Autonomous Mobile Robot Navigation Tracking Problem

Saddam Hossain
School of SEE, University of Salford, Manchester M5 4WT, UK
s.hossain1@edu.salford.ac.uk

***Abstract* - This project presents an obstacle avoidance algorithm and an odometry model for mobile robot navigation. The navigation system under consideration includes three navigation subsystems: angular and linear velocities, as well as a navigator system. We assume that the robot can detect and avoid various types of obstacles such as furniture. the robot used to turn and free from obstacles, collisions and corners. We're working on a viable autonomous mobile robot navigation system that can conduct obstacle avoidance, odometry model, and solve the target tracking problem by Kalman tracker and Zone tracker tasks as part of our present study. To complete the navigation job, a fuzzy controller is used to handle the door traversal problem by evaluating linguistic input and output variables using fuzzy sets utilizing a rule foundation. This method's characteristics are examined, and the result is presented.**

Index Terms - Robot Navigation, Odometry Model, Robot Target tracking, Robot obstacle avoidance, Door traversal, Subsumption coordination.

I. INTRODUCTION

An autonomous mobile robot's most important characteristic is its capacity to function autonomously in new or partially understood surroundings.[2] In robotics, the tracking or target tracking issue is well-known, and it plays a crucial role in autonomous mobile robot navigation (artificial intelligent mobile robots).To solve the tracking problem use a vision-based method that allows the robot to see a colour blob selected by the user for tracking. Furthermore, a stereo camera can partially solve this problem, but the computation is time-consuming and the precision is poor. [3] Instead, a laser range finder is an excellent option. It calculates distances quickly and precisely while being unaffected by environmental factors. To build the target track there have been used Kalman tracker and zone tracker. The robot must adapt to changes in the environment, such as the rapid emergence of obstacles, during local navigation. This endeavour has been proved to be successful with sonar sensors. the navigation system inspired to used an odometry-based navigation method and investigation the obstacle avoidance to pursue make decolider, avoidance and unstrapped method which helps to pass the corner place on the floor. There have been also inspired us to solve the door traversal problem by fuzzy controller system. the primary objectives of this project to deal with lost tracker target, handle obstacle, accurate delivery at destinations of a navigation system. It is important to guide the robot to properly follow the navigation and pass the obstacle. The object target tracking is essential that enables the robot to track: identify the object and follow turn and move the robot towards the target and approach. For the contribution, there have been used some methods those help to implement this project. [4] In reality, there have been several studies on the tracking challenge of mobile robots published in the literature. The rest of the project is planned out as follows. The Literature Review and Methods will be presented in the next part.[4] The results include simulation and visualisation. Finally, the conclusion is presented.

A. *Literature Review*

[5]Recent technological advancements in the field of robotics have made significant contributions in a variety of industrial and social arenas. [5] For many such applications, the autonomous mobility of robots is a mandatory key issue. [6] Throughout the next few years, robotics in general, and mobile robotics in particular, will continue to evolve. [6] The current trend suggests that the market for autonomous automobiles will be driven by technical advancements in mobile robotics in the next years and that open-source platforms, together with lower sensoring cost, would stimulate demand for home mobile robots like vacuum cleaners.[7] Based on the analysis of the vehicle model's characteristics and the target tracking control algorithm, a target tracking control method to the tracking problem of an autonomous mobile robot is proposed.[8] The existence of moving impediments is another cause of uncertainty in mobile robot navigation. While some of these items' behaviour (for example, a car going down the road) can be expected, others (for example, a child sprinting across a field) may move at random. [8] The approach to unexpected obstacles and unknown trajectories, also known as collision avoidance or collision detection, is based on reactive navigation and is not constrained by the planning problem's hardness restrictions.

B. *Paper Outline*

All of the methods have been described in the next Section where have written about the method model description, Implementation and Pseudocode of all methods. We demonstrate the trajectory graphs and correlation graphs to find the best result outcome. The result reported in tables those will discuss on Result Section.

