Line following robots on factory floors: Significance and Simulation study using CoppeliaSim

Saharsh Oswal* and Saravanakumar D

1School of Mechanical Engineering, Vellore Institute of Technology, Chennai, India

E-mail:saharshrajoswal@gmail.com

Abstract. Line following robots has been around for many years now. Research and development of these robots have been keenly studied and examined for many industrial applications. Line following robots can provide maximum effectiveness to in-house transportation of jobs inside a factory, which is traditionally done by forklifts and different types of cranes. These traditional methods are high maintenance and extremely unsafe for the employees. This paper presents the design, assembly and dynamic simulation of a line following robot integrated with a proximity sensor for collision avoidance and vision sensors for line tracking. The model is designed using Autodesk Inventor 2018 and the assembly and simulation is carried out using CoppeliaSim software. This simulation study intends to provide significance of the implementation of such robots on a factory floor, which can cover a path of distance 10 meters in approximately 8 seconds.

1. Introduction

Logistics and supply chain management has become a crucial part of the entire manufacturing operational timeline. But, we often tend to forget the logistics dealt within the factory where products are manufactured, tested and packaged. Traditionally, jib cranes, conveyor belts, overhead cranes and forklifts are used to transfer material or products between different production and machining operations. Most of the aforementioned methods are regulated and inspected manually. Currently, with Industry 4.0 at its peak, autonomous technologies are used in every aspect of product development, logistics and management. In modern manufacturing, the ultimate aim is to reduce the cycle time and enable the possibility of manufacturing more complex and customizable products [1]. Along with these parameters, the operational efficiency has to not be compromised so as to ensure minimal financial loss and ill-usage of raw material & energy.

Most defects in manufacturing are caused due to processing errors carried out by individuals who fail to follow operating schedules and instructions, which generally happen when companies fail to fix an established set of rules or “kaizens”. Most manufacturing processes consist of more than one operations in the assembly line which makes the transfer of unfinished jobs difficult, complex and even unsafe due to high temperatures and heavy payloads. Owing to the concerns and risks arising due to manual errors and delay between two or more operations, the methodology of “lights-out manufacturing” has been entrenched.
Although now, most production processes are being carried out autonomously using CNC operated machines, SCADA controlled systems using G-codes and PLC programmed panels. The performance gets compromised when one specific job has to be transferred between two or more manufacturing and machining processes. This in-house multi-variable logistic problem has given a kick-start to autonomous mobile robots which help in picking and delivering jobs on factory floors. Conventionally, manually operated cranes (jib or overhead), conveyors or forklifts or even combination of these machines are used for the transit of jobs on a factory floor. The biggest concern while embedding automation in manufacturing is the risk of quality control and the pressure on the equipment. Not only do these run operational risks, but also leads to complicated labor management, schedule delays and increased accidents which also threatens safety issues of the employees.

The basic requisites in the motion planning of mobile robots are path plans, localization and collision avoidance [2]. The path planning tends to get very complicated in a dynamic or static environments such as markets or shopping malls, and also automated factories and warehouses, where autonomous mobile robots can be conveniently used for in-house logistics. The navigation and localization of the auto-vehicle have to be pre-determined, regulated and tested to ensure minimal errors. We are well aware that some robots can perform tasks such as milling, welding and turning with higher precision and speed as compared to humans. Also, robots can withstand high temperatures and survive in conditions where humans are unable to. Similarly, mobile robots have proven to be far more efficient than human-operated cranes and forklifts so as provide lesser time-delays and high safety measures.

Line following robots have been a part of many industrial and commercial applications for a long time. The idea of using this kind of an approach for motional dynamics is due to the simple construction and programming. These robots can be configured conveniently for any tracks inside a factory, commercial space and even hospitals [3]. The usage of line following robots on factory floors is an integral part of automating the material handling and unfinished goods. The hardware of the robot consists of the robot body, motors, sensors and control boards while the software includes the program code, controller design, sensor parameters and method design [4].

The low-cost wheeled mobile robot (as shown in figure 1) which is proposed in this paper serves as a good example on which manufacturers can understand the significance of autonomous path following robots in factories.

![Figure 1. Mobile robot for factory floors](image)

Line following robots can be used in various environments such as hospitals, commercial spaces and also, factory floors. These robots can be conveniently used to substitute manual workforce involving the transportation of goods in factories and warehouses. Optimizing manufacturing process has been the core requirement with the increase in population, demand and customizable priorities [5].
The objective of this paper is to provide a comprehensive as well as a simulation study on a state of the art mobile robot which tracks and follows a pre-defined path so as to provide transit amidst the walls and machinery on the factory floor. It intends to give aid in the better perception of the path planning robot technologies and also guide manufacturers to combine and optimize techniques for better line following executions in both single and multi-robot environments.

