Real-Time Optimal Trajectory Correction (ROTC) for Autonomous Omnidirectional Robot

Noorfadzli Abdul Razak, Nor Hashim Mohd Arshad, Ramli bin Adnan, Norashikin M.Thamrin, Ng Kok Mun.

Universiti Teknologi Mara , Shah Alam , Selangor 40450 , Malaysia
noorfadzli@gmail.com

Abstract. This paper proposed a Real-Time Optimal Trajectory Correction (ROTC) algorithm designed to be applied for autonomous omnidirectional robot. It is programmed to work when a robot undergoes a deviation, an admissible trajectory correction path is generated for the robot rapidly returns to the route line. For the algorithm to do this, initially a deviation scheme is employed to sense deviation and formulates a vector consists of displacement and angle. Via the vector, an admissible correction path is originated utilizing Hermite cubic spline method fused with time and tangent transformation schemes. A Dead Reckoning (DR) technique is applied for robot to pursue the path. Several experiments are arranged to evaluate the reliability of robot navigation with and without the algorithm. It motion is mapped in Graphical User Interface (GUI) window using data from Laser Range Finder (LRF) sensors as attached to the robot controller. Using the map, the performances of the algorithm are evaluated in terms of distance travel and duration to return on the line. The results signify robot navigation with the algorithm required shorter distance and duration as compared to robot navigation without ROTC. Thus, it justifies the algorithm is feasible in the navigation system where it can assist robot effectively to move back to the route line after experiencing a deviation caused by a disturbance.

Keywords: Deviation, Line Tracking Technique, Autonomous Control, Omnidirectional Robot; Trajectory Correction Path.

1 Introduction

Applying omnidirectional robot to handle logistic tasks such as transport, sorting, delivery of products and others in industry becomes relevant nowadays. Implementing autonomous navigation such as line tracking technique either using several photo sensors or a camera increases the robot usability to perform the tasks, especially in tight areas. However, there is a major drawback encountered by this robot where when it diverges from route line due to disturbance, it suffers difficulty to navigate back to the line. The disturbance in this situation is referred to false reading from sensors or camera when tracking the line or may result from robot motion itself performing obstacle avoidance. There are several researchers did propose a corrective motion technique which can be employed to enable the robot to return to the route

R. Alfred et al. (eds.), *Computational Science and Technology*, Lecture Notes in Electrical Engineering 481, https://doi.org/10.1007/978-981-13-2622-6_27

line. Such as Seiler et al [1] proposed a motion technique using Lie group of symmetries. Pham et al [2] presents corrective motion algorithm based on affine transformation. For Sprunk et al [3], a kino-dynamic trajectory generation technique is suggested. Meanwhile, Künemun et al [4] designed a fast and accurate generation of cubic spline trajectories.

As for this research, their works become an inspiration for us to introduce a new technique that able to perform the similar action. Hence, a novel algorithm name as ROTC is presented. This algorithm is designed for situation when the robot diverges from route line because of disturbance, then it will generate an admissible trajectory path for the robot to pursue in order to return to the line. Admissible path refers a closest path to the line and can be pursued by robot where it must not exceed its maximum velocity. Deviation vector comprises of distance, and angle is used by the algorithm to yield the said path. It will be a cubic type as applied by Zhou et al [5] and Hermite interpolation technique is fused in the algorithm to yield it. This type of path is selected since it can preserve the robot navigation momentum and evades backlash to the drive mechanisms. Meanwhile, using Hermite technique allows the path easily can be produced using two points and tangent vectors. Special transformation schemes namely, tangent and time are integrated in the algorithm. For a tangent transformation scheme, it used to alter the target tangent vector so that the correction path becomes near to route line. In the meantime, time transformation scheme will ensure velocity of each waypoint on the path can be pursued by the robot. Once the admissible path is obtained, a DR technique is employed to move the robot to each waypoint on the path. The robot is tested in three deviation acceleration categories to justify the algorithm dynamic functionality. Comparative performance is carried where robot navigation with and without the ROTC algorithm are evaluated.

The process implemented to develop and evaluates the ROTC algorithm performance are described throughout this paper according to arrangement as follows. The workflow of the ROTC algorithm is explained in Section 2. Next, methodologies perform in the research are particularized in Section 3. Section 4 elaborates the experiment setup, and approach arranges to evaluate the ROTC performance. In Section 5, it will present and discuss outcomes gained from the experiments. At last, Section 6 concludes the overall works in this research.