II. METHODS

In this project, we have been used so many methods such as - odometry model, obstacle avoidance model, target tracking model, door traversal model and subsumption behaviour coordination. All methods are going to describe below.

Odometry Model: Odometry is a technique for estimating a mobile robot's location and orientation [9]. It is one of the most often used methods for estimating relative location.

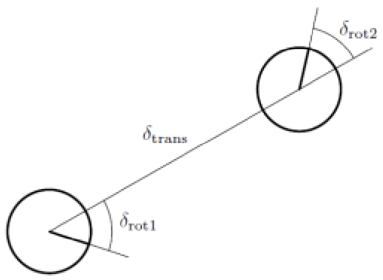


Figure 1. Odometry model: A rotation &rot1, followed by a translation & trans, and a second rotation & rot2, approximates the robot motion in the time span (t – 1, t]. The turns and translations are a bit claustrophobic.

A. Model Description

By integrating over the internally measured motion of the actuators or the signals of an inertial measurement unit, the odometry system typically predicts relative changes in the robot position[10]. Odometry is widely recognised for providing good short-term accuracy, being affordable, and allowing for extremely high sample rates. For a variety of reasons, odometry is employed in practically all mobile robots: To offer a better and more trustworthy location estimation, odometry data can be combined with absolute position measurements. However, Many mapping and landmark matching methods presume that the robot can retain its position well enough to seek landmarks in a confined region. [11] Odometry can be utilised in between absolute position updates with landmarks. Odometry may be accomplished without the need for external information. As a result, odometry is simple, affordable, and quick to do in real-time. [9] Internal systematic and non-systematic variables, on the other hand, have a tremendous impact on odometry, causing an infinite accumulation of errors.

B. Architecture

In this work, The model requires that the robot's odometry (x, y, θ) be calculated using wheel encoder sensors and the robot's geometry. The model generates an angle transition (based on the arctangent function) that rotates the robot towards the node of coordinates, as well as a linear transition (based on the Euclidean distance) that pushes the robot node of coordinates to move across those nodes. This Architecture, describing the pseudocode algorithm which will allow the robot to move in a predefined pair of coordinates.

```
Odometry Model Algorithm:
 TURN ← 0
MOVE ← 1
 step ← MOVE
 i ← 0
 ni,j ← [500 500 0 0 500 500 0 0]
method navigator(vel)
 x ← ni,0
 y ← ni,1
 th ← angle(x, y)
 switch step
 case TURN:
 turnSpot(-vel / 2)
 if angularDestination(th) = true then
 stop( )
 step ← MOVE
 end if
 break
 case MOVE:
 move(vel)
 if linearDestination(x, y) = true then
 stop( )
 step ← TURN
 if ++i = n.length then
 return true
 end if
 end if
 break
 return false
 end method
```

C. Implementation

To implement this algorithm, we have to make a new Odometry class in our project, where build many methods to fulfil the robot to touch the nodes. The odometry class of all methods is called in a run class for testing purpose. At first, we have written a getAngle() method to get the tangent angle that points to a pair of coordinates then make a get360() methods to get the reset the angle after exceeding 360 degrees. Secondly, For the angular test create a new method isAngularDestination() to check if the robot has reached the node's angular destination, when this method called in run class we have seen it is working on a MobileSim area simulator. Thirdly, For the linear test create a new method isLinearDestination() to check if the robot has reached its linear destination using Euclidean distance, when this method called in run class we have seen it is working on a MobileSim area simulator. Finally, For the navigator test create a new method navigator() to implement waypoint navigation method where we have used as following that algorithm node and used switching statement case to check the linear or angular position to go the robot and touched all nodes when this method called in run a class we have seen it is working on a MobileSim area simulator and touch all of nodes point.

Obstacle Avoidance Model: [12] Obstruction avoidance algorithms are used to avoid colliding with barriers. Obstacle avoidance algorithms deal with the robot's movement depending on data from its sensors. The mobile robot's trajectory is being changed in real-time by an obstacle avoidance algorithm to prevent collisions with obstructions on its route.

