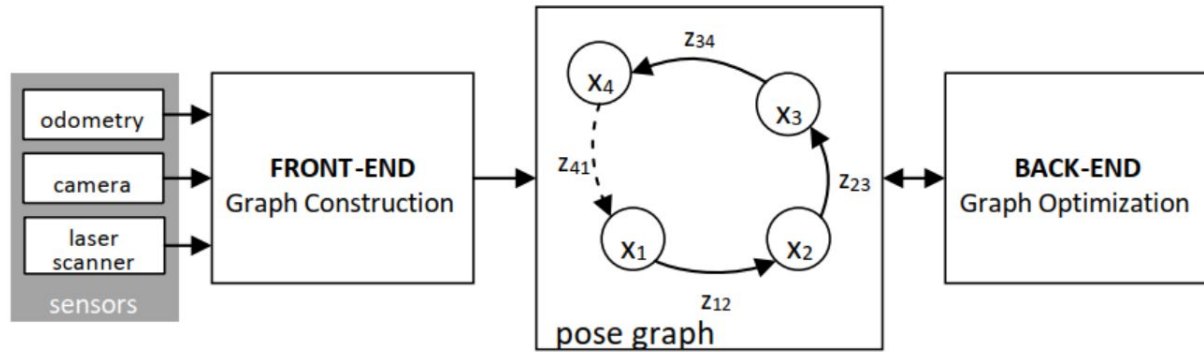


Fig. (2): Pose Graph SLAM system. The front-end creates nodes as the robot travels, and creates edges based on sensor data. The back-end adjusts the node positions to minimize total error.

6. Mathematical Description.

This research addresses the SLAM problem in two dimensions using graph optimization, following Stachniss et al. [7]. The graph's nodes are stored in a state vector, with node pose and landmark location described by position and orientation. The nodes of the graph are stored in the state vector $\mathbf{x} = (x_1, \dots, x_T)^T$ where x_i describes a pose or the location of a landmark. The edge of nodes carries measurement from observation and is denoted by $z_{ij} = (x_{ij}, y_{ij}, \theta_{ij})$. Figure (3) illustrates the relationship between nodes \mathbf{x}_i and \mathbf{x}_j , with the dashed edge representing the measurement at the pose i . The expected measurement \hat{z}_{ij} computed from the relative pose of the two nodes, representing the relative pose of \mathbf{x}_j seen in the frame of \mathbf{x}_i . The error function (x_i, x_j, z_{ij}) computes the spatial difference between the measured poses. The error function can be calculated, notations and rotate coordinates. Homogeneous coordinates. The calculation of the expected measurement of a landmark and the location error is less complex, with the expected observation calculated using a sensor. In addition, bearing sensor. The goal of graph-based SLAM is to optimize the constructed graph by using the method of least squares errors, which computes the best state vector \mathbf{x} where the squared errors are minimal.

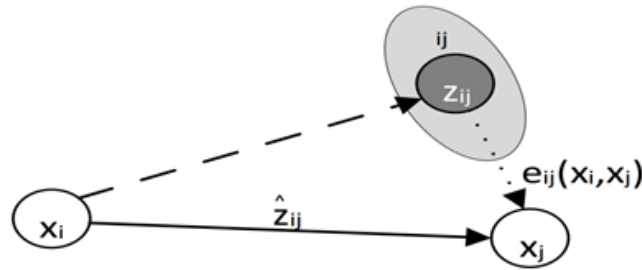


Fig. (3): Relation between the nodes x_i and x_j . The dashed edge originates from the real measurement.

7. Mathematical Theories.

This research provides an overview of the mathematical theories used in this research. The following part briefly explain the topics triangulation, least-squares, Hough transform, homogeneous coordinates, the Helmert transformation and the isoperimetric inequality.

7.1. Triangulation.

Triangulation is a geometry method used to calculate the position and orientation of a robot within a map through known landmarks. It differs from sectioning, which measures the bearing to the unknown position from each landmark [3]. There are three types of triangulation: backward cut, forward cut via triangle, and bow cut, which use distance values. The bow cut has three possible solutions, depending on the relationship between landmarks. The backward and forward cuts use only bearing angles to calculate the pose, but only bearing has the advantage of not requiring the distance to the landmark. The equation system can solved using the method of least squares. It is possible to consider more than two landmarks for calculating the robot's position, as used by GPS. However, sensor data can be noisy, and the optimal position is within the sectional area of the three circles.

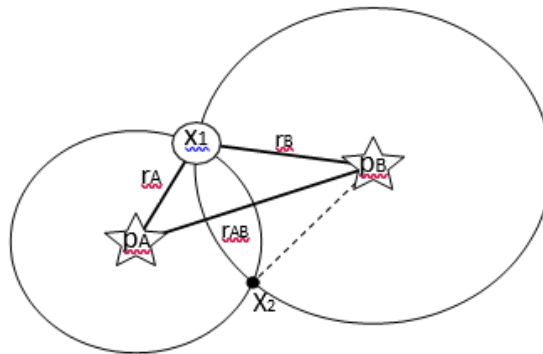


Fig. (4): Triangulation with bow cut.

The intersection point of the circles A and B represents the searched position

7.2. Method of Least Squares

The least squares method is a statistical technique used to find the best-fitting curve or line of best fit for a set of data points by reducing the sum of the squares of the offsets of the points from the curve. This process is known as regression analysis and is used in data fitting to find a curve with a minimal deviation from all measured data points. The method is widely used in evaluation and regression and is a standard approach for approximating sets of equations with more equations than the number of unknowns. There are two basic categories of least-squares problems: ordinary or linear least squares and nonlinear least squares. The least squares method states that the curve that best fits a given set of observations is a curve with a minimum sum of the squared residuals from the given data points. The formula for determining the equation of the line of the best fit is given by $Y = a + bX$. It is quite obvious that the fitting of curves for a particular data set is not always unique. Thus, it is required to find a curve with a minimal deviation from all the measured data points. This is known as the best-fitting curve and is found by using the least-squares method [8].

