An algorithm for Path planning with polygon obstacles avoidance based on the virtual circle tangents

Zahraa Y. Ibrahim
Electrical Engineering Department
University of Basrah
Basrah, Iraq
zahraaalkatrany@gmail.com

Abdulmuttalib T. Rashid
Electrical Engineering Department
University of Basrah
Basrah, Iraq
abdturky@gmail.com

Ali F. Marhoon
Electrical Engineering Department
University of Basrah
Basrah, Iraq
alifm60@gmail.com

Abstract: In this paper, a new algorithm called the virtual circle tangents is introduced for mobile robot navigation in an environment with polygonal shape obstacles. The algorithm relies on representing the polygonal shape obstacles by virtual circles, and then all the possible trajectories from source to target is constructed by computing the visible tangents between the robot and the virtual circle obstacles. A new method for searching the shortest path from source to target is suggested. Two states of the simulation are suggested, the first one is the off-line state and the other is the on-line state. The introduced method is compared with two other algorithms to study its performance.

Index Terms — Path planning, polygonal obstacles avoidance, randomized incremental algorithm, visible tangent graph.

I. INTRODUCTION

The fundamental thing that the robotics need to have are algorithms to convert the human orders to movement, these kinds of algorithms called path planning algorithms [1]. Path planning of mobile robot is one of the most challenges in robotics because it is applied in different applications such as routing transportation, organizing allocation of machines in factories, controlling robots, intelligent agents [2] and also in military applications [3].

The concept of path planning is very simple: the mobile robot starts from an initial position and has final destination need to be reached. Depending on how much information the robot is known about the environment path planning algorithms can be classified into two types: off-line and on-line path planning algorithms. In the off-line path planning algorithms the robot has global information about the environment that's mean the robot know the location, shape and size of each obstacle, therefore, the robot can know if the target can be reached or not in advance and the shortest path can be computed. In on-line path planning algorithms, a robot gets information about the environment from its sensors during its movement, therefore, this robot at start point does not know if the target can be reached or not and the shortest path not always can be reached [4].

Many path planning algorithms have been introduced and some of them use the principle of the artificial potential field. In these approaches, the obstacles generate repulsive force to the robot, while the target has an attractive force. This robot navigates in the environment under the influence of these two forces so, the sum of these two forces determine the speed and direction of that robot [5]. The artificial potential field simple to implement because of its simple mathematical structure , but it is not suitable in some situations: if the obstacle very close to the target then the repulsive force become more than the attractive force, therefore, the target cannot be reached this problem was called goals non-reachable [6]. Also, with obstacle nearby ,when the robot, obstacle, and target are aligned then the attractive force equal to the repulsive force and the sum of forces influence the robot equal to zero which make this robot stop [7]. Finally, when the robot is far away from the target, the attractive force very great which make this robot move near the obstacles then it may collide the obstacle [8]. Traditional artificial potential field method was improved using the wall-following mode to

overcome the problems mentioned above. The wall-following mode overcomes the problems, but the path length increases and there are some cases the robot fails to escape from the local minimum [9]. Wall-following method was improved to overcome the problems mentioned above and regression search algorithm was introduced to shorten the planned path, but the computing time still increases [10].

Other class of path planning methods based on Voronoi diagram which can be implemented either as off-line algorithm [11] or on-line algorithm [12]. In Voronoi diagram based algorithms the resultant path consists of line segments which make the robot stop at each line segment, change its direction and continue. Such movement causes power consumption to the robot. In order to overcome this problem and satisfy initial position and shortest path conditions two Bezier curves are used [13, 14].

The graph search algorithms are used to determine the shortest path by other path planning algorithms. In these algorithms, the first step is the construction of all the possible trajectories, then the graph search algorithm is applied to choose the shortest path. In a case of polygonal obstacles the graph constructed from the visible vertices. If the line connects these two vertices do not intersect any obstacle, then such graph is called the visibility graph [15, 16]. The visibility graph also can be applied to a circular environment in such case the vertices is the tangents point of the obstacles and the tangents lines are the edges [17]. The main drawback of visibility graph based algorithms is the time consumed searching for the shortest path and the large memory required to store the edges and vertices of visibility graph. These problems can be reduced by reducing the number of edges in the visibility graph as in tangent. The vertices of the tangent graph are the tangents points on obstacle boundaries, and its edges are the tangents of the obstacles or convex boundary segment between the tangent points [18]. There are many algorithms used for selecting the shortest path, one of them the Dijkstra's algorithm. This algorithm gives an optimum path when it is run to reduce the visibility graph [19].

This paper proposes a new simplified path planning algorithm. The proposed algorithm can be applied in an off-line and on-line scenario in an environment with polygonal shapes obstacles. The polygonal obstacles represented by virtual circular obstacles, after that the visible tangents graph constructed by computing the tangents of each collides obstacle. In order to find the shortest path from source to target, a new shortest path finding algorithm is suggested. The low-level trajectory planning is implemented by using the digital differential analyzer algorithm (DDA), which requires simple resources for implementation. The rest of this paper organized as follow. Section II describe the low-level path planning of the mobile robot and section III give a comprehensive view of the proposed algorithm. In section IV the proposed algorithm is compared with two other algorithms in two scenarios. Finally, section V deals with the conclusion of the paper.