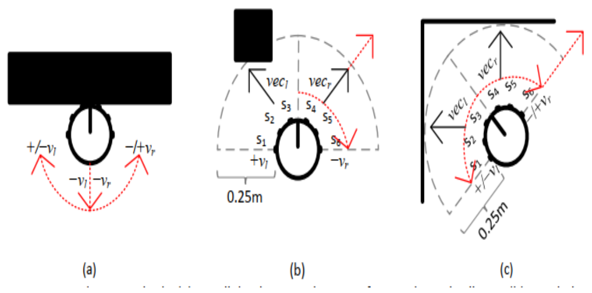


Figure 2. Avoidance methods. (a) Decollide: detect and recover from a physical collision, (b) Avoid: detect and avoid obstacles, (c) Untrap: detect and recover from corner traps.

A. *Model Description*

[6]During robot motion, collisions between the mobile robot and obstacles must be avoided. Robot navigation refers to a mobile robot's capacity to travel through a known or unfamiliar area in order to reach a goal without colliding with any impediments.[6] A competent motion planner must also be able to detect collisions between the robot and barriers in the work area so that the robot may adjust its direction or stop before colliding. Geometric models or topological maps of the environment are used to power these algorithms. The environment is modelled by the robot. It can detect collisions by calculating distances since it knows where it is at any given time. Furthermore, Path planning algorithms seek to produce a collision-free mobility trajectory for the robot over time by constructing a route from one spot in the environment to another. However, [13] Robot already knows the way to take, which is provided by a human and uses the knowledge about the way and the presence of obstacles to devise a new way to the destination on its own.

B. *Architecture*

The robot's sonar ring, which is a collection of 8 ultrasonic range sensors (range: 10-5000mm), is employed in the avoidance algorithm, although only 6/8 sensors will be employed: s1 - s6.In this exercise, we created multiple algorithms that allow the robot to avoid obstacles by identifying and turning away from them, decollide by identifying and avoiding collisions and untrap by identifying and avoiding corner traps. The mathematical notation for each avoidance approach is shown in the table below. The wheel velocities that are utilised to turn and free the robot from obstacles, collisions, and corners are updated in the vectors velocity of decollide, the velocity of avoiding, and velocity of untraper.

```
Decollision Algorithm:
method decollide(vel)
 initPose ← robotX = 0 and robotY = 0
 zeroVel ← robotlvel = 0 and robotrvel = 0
 if zeroVel and ¬initPose then
 move(-vel)
 delay(2000)
 p ← random( )
 if p < 0.5 then
 turnSpot(vel)
 else
 turnSpot(-vel)
 end if
 delay(2000)
 return true
 else
 move(vel)
 end if
 return false
 end method
```

```
Avoidance Algorithm:
GAMMA_A ← 250.0
 vel ← 0.0
 method avoid(vel)
 l_vec ← [s1 s2 s3]T
 r_vec ← [s4 s5 s6]T
 l_min ← min(min(l_vec0, l_vec1), l_vec2)
 r_min ← min(min(r_vec0, r_vec1), r_vec2)
 if l_min < GAMMA_A then
 this.vel ← +vel
 turnSpot(this.vel)
 return true
 else
 if r_min < GAMMA_A then
 this.vel ← −vel
 turnSpot(this.vel)
 return true
 else
 decollide(vel)
 end if
 return false
 end method
```

```
Untrap Algorithm:
 l_cnt ← 0
 r_cnt ← 0
 l_trap ← false
 r_trap ← false
 method untrap(vel)
 If this.vel ≥ 0 and ¬l_trap then
 l_cnt++
 l_trap ← true
 r_trap ← false
 else
 if this.vel < 0 and ¬r_trap then
 r_cnt++
 r_trap ← true
 l_trap ← false
 end if
 if l_cnt ≥ 2 and r_cnt ≤ 1 then
 turnSpot(vel)
 delay(2000)
 l_cnt ← 0
 r_cnt ← 0
 l_trap ← false
 r_trap ← false
 return true
 else
if r_cnt ≥ 2 and l_cnt ≤ 1 then
 turnSpot(-vel)
 delay(2000)
 l_cnt ← 0
 r_cnt ← 0
 l_trap ← false
 r_trap ← false
 return true
 else
 avoid(vel)
 end if
 return false
 end method
```