8. PID Controller.

The PID controller is the most commonly used algorithm in control system design, with over 80% of dynamic controllers coming from various types. The PID controller, also known as a three-term controller, involves computations involving three constant parameters: P, I, and D. P depends on the current error, I on past errors, and D predicts future errors as shown in figure (5). The weighted sum of these actions is used to adjust processes via control elements like valve positions or heating elements. Microcontrollers have gained strength due to their widespread use in industry, low cost, free software development, and extensive internet information. Ziegler-Nichols Rules for Tuning PID controllers were implemented using MATLAB software to control a lab-volt temperature process based on the reaction curve method as shown in table [1].

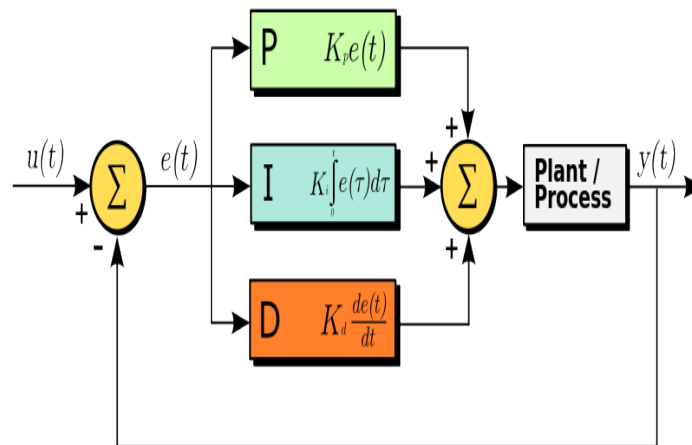
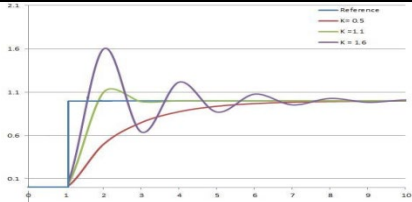
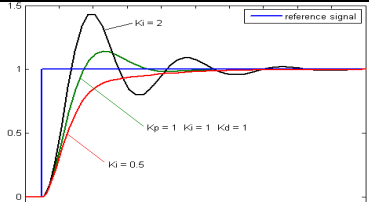
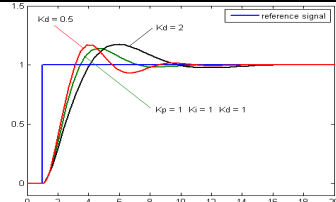


Fig. (5): PID controller design

Table [1] shows the type of controllers:

Proportional Term $P_{out} = K_p e(t)$	Integral Term $I_{out} = K_i \int_0^t e(t) dt$	Derivative Term $D_{out} = K_d \frac{d}{dt} e(t)$
 <p>The effect of add K_p (K_i, and K_d) held constant.</p>	 <p>The effect of add K_i (K_p, and K_d) held constant.</p>	 <p>The effect of add K_d (K_p, and K_i) held constant.</p>

This text focuses on tuning PID controllers, focusing on typical system responses and the effects of changing PID gains. Table [2] outlines trade-offs between individual changes in PID coefficients, emphasizing the need for joint tuning for optimal performance as each gain affects the other.

Table [2]: Effect of increasing parameter independently

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
K_p	Decrease	Increase	Small Change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor Change	Decrease	Decrease	No Effect	Improve if K_d small

8.1. Tuning Methods.

Tuning a PID loop involves developing a process model and choosing parameters like P, I, and D based on dynamic model parameters. Manual tuning methods can be inefficient, especially for loops with long response times. The choice of method depends on whether the loop can take offline and the system's response time. If the system can take offline, the best tuning method involves a step change in input, measuring output, and determining control parameters. Performance requirement defined before tuning as shown in figure (6).

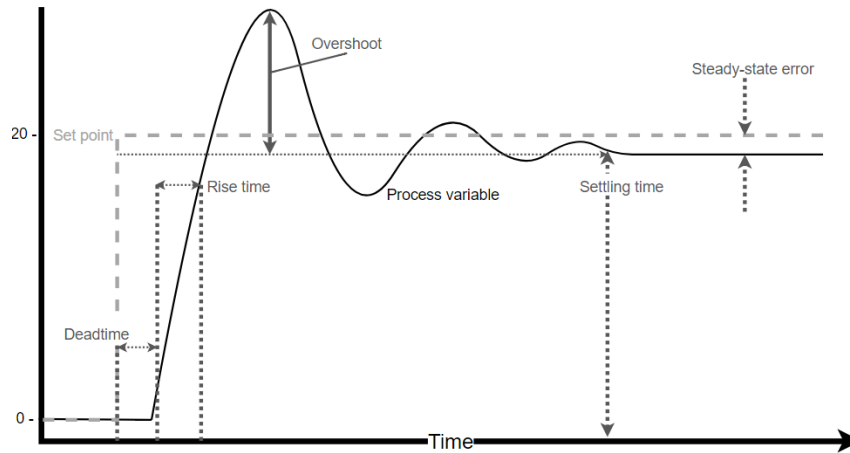


Fig. (6): PID-controlled closed loop response to a change in set point

8.2. Standard feedback systems with disturbances

Usually, in addition to the input u the system is also influenced by an external disturbance v as illustrated in Figure (7).

Hence, the plant may be influenced both from the control input u and an external disturbance v . The plant illustrated with the dashed box in Figure (7).

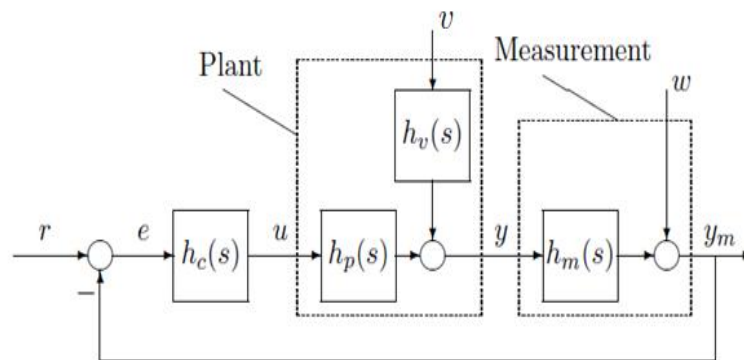


Fig. (7): Standard feedback system with disturbance v at the output.