2. Limitations

Logistics Implementation of any techniques in a conventional space takes time to develop and get accustomed to. Even though implementing line following robots in factories is beneficial in safety, productivity and finances, there are several challenges to be faced.

- Skill gap: The employees at most factories are used to the forklift and crane system of transportation of “work-in-progress’ items. Implementing such robots can cause a gap in pre-existing skills, which might take some time to settle in.
- Factory plan adjustments: The Manufacturing Resource Plan (MRP) has to be altered integrating the line following action of the robot. Besides, the layout has to be changed throughout the enterprise so as to promote data standardization and interoperability.
- Disruptive technology: Most blue collars have found comfort in using forklifts and cranes for in-house logistics, changing the mindset and familiarizing the workers with this technology can be a difficult task. Not only humans but even other machines might also have to be customized in accordance with the line following tracks.

3. Robot Design

The robot used for the simulation is a differential drive robot, which is basically a mobile robot with wheels placed on both sides of the main base. The movement of the entire robot body is based on these two separately placed wheels. In addition to these wheels, a castor wheel is added to the center-back of the robot body for static and dynamic balance, which orients itself in accordance with the other two wheels [6]. The robot steers itself autonomously with the help of necessary sensors and motors. The parts are modelled individually in Autodesk Inventor 2018.

The main base of the robot is a tray-type design so as to provide better assistance to carry unfinished jobs from one machine to another. Many items in a factory are either heated to high temperatures, or too fragile or maybe even radioactive. Hence this robot offers assistance and control over most logistical features on the factory floor, which cannot be carried out by humans.

The dimensions of this robot are assigned in accordance with the average payload for most components manufactured (as shown in figure 2 and 3). The differential wheels are set as 200mm in diameter while the castor wheel is set as 80mm in diameter. The robot is designed with a base of 1000mm by 750mm. The total height of the model base is 600mm, which gives it an appropriate structure to travel over a line in a dynamic environment.
The dynamic assembly of this robot is done in CoppeliaSim Edu as it allows us to add the necessary joints, sensors and the necessary scaling factors (as shown in figure 4). The same software is also used for the simulation of the line following action in a factory environment, to be demonstrated later in the paper. CoppeliaSim is a simulator software specifically for robotics and automated systems, in which, every object of the system can be individually controlled via an embedded script written in many programming languages. The script for this robot is written in Lua programming language.

CoppeliaSim allows importing the ‘.stl’ files to its scene and orient all parts as per the robot design. All parts are converted to convex shapes and made invisible using the layer properties. The entire robot is set as collidable, measurable, detectable and renderable. Revolute joints are added to the side wheels, castor holder and the wheel, with the motors, enabled only for the differential wheels. A cone-type proximity sensor and three orthographic vision sensors are added to the front and base of the robot respectively. A non-threaded child script (program) is linked to the model base of the robot body.

After the assembly is completed, the objects are arranged in a particular hierarchy for the simulation (as shown in figure 5). Only the imported parts are visible and the convex shapes and sensors are set to invisible using the layer option, as they are used for only dynamic simulation.
Figure 5. Hierarchy of robot objects. The sensors and motors have been set to invisible.

The mass of the entire robot body is set as 218.3kgs which is optimum for a line following mobile robot carrying a heavy payload. The corresponding principle moments of inertia in the x, y and z-direction are formulated as shown in table 1. Lesser the moment of inertia, lesser the force required for a body to perform the non-translational motion, which works in favor of a line following mobile robot.

Table 1. Principle moments of inertia of the robot

| Direction | Moment of Inertia (kg m^2) |
|-----------|---------------------------|
| I_{xx}    | 19.288                    |
| I_{yy}    | 17.217                    |
| I_{zz}    | 9.439                     |

4. Program Instructions

CoppeliaSim allows the dynamic assembly of the robot integrating features such as sensors, joints, motors and even embedded scripts so as to link the code to the robot. Other robots simulating software such as Webots, Gazebo and RobotStudio do not allow engineers for remote monitoring or safety double-checking. CoppeliaSim supports a number of programming languages such as C/C++, Python, Java, Lua, Matlab or Urbi and also allow integrating robots designed by other third-party companies [7]. This simulator not only facilitates the design of a factory environment and the mobile robot but also permits the program to execute in accordance to the robot features.