C. *Implementation*

To implement these algorithms, we have to make a new ObstacleAvoidance class in our project, where build many methods to fulfil the robot avoid the obstacle and pass the corner place on the floor. The ObstacleAvoidance class of all methods is called in a run class for testing purpose. At first, we have written a decolider() method to Perform collision detection and avoidance. It will send true if a collision is detected then will go backwards, otherwise false that will go forward. Secondly, For the avoidance test create a new method avoid() to Validate if an obstacle is detected left and turn right or right and turn left otherwise that will follow the decollider method, when this method called in run class we have seen it is working on a MobileSim area simulator. Finally, For the untrap test create a new method untrapper() to Perform corner trap detection and avoidance, where if a trap is detected it will try to escape phase, when this method called in run class we have seen it is passing the corner place on the floor in a MobileSim area simulator visualiser.

Target Tracking Model: The suggested controller's validity is demonstrated by its application to target tracking of a mobile robot in an unknown environment.

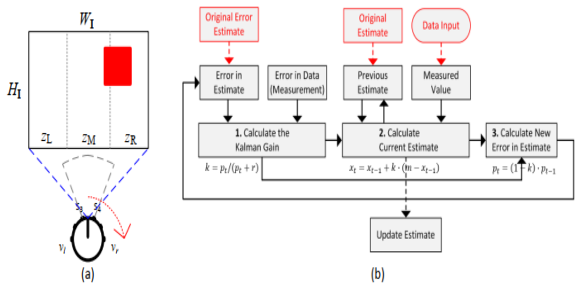

W_I
H_I
z_L
z_M
z_R
s_3
s_4
v_l
v_r
(a)
Original Error Estimate
Original Estimate
Data Input
Error in Estimate
Error in Data (Measurement)
Previous Estimate
Measured Value
1. Calculate the Kalman Gain
2. Calculate Current Estimate
3. Calculate New Error in Estimate
$k = p_t/(p_t + r)$
$x_t = x_{t-1} + k \cdot (m - x_{t-1})$
$p_t = (1 - k) \cdot p_{t-1}$
Update Estimate
(b)

Figure 3. Tracking methods. (a) 3-zone image tracker and follower, (b) 1D Kalman filter diagram.

A. *Model Description*

The target tracking behaviour is a vision-based mechanism that allows the robot to perceive and track a colour blob that has been chosen by the user. By default, the blob colouring method returns the discovered blob's centre of mass as picture coordinates xc, yc. [14] The usage of the rg-chromaticity colour space, adaptive colour processing edge detection, and colour segmentation as part of the colour image identification routine is proposed in this section. This method will be used to track and navigate the mobile robot's targets. [15] The suggested tracking robotic system, which is led by sonars and a camera, is capable of detecting, tracking, and following the target robot while avoiding obstacles.

B. *Architecture*

We built a tracker technique in this exercise that allows the robot to track: detect the item among three horizontally divided sections in the picture, follow: turn and drive the robot towards the target, and approach: approach the target and halt at a safe distance. Implement a tracking system that employs a Kalman filter, which is an iterative mathematical procedure comprised of a set of equations and sequential data inputs that quickly predicts the real value, location, velocity, and other properties of the item being monitored when the observed values contain unpredicted or random error, uncertainty, or fluctuation.