2 Design Concept

2.1 ROTC Workflow

Fig.1 demonstrates the pictorial diagram to illustrate the ROTC algorithm workflow to generate an admissible trajectory correction path when an autonomous robot experiences a deviation. According to Fig.1, an omnidirectional robot equipped with line tracking technique via a camera is arranged to navigate and pursuing the line. In the event, a disturbance causes the robot to diverge from the route; a deviation vector is estimated using an approach as introduced by Razak et al [6]. At that moment, ROTC algorithm is activated and begins to generate an initial linear trajectory path

based on deviation vector, $\vec{\sigma}$ that comprises displacement, s_σ and angle θ_σ. Then, there are two scheme are designed namely the tangent and time transformations to correct the path to become admissible.

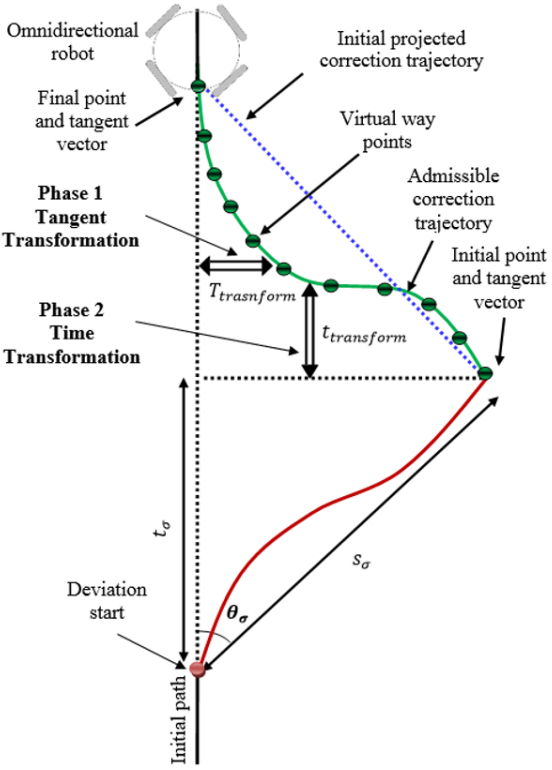

Fig. 1. ROTC workflow

In the tangent transformation phase, the algorithm will adaptively adjust the trajectory final tangent vector. The tangent is varied until the resulting trajectory achieves maximum proximity to the original route. At the same time, the series of points along the trajectory must be ensured ascending to each other. This is for attaining the continuity and smooth transitions throughout the vehicle maneuvers. Once the first scheme establishes the correct tangent for the trajectory, the second scheme which involves time transformation will commence. The average velocity at each segment of two points along the trajectory is computed. Then it will be validated with maximum velocity that allowable to be enforced by the robot. This maximum velocity is theoretically calculated based on the specification of motors employed in the vehicle system. If one of the segments has velocity exceeds the limitation, therefore, the trajectory yielded from first phase becomes invalid. Hence, automatically the algorithm makes an adjustment on the time navigation. At this stage, the time navigation will be increased. Simultaneously, the first scheme is activated again. This process will recur-

rently be executed until both schemes able to identify the best final tangent vector and optimal navigation time for the trajectory to lead back the vehicle to original route.
At last, the trajectory correction that is expressed as admissible produced. The average velocity at each segment of the trajectory is also secured. Alongside, via the velocity, the navigation course or angle at the segment can be formulated. Hence, with the data, it is now conceivable to activate following framework sequence namely DR navigation technique. This technique is chosen because the technique is simple and practicable enough for directing the robot to pursue on the generated path. Therefore, the robot will be moved until the final point of trajectory. At the same time, once the camera able to sense the route line back, then navigation via line tracking is activated again, and all the RATC sequences will be reset. The mentioned sequences and process will continuously run every time the robot experienced a deviation caused by the disturbance. This is also applicable either deviation happened to the left or right from the route line.

3 Methodology

This section reveals the process to design the proposed ROTC algorithm. It includes on how the trajectory is developed using a cubic Hermite interpolation method along with respective equations. In addition, the ways the cubic trajectory path is corrected to be admissible by tangent and time transformations are clarified. Then, a custom-made controller utilized to implement the algorithm, line tracking and DR navigation techniques is exposed. This goes along with devices installed on it. Afterward, the section confers a customize robot as an experimental platform, and its control model. At last, a GUI window employed for data acquisition and maps robot motion are presented.