The code for the robot is written in Lua programming language. Lua is a powerful, fast, lightweight language designed to support procedural programming and it is embedded in CoppeliaSim software. A non-threaded child script is used here which follows a precise execution order as specified by the user. The code consists of three main parts:

- Initialization: Here the objects are retrieved individually using ‘sim.getObjectHandle’ and values are assigned to certain variables such as the speed of the robot. The User Interface is also designed here using ‘simUI.create’, which in this case consisted of options to stop the robot, resume the robot, decrease and also increase the speed of the robot [8].
- Actuation: This part of the code deals with the execution and implementation of various sensors and motors initialized before. The varying speeds for the wheels are computed as per the path detected. On basis of the principles of the line following robot, the code is generated. The parameters for the sensors and motors are specified and called.
Other functions: These are basically the functions associated with the UI initialized before.

5. Sensors

The sensors play the most vital role in the implementation of the line following action of the robot. A single cone-shaped proximity sensor and 3 different vision sensors are used for this particular simulation. The proximity (ultrasonic) sensor allows the robot not to collide with an obstacle in case the path has been blocked or obstructed. The vision sensors are used to detect and differentiate the color of the path from the surroundings [9]. Moreover, as sensors are usually light-weight and reliable, consequently not causing any alterations to the weight, inertia and torque of the robot.

The proximity sensor is set as invisible for the simulation. The radius of the cone is 5 meters and the range is set to 1 meter, with an angle of 90 degrees (as shown in figure 6). For the collision simulation, all objects have to be set to detectable, so that they are on par with this ultrasonic sensor. If an object is detected by this sensor, both the left and right wheels move backwards at different speeds, hence in a curve. This helps the robot to re-trace itself onto the path.

An ultrasonic sensor can be used as a proximity sensor which can regulate the distance of an obstacle by emitting ultrasonic waves. It converts the reflect sound wave into an electric signal and this time is used to calculate the distance between the robot and the object. The sensor has two main parts, namely the transmitter and the receiver [10]. The transmitter emits the sound wave and the receiver is where the sound waves travel back to. These sensors can easily detect the surface of an obstacle and perform collision avoidance algorithm in order to facilitate maximum safety in the factory environment.

![Figure 6. Proximity sensor detecting obstacle](image)

The vision sensors (as shown in figure 7) are the most important components of the robot architecture as they help with the detection of the path, and in turn, allow the robot to move forward. All three sensors act as cameras placed 200 mm apart from each other. The orthographic nature with the resolution of 1x1 helps to detect the color beneath. The far clipping plane is 0.15 meters which associates itself with the code guiding this robot to move straight, turn right or left accordingly [11].

Vision sensors are used in this application just as a camera so as to return 1x1 images of the surface underneath the robot. The line to be followed has to be segmented from the factory floor using a set of three such sensors, working simultaneously. Infrared sensors can be used to transmit and receive IR rays which are in-turn collected by the photodiodes based on the Planck’s radiation law, Stephan Boltzmann law, and Wien’s Displacement law [12]. As IR sensors reflect back when a white surface is detected, it provides the necessary inputs needed for the line following action to take place. These
sensors are used to inspect the track to be followed using a combination of both image acquisition and image processing.

![Image of vision sensors](image)

**Figure 7.** Left, middle and right vision sensors

The vision sensors are placed 20 cm apart from each other keeping the line following operation under consideration. The track is 25 cm in width which allows the middle sensor to almost always stay with the edges of the track. As soon as the left or right sensor detects a black image i.e. the line underneath, the middle sensor will detect the factory floor. This allows the steering action to take place almost instantly when the middle IR sensor detects a non-black image [13]. Most line following operations have a narrower track to be followed, but as this takes place inside a factory, maximum safety and minimum delay has to be ensured. These configurations allow the robot to follow the line at sharp turns and even U-turns as the steering delay is almost negligible.

The combination of these two types of sensors not only help in avoiding collision with factory equipment but also navigate the robot throughout the track. These sensors detect the necessary information and corresponding processors convert and analyses the raw data into valuable information which in turn, guides the mechanical action of the robot. The combination of this signal processing technology and dynamic configuration of motors enables the robot to manoeuvre over a pre-defined line in a factory environment.