Approach Algorithm:

```
 GAMMA_A ← 350.0
method approach(vel)
 f_vec ← [$s3$ $s4$]$T$
 f_min ← min(f_vec0, f_vec1)
 If f_min ≤ GAMMA_A then
 stop( )
 return true
 else
 move(vel)
 return false
 end if
 end method
```

3-zone tracking Algorithm:

```
 method zoneTracker(vel)
if blobX > 0 and blobX < (IW / 3) then
 turnSpot(+vel)
 return true
 else
 if blobX > (2 · IW / 3) and blobX < IW
 turnSpot(-vel)
 return true
 else
 approach(vel)
 return false
 end if
 end method
```

Set mapping algorithm:

```
method map(x, A1, A2, B1, B2)
 return (x − A1) · (B2 − B1) / (A2 − A1) + B1
end method
```

1D Kalman Tracking:

```
 q ← 0.1
 r ← 0.1
 p ← 0.1
 k ← 0.1
 x ← blobX / 2
 method kalmanTracker(vel)
 m ← blobX // Get measurement.
 p ← p + q // Kalman prediction update.
 k ← p / (p + r) // Kalman measurement update.
 x ← x + k · (m − x)
 p ← (1 − k) · p
 velr ← map(x, 0, IW, -vel, +vel)
 if velr ≥ −(vel/4) and velr ≤ +(vel/4) then
 approach(vel)
 return false
 else
 turnSpot(velr)
 return true
 end if
 end method
```

C. *Implementation*

To implement these algorithms, we have to make a new Targettracking class in our project, where build many methods to perform target tracking using 3 discrete zones, 1d Kalman tracker and let the robot approach the target at a safe distance. The Targettracking class of all methods is called in a run class for testing purpose. At first, we have written a zonetracker() method to Perform target tracking using 3 discrete zones. It will send true if detected a target object, otherwise false. We have also used a camera sensor and a blob detector to identify the object and follow the object motion. If the object will take the left image zone the robot will take a turn left or If the object will take the right image zone the robot will take a turn right. Secondly, For the Kalman tracker test create a new method Kalman tracker() to perform target tracking using a 1D Kalman filter, when this method called in run class we have seen it is showing Kalman estimate and the measurement with turn to the estimate detection on a MobileSim area simulator. Finally, For the safe distance test create a new method approach() to let the robot approach the target at a safe distance, when this method called in run class we have seen it is controlled at safe distance in a MobileSim area simulator.

Door Traversal Model: [16] This work presents a way for a mobile robot to discover the position of doors in complicated surroundings and securely transit the door in order to improve performance.

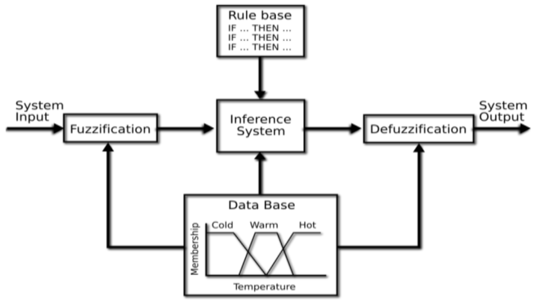


Figure 4. Fuzzy controller scheme. Fuzzification is the conversion process of crisp input values into fuzzy values using fuzzy sets. The Fuzzy Inference System (FIS) is the rule-based evaluation process. Defuzzification is the conversion process of fuzzy output values into crisp values using fuzzy sets.

D. *Model Description*

The fuzzy controller technique used to solve the door traversal problem is presented in Figure 4. Fuzzy Logic (FL) is a system that employs a rule basis to analyse linguistic input and output variables that employ fuzzy sets in order to provide crisp and precise numerical results. It is based on human thinking.

E. *Architecture*

In this exercise, we used the jFuzzy Logic Java package to create a fuzzy controller for the door traversal issue. By auto-centring and going through a tiny or broad door, our system guides the robot through it. The fuzzy controller's main design must be implemented within the controller. We generated or corrected our own fuzzy input/output membership functions and rules using the controller.fcl file.