3.1 ROTC algorithm

Once the algorithm activated, it begins to project a linear virtual path, σ_{lin}. This path is projected inversely relative to last location of vehicle deviation toward to original navigation route. In time, this location becomes an initial navigation point, p_{init} for the vehicle to maneuver back on the route. The displacement, s_{lin} of the σlin have similar values of s_σ as recorded by the deviation scheme. The same thing also happened to the angle, θ_{lin} of the σ_{lin}, where the amount is set equally to deviation angle, θ_σ. However, the direction of the angle at this moment is overturned. The σ_{lin} is deliberately yielded to fix the 2D maximum permissible position, p_{max} on the line route for the vehicle to enter. Likewise, this position will become the target location, p_{trgt} for the vehicle to pursue. Therefore, it is essential to determine the target location, in XY coordinate system and listed are formulas utilized.

$$p_{y\_trgt} = s_\sigma \sin\theta_\sigma \tag{1}$$

$$p_{x\_trgt} = s_\sigma \cos\theta_\sigma \tag{2}$$

Hence, p_{init} and p_{trgt} for robot navigation in the 2D coordinate system at this instant are recognized. Next, the action will be executed by the algorithm is formulating an admissible trajectory correction path. To do this, since the research does fix the path will be in form of cubic, therefore, it equations in 2D coordinate as follows:

$$p_y(t) = a_{y3}t^3 + a_{y2}t^2 + a_{y1}t + a_{y0} \tag{3}$$

$$p_x(t) = a_{x3}t^3 + a_{x2}t^2 + a_{x1}t + a_{x0} \tag{4}$$

The t in the equations is concerned with the time variable. Meanwhile, a_3, a_2, a_1 and a_0 are the coefficients for the equations. These coefficients must be resolved since they will influence the form of the path. Coefficients for $p_y(t)$ will be resolved first and at the same time, similar ways are implemented to find coefficients for $p_x(t)$. First, a_0 in Equation 1 will be identified. In Fig.1, it is shown that the initial point of the trajectory path is p_{init}. Correspondingly, the $p_y(t)$ in Equation 1 turns out to be $p_{y\_init}(t)$, where the value is set to be zero. Furthermore, at that point, t is 0 second. Hence, substituting the $p_{y\_init}(t)$ and t in Equation 3 causes a_{y0} established as listed below:

$$a_{y0} = p_{y\_init}(0) = 0 \tag{5}$$

Next, to find a_{y1}, Equation 1 is differentiated and result in following equation is obtained.

$$p_y{}'(t) = 3a_{y3}t^2 + 2a_{y2}t^1 + a_{y1} \tag{6}$$

This equation actually presents a tangent vector denoted as m_x for the path in X-axis domain. Via this equation, the a_{x1} now can be identified via following means. The initial tangent vector, m_{y_init} is formulated by substituting t as 0. This is done because it is located at p_{init}. As a result, a_{y1} becomes as shown:

$$a_{y1} = p_y{}'(0) = m_{y\_init} \tag{7}$$

To this extent, only a_{y2} and a_{y3} are still unresolved. In order to determine them, it is necessary to obtain the target point, $p_{y\_trgt}(t)$ and target tangent vector, m_{y_trgt} equations. To do that, it is assumed the robot arrived at *ptrgt,* at 1 second, which make t can be set as 1. Therefore, substituting the t along with a_{y1} obtained previously in Equations 3 and 6 give the desired equations appear as follows:

$$p_{y\_trgt}(1) = a_{y3} + a_{y2} + p_y{}'(0) \tag{8}$$

$$m_{y\_trgt} = p_y{}'(1) = 3a_{y3} + 2a_{y2} + p_y{}'(0) \tag{9}$$

Noticeably, Equations 8 and 9 turn to be in form of linear equations. This permits matrix manipulation method can be employed to find a_{y2} and a_{y3}. Applying the said method gives both coefficients are recognized as stated:

$$a_{y2} = 3p_{y\_trgt}(1) - 2p_y{}'(0) - p_y{}'(1) \tag{10}$$

$$a_{y3} = -2p_{y\_trgt}(1) + p_y{}'(0) + p_y{}'(1) \tag{11}$$

All the coefficients at present are established, hence substituting them in Equation 3 results in the cubic trajectory path in Y-axis domain can be finalized as follows:

$$p_y(t) = \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ t^0 \end{bmatrix}^T \begin{bmatrix} -2p_{y\_trgt}(1) + p_y'(0) + p_y'(1) \\ 3p_{y\_trgt}(1) - 2p_y'(0) - p_y'(1) \\ p_y'(0) \\ 0 \end{bmatrix} \tag{12}$$

Meanwhile, similar approaches above are utilized to recognize coefficients for trajectory path in X-axis domain. Therefore, this path equation is presented in Equation 13.

$$p_x(t) = \begin{bmatrix} t^3 \\ t^2 \\ t^1 \\ t^0 \end{bmatrix}^T \begin{bmatrix} -2p_{x\_trgt}(1) + p_x'(0) + p_x'(1) \\ 3p_{x\_trgt}(1) - 2p_x'(0) - p_x'(1) \\ p_x'(0) \\ 0 \end{bmatrix} \tag{13}$$