II. LOW LEVEL OF PATH PLANNING

This section deals with the low level of path planning to aims the robot to follow a given path. This path is represented by the motion of the robot over a straight or curved lines using the digital differential analyzer algorithm (DDA). This algorithm is an incremental method and at each step has the result that comes from the preceding step. In DDA algorithm, the coordinate with higher change is increased by unit step and then the value of step in another coordinate is computed according to the procedure of the algorithm.

A. DDA line drawing

The DDA algorithm is faster than the direct line equation in determining the line pixels since it eliminates the number of multiplications by using the raster characteristics [20]. The calculated pixels positions may be drifting away from the true line path for long line segments because of the round off error in successive additions of the floating-point increment. Furthermore, the rounding operations and floating-point arithmetic in procedure line DDA are still time-consuming. The procedure for line drawing in DDA algorithm starts from the line equation:

$$y = m * x + b \quad (1)$$

Where m is the slope of the line and b is the step value. Let (x_1, y_1), (x_2, y_2) be the end points of the line shown in Fig.1

$$m = dy / dx = (y_2 - y_1) / (x_2 - x_1) \quad (2)$$

$$b = y_1 - m\, x_1 \quad (3)$$

Depending on the value of m we have two cases:

Case 1: *($m < 1$)*
When x increases more than y then dy / dx (m) is less than one. According to Fig.1 in such case the line is drawn using the following steps:

1. Draw the first point on the line (x_1, y_1) and let $x=x_1$ and $y=y_1$.
2. Compute the new values to x and y using the following equations:

$$x = x + 1 \quad (4)$$

$$y = y + m\, x \quad (5)$$

Since x increased by one then

$$y = y + m \quad (6)$$

3. Draw the new point (x, round(y))
4. if $x \neq x_2$ then go to step 2

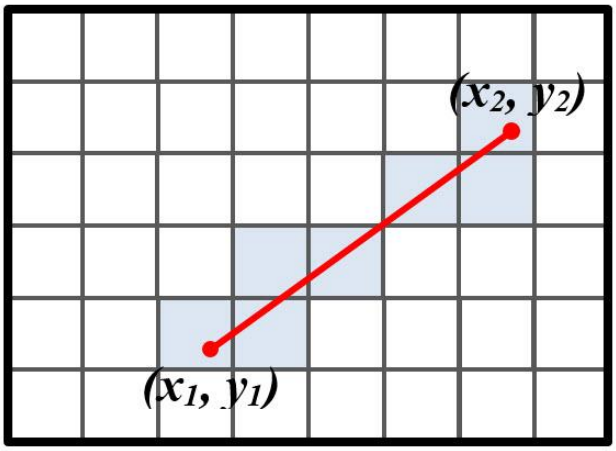


Fig.1 DDA line drawing algorithm ($m < 1$).

Case 2: *($m > 1$)*
When y increase more than x then dy / dx (m) is greater than one. According to Fig.2 in such case the line is drawn using the following steps:

1. Draw the first point on the line (x_1, y_1) and let $x=x_1$ and $y=y_1$.
2. Compute the new values to x and y using the following equations:

$$y = y + 1 \quad (7)$$

$$x = x + 1 / m \quad (8)$$

3. Draw the new point (round (x), y)
4. if $y \neq y_2$ then go to step 2

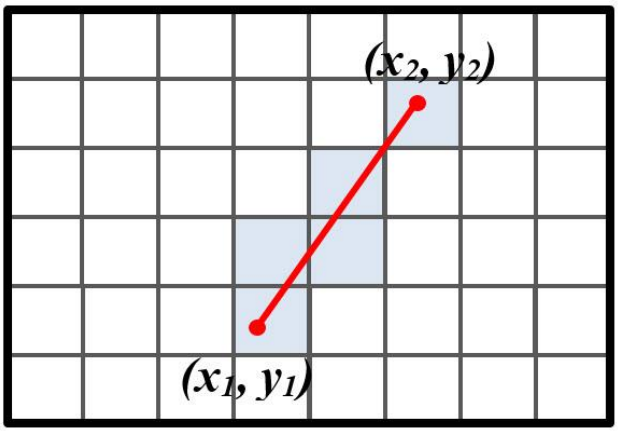


Fig.2 DDA line drawing algorithm ($m > 1$).

The performance of the DDA algorithm can be improved by separating the increments m and l / m into integer and fractional parts so that all calculations are reduced to integer operations. The increment of y by m and x by $1/m$ can be done as integer operation. Since the slope equal to the ratio of two integers is $m=dy/dx$, then the increment of x by dx/dy can be done by setting the counter to zero and increment it by dx each time a new pixel is computed. When the counter equal to or greater than dy, then the intersection point of x increment by one and the counter decrease by dy [20].