A more general situation is as illustrated in Figure (7) where the measurement system is illustrated in the dashed box and where the actual measurement y_m is related to the desired output y as $y_m = h_m(s)y + w$ where w is measurement noise, and $h_m(s)$ a model for the measurement system. Usually, as described above we assume $h_m(s) = 1$. The measurement noise w is usually high frequent noise but may also represent drifts etc.

8.3. PID tuning rules.

This section presents a simple and direct method for PID controller tuning and design, based on an approximate model of 1st or 2nd order with a time delay or inverse response. The method is practical and robust, providing good results. The starting point is the approximate model, which is stable models. If the process approximated with a 2nd-order model with a time delay, a PID controller is the result, while a 1st-order model with a time delay results in a PI controller. The method can be used if the second time constant, T_2 , is zero.

If the starting point is a more complicated higher-order model or a higher-order model from data, a 1st or 2nd-order model can be constructed using model reduction techniques or system identification methods. In these low-order models, the parameter T represents the effective time delay or inverse response time, and the dominant time constant is $T_1 \geq T \geq 0$. Many high-r order models with time delay can approximate by a 1st or 2-d order model with time delay. 2nd order model with time delay given by the following transfer function:

$$h_p(s) = K \frac{e^{-ts}}{(1+T_1s)(1+T_2s)} \dots\dots\dots (1)$$

8.4. Ziegler–Nichols rules for tuning PID controllers.

Over half of industrial controllers today use PID control schemes or modified PIDs, which are mainly hydraulic, pneumatic, electronic, or electrical. Many of these controllers are transformed into digital form using microprocessors. A PID controller can be expressed as a frequency domain equation, where $e(t)$ is the error signal, $u(t)$ is the control input, K_p is the proportional gain, T_i is the integral time constant, and T_d is the derivative time constant. The Ziegler and Nichols method can be used to determine the values of these parameters or tuning parameters.

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t) \partial t + K_p T_d \frac{\partial e(t)}{\partial t} \dots\dots\dots (2)$$

$$U(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) E(s) \dots\dots\dots (3)$$

8.5. Ziegler–Nichols open loop method.

The transfer function in equation (4) can be used to define many processes in an open loop system. The coefficients K , τ_d , and τ are obtained from the system's response to a step input in figure (8). The system is stabilized in $y(t) = y(0)$ for $u(t) = u(0)$, and the output response is recorded to stabilize at the new operating point. The parameters K , τ_d , d , and τ can be obtained from equation (5). The S-shaped response curve can be generated experimentally or from a dynamic simulation of the plant. The S-

shaped curve is characterized by two constants: delay time τ_d and time constant τ . The Ziegler and Nichols open-loop method suggests setting the values according to the formula.

$$\frac{Y(s)}{U(s)} = \frac{ke^{-\tau_d s}}{\tau s + 1} \dots\dots\dots (4)$$

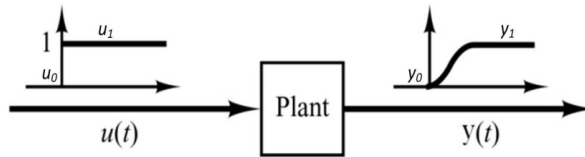


Fig. (8): Unit-step response of a plant.

The parameters K , τ_d and τ can be obtained from the response shown in Figure (9).

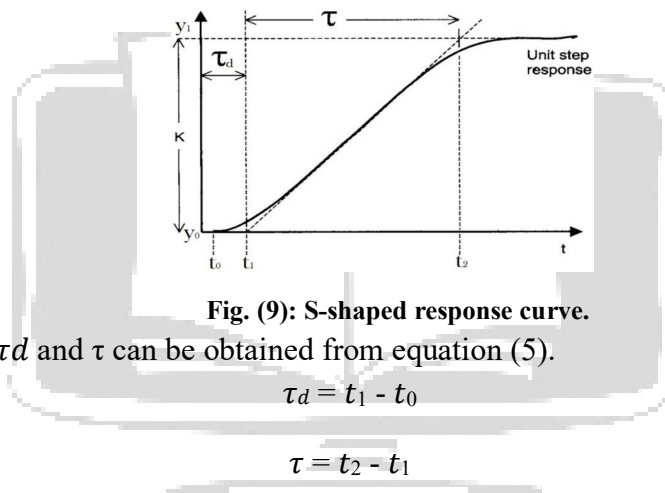


Fig. (9): S-shaped response curve.

Where, the values of K , τ_d and τ can be obtained from equation (5).

$$\tau_d = t_1 - t_0$$

$$\tau = t_2 - t_1$$

$$K = \frac{y_1 - y_0}{u_1 - u_0} \dots\dots\dots (5)$$

This method applies if the response to a step input exhibits an S-shaped curve. Such step-response curves may be generated experimentally or from a dynamic simulation of the plant.

The Ziegler and Nichols open loop method is used to determine the values of K , τ_d , and τ in a step response curve. These curves can be generated experimentally or from dynamic simulations. The S-shaped curve is characterized by two constants: delay time τ_d and time constant τ . The values are set according to the formula in Table [3]. The method is applicable when the response to a step input exhibits an S-shaped curve.

Table [3]: Ziegler–Nichols tuning rule based on step response.