Fuzzy Door Traversing setup Algorithm:

```
MAX_RANGE ← 2000.0
VEL_GAIN ← 3.0
MIN_VEL ← 50.0
method controller(vel)
l_vec ⟵ [$s0$ $s1$ $s2$]$T$
m_vec ⟵ [$s3$ $s4$]$T$
r_vec ⟵ [$s5$ $s6$ $s7$]$T$
l_min ← min(min(l_vec0, l_vec1), l_vec2)
m_min ← min(m_vec0, m_vec1)
r_min ← min(min(r_vec0, r_vec1), r_vec2)
if l_min > MAX_RANGE then l_min ← MAX_RANGE end if
if m_min > MAX_RANGE then m_min ← MAX_RANGE end if
if r_min > MAX_RANGE then r_min ← MAX_RANGE end if
fuzzifier.set("l_vec", l_min)
fuzzifier.set("m_vec", m_min)
fuzzifier.set("r_vec", r_min)
fis.evaluate([1 ⋯ $n$]$T$)
$vL$ ⟵ defuzzifier("l_vel")
$vR$ ⟵ defuzzifier("r_vel")
if $|vL|$ < MIN_VEL then $vL$ · VEL_GAIN else $vL$ end if
if $|vR|$ < MIN_VEL then $vR$ · VEL_GAIN else $vR$ end if
setVel($vL$, $vR$)
end method
```

F. *Implementation*

To implement these algorithms, we have to make a new Doortraversal class in our project, where build a method to fulfil the robot overcomes the door problem. The Doortraversal class of method is called in a run class for testing purpose. At first, we have to collect left/middle/right sensor ranges then Calculate the left/right min sensor vector also limit the input's max range. Secondly, do the fuzzification set crisp inputs in fis file, where set all of the fuzzy rules with evaluating rule base. Finally, amplify velocities at min threshold and set all velocities to a robot.

Subsumption Behaviour Coordination Model: [17]The subsumption architecture is a strong tool for building intelligent robot control systems by layering basic actions together.

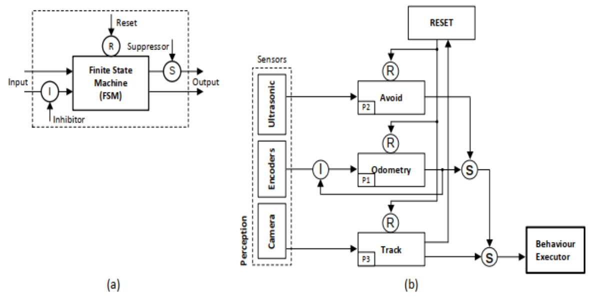


Figure 5. The Subsumption coordination architecture. (a) Augmented Finite State Machine (FSM), (b) Behaviour Executor

G. *Model Description*

Reactive paradigms impacted the Subsumption Architecture, where a behaviour is a network of sensing and acting modules that complete a job, and the modules are Augmented Finite State Machines (AFSMs or FSMs). An FSM is made up of simple or Inhibition inputs that restrict signals/data/directives from being inserted, simple or Suppression outputs that prohibit signals/data/directives from being exerted, and a Reset input that returns the behaviour to its initial state (Figure 5).

H. *Architecture*

In this activity, we used any of the following coordinating methods: (1) The Flag-Based Behaviour Coordinator, described in Algorithms, is a simple approach for prioritising the execution of numerous behaviours, or (2) The Subsumption Behaviour Coordinator, described in Algorithms, is an FSM-based system that leverages levels of intelligence or competence to encapsulate fundamental survival functions (collisions) and generate goal-directed actions (odometry model, tracking).

Subsumption Behaviour Coordinator Algorithm:

```
avoid ← Avoid(prior=75, vel=100) // Instantiate Avoid behaviour.
odom ← Odometry(prior=100, vel=100) // Instantiate Odometry behaviour.
method update(vel)
priorn ← {n=2}
sonarDatai ← [s1 s2 s3 s4 s5 s6]T
odomDatai ← [robotX robotY]T
prior0 ← avoid.init(Entry[Entry(sonarData, false)], Entry[Entry([-1.0], false)], false)
prior1 ← odom.init(Entry[Entry(odomData, odom.getInhibitor( ))], Entry[Entry([-1.0], avoid.getSuppressor( ))], false)
max ← max(prior)
if max = avoid.getPriority( ) then
Exercise2.avoid(vel)
else
if max = odom.getPriority( ) then
Exercise1.navigator(vel)
end if
end method
```