B. DDA arc drawing

There are many algorithms that use the concept of DDA arc drawing. These algorithms which differ from each other in the trigonometric transformations have described the rotation of the

vector in the x-y plane. The rotation matrix which is used for vector rotation is described as follows: By using equations 9 and 10, we can compute both x_{n+1} and y_{n+1} as shown in Fig. 3.

$$x_{n+1} = R\sin(\theta + \Delta\theta) = R\sin\theta\cos\Delta\theta + R\cos\theta\sin\Delta\theta = x_n\cos\Delta\theta + y_n\sin\Delta\theta \quad (9)$$

$$y_{n+1} = R\cos(\theta + \Delta\theta) = R\cos\theta\cos\Delta\theta + R\sin\theta\sin\Delta\theta = y_n\cos\Delta\theta - x_n\sin\Delta\theta \quad (10)$$

By making $\Delta\theta = \varepsilon = 2^{-m}$ where m is an integer (usually $m \geq 3$)

$$x_{n+1} = x_n\cos\varepsilon + y_n\sin\varepsilon \quad (11)$$

$$y_{n+1} = y_n\cos\varepsilon + x_n\sin\varepsilon \quad (12)$$

These two equations can be expressed in a matrix called rotation matrix A.

$$A = \begin{bmatrix} \cos\varepsilon & -\sin\varepsilon \\ \sin\varepsilon & \cos\varepsilon \end{bmatrix} \quad (13)$$

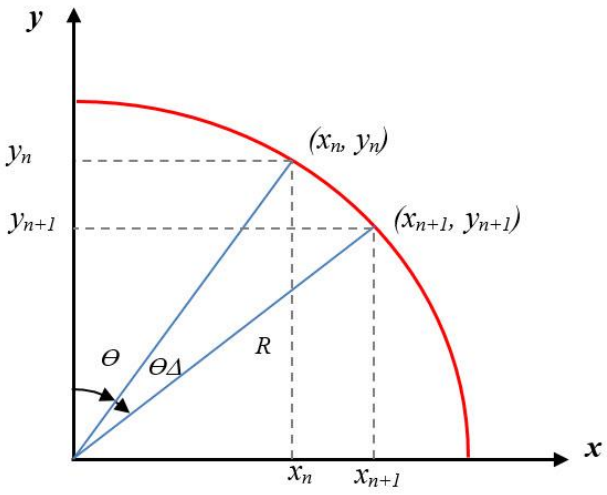


Fig.3 The rotation of vector R in the x-y plane.

One of the simplest methods used for drawing an arc in computer graphics, computer controlled printing and automated control is the fast exact DDA algorithm [21]. DDA algorithms use simpler expressions in order to eliminate the expensive computation of trigonometric functions. All DDA algorithms give rotation matrix determinant approximately equal to one. The algorithm which gives determinant closer to one is the more accurate.

Two steps DDA algorithm called fast exact DDA algorithm gives maximum accuracy in circle interpolation, although it is simple as one step DDA algorithm and can be implemented using shift and rotate operations [22]. This algorithm assumes the following initial conditions:

$$x_o = R \qquad y_o = 0 \quad (14)$$

and

$$x_1 = R(1 - \varepsilon^2)^{0.5} \qquad y_1 = R\,\varepsilon \quad (15)$$

Where $R=2^m$

Then x_{n+2} and y_{n+2} shown in Fig.4 are computed as follow:

$$x_{n+2} = x_n - 2\,\varepsilon\,y_{n+1} \qquad y_{n+2} = y_n + 2\,\varepsilon\,x_{n+1} \quad (16)$$

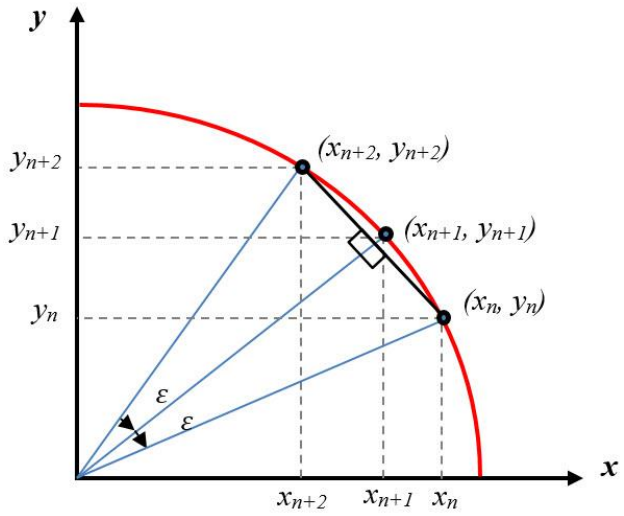


Fig.4 The generation of three subsequent points along the circle.