At this level, the path equations in Y and X axis domains with respect to *t* are obtained. Now, it is organized; these paths comprise numbers of waypoints labeled as *n*. Correspondingly, each *n* can be computed via these equations by setting the time as $t \in \left(\frac{t_{nav}}{10}\right) n$ where n $\in \{0,1,2 \cdots, 10\}$. Giving that the p_{init}, p_{trgt} and m_{init} are known, hence leaving m_{trgt} becomes indefinite. In concern about this matter, the algorithm has introduced tangent and time transformation schemes to find the correct for m_{trgt}. This is to ensure the path generated will near to line route and can be pursued by robot at optimum velocity. The flowchart in Fig.2 demonstrates the workflow of the schemes.

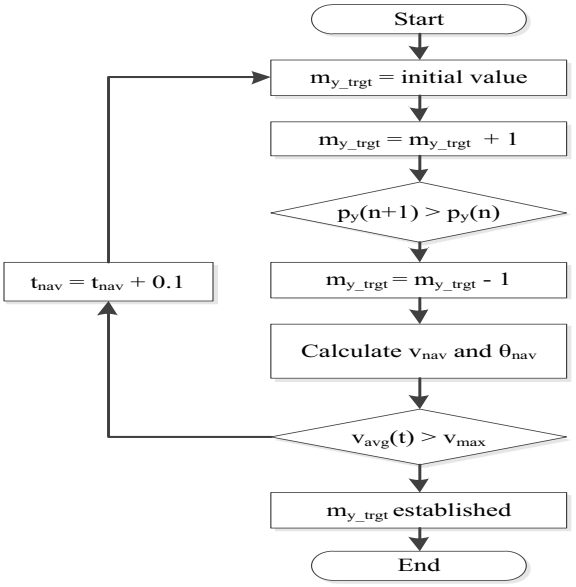

Fig. 2. Workflow of tangent and time transformation scheme

In the tangent transformation scheme, a preliminary value for m_{trgt_y} is assigned. Then it will be elevated one causes the path form to change. At moment, each position of n on the path will be reviewed. The review process must obey the rule where every post point must be more than prior point. If the rule obeyed, than m_{trgt_y} will be increased one. The process continuously runs until the rule is defied. Once happened, the current m_{trgt_y} value will be minus by one since the previous value is actually the optimum m_{trgt_y} that causes the path nearer to line. Next, time transformation scheme is activated. The average velocity, v_{avg} between each segment of waypoints on the X and Y axis paths are calculated using Equation 14 and 15

$$v_{y_{avg}}(t) = \frac{p_{y[n+1]}\,(t+0.1) - p_{yn}\,(t)}{(t+0.1)-t} \tag{14}$$

$$v_{x_{avg}}(t) = \frac{p_{x[n+1]}\,(t+0.1) - p_{xn}\,(t)}{(t+0.1)-t} \tag{15}$$

Respectively, the average velocity for each segment on 2D path and also signified as navigation velocity, v_{nav} can be obtained with subsequent equation:

$$v_{nav}(t) = \sqrt{\left(v_{xn\_avg}\,(t)\right)^2 + \left(v_{yn\_avg}\,(t)\right)^2} \tag{16}$$

Meanwhile, the navigation angle, θ_{nav} of the segments are also computed via following equation

$$\theta_{nav}(t) = tan^{-1}\left[\frac{v_{yn_{avg}}(t)}{v_{xn_{avg}}(t)}\right] \tag{17}$$

Next the scheme will check either the v_{avg} of each segment is less than v_{max}. If yes, then the m_{trgt_y} value is established and the path generated is optimum for robot to return to the lone. Contrary, if not, subsequently the time is increased by 0.1s. Tangent transformation scheme is started again, and this process continuously runs until m_{trgt_y} value that ensures velocity at each segment less than v_{max} is gained.