Type of Controller	K_p	T_i	T_d
P	$\frac{T_1}{-K\tau_d}$	∞	0

PI	$\frac{0.9\tau}{K\tau_d}$	$3.3\tau_d$	0
PID	$\frac{0.2\tau}{K\tau_d}$	$2\tau_d$	$0.5\tau_d$

According Ziegler-Nichols formula shown in Table [3], the relationship of these parameters with coefficients controller are shown in equation (7).

$$K_p = \frac{1.2\tau}{K\tau_d} \quad \& \quad T_i = 2\tau_d \quad \& \quad T_d = 0.5\tau_d \quad \dots\dots\dots (6)$$

Where K_p is the proportional gain, K_i is the integral gain ($K_i = \frac{K_p}{T_i}$), K_d is the derivative gain ($K_d = K_p \times T_d$), T_i is integral time constant and T_d is the derivative time constant.

Notice that the PID controller tuned by this method of Ziegler–Nichols rules gives equation (7) by substitute equation (6) in equation (3) results:

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right)$$

$$G_c(s) = \frac{1.2\tau}{K\tau_d} \left(1 + \frac{1}{2\tau_d s} + 0.5\tau_d s \right)$$

$$G_c(s) = \left(\frac{1.2\tau}{K\tau_d} + \frac{1.2\tau}{2K\tau_d^2 s} + \frac{0.6\tau\tau_d s}{K\tau_d} \right) \quad \dots\dots\dots (7)$$

8.6. Auto tuning method for PID controller

Automatic PID tuning is a process that adjusts controller gains based on a plant model or plant data. It can be done using Simulink Control Design TM or a PID auto-tuning algorithm for real-time tuning against a physical plant. Model-based PID controller tuning allows users to tune controller gains based on a Simulink model of the control system. Tools like PID Tuner can be used to interactively tune PID gains while examining relevant system responses to validate performance. Real-time PID auto tuning allows users to deploy an embedded automatic tuning algorithm as a standalone application for model-free tuning against a physical plant. The Simulink Control Design PID auto tuning algorithm estimates plant frequency response and computes PID controller gains to balance robustness and performance as in figure (10).

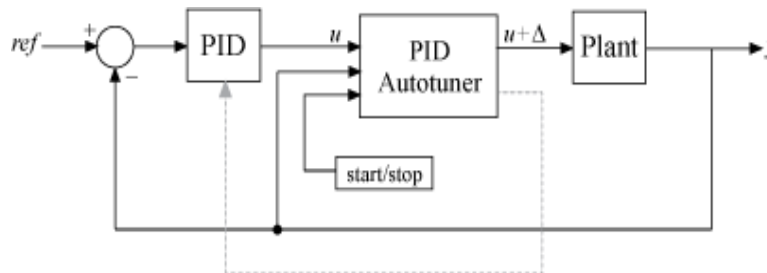


Fig (10): PID auto tuner block fits into a control system.

9. Landmark Detection Module

Landmark Detection is a method in the LabVIEW library that determines the position and ID of landmarks in an image of the workspace. It involves three levels: acquisition of all objects, processing of these objects, and reconstruction by identified vertices. The library offers various ready-to-use modules for camera applications, including custom modules like the Hough Transform. Express VI Vision Acquisition is used to obtain images from the webcam, with settings adjusted by a wizard application.

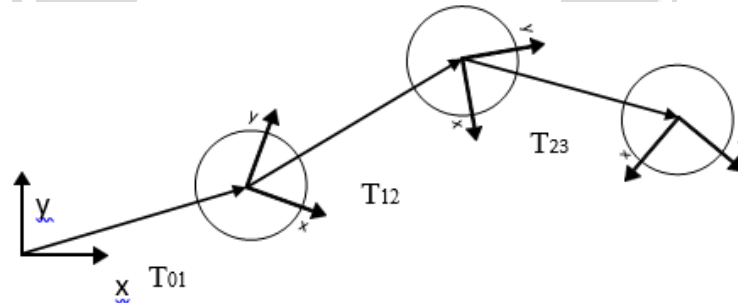


Fig. (11): Sequence of consecutive poses of a vehicle.

LabVIEW also provides a discrete implementation of image acquisition through multiple VIs from the submenu NI-IMAQ. The Canny Edge Detection extracts edges from gray scale images using parameters determined euphorically. The segmented edge image is processed with the self-developed VI Square Detector, converting the segmented edge image into a vector of objects. The Hough Transform examines selected objects by transforming them into the Hough space, which represents the lateral surface of a cylinder. The performance of the Hough Transform depends on the number of processed pixels and the number of iterations. SubVI clustering sorts selected elements into groups corresponding to the distance metric, with the maximum distance set to 30. The VI Landmark Transformation extracts the determined area of the image and reconstructs the landmark. The SubVI Grid Transformation calculates a grid from the four vertices, and the determined 9×9 array represents the elements of the detected landmark. The SubVI Landmark Interpretation uses several characteristics of the landmarks to decrease the uncertainty of the output of the Landmark Detection.

10. Pose Estimation and Simulation

The VI Pose Graph SLAM is a simulation structure that aims to construct a graph using data from visual odometer and detected landmarks. It is implemented with two Sub VIs: front-end and back-end. The front end consists of several Sub VIs, including the Observation Handler, which manages an observation database of all observed positions of landmarks, and the Scan-Matching module, which estimates the new pose and adds new nodes (poses and landmarks) to the graph. The back-end of the Pose Graph SLAM is not implemented due to the time-consuming and effort-intensive optimization process. However, the function of the front end can be demonstrated through a simulation. The simulation consists of three parts, plus several modules for the illustration of the graph: Simulate Vehicle Pose, Simulate Projection, and Simulate observation. The vehicle model is used to describe the movement of a wheeled vehicle, which is determined by a sequence of changes in distance and heading. The trajectory of the vehicle is generated from this trajectory, which can be used for both image for landmark detection (simulate projection) and a set of landmark positions (simulate OBs). In addition, it is possible to generate a random or a predefined trajectory, e.g. a circle. Figure (11) shows the consecutive poses described by the vectors ($T_{01}, T_{12}, T_{23} \dots T_{mn}$).

In summary, the VI Pose Graph SLAM is a comprehensive approach for estimating poses and landmarks using visual odometry and landmark detection.