III. THE VIRTUAL CIRCLE TANGENTS ALGORITHM

This section describes a new algorithm called virtual circle tangents that is used to search for the shortest path from source to target in the environment with several polygonal shape obstacles. The tangents graph is constructed by drawing tangents from source to visible vertices on polygon obstacles, This process requires complex mathematical computation and consume time, so for simply we represent each polygonal obstacle by a virtual circle using the randomized incremental algorithm [23]. Now, a tree of tangent paths from source to target through the virtual circular obstacles are drawn, and one of these paths is chosen as the shortest path. The virtual circle tangents algorithm can be applied either in the on-line scenario that dependent on the sensing range of robots or in the off-line scenario.

A. The randomized incremental algorithm

Path planning with polygon shape obstacles has some complexity in determining the shortest path to target especially, when we used the tangent paths with these obstacles. In this case, we need to know the coordinates of the visible vertices in polygons that attach the tangent paths. For simplicity, we assume that each polygonal shape obstacle is surrounding by a virtual circle with all its vertices inside this circle. This process is done by using the randomized incremental algorithm. This algorithm determines the smallest enclosing circle that surrounding the vertices of a given polygon. The key points of Randomized Incremental Algorithm are:

- Randomized: use a random order to add the points.
- Incremental construction: add the points one by one and maintain the solution so far.

Assume that the obstacle consists of n vertices and the smallest enclosing circle either has at least three points on its boundary or it has two points which form the diameter of the circle. If there are three points then they subdivide the circle into arcs of an angle at most. The randomized incremental algorithm is implemented according to the following procedure:

Randomly permute the vertices in any polygon shape yielding the following sequence $V=< v_1, v_2... v_n>$. Then call the function MinCircle(V).

MinCircle(V)

1-Let C_2 be the minimum circle enclosing $\{v_1, v_2\}$

2-For i = 3 to |V| do
  If $\mathbf{v}_i \in C_{i-1}$ then
    $C_i = C_{i-1}$
  Else
    C_i=MinCircleWithOnePointBoundary
      ($V_{[1,2,...i-1]}, \mathbf{v}_i$)

MinCircleWithOnePointBoundary(V,q)

1-Randomly permute the points in V. Let C_1 be the minimum circle enclosing $\{q, v_1\}$

2- For i = 2 to |V| do
  If $\mathbf{v}_i \in C_{i-1}$ then
    $C_i = C_{i-1}$
  Else
    C_i=MinCircleWithTwoPointsBoundary
      ($V_{[1,2,...i-1]}$, $\mathbf{q}, \mathbf{v}_i$)

MinCircleWithTwoPointBoundary(V,q_1, q_2)

1-Randomly permute the points in V. Let C_0 be the minimum circle enclosing $\{q_1, q_2\}$

2-For i = 1 to |V| do
  If $\mathbf{v}_i \in C_{i-1}$ then
    $C_i = C_{i-1}$
  Else
    C_i = Circle ($q_1, q_2, \mathbf{v}_i$)

All the polygonal obstacles in the environment have been converted to a circular shape in off-line path planning scenario, where the robot has complete knowledge about the environment [23]. Fig.5 shows a polygonal shape obstacle with six vertices and the smallest enclosing circle of that obstacle.

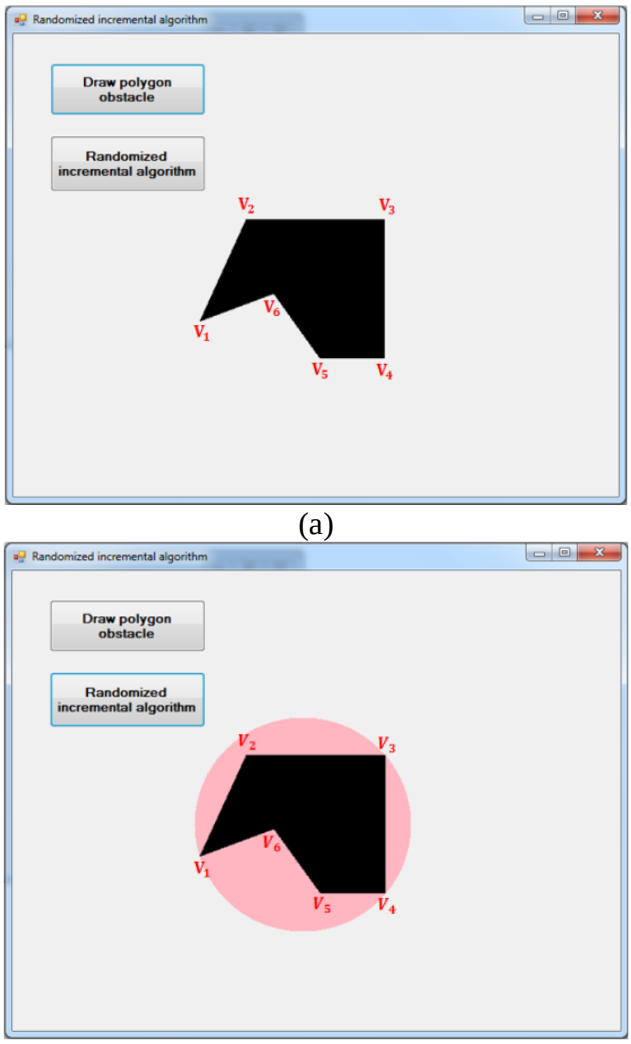


Fig. 5 The polygon shape obstacle with six vertices.
(a) Before using the randomized incremental algorithm.
(b) After using the randomized incremental algorithm.