3.2 Robot control model and navigation techniques

A custom-made omnidirectional robot using four wheels is built to be employed as an experiment platform. Acrylic sheet and aluminum bar is used as robot chassis. The chassis has dimension of 30 cm height and 60 cm for it width and long. The robot is driven using four 7.2V DC motors and every motor coupled with 2 inches transwheel. Overall robot weight is 3.2 kg. Meanwhile to enable the robot to move according to navigation inputs namely v_{nav} and θ_{nav}, a control model for the robot is required. To obtain the model, Fig.3 shows a free body diagram of the omnidirectional robot. According to the diagram, symbol O as located at the chassis center is meant for robot gravity. Thus, robot local coordinate in X and Y axis becomes X_LOY_L. As for robot global coordinate, it turns to be X_GOY_G. The α_i is angle between the axles of the wheels relative to X_LOY_L. Character i is related to the number of wheel used by the robot. Since the robot used four wheels hence it is fixed to $\alpha_1 = 90^0$, $\alpha_2 = 180^0$, $\alpha_3 =$

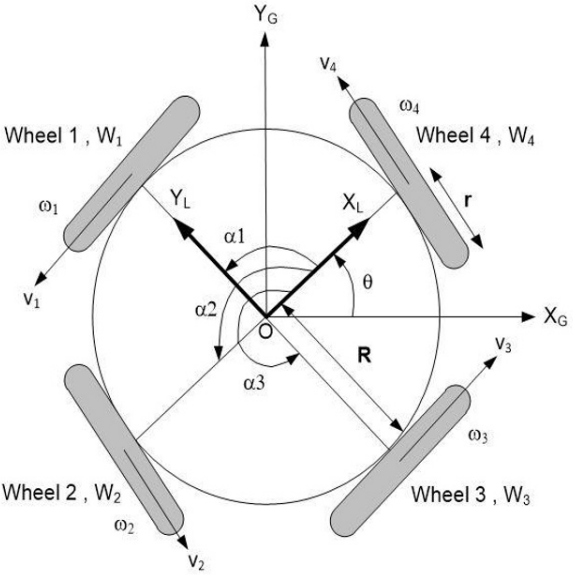

Fig. 3. Omnidirectional robot free body diagram

270^0 and $\alpha_4 = 0^0$. The ω_i is denoted for the angular velocity of each wheel, and v_i signifies the direction of linear velocity of the center of the wheel with respect to X_LOY_L. This robot has a chassis radius, R of 30 cm and radius of the wheel, r is 2.54 cm. Meanwhile, θ appears to be angle between the coordinates X_LOY_L and X_GOY_G. Hence, via the diagram and applying the methods as carried out by Alves et al [7], the control model for the robot is like so:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}-\sin(\theta+\propto_1) & \cos(\theta+\propto_1) & R\\ -\sin(\theta+\propto_2) & \cos(\theta+\propto_2) & R\\ -\sin(\theta+\propto_3) & \cos(\theta+\propto_3) & R\\ -\sin(\theta+\propto_4) & \cos(\theta+\propto_4) & R\end{bmatrix}\begin{bmatrix}V_{Gx}\\ V_{Gy}\\ \dot{\theta}\end{bmatrix} \tag{18}$$

In this model, the V_{Gx} and V_{Gy} represent velocities of X and Y axis in X_GOY_G. However, to appropriate steers the robot, it is best to translate V_{Gx} and V_{Gy} into velocities in X_LOY_L. This can be done via Equation 19 to 21.

$$\begin{bmatrix}V_{Gx}\\ V_{Gy}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos(\theta) & 0 & 0\\ 0 & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}V_{Lx}\\ V_{Ly}\\ \dot{\theta}\end{bmatrix} \tag{19}$$

where:

$$V_{Lx} = V_{nav}\cos\theta_{nav} \tag{20}$$

$$V_{Ly} = V_{nav}\sin\theta_{nav} \tag{21}$$

With these equations, it is possible for the model to receive v_{nav} and θ_{nav} and then determine correct ω_i so that robot can move to desired direction and velocity. In the meantime, the robot is arranged to perform two navigation techniques line tracking and Dead Reckoning (DR). To enable the robot to navigate via line tracking technique, an analog camera is utilized. Camera will capture the line image and then a selective video line technique designed by Arshad et al [8] to track the line. The technique will produce digital values represent the location of the line. Via the values, robot is programmed to move at constant v_{nav} and θ_{nav} is regulated so that robot steers in proportion to the line. As for navigation via DR technique, it is used to steer the robot to move along the correction path. Each segment of waypoints on the path has it own navigation inputs namely v_{nav} and θ_{nav} as calculated via Equations 16 and 17. Therefore, a timer interrupt is exploited by the controller to supply these inputs to the control model according to interval t. By doing this robot is ensured to perform the desired motion to follow the path.

