**Introduction**

**Mobile Robot & Navigation:**

Have you ever seen service robots in a restaurant? Or let a cleaning robot clean your room? In fact, these kinds of robot have been changing our world more than yours think. Many new types of mobile robots have been developed to work in places like hospitals, factories, or other public areas. To work safely and efficiently for such robots, they have to evade many obstacles like chairs, tables, or even people, so a powerful navigation system is important.

Hello everyone! Let me start with a question: have you ever seen robots at work in a restaurant or maybe used a cleaning robot at home? These mobile robots are more than just cool gadgets; they have been reshaping many industries greatly. From hospital, factories, to other public areas, mobile robots are becoming essential. But here’s the problem: for those robots to operate safely and efficiently, they need to navigate complex environments filled with obstacles like chairs, tables, and even people. This is where a powerful navigation system comes into play, and that’s exactly what I have been researching.

**Navigation system:**

Just like the google map, in the robotic field, conventional navigation systems can generate a clear path from point A to point B, telling the robots how to avoid static obstacles and reach their destinations. Dozens of strategies have come out to deal with this issue, and many of them have found big success in dealing complicated environments even like a maze. However, if there are people, cars, or other moving obstacles in robots’ working environment, none of a fix path could be guaranteed safe. Therefore, a different type of navigation system should be applied to overcome this situation.

Think of navigation system as the robot’s version of Google Maps. Traditional systems do a great job generating a clear path from point A to point B, telling the robots how to avoid static obstacles. But the problem will become more difficult when the environment is dynamic. When there are moving objects like cars, people or pets, fixed paths are no longer safe. Mobile robots need something smarter—a robust system that can quickly adjust to changes in its environment.

**Method**

**An obstacle-avoidance algorithm for dynamic environment:**

So, my research focuses on developing an obstacle-avoidance algorithm for mobile robots in dynamic environment. This system is different from other navigation tools. Instead of generate a clear route, our algorithm generates a motion command for robots in each time period. In other words, in every second, our method is going to tell robots how to move forward or which direction I should turn. This system can help the robot develop a kind of “special awareness”, by following the three steps: observation, prediction, and making decisions. Let me explain more detail in the following slides.

So, my research focuses on developing an obstacle-avoidance algorithm for mobile robots in dynamic environment. Unlike traditional system, this approach doesn’t create a fixed path. Instead, it generates motion commands for robots in each time period. In other words, it gives robots moment-to-moment instructions like “turn left” or “move forward”, based on real-time data. Essentially, this method should help the robot develop a kind of “spatial-awareness”, by following the three steps: observation, prediction, and decision-making. Let me explain more detail in the following slides.

**Three steps in the method:**

Step1 Observation:

In general, mobile robots use many sensors like radars, LiDAR, or cameras to detect environment changings. They are similar to people’s eyes, ears, or nose. However, those sensors are too sensitive and have many noises in their row data. Therefore, we can’t use those signals directly. We have to process those signals by a filter and get the relatively clear data.

In general, mobile robots use sensors like radar, LiDAR, and cameras to scan its surroundings and detect changing in environments. They are similar to people’s eyes and ears. But here’s a challenge: raw sensor data is noisy and unreliable. To tackle this, the system must filter the data to get a cleaner, more accurate picture of obstacles.

Step2 Prediction:

In step2, we assume that the obstacles are linear models. That mean, we guess those obstacles’ movement have some trends. We can evaluate where they will be by finding their moving trends. Just like typhoon predictions in a weather forecast, the prediction of obstacles will be more unreliable as we look further in future.

In step2, once the robot knows where obstacles could be, this algorithm assumes those obstacles follow a certain pattern—(maybe a straight line or curve)—and estimates their future positions. The farther into the future we predict, the less reliable it gets. So, we have this picture, just like a typhoon’s prediction in weather forecasts.

Step3 Making decision:

Now, our system knows where obstacles could be in the future. So, in the final step, we want to find the best motion command for robot to reduce the chance of collision. Nevertheless, this motion should let the robot reach its destination as fast as possible. Because of this, optimization technics was integrated in our method. Just like this picture, the optimization tool can help the robot pick the smartest option to move forward.