B- Visible tangents graph construction

The visible tangents graph is constructed by drawing a straight line from the robot source position to the target. If this path passes through any obstacle in the environment then we considered it as a not safe path and must be replaced by two possible paths to the colliding obstacle. These paths are results from drawing two tangent lines from robot source position to the virtual circle that represents the colliding obstacle. The distance length from the source point to each tangent point on the virtual obstacle must be computed and new paths from these tangent points to target must be drawn. If any new path is not safe and has a collision with another obstacle then this path must be replaced by two tangents and the process is repeated until reach to the target. As a result, a visible tangents graph consist of several paths from source to target with their distances was obtained. The shortest path from source to target is chosen from these paths to be a safe path for the robot to reach the target without any collision. The proposed algorithm can be described by the following steps.

Step1: detecting the location, radius, direction, and velocity of robot $(P_s(x_s, y_s), r_s, \theta s, v_s)$,and the target location $(P_g(x_g, y_g))$.

Step 2: represent each polygonal shape obstacle by virtual circle using the randomized incremental algorithm. The radius of each virtual circle must be increased by the radius of the robot to produce a safe path to that robot.

Step 3: calculate a direct robot path from source to goal. This path has two sub paths: a curve path, necessary to direct the differential drive robot in the direction of the goal, and a straight line which is the tangent line between the circular path of the robot and the goal position, as shown in Fig. 6

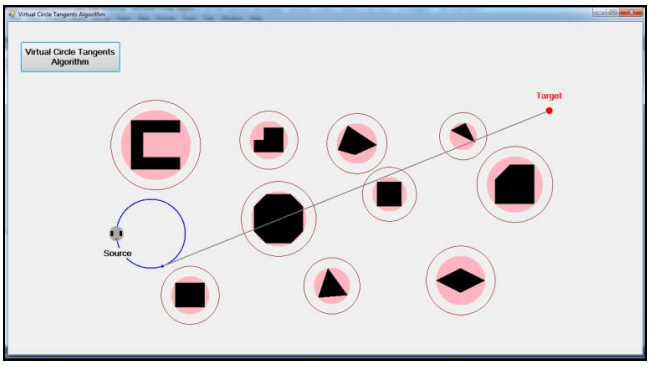


Fig.6 The standard robot path from start point to the goal.

Step 4: check if the robot path has an intersection with any obstacle. The intersection occurs between the robot and any polygonal obstacle when the distance between the standard trajectory and the virtual circle center of the obstacle is smaller than the radius of the virtual circle.

Step 5: if there is an intersection, then draw the inner and outer tangent lines from the arc path of the robot to the virtual circle. The procedures

used in construction the outer and inner tangent lines are as follows:

A. Outer tangent lines

Fig.7 is used to indicates the two outer tangent lines with their four end points $p_3(x_3,y_3)$,$p_4(x_4,y_4)$, $p_5(x_5,y_5)$,$p_6(x_6,y_6)$**.** The points $p_o(x_o, y_o)$ and $p_s(x_s, y_s)$ represent the center points of the virtual circle of obstacle and the circular trajectory of robot and also, R_o and R_s are the radiuses of both circles respectively. In order to find the four end points of the outer tangent lines, firstly we draw third circle with the same center point of the virtual circle and has radius R such that:

$$R = R_o - R_s \qquad (17)$$

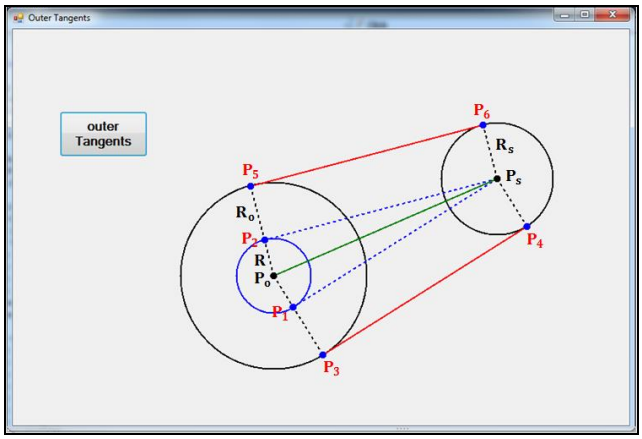


Fig.7 Drawing the outer tangent lines.

Then draw the tangent lines from the external point $p_s(x_s, y_s)$ to the third circle and compute the tangent points p_1 and p_2 as shown in Fig. 8.