3.3 Controller board and GUI window

To execute the ROTC algorithm and autonomous navigation of the robot, a customize controller board based on PIC32MX795F512L microcontroller is built. The board is equipped with an accelerometer, a camera, four units of 10A motor driver, two units of LRF sensors and a XBee 2.4GHz module. All of these devices are needed so that the desired functions can be performed. Like an accelerometer, it is meant for deviation scheme to compute the vector. A camera is employed for line tracking technique. Motor driver is used to drive each motor according to respective ω_i . Temporarily, the LRF sensors are used to provide 2D distance data. These data are then transmitted wirelessly to a custom-made GUI window via the XBee device. The GUI window is developed using Microsoft Visual Basic 2015 and installed in a PC. A same XBee module installed on the controller board is plugged to a PC. Hence, the distance data received are exploited to map the robot motion. The motion includes robot motion when autonomously pursuing the line route, deviations that happened and navigates on the trajectory correction path to return to the route. Meanwhile, the controller and robot itself are powered by 11.1V, 2.2A Lithium Polymer (Li-Po) battery.

4 Experimental setup and data validation

An indoor controlled environment field as shown in Fig.4 is utilized to test the proposed algorithm performance. The field is built using a grey rubber mat where the dimension is 6 meters long and 3 meter width. Additionally, a white line with 3cm width is taped at the center of the mat. With this setup, it is possible for a camera to track the line and enable the robot to perform autonomous navigation. Meanwhile, there are few wood blocks positioned within the field. They are utilized as object to reflect laser beams from LRF sensors to gain 2D distance data. These data are neces-

sary to map robot motion trail when moves along the line. Consequently, the map is used for evaluation purpose afterward.

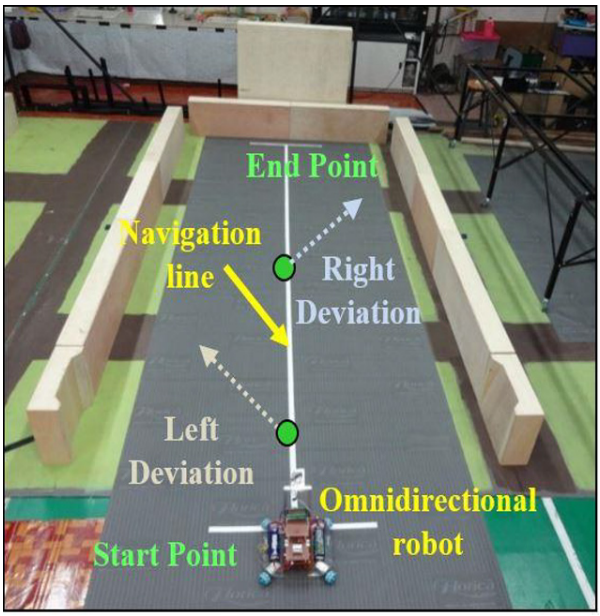

Fig. 4. Experiment field

The trials begin by positioning the robot at the start point of navigation line. Then, robot is activated to move at a constant speed. At a distance from the start point, the robot is pulled via a string to make it deviate to the left from the route line. Doing this action actually simulates the same effect when the robot suffers deviation due to disturbance in real practice. At that moment, the process where line tracking technique is disabled, a deviation scheme will sense deviation and computes the vector; ROTC generates the path and finally, DR technique moves the robot to return to the line are monitored. Concurrently, the respective data are captured in GUI window. As the robot able to return to the line, it is given a time to move steadily on the line. Then the string is pulled once more to force the robot to diverge to the right from the line path. Similar robot process as mentioned above is monitored again and data are captured in GUI window. The experiment is stopped once the robot reaches the end position on the line route.

Subsequently, the experiment is performed again whereas at this time, robot autonomous navigation without the algorithm is implemented. The experiment is not fully executed as applied in experiment before. The vehicle will be moved manually at end position of deviation. The position is easily tracked by using LRF modules. The navigation path before and during deviation occurred in prior experiment is presumed to be similar for this navigation. This goes along with the duration of the navigation. Next, the vehicle is activated again to move and senses the route via line tracking technique. Simultaneously, navigation time is resumed and the GUI maps the naviga-

tion path. The vehicle is programmed to stop once the camera able to track the line of route. Once it stops, the motion path is unified with the navigation path as recorded earlier. As for the duration of the navigation, it will be added to time as recorded before. In the meantime, these steps will be repeated for the situation of second deviation that occurred toward the right of the route.

Accordingly, the methods as explained above is decided since the methods are the best way to observe and compare the path navigated by vehicle with and without ROTC algorithm. Particularly, to ensure both navigation systems experience the same deviation distance and direction. Meanwhile, this experiment has been set up to run in three different categories of deviation acceleration. The categories are high, moderate and low having the range of acceleration as specified in Table 1. Such setup is arranged to test the dynamic operation of the designed algorithm to perform in different acceleration of deviation. To validate the algorithm performance, the map that plots robot navigation with and without the algorithm is examined. Via the map, analytical evaluation can be carried out to differentiate performance of robot navigation with and without the algorithm in terms of distance and time taken to return to the original path after experiencing deviation.