6. Simulation and Results

The simulation for this robot is carried out using CoppeliaSim. The software allows us to check the dynamic properties of the robot and assign different actuation codes to every component. The basic principles used for this simulation:

- If the middle vision sensor detects a black image, both the wheels move forward with the same speed (v).
- If the left sensor detects a black image, the left wheel moves with a decreased speed (0.2v), hence steering the robot to the left.
- Similarly, if the right sensor detects a black image, the right wheel moves with a decreased speed (0.2v), hence steering the robot to the right.

Therefore, this enables the robot to take travel over a certain “track”, turning left or right and even take U-turns. The vision sensors allow the robot to travel by differentiating the line color and its surroundings. This difference between black and non-black pixels is detected and processed to accede the motion of the robot.

For the simulation, a factory environment is created (as shown in figure 8) consisting of a black track for the robot to trace and follow, a stacking and storage area and grey blocks to portray
machinery. A closed path is created with a horizontal segment including both left and right turns. A floating view (a display screen) is also added and associated with the middle visual sensor of the robot, so as to check the image being detected.

The main base of the robot is a tray-type design so as to provide better assistance to carry unfinished jobs from one machine to another. Many items in a factory are either heated to high temperatures, or too fragile or maybe even radioactive. Hence this robot offers assistance and control over most logistical features on the factory floor, which cannot be carried out by humans.

![Figure 8. Factory scene set in CoppeliaSim](image)

The robot moves along the track designed on the factory floor with varying speed due to the curves and turns. The linear and angular velocity varies in accordance with the path being followed. All sensors and motors work simultaneously which enables the robot to navigate itself and travel efficiently in the environment.

![Figure 9. Top view of path to be taken](image)

![Figure 10. Path curve formulated on an X-Y graph](image)

An X-Y graph is added to trace the path being taken by the robot. The X-axis indicate the absolute x-positions of the robot model base and similarly the Y-axis indicate the absolute y-positions of the robot model base. The points generated are linked so as to form a curve on the graph. This graph determines the path being followed by the robot in real-time and helps in the monitoring and verification of the line following action. Figures 9 and 11 are top views of the pre-defined line on the factory floor, the robot has to detect the line and follow, once the code is run. The sensors detect the line and enable the motors to move along the line as shown in the graphs (figure 10 and 12).

For the first case (figure 9), the speed of the robot is set as high as 1.5 m/s, taking a total time of 75 seconds, covering a path 90 meters in length. The line followed has 2 turns, one towards the right and another to the left. The velocity varies all throughout, due to the proximity between the middle sensor
and the line to be detected underneath. As the robot reaches the first turn, the right sensor detects a black image which executes the actuation code so as to steer the robot to the left by decreasing the speed of the right differential wheel. Correspondingly, during the next turn, the left sensor detects a black image, hence the left wheel decreases its speed to obtain a left steering motion. As no obstacles are detected by the embedded proximity sensor, no disruption is anticipated in the kinematic motion. The simulation runs as per the code provided in the child script. The robot can be stopped at any time using the user interface, and the speed can also be increased or decreased so as to support the payload.

Figure 11. Top view of path to be taken

Figure 12. Path curve formulated on an X-Y graph

In the second case (figure 11), with the same speed, this robot covers a path of 80 meters in 62 seconds. As noticed, this path consists of a steep turn to the left, which corresponds to a black image being detected by the left sensor, leading to a decrease in speed in the left wheel, causing a left steering motion. Even though the robot passes along different machinery, it is at a safe distance from any damage or else the proximity sensor would detect and source a feedback motion. This path takes less time than the path in figure 9 as it consists of only one turn. The resultant speed during a turn is considerably less in case of a differential drive line following robot. On taking the average for the entire path in the environment, this mobile robot manoeuvres over a pre-defined path with an average of 1.25 meters per second. The decrease in speed is due to the varying speeds of specific motors at the turns in the designed path.

The main aim of this simulation is to facilitate the robot to travel over the pre-defined path on the factory floor. The model base moves as per the data collected from the sensors and the actuation followed by the corresponding motors. This paper signifies the possibility of line following robots on a factory floor by assembling a state of the art robot with necessary components and a code which helps the simulation to succeed.

7. Conclusion and Scope

The advancement in manufacturing technology and automation has set its mark in the era of Industry 4.0. This paper describes all the instructions and steps to be followed to design, assemble and simulate a line following robot in a factory environment. On simulating, the robot travels over the pre-defined path which promises to be beneficial in most industrial scenarios. This method of transportation is far more efficient and superior than traditional methods. It requires minimal maintenance, almost no workforce and no hazards to safety and security.

