The suggestion method for self-localization by using intelligent graphs

Mohammad Hossein Saeidinezhad1, Bahareh Karamsichani2, Ehsan Movahedi3
1Islamic Azad University, Najafabad Branch, Saidinezhad@yahoo.com
2Islamic Azad University, Najafabad Branch, Branch.karamsichani@yahoo.com
3Islamic Azad University science and research Branch, eh.movahedi@yahoo.com

Abstract: One of subjects that always has been considered about artificial intelligence is find the shortest direction to destination in space with several obstacles. An effective algorithm in this connection should overcome problems such as time-consuming of calculation and uncertainty of answer. In relation to this, in this article we present an algorithm and try to change the real environment to a graph and then can find the shortest direction from source to destination by analyzing that graph. The proposed algorithm act immediately, and solve the problems such as the manner of encounter with co-potential points. This algorithm can find the most constant direction in dynamic environment by estimating the obstacles movement too and by making unreal obstacles in places that their distance to unreal obstacles is less than certain amount, exit them from possible selections domain. The direction have been chosen by use of knots that calculate in the graph and one graph can connect to other if no obstacle or unreal obstacle don’t cut off their connecting mane. In addition to desirable accuracy which is about 95%, this algorithm able to act in motion environments regularly and with the least number of knot.

Keywords: self-localization, robot self localization, intelligent graph, optimal direction, estimation of direction

1. Introduction

One main problem in automatic of robots is finding the desirable direction between the primary and last place of robot that also prevent it to encounter with obstacles, which called self localization of the robot. There are several methods for solving this problem such as using artificial intelligence-based methods, and geometrical estimation which act based on the programmed principles. Time complexity of the geometrical solutions is proportional with robot degree of freedom. Therefore, the geometrical methods generally are practical when the degree of robot freedom is low. This fact results in the exploratory methods which is based on several artificial intelligence techniques. These methods have desirable speed but do not guarantee the self-localization.

Another method for self-localization is using polyrobot realist topography. Many of available methods act like sonars but at the same time try to perform the information transferring to comprehensible framework of user[1] and use of structural relation of information too[2,5,7]. For example, for this object callman filter have been used in[3,6].

In moving robot, some researchers have considered the help and assistance in self-localization. To do that, different sensors have been used such as ordinary cameras, infrared and thermal cameras[8]. For example, a pair of robots equipped with sighting system with active store and light disting wishing have been used in[4] for tracing one robot by another. There are several sensors of this type available in the high tech market.

For example some of them are able to evaluate the distance between two robots accurately and some others are able to evaluate the relative direction of the observed robot relative to observer robot and some even are able to estimate the accurate direction of the observed robot[8,4].

2. The proposed algorithm

Now, we consider the phases of self localization algorithm which have been proposed and it’s executing results.

2-1. obtain the information from environment

We assume that we can obtain all the necessary information about environment, obstacles and destination. These information include the accurate place of obstacles and destination too. In dynamic environments, the information about speed and direction of robot movement will add too. Therefore, we suppose the operation environment of robot will be a dynamic observable environment.
2-2. establish the virtual obstacles

virtual obstacles are obstacles that set at impossible places. These places are specified by calculation of the distance between adjacent obstacles in the manner that if this distance will be less than certain amount, that place will obstruct so don’t establish a direction by mistake and increase the accuracy of algorithm. Figure 1, shows the profile of unreal obstacle.

![Figure 1: establish the unreal obstacle](image1)

The result of execution this phase of algorithm show in figure 2. Just as see in figure, because the distance between knots (15,4), (10,3), (0,2), (11,1), (12,8), (13,7), (6,14) is less than cartion threshold, it obstruct by unreal obstacle so robot don’t try to passing between them by mistake. This threshold is estimatable according to robot dimensions.

![Figure 2: obstruct the space by unreal obstacle](image2)

obtain the optimal direction. In this connection, first draw a line from starting point to destination so obtain the angle between this line relative to horizontal line. This angle specify in figure 3.