Table 1. Deviation acceleration categories

Category	Deviation acceleration , a (ms^{-2})
High	$a > 10$
Moderate	$5 \leq a \leq 10$
Low	$a < 5$

5 Results and Discussion

Fig. 5 to Fig. 7 show the maps that plots robot motion either it autonomous navigation is integrated with and without ROTC algorithm in all deviation acceleration categories. Apparently as seen in the figures, robot with autonomous navigation equipped with ROTC algorithm performs well when maneuvers on the route in all the categories. Even the robot endures deviation either to the left or right, somehow the algorithm guarantees the robot to return nearly to the original line Moreover, it also can be realized, the distance taken for the robot where it navigation fused with ROTC algorithm appears to be shorter. Consequently, robot consumes lesser time to return to the line. This achievement is compared to robot navigation without the algorithm where the distance and duration turn out to be longer. All of above declarations actually justifies decisively according to data as summarized in Table 2. Meanwhile, the table also reveals the performance of robot navigation with the algorithm becomes deteriorated from high to medium and subsequently to low category. The duration for the robot to return to the line appears to increase slightly from one category to another. This goes along with the distance traveled by the robot. This matter happened because of inaccuracy of deviation scheme to estimate the vector. According to Razak et al [6], the accuracy of the deviation scheme that designed by them becomes de-

clined as the deviation acceleration decrease. Due to this drawback, the correction path generated by the ROTC algorithm actually doesn't reach the line route.