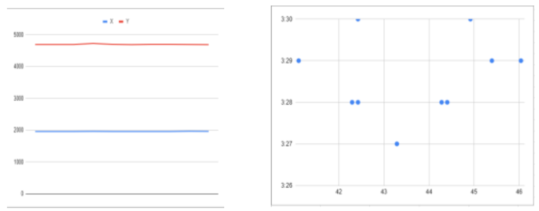
```
Flag-Based Behaviour Coordinator:
flag ← false
method flagCoordinator(vel)
if ¬flag = true then
flag ← Exercise1.navigator(100)
else
if ¬Exercise2.avoid(100) = true then
Exercise3.tracker(100)
end if
end if
end method
```

I. Implementation

To implement these algorithms, we have to make a new SusumptionCoordinator class in our project, where build a method to run a custom subsumption architecture that will work at the same time with odometry, obstacle avoidance. The SusumptionCoordinator class of all method is called in a run class for testing purpose. At first, For the run test create a new method run() to select the priority odometry and avoidance, when this method called in run class we have seen it is following the odometry nodes with obstacle avoidance on a MobileSim area simulator.

III. Results

Using the control panel for odometry, lead the robot to node n0 and through all specified nodes till (n0 to n5), recording the spatial odometry coordinates (x, y) for each node. The odometry model is started by the robot at node n0 and concludes at node n5, where the odometry route finishes. collect the destination coordinates (odometry: $n5$) and time taken at the last point.



(a) (b)

Figure 6. Odometry performance. (a) X and Y correlation, (b) Euclidean correlation

The table illustrates a correlation between distance and time with euclidean :

TABLE I
EXPERIMENTAL TABLE FOR THE ODOMETRY PHASES.

Samples	***Odometry(mm)***		***Euclidean***	***Time(sec)***
#	X	Y	d(x,y)	t
1	1958	4694	42.4264	3:30
2	1955	4694	45.3982	3:29
3	1958	4694	42.4264	3:28
4	1962	4726	46.0434	3:29
5	1958	4695	42.2965	3:28
6	1957	4687	44.9221	3:30
7	1956	4695	44.2831	3:28
8	1957	4695	43.2831	3:28
9	1965	4694	44.4072	3:28
10	1961	4687	41.1096	3:29
μ σ				

Using the control panel for tracking, lead the robot to node n5 and through all specified nodes till (n5 to n6), recording the spatial odometry coordinates (x, y) for each node. The odometry model is started by the robot at node n5 and concludes at node n6, where the odometry route finishes. collect the destination coordinates (odometry: $n6$) and time taken at the last point.

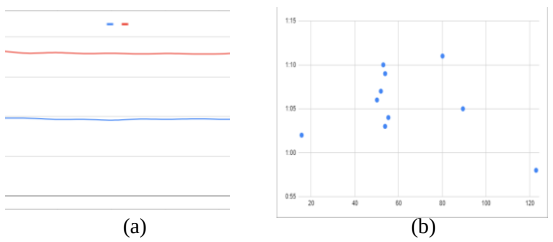

(a) (b)

Figure 6. Tracking performance. (a) X and Y correlation, (b) Euclidean correlation

The table illustrates a correlation between distance and time with euclidean :

TABLE I
EXPERIMENTAL TABLE FOR THE TRACKING PHASES.

Samples	***Odometry(mm)***		***Euclidean***	***Time(sec)***
#	X	Y	d(x,y)	t
1	3915	7322	122.9186	0:58
2	3912	7190	15.6204	1:02
3	3853	7222	51.8941	1:07
4	3855	7172	53	1:10
5	3812	7184	89.4427	1:05
6	3869	7156	53.8237	1:03
7	3855	7178	50.0899	1:09
8	3880	7150	53.8516	3:28
9	3858	7164	55.3172	1:04
10	3896	7280	80.0999	1:11
μ σ				

IV. CONCLUSIONS

[15] In this paper, we investigate target tracking control for autonomous mobile robot navigation. Obstacle avoidance Collision is also considered for practical use and the algorithm for collision-avoidance can be easily augmented to the proposed target-tracking controller.