![Figure 3: calculating the knot](image3)

Then draw a line perpendicular to the line between starting point and destination and obtain angle b which is angle between vertical line and horizontal axis, by angle a.

Now, if we can place an ellipse, which it’s large diameter’s angle is equal to a and it’s small diameter’s angle is equal to b, analog with center of one obstacle in the manner that don’t encounter with no another obstacle, that place in our algorithm suppose as a knot.

After executing this phase of algorithm, the results such as figure 4 observed which in this figure, the white knots are knots that calculated as measurement graph knots.

![Figure 4: knots of direction](image4)

2-3. calculation of graph

knots of direction this phase of algorithm is very important because in fact by specify the knots, travelling direction of robot will determine. As a result, we should choose the place of knot appropriately and optimally so
This algorithm in static state have ability to find the direction in conditions that obstacles are in bar-shape. It do this in this manner that consider the beginning and end of bar-shape obstacle as two usual obstacle and behave them like point obstacles. In this state, no direction have permission to passing over this bar-shape obstacle. Figure 5 shows a sample of bar-shape obstacle and the formation of knot around these obstacles.

2-4. determine the adjacent knots of each knot in the course of previous stages we established one graph from probable measurement direction. Know, we connect a mane from each knot to another knots to determine the adjacent knots of each knot but on the condition that one knot of graph can connect to another knot in the manner that their connected mane haven’t been interrupt by any obstacle or unreal obstacle. As a result, for determine the adjacent knots, we begin from one knot and connect a mane to another knots. If this mane don’t interrupt by obstacles in the course of direction, above-mention mane records and that knot records as adjacent knot of beginning knot.

Figure 6 show the execution results of algorithm in static environment such as bar-shape and point obstacles. Jus as this figure, the obtained direction don’t passing over any obstacle.
Figure 8 shows the adjacent knots that obtain for our example algorithm graph.

![Figure 8: Manes from one knot to adjacent knots](image)

2-5. **find the best direction**

Each knot have information such as distance to adjacent knot and distance to destination. All first, amount of distance to destination give primary quantity for each knot by a too much quantity. Then add the source and destination to the knots list for finding the best direction and start of destination, place it’s adjacent knots in one mound because these knots connect to destination directly, so have the least distance to destination and this distance save at each knot. Then one knot exit from the first of mound and place adjacent knots in mound, for this knot too. As see in figure 9, for example if the distance between Ne to Na is more than total distance a+b, so this quantity have been considered as Ne distance to the destination and the knot that establish this condition, set as next knot of Nb at knot information. This process continues until reaching to source. As a result, the shortest distance to destination and next place that it should pass to reaching the destination, have been saved for each knot. This algorithm have similarity to direction passing algorithms which is base on the heuristics of direct line distance.

In environment that movement direction of obstacles is unforeseeable, the shortest direction isn’t always the safest direction. The direction that is permanent for much time, is the safest direction which in this algorithm, direction that have less compression of obstacles have been considered as safer direction. Of course, the expense of distance has basic role for finding the safest direction and as a result, we consider the function of distance expence as as following with attention to importance of direction length and security of direction at different conditions:

\[
(\text{distance} \times (1-\alpha) + (\text{security of distance} \times \alpha) = \text{the expense of direction})
\]

\(\alpha\) is a coefficient between 0 to 1 that specify the ratio of importance of direction’s security to distance. Naturally, if this coefficient will be 1, the distance haven’t role for finding the direction and the shortest direction obtain in according to least expense quantity.
2-5-1. The calculation method of security amount of each direction