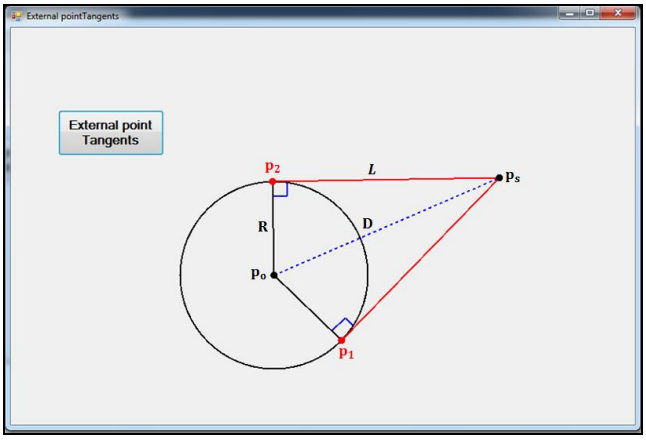


Fig.8 Tangency points from external point.

Since each tangent line drawn from external point is perpendicular to the radius R, then L computes as follow:

$$L = \sqrt{D^2 - R^2} \qquad (18)$$

Where

$$D = \sqrt{(x_s - x_o)^2 + (y_s - y_o)^2} \qquad (19)$$

p_1 and p_2 represent the intersection points between two circles as shown in Fig.9. The first one has a center point at p_o with radius equal to R and the other circle has a center point at p_s with radius equal to L.

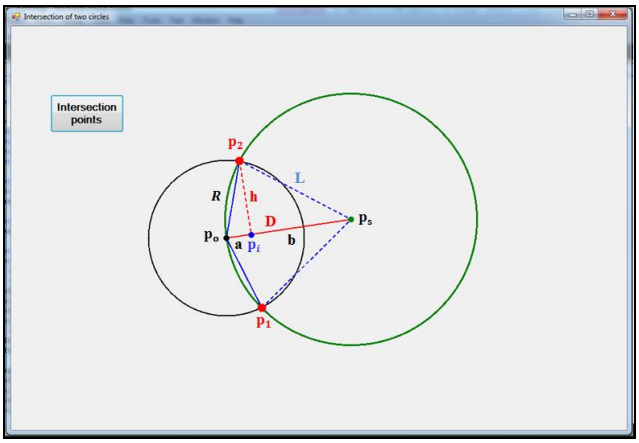


Fig.9 The intersection points of two circles.

According to Fig.9 we found that

$$D = a + b \qquad (20)$$

The length of the perpendicular line h between line D and the intersection point p_2 is computed as follows:

$$h^2 = L^2 - b^2 \qquad (21)$$

And also

$$h^2 = R^2 - a^2 \qquad (22)$$

So that

$$L^2 - b^2 = R^2 - a^2 \qquad (23)$$

By substituting $a = D - b$ in equation 23:

$$b = \frac{L^2 - R^2 + D^2}{2D} \qquad (24)$$

Similarly:

$$a = \frac{R^2 - L^2 + D^2}{2D} \qquad (25)$$

Now the coordinate axis of p_2 point can be computed. The process of computing the coordinate axis of p_5 point becomes easy as shown in Fig. 9.

$$x_2 = x_i + \frac{h(y_o - y_s)}{D} \qquad (26)$$

$$y_2 = y_i - \frac{h(x_o - x_s)}{D} \qquad (27)$$

Then

$$x_5 = x_2 + \frac{R_s(y_s - y_2)}{L} \qquad (28)$$

$$y_5 = y_2 - \frac{R_s(x_s - x_2)}{L} \qquad (29)$$

The same procedure can be used to compute the coordinate axis of both points p_1 and p_3.

B-Inner tangent lines

The inner tangent lines with their four end points $p_3(x_3, y_3)$, $p_4(x_4, y_4)$, $p_5(x_5, y_5)$ and $p_6(x_6, y_6)$ shown in Fig. 10. The procedure used to computes the coordinate axis of these points is similar to the procedure used by the outer tangent lines but the difference is that the radius of the third circle is the sum of the radiuses of the other two circles as shown in Fig.10. According to this figure, the coordinate axis of the inner tangent point p_5 is computes by using equation 30 and equation 31.

$$x_5 = x_2 - \frac{R_s(y_2 - y_s)}{L} \qquad (30)$$

$$y_5 = y_2 - \frac{R_s(x_2 - x_s)}{L} \qquad (31)$$

The same procedure can be used to compute the coordinate axis of both points p_1 and p_3.

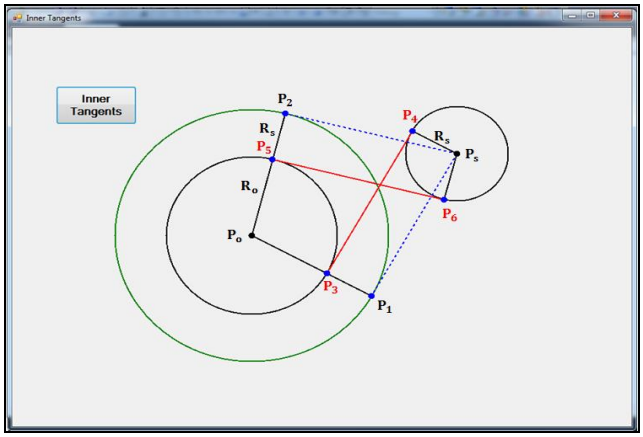


Fig.10 Drawing the inner tangent lines.