11. Software requirement (LabVIEW program)

LabVIEW, a graphically based programming language developed by National Instruments as in figure (12), is a powerful tool for increasing efficiency and productivity in various applications. It is ideal for Test and Measurement (T&M), automation, instrument control, data acquisition, and data analysis. Engineers can create user-defined solutions using LabVIEW, which offers a great alternative to proprietary, fixed-functionality traditional instruments. Virtual instrumentation capitalizes on the increasing performance of personal computers, allowing engineers to downsize automated test equipment (ATE) while experiencing up to a 10-fold increase in productivity gains at a fraction of the cost of traditional instrument solutions.

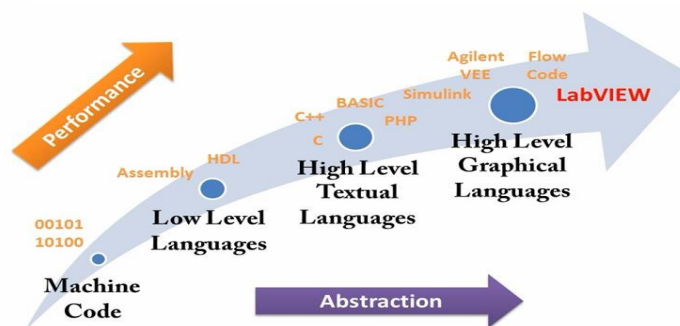


Fig. (12): program languages timeline.

11.1. Virtual Instrument

A Virtual Instrument (VI) is a LabVIEW programming element with a front panel, block diagram, and icon representing the program. It displays controls and indicators for the user; handles function inputs and outputs, and can be used as a user interface or subroutine in large-scale applications.

11.2. The front panel

Figure (13) illustrates the front panel of a LabVIEW VI. It contains a knob for selecting the number of measurements per average, a control for selecting the measurement type, a digital indicator to display the output value, and a stop button. An elaborate front panel can be created without much effort to serve as the user interface for an application [6].

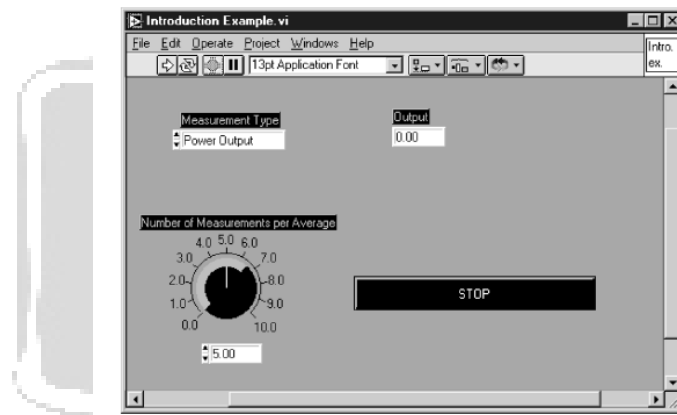


Fig. (13): the front panel.

11.3. The block diagram for simulation

LabVIEW is a graphical simulation tool that consists of a block diagram, a case structure as shown in figure (14), and a virtual instrument (VI). It is compiled by LabVIEW's execution engine, similar to Java. When a VI is changed, it creates a wire table, identifying elements in the block diagram that require inputs. If all wire tables are created, the VIs can be executed. If not, a broken arrow is displayed. LabVIEW runs on multiple subsystems and is not an interpreted language. It is a powerful tool for simulation and analysis.

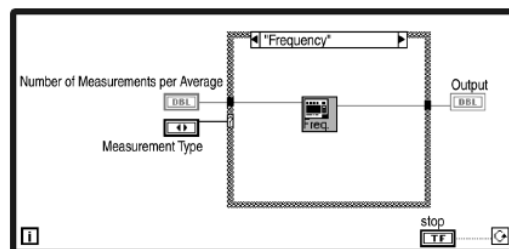


Fig (14): the bock diagram for the LabVIEW simulation.

12. Hardware requirement:

Figure (15) shows the Hardware requirement if anyone want to applied in real time.

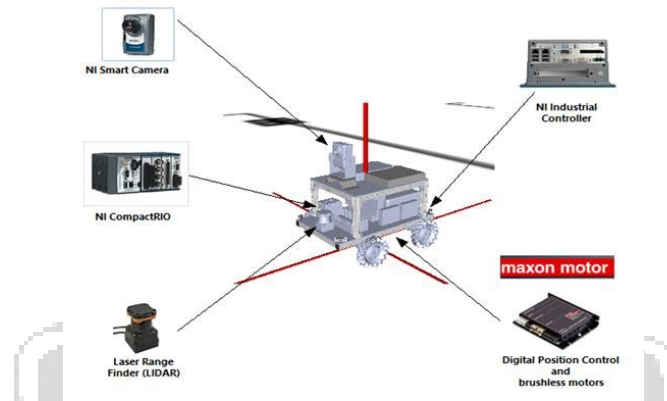


Fig. (15): Hardware requirement.

13. DC Motor Mathematical Model:

We will calculate transfer function of DC motor in figure (16).

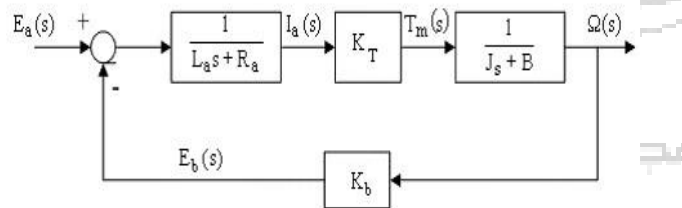


Fig. (16): Block diagram of PI controller.

A PID controller is a feedback mechanism used in industrial systems to minimize errors by adjusting processes through manipulated variables. It can be referred to as PD, PI, I, or P controller. PID tuning involves selecting the ideal set of numerical values for P, I, and D, which can be achieved using various methods. The mathematics behind PID control is complex.

- Process reaction curve.
- Tyreus and Luyben.
- Ziegler Nichols method.
- Cohen and Coon.
- Model of the DC Motor represented in the Lab VIEW Math-Script.
- Node.
- Model of the DC Motor control using PID in Block diagram window.