**Application on robot:**

We apply the navigation system to a specific type of mobile robot called General Bicycle Mode or GBM. The GBM has a unique design. Unlike regular bicycles that steer by turning the front wheel, the GBM has two wheels that can rotate and move independently. This design allows it to move in any direction without changing its orientation—as you can see in this gif picture! This flexibility makes the GBM highly maneuverable.

To test this navigation system, I applied it to a specific type of mobile robot called General Bicycle Mode or GBM. The GBM has a unique design. Unlike regular bicycles that steer by turning the front wheel, the GBM has two wheels that can rotate and move independently. This design allows it to move in any direction without changing its orientation—as you can see in this gif picture! This flexibility makes the GBM highly maneuverable.

**Experiment:**

So, we want to test whether this algorithm is worth developing or not. Therefore, we build a Python simulation for experiment. In this experiment, we give the GBM robot a simple mission, move from point A to point B. However, the robot will encounter obstacles in different directions and speeds along their path. As the result, all the GBM has to do is try to avoid those obstacles and reach the goal safely. Furthermore, we record the time cost in the hall process to test the efficiency of the algorithm.

We only consider two dimensional movement in this simulation. Also, we assume that every obstacle is well detected and the wheel skid on the GBM robot was ignored.

To see how well this algorithm works, I ran a Python simulation. In the experiments, the GBM had one simple mission: move from point A to point B. Along the way, it encountered obstacles moving in different directions and at different speeds. The robot’s job was to avoid these obstacles and still reach its goal. I also measured how long it took to complete the task to evaluate the system’s efficiency.

Now, here are some assumptions I made for this simulation:

First, everything happens on a flat, two-dimensional plane.

Second, obstacles are perfectly detected. No missed spots

Finally, wheel skidding on the robot was ignored to keep things simple.

**Result**

**Experiment result:**

After testing our navigation system in several scenario, we have two significant results.

First, our algorithm can bring the robot to it destination successfully in almost every scenario. The passing rate is about 88%, indicating our navigation algorithm is a feasible solution.

Second, we compare our method with an exit method. The result shows that our algorithm can reduce around .8 second to let the robot reach its destination. This comparison suggest that our new navigation system is a more efficient strategy.

After testing our navigation system in several scenario, we have two significant results.

First is success rate: our algorithm can bring the robots to it destination successfully in almost every scenario. The passing rate is about 88%, indicating our navigation algorithm is a feasible solution.

Second is efficiency: compared to an existing method, the new system reduced the average travel time by 0.8 seconds. This might not sound like much, but in robotics, small gains in efficiency can lead to big improvements overall.

**Conclusion**

**Contribution:**

In this research, we have several contributions. First, we propose an obstacle-avoidance algorithm for mobile robots, let the robots have the ability to navigate in dynamic environment. Second, we apply our method to the specific type of mobile robot – General Bicycle Model and take several tests on it. Moreover, in the future, mobile robots with this algorithm could operate safely and efficiently in complex environments, providing support in Logistics industry, Manufacturing industry, and medical field or so on.

Here is why this research matters:

First, it introduces a practical obstacle-avoidance algorithm for dynamic environments, let robots have the ability to evade moving obstacles.

Second, it applies this method to a versatile robot, the GBM, and demonstrates its effectiveness through testing.

Moreover, it allows mobile robots to operate safely and efficiently in complex environment, providing support for logistics, manufacturing or other industries in the future.

**Future works:**

My research still has some works need to be done in the future:

First, more modification needs to be done in our algorithm. Provided the system is polished, the passing rate of the GBM in the previous simulation could be rose.

Second, a more complicated simulation is needed to test our algorithm’s performance, such as: adding more obstacles in the simulation environments, and let the obstacles’ movement more unpredictable.

Last but not least, we are going to apply the algorithm to a realistic robot hardware to evaluate the performance of our method.

Additionally, there are some works need to be done:

First, the algorithm needs further refinement to improve its success rate. Our goal is to insure the security for both robot and people in same environments.

Second, future simulation should be more complex in order to test our algorithm’s performance. Here is something I can do like adding more obstacles in the simulation environment and let the obstacles’ movement more unpredictable.

Last but not least, this algorithm should pass the ultimate test—implementing this system on a real robot. This could provide critical insights into its real world performance.