Furthermore, we have learned the fuzzy control system with door traversal method which helped to robot overcome the door problem. In addition, the subsumption behavior coordination help to run FSM-based method.

ACKNOWLEDGMENTS

The Lecturer would like to thank Dr Theo Theodoridis for the carefully guide us. We also thank very much classmates of the University of Salford for their useful suggestions and support of this project.

REFERENCES

[1] Roland siegwart and Illah R.Nourbakhsh, "An introduction of Autonomous Robots", Mit Press, 2004.

[2] N.Sariff; N. Buniyamin, *An overview of Autonomous mobile robot planning algorithm,*IEEE publisher, Saha alam, Malaysia.2006.

[3] T.T.Hoand, D.T. Heip,P.M. Duong, "Proposal of Algorithms for Navigation and Obstacles Avoidance of Autonomous Mobile Robot,"University of engineering and technology Vietnam National University,Hanoi,, pp. 1-3.

[4] Faten Cherni: yassine Boutereaa; Chokri rekik, Autonomous mobile robot navigation algorithm for planning collision-free path, IEEE published, Amman, Jordan, 2015.

[5] Amitava chatterjee,Anjan rakshit, N.nirmal singh, "Mobile robot navigation" , Studies in computational intelligence book series(SCI , volume 455).

[6] Francisco Rubio, Francisco Valero, Carlos Llopis-Albert, "A review of mobile robots:concepts,methods and application", journals.sagepub, 2019.

[7] Xinxin Yang, Kezhong he, Muhe Guo, Bo Zhang, "An intelligent predictive control approach to path tracking problem of Autonomous mobile robot", Tsinghua University, Beijing, China, 1998.

[8] Amit Singhal, "issues in Autonomous mobile robot navigation", University of Rochester, 1991.

[9] Weihua Chen, Tie Zhang, " An indoor mobile robot navigation technique using odometry and electronic compass, journals.sagepub, 2017.

[10] J.A.M. Zaki, "Mobile robot positioning using odometry and ultrasonic sensors", Journal of Cybernetics and Informatics,volume 13, 2012, pp. 4-6

[11] F. Chenavier and J. Crowley, " Position Estimation for a Mobile Robot using Vision and Odometry ", Proc. of IEEE Inter. Conf. on Robotics and Automation , ICRA´92, Nice, France, May 12-14, pp2588-2593.

[12] Veomir Kunchev,lakhmi jain, Vladimir lvancevic, Anthony finn,"Path Planning and Obstacle Avoidance for Autonomous Mobile robots", Lecture notes in computer science book series(LNCS, Volume 4252).

[13] Masako Kumano, Akihisa ohya, "Obastacle avoidance of Autonomous mobile robot using stereo vision sensor, University of tsukuba, lbaraki,305-8573 Japan.

[14] Patrick Benavidez; Mo jamshidi, "Mobile robot navigation and target tracking system, IEEE published, Albuquerque,NM,USA,2011.

[15] Sung-on lee, young-jo cho, myung hwang-bo, " A stable target-tracking control for unicycle mobile robot, Cheongryang,seoul 130-650,korea,2000.

[16] C.Eberst, M.anderson and H.I. Christensen,"Vision-Based Door-Traversal for Autonomous Mobile Robots, IEEE published, Takamatsu,Japan, 2000.

[17] J.keneth Rosenblatt and David w.payton, " The subsumption Architecture for Mobile Robot Control",Malibu,California 90265.

DEMO:

Video file link: **https://drive.google.com/file/d/1Y4Cx1xjvOmTs4Td3DwjJliTl4sWjUBDz/view?usp=sharing**

Project File LInk: **https://drive.google.com/file/d/1KEc4JerHUk-MVW-OLB9KdgCiydBQNnuw/view?usp=sharing**

Google doc link: **https://docs.google.com/document/d/1NH90IgPQ-pRITk7wdusXqg4AucFv-p2Za-lPRnhpeno/edit?usp=sharing**