Step 6: In order to complete the visible tangents graph and compute the shortest path the paths set divided into two parts upper paths set and lower paths set. The upper paths start from the source through the inner tangent line to the first collide obstacle and continue to process any obstacles intersects the robot trajectory to target. The lower paths set start from the source to the target and pass through the outer tangent line to the first collide obstacle.

Step7: The construction of the upper paths set passes through several steps to reach the target. At each step, the robot must manipulate an obstacle in his trajectory by drawing to paths, The first one is an arc path drawn around the virtual circle of that obstacle and the other one is the tangent straight line which may be inner or outer tangent. The lengths of these paths must be computed. Fig. 11 shows the complete upper paths set from source to target.

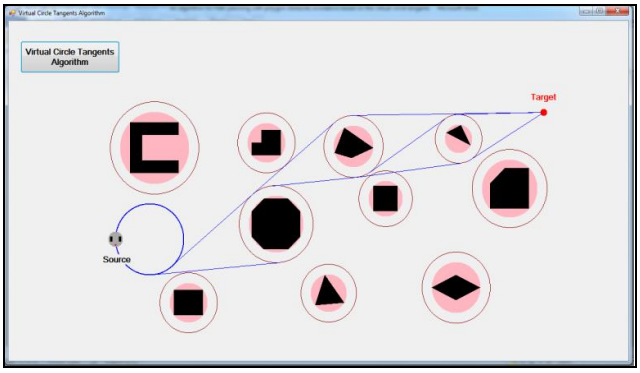


Fig.11 Complete set of upper paths.

Step 8: The upper paths set has several paths start from source and reach to the target. The length of each path is computed in the last step. The shortest one can be chosen to be the short path to the upper paths set which indicates by magenta color in Fig. 12.

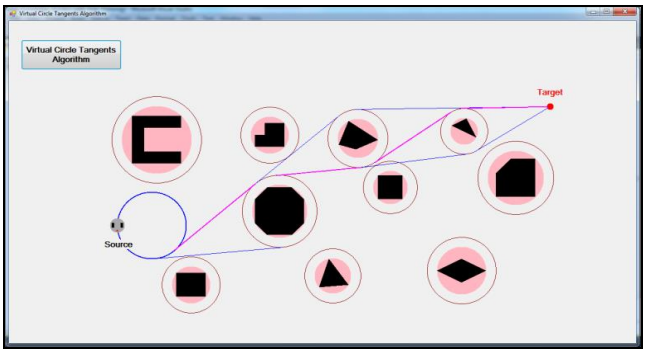


Fig.12 The shortest path of the upper paths set.

Step 9: The process used to compute the lower paths set is similar to that used by the upper paths set but it starts from the outer tangent line of the first collide obstacle. Fig.13 shows the complete lower paths set with the upper paths set. Also, the lengths of each arc and straight line paths at each step must be computed.

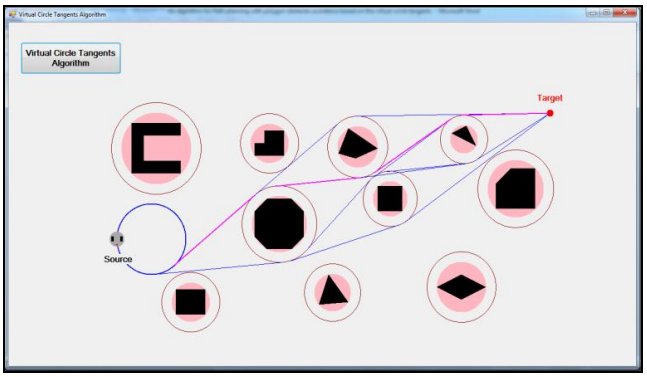


Fig.13 The complete set of lower paths set.

Step10: At last step, the length of each lower path from source to target is computed. The shortest one of these paths is chosen to be the shortest lower path from source to target as shown in Fig. 14 with green color.

Step11: Fig. 14 shows both the shortest path of the upper paths set (magenta color), and the lower paths set (green color). Compute the shortest path from these two paths to be the shortest path to the visible tangents graph. The shortest path in visible tangents graph is shown in Fig. 15.

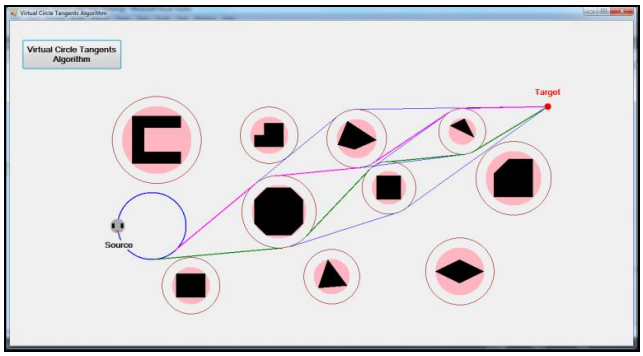


Fig.14 The shortest path in the lower paths set.