- Simulation results.
- The figure (17) shows the block diagram for DC motor which is designed by Simulink Math lab.

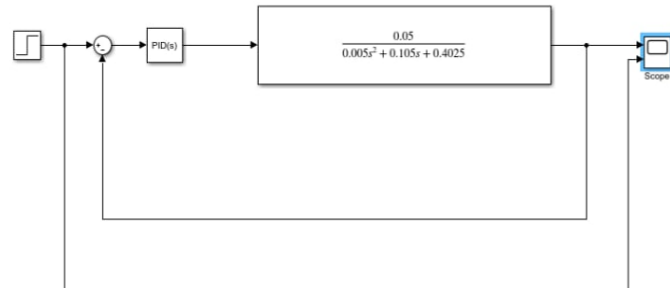


Fig. (17): Transfer Function of DC Motor (Simulink).

14. Simulation results:

14.1.First: PID tuning with ziegler nichols method .

Plot the transfer function response without PID controller to get the values of (K,t2,td) from Figure(18).

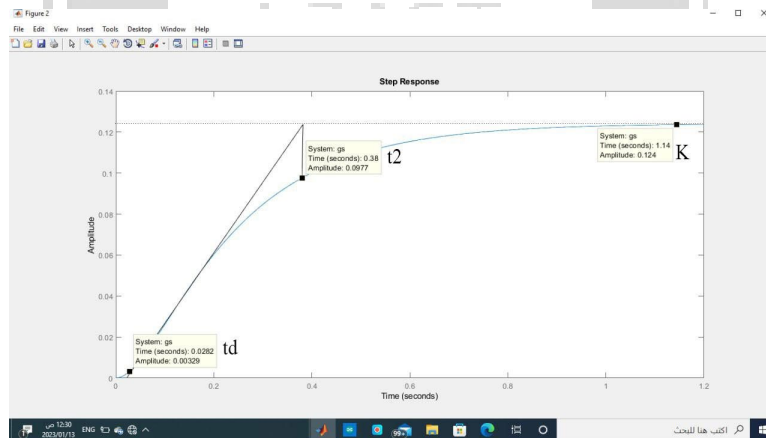


Fig.(18): Response of transfer function without PID.

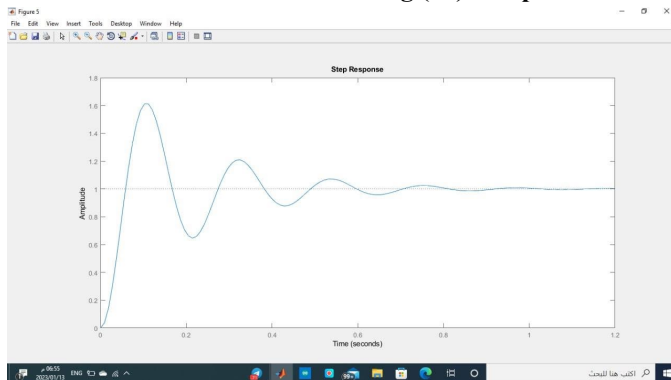


Fig.(19): Response of transfer function with PI controller using ziegler nichols.

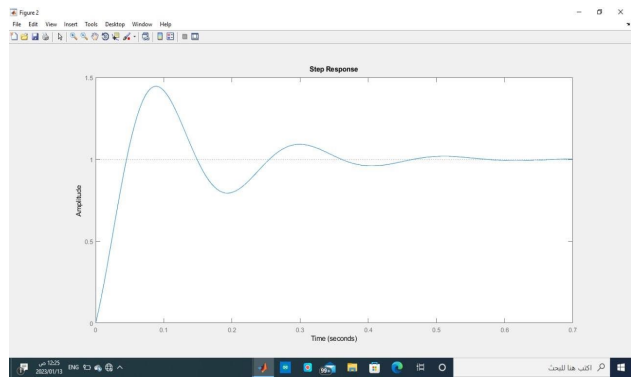


Fig.(20): Response of transfer function with PID controller using ziegler nichols.

Notes: When using PID controller the response of the system is better than use PI controller, but even with that improve using ziegler nichols method the system still need some develop to get rid of the error .

14.2. Second: the PID auto-tuning method for math lab Simulink.

The PID auto-tuner blocks work by performing a frequency-response estimation experiment. The blocks inject test signals into your plant and tune PID gains based on an estimated frequency response. Response of the DC motor system.

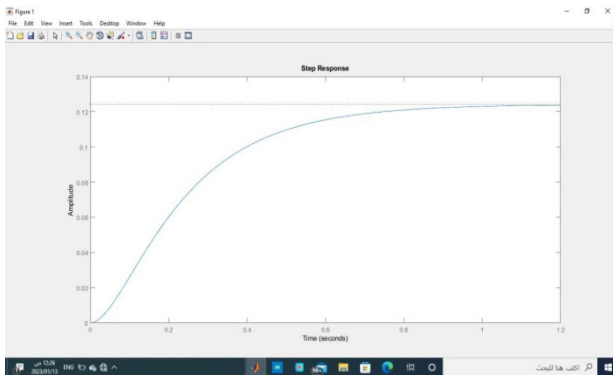


Fig.(21): Response of dc motor system without PID controller.

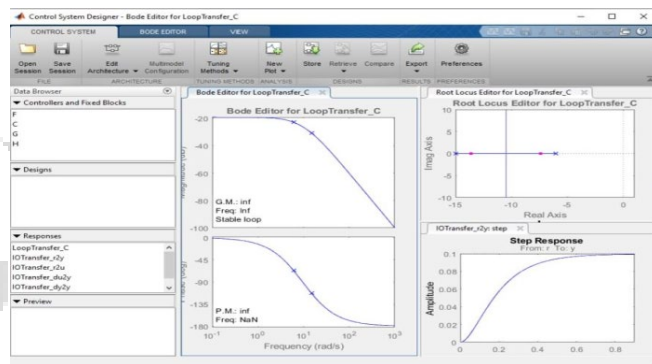


Fig.(22): Bode plot for Response of dc motor system without PID controller.