The security calculate for maximum distance between two obstacles (D) and distances that are more than this amount don’t consider. This distance is determinable in according to environment and speed of obstacles movement. The amount of D can determine base on distance from source constantly or variably that as distance from source increase, so this amount will be more too. As a result between two obstacles that place in distance less than D amount from each other we giving weight in the manner of a linear function. The quantity of this weight calculate as following:

\[
D = \text{possible greatest distance} \\
\text{DIST} = \text{distance between two obstacles} \\
\text{MDL} = \text{DIST} / 2; \\
\text{Wd} = \text{calculating weight according to distance} \\
\text{If ( DIST < D )} \\
\quad \text{For (i=0;i<=DIST;i++)} \\
\quad \text{Wd} = (\text{abs}(i - \text{MDL})/\text{MDL}) \times (\sqrt{\text{SQR}(D) - \text{SQR}(\text{DIST}))});\\
\]

Just as observe above, we calculate the weight that should obtain according to distance of obstacles on weights surface and then move from point to another point, each place that weight of surface is more than zero in it, add up with variable that can it security of direction. If \( (w[x][y] > 0) \)

\[
\text{RE} = \text{RE} + w[x][y];
\]

\( x, y \) is the place that now are on it, \( w[x][y] \) is the correspondence weight to point \( x, y \) at weight surface. \( \text{RE} \) is security amount of direction that is zero at the first of each direction and save as safety coefficient of that direction at the end of a direction. A sample of giving weight have been see in figure 11.

Figure 11: The weights surface and giving weight to each point

If safety coefficient don’t consider, the calculating direction will be as figure 12 and if this coefficient consider, the direction calculate as figure 13.

Figure 12: Calculating the direction without considering the safety coefficient

Figure 13: Calculating the direction by considering the safety coefficient
2-6. estimate the movement for dynamic environment in proposed method, the estimation of movement have been performed on 3 basis:

1. calculate the movement of obstacles randomly and according to time.
2. suppose the perform of movement preventably for preventing of reaching robot to destination.
3. estimating the movement on the base of direction and speed of obstacles movement

One of the important principles that should consider in localization is find the reliable direction. this means that the shortest direction isn’t always the most reliable and most constant, competition of footballer robots, we always should select the shortest direction from the most constant direction. the constant direction is the direction that from beginning of movement until reach to destination is still reliable. For example, the movement of obstacles don’t prevent from their continuation. Therefore localization in dynamic environment is very important because of place-change of obstacles. In these environments, the conditions of take a decision is variable constantly therefor we can’t find the direction that will be constant until reach to destination. For solving this problem we try to estimate the place of obstacle at time that we reach to it and find the direction. for this object we need to speed of robot movement and direction of movement and speed of each obstacles too.

The stages of proposed algorithm for localization in dynamic environment is as following that a sample of it’s output have seen in figure 14.

1. Calculate the direction of obstacles movement that obtain by tracing the movement changes of each obstacles at successive frame
2. Calculate the speed of obstacles movement
3. Calculate the distance of each obstacles to robot
4. Calculate the time of reaching to obstacle according to robot speed

For obstacle 1 in Figure 14, the algorithm stages executed as are following:

1. \( X_1 \), The distance to obstacle 1
2. \( V_1 \), The speed of obstacle 1
3. \( V \), The speed of robot
4. \( T_t = \frac{x_1}{v} \) the time period that is necessary to reach obstacle 1
5. \( X_1 = v_1 * t_1 \) The amount of obstacle displace in direction of movement vector at time \( t_1 \)

After performing above stages for dynamic localization, we estimate the name place of moving obstacles. We can behave with list of new obstacles like static obstacles by use of these new places because the new place of obstacles in according to \( x_1, t_1 \) estimated accurately. Figure 15 shows the new place of moving obstacle after executing the above mentioned algorithm.
Conclusion

In this article, we proposed a novel a method for optimal localization of robots in dynamic and full of obstacles environments. The proposed method which act based on the modeling of the middle points of the path us the graph nodes, and move between these nodes, can solve the problem of passing between bar-shape obstacles. In addition, the proposed method have got the optimal localization of robots ability in dynamic environment with moving obstacles by estimating the movement direction of obstacles.

By successive tests, it is specified that the accuracy of finding the optimal direction in this method is about 95% and robot could find the optimum path in almost 93% of cases.

References