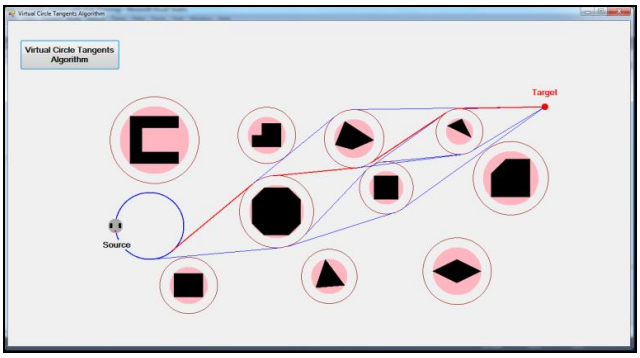


Fig.15. The shortest path of the visible tangents graph.

C- On-line path planning scenario

The virtual circle tangents algorithm is designed to work in off-line and on-line path planning scenarios. The off-line path planning means that the robot has a global knowledge of the environment and there is not any limitation on the sensing range of the robot as we described in the previous section.

The on-line path planning scenario means that the robot has a limited sensing range that leads to limited knowledge about the polygonal obstacles in the environment. That's means the robot knows the locations of the start point and target point and also, the polygonal obstacles within the sensing range of this robot. The procedure used for on-line path planning can be described by the following steps:

Step 1: According to the location of the source, location of the target and the current

orientation of the robot construct a direct path from source to target.

Step 2: When the robot detect that there is a new obstacle in its sensing range, then this range is increased by a value equal to the diameter of the largest obstacle in the environment. This process is done to enable the robot to view all the vertices of that obstacle in order to calculate the virtual circular obstacle for the polygonal obstacle using the randomized incremental algorithm.

Step 3: Outer and inner tangent lines are drawn from current location of the robot to the virtual circle of the sensed obstacle. The robot completes the paths from virtual circle to target by drawing another tangent between the virtual circle and target.

Step 4: The robot computes the lengths of two paths from the current location of the robot to the target through the inner and outer tangents and then it is choosing the shortest one to complete its movement to the target.

Step 5: When the robot detects another obstacle then it repeats steps 2, 3 and 4 until reach to the target.

IV.SIMUIATION RESULTS

The virtual circle tangents algorithm has been tested in windows operating system and implemented with the visual basic language. Differential drive mobile robot was assumed in our simulations. The proposed algorithm tested in on-line and off-line scenarios in different environments with a different number, size, and shapes of polygonal obstacles. Two path planning algorithms are used for comparison: Vis-bug algorithm [24] and Tangent but algorithm [25]. The first algorithm has two motion modes: the movement toward the goal mode and obstacle border line-following mode. In the first mode, the robot moves toward the target until hitting an obstacle and then changes its motion to border line-following mode. During this mode, the robot computes the distance to the target and return to the target motion mode when the distance decreases. The Tangent-bug algorithm has the same modes of operations of VisBug (Target motion mode and obstacle border line-following mode). The main difference between the Tangent but and VisBug is that the Tangent but use the local data in the local tangent graph to compute the shortest path. The local tangent graph is enclosing the obstacles within the sensing range of the robot. During motion toward Target mode, the robot follows the trajectory calculated in the local tangent graph. Through the obstacle border line-following mode the robot checks for exits this mode and return to the motion toward the target mode. The two algorithms have global information represented by the target position and local information represented by the obstacles, however when the sensing range of the robot increase the local information become global information. The virtual circle tangents algorithm introduced in this paper use graph like TangentBug algorithm but it differs in the way used for computing the graph which gives benefit when the target information is limited.

The on-line scenario is discussed in the first simulation by using a robot with limited sensors range. Different conditions were discussed such as the obstacle shapes, their numbers, their dimension, and the sensing range of the robots. A scenario with 20 obstacles of different shapes and dimension is simulated. Several snapshots on the same scenario are implemented on the trajectories generated by the three different algorithms as shown in Fig.16-18. The robot speed was assumed to be 100 pixels/s and the sensing range of robot is 100 pixel and increase to 220 pixels when a new obstacle is detected. The increasing in the sensing range of the robot is related to the diameter of the largest obstacle in the environment. From this simulation results, we found that the virtual circle tangents algorithm provides the shortest path while the VisBug algorithm provides the longest path.

For more comparison, this scenario is repeated for obstacles number range from 0 to 40 increasing by 5. For each value of the obstacle numbers, we take about 20 distributed that arranged randomly and the mean value of the distance between source and goal have been calculated for each algorithm and indicated in Fig.19. From these simulations, we found that the introduced algorithm give better performance than the two other algorithms for different obstacles number.