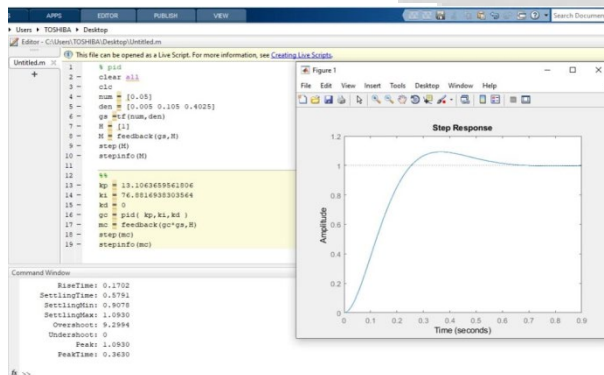


Fig. (23): Simulation Result for $K_p=13.10$,
 $K_i=76.88$, $K_d=0$

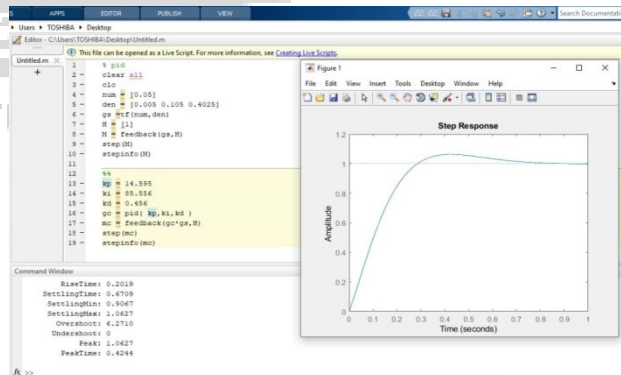


Fig. (24): Simulation Result for $K_p=14.595$,
 $K_i=85.556$, $K_d=0.456$

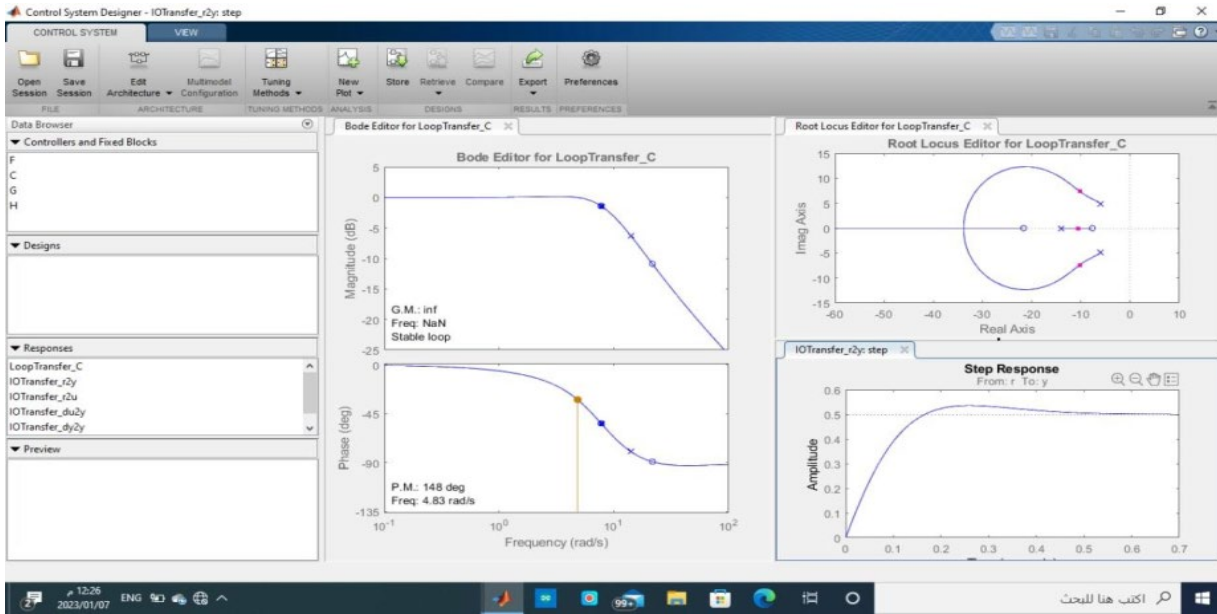


Fig. (25): Bode plot for Response of Auto tuning PID controller.

Table [4]: Result

Methods \Rightarrow	AUTO TUNING		Ziegler Nichols		Error percent in Ziegler (%)	
Response specification	PI	PID	PI	PID	PI	PID
Rise Time(s)	0.1702	0.2019	0.0393	0.0351	76.9%	82.6%
Peak Time(s)	0.3630	0.4244	0.1117	0.0890	69.2%	79%
Settling Time(s)	0.5791	0.6709	0.7752	0.4433	25.2%	33.9%
Overshoot	9.2994	6.2710	61.0886	44.5997	51.8%	38.32%
Peak	1.0930	1.0627	1.6109	1.4460	32%	26.5%

The simulation involved a wheel mobile robot on uneven stochastic ground with a pit and mound. The robot designed to avoid lateral movement and turn in place at the inflection point of the planned path. As it moved forward, LiDAR data detected, and the robot's path was constantly changing. The robot

rounded the mound, passed through the gap, and reached the target point. The movement trajectory recorded and displayed on a screen as shown in Figure (26 - 27) and the results table [4].