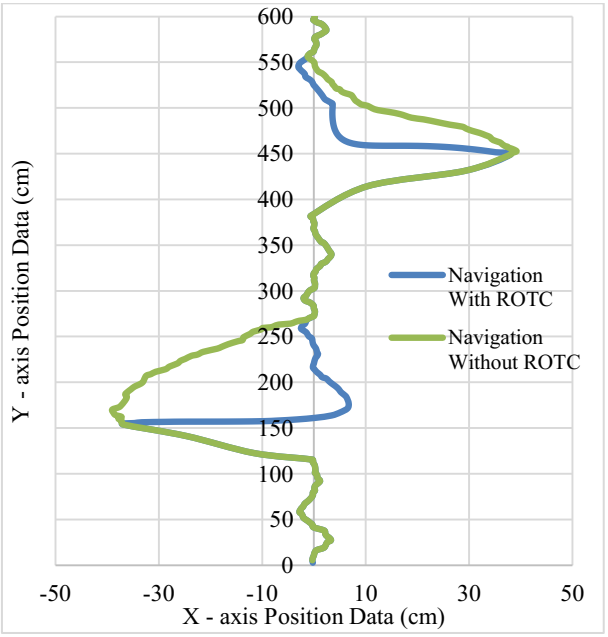

Fig. 5. Robot motion in high acceleration category

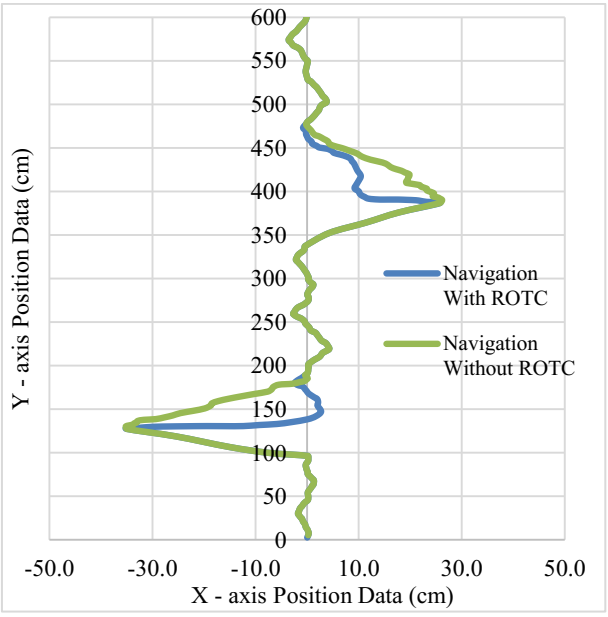

Fig. 6. Robot motion in moderate acceleration category

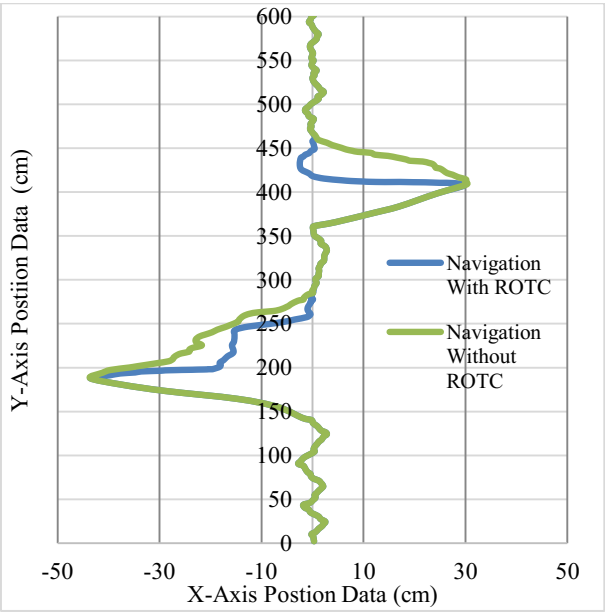

Fig. 7. Robot motion in low acceleration category

Table 2. Results for robot navigation with and without ROTC

Deviation		Navigation Time (s)		Navigation Distance (cm)	
Category	Direction	With ROTC	Without ROTC	With ROTC	Without ROTC
High	Left	2.1	2.7	81.0	89.3
	Right	2.5	3.0	79.5	82.5
Moderate	Left	2.9	4.3	85.3	95.4
	Right	3.3	4.0	87.3	91.4
Low	Left	3.8	4.8	94.0	110.3
	Right	3.7	4.7	97.4	113.5

This causes when the robot arrives at end location of the correction path, more time is needed for the camera to track the line. Therefore, robot navigates further to reach the line again and result in the duration turn out to be increased. Nevertheless, at moment, the research does justify with these achievements, the usability of the ROTC algorithm to assist robot navigation to return to lined route after experienced is practically relevant.

6 Conclusion

A ROTC algorithm designed using Hermite Spline Interpolation technique is established. Integrated with tangent and time transformation schemes enables it to generate

an admissible trajectory path. Using DR technique, robot can pursue the path to return rapidly to the lined route after experienced deviation due to disturbance. Three categories of deviation acceleration are carried out to test algorithm performance applied to an autonomous omnidirectional robot. It performances are compared with robot navigation without the algorithm. The outcomes do reveal, the dynamic practicality of the algorithm to work in different deviation acceleration. Moreover, the results prove robot navigation with ROTC performs better where the distance and duration for robot to return to route line shorter compared to robot navigation without it. Even there is performance deterioration of the robot when experimented in high, medium and then low category, however, this achievement is accepted since via the algorithm robot still able to return the lined route.

Acknowledgment

This research is supported by Ministry of Education fund (600-RMI/RAGS 5/3 (42/2015)). The authors would like to express appreciation to Faculty of Electrical Engineering and Institute of Research Management and Innovation, Universiti Teknologi Mara (UiTM) for supplying the financial supports and equipment to conduct this research.

References

1. Seiler, K. M., Singh, S. P., Sukkarieh, S., Durrant-Whyte, H.: Using Lie group symmetries for fast corrective motion planning. The International Journal of Robotics Research 31(2), pp. 1-16 (2012).
2. Pham, Q. C.: Fast trajectory correction for nonholonomic mobile robots using affine transformations. In Robotics: Science and Systems VII, pp. 265-272 (2012).
3. Sprunk, C., Lau, B., Pfaffz, P., Burgard, W.: Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In IEEE International Conference on Robotics and Automation (ICRA), pp. 72-77 (2011).
4. Künemund, F., Kirsch, C., Heß, D., Röhrig, C.: Fast and accurate trajectory generation for non-circular omnidirectional robots in industrial applications. In 7th German Conference on Proceedings of ROBOTIK, pp. 1-6 (2012).
5. Zhou, F., Song, B., Tian, G.: Bezier Curve Based Smooth Path Planning for Mobile Robot. Journal of Information &Computational Science 8(12), pp. 2441-2450 (2011).
6. Razak, N. A., Arshad, N. H. M., Adnan, R., Misnan, M. F., Thamrin, N. M., Mahmud, S. F.: A Real-Time deviation detection and vector measurement technique for straight line quadrocopter navigation using accelerometer. In IEEE 5th Control and System Graduate Research Colloquium (ICSGRC), pp. 69-74 (2014).
7. Alves, S. F., Rosario, J. M., Ferasoli Filho, H., Rincon, L. K., Yamasaki, R. A.: Conceptual bases of robot navigation modeling, control and applications. In Advances in Robot Navigation (2011).
8. Arshad, N. M., & Razak, N. A.: Vision-based detection technique for effective line-tracking autonomous vehicle. In IEEE 8th International Colloquium on Signal Processing and its Applications (CSPA), pp. 441-445 (2012).