Forklifts require fuel and tank replacements, which increases maintenance costs and duties. Cranes take up space in a manufacturing layout and overhead cranes are extremely unsafe and heavy-duty.
Using line following robots overcomes all these issues and delivers promising results for the transportation of “work-in-progress” goods on factory floors.

These robots can be an essential aspect to optimize a factory operation and the manufacturing quality can be enhanced to a greater level. However, safety regulations regarding the same are extremely necessary and should be evaluated keeping all constraints in mind. According to the US Department of Labor, there were only 38 robot-related accidents in the United States between 1987 and 2016 [14]. Despite the possibility of these accidents is low, safety measures should be taken for the protection of the employees and the machinery involved. Laser scanners can also be used to guarantee collision avoidance between the robots and workers on the factory floor. All mobile robots should be assessed individually and all dangers should be checked regularly.

The sensors and actuators need multiple tests and re-planning which ensure the efficiency of the robot to perform the operation [15]. The simulation study ensures that this following robot with differential wheels, a castor joint, a proximity sensor and 3 vision sensors can operate over a globalized path in a factory environment. Considering suitable turns and shifts in the path, the robot if found to cover a distance of 10 meters in 8 seconds, when the speed is set as 1.5 m/s.

Optimizing a facility’s configuration increases the manufacturing outcome, quality and the supply output. Automated machinery, Implementation of Industrial IoT and cloud-based design and the usage of autonomous robots can contribute to an enterprise’s journey towards quality assurance, vigorous inspection and extensive production. The assistance provided by these robots can help manufacturers to contrive optimized factory and assembly layouts, with incremented operational efficiency.

References
[1] “How replacing forklifts with AMRs increases safety”, Mobile Industrial Robots (https://www.mobile-industrial-robots.com/en/insights/amr-safety/how-replacing-forklifts-with-amrs-increases-safety/)
[2] S. Sedhumadhavan, E. Niranjana, “An Analysis of Path Planning for Autonomous Motorized Robots”, International Journal of Advance Research, Ideas and Innovations in Technology (Page 1234-1257, Volume 3, Issue 6)
[3] Beatriz Arruda Asfora, “Embedded Computer Vision System Applied to a Four-Legged Line Follower Robot”, 23rd ABCM International Congress of Mechanical Engineering, December 6-11, 2015
[4] Shomitro Kumar Ghosh, Toheen Bhuiyan, Raihan Chowdhury, Ismail Jabiullah, Daffodil International University, “A Line Follower Robot with Obstacle Detection by Ultrasonic”, A Steganographic Apps-based Patient’s Information Encryption-Decryption.
[5] Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z, “A Simple Local Path Planning Algorithm for Autonomous Mobile Robots”, International Journal of Systems Applications, Engineering & Development (Page 151-159, Issue 2, Volume 5, 2011)
[6] S Samuel Abhishek, K.Veladri, “Trajectory Planning of a Mobile Robot”, National Conference on Technological Advancements in Mechanical Engineering (Page 296-302)
[7] “List of Robotic Simulation Softwares”, Ricky Ruijiao Li, University of Essex (https://sites.google.com/site/ruijiaoli/resources/roboticsimulationsoftwarelist)
[8] Regular API function list by CoppeliaSim (https://www.coppeliarobotics.com/helpFiles/en/apiFunctionListAlphabetical.htm)
[9] “Line following robot using Arduino”, Saddam, Circuit Digest (https://circuiddigest.com/microcontroller-projects/line-follower-robot-using-arduino)
[10] “Sensors”, Fierce Electronics (https://www.fiercenelectronics.com/sensors/what-ultrasonic-sensor)
[11] M. S. Islam & M. A. Rahman, “Design and Fabrication of Line Follower Robot”, Asian Journal of Applied Science and Engineering, Page 27-32, Volume 2, No 2 (2013)

[12] Jagruti Chaudhari, Asmita Desai, S Gavarskar, MGM’s Jawaharlal Nehru Engineering College, “Line Following Robot Using Arduino for Hospitals”, 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), September 2019.

[13] “Vision sensors and their modern uses”, AZO Sensors (https://www.azosensors.com/article.aspx?ArticleID=1129)

[14] “The Importance of Robot Safety on Your Factory Floor”, Genesis Systems, IPG Photonics (https://www.genesis-systems.com/blog/importance-robot-safety-factory-floor)

[15] Yitao Ding and Ulrike Thomas “Collision Avoidance with Proximity Servoing for Redundant Serial Robot Manipulators” IEEE Systems Journal PP(99): 1-11, September 2018