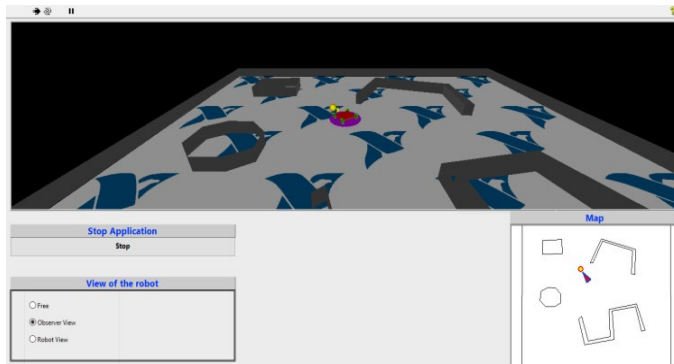


Fig.(26): Robot simulation

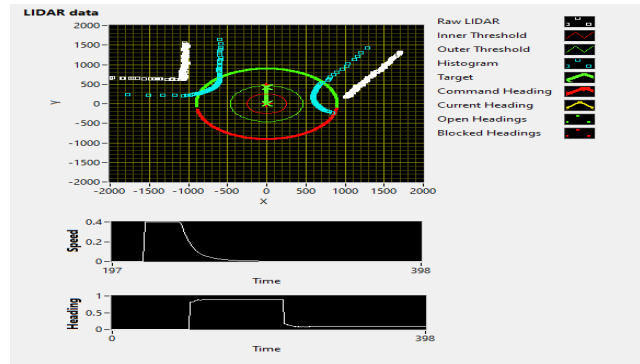


Fig.(27):Robot response

15. Data flow:

The navigation system, illustrated in Figure (28), consists of object detection, tracking, and path planning. It moves towards an object if detected and plans a new path if not. Decision-making relies on image processing. The system's main components explained using flowcharts.

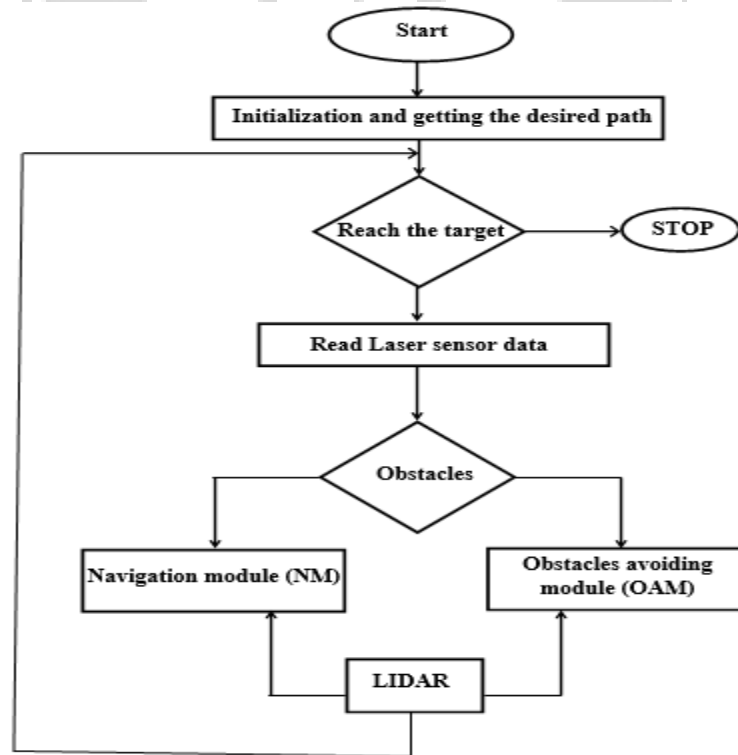


Fig. (28): Data flow.

16. Conclusion

This investigates the development and implementation of modules for landmark detection and pose estimation in LabVIEW, aiming to investigate the feasibility of navigation components for a mobile robot system with graphical programming. The robotic platform features a webcam and an omnidirectional drive. The research explores the theories of navigation components, describes the concepts of landmark detection, and pose estimation modules. Landmark detection focuses on recognizing square-shaped landmarks and calculating their position and unique ID. The process involves processing an image of the workspace with a webcam, segmenting it to achieve individual boundaries, selecting credible subareas, and analyzing the selected lines and pixels. The concept of self-localization and pose estimation proposes the method of Scan Match for map-based localization and obtaining visual odometry data. The study also evaluates the Pose Graph SLAM as a suitable solution according to the defined specifications. The investigation implemented the concept in two steps, with the components of landmark detection realized in several self-developed subVIs. The Ziegler-Nichols control method used for automatic tuning to improve system response. The LabVIEW program used for a limited time, and to continue using the simulator, paid versions are required.

17. References:

- [1] B. Graf and M. Hägele, "Automatisierung und Robotik-Systeme," Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA), Karlsruhe, 2016.
- [2] J. Guldner, "Autonomous Navigation System for a Mobile Robot or Manipulator". Germany / Munich Patent 5,758,298, 26 05 1998.
- [3] P. Croke, Robotics, Vision and Control, Brisbane, Queensland: Springer International Publishing AG, 2017.
- [4] SRI International's, "Shakey Images, Artificial Intelligence Center," [Online]. Available: <http://www.ai.sri.com/shakey/images.php>. [Accessed 10 7 2022].
- [5] L. Libuda, Wissensbasierte Szenenanalyse für Navigationsaufgaben mobiler Roboter in Innenräumen, Aachen: Rheinisch-Westfälische Technische Hochschule Aachen, 2007.
- [6] N. Manske, Kamerabasierte Präzisionsnavigation mobiler Systeme im Indoor-Bereich, Hamburg: Hamburg University of Applied Sciences, 2008.
- [7] S. Grünwedel, Robuste Lokalisierung von autonomen Fahrzeugen mittels Landmarken, Chemnitz, 2008.
- [8] C. Sprunk, Highly Accurate Mobile Robot Navigation, Freiburg: University of Freiburg, 2015. [9] G. Gallegos and P. Rives, "Indoor SLAM Based on Composite Sensor Mixing Laser Scans and Omnidirectional Images," 2010 IEEE International Conference on Robotics and Automation, pp. 3519-3524, May 2010.
- [10] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss and W. Burgard, "Autonomous Robot Navigation in Highly Populated Pedestrian Zones," Journal of Field Robotics, pp. 565- 589, 2015.
- [11] DC motor tf <https://www.collimator.ai/tutorials/dc-motor-speed-controller-design>.
- [12] <https://instrumentationtools.com/ziegler-nichols-open-loop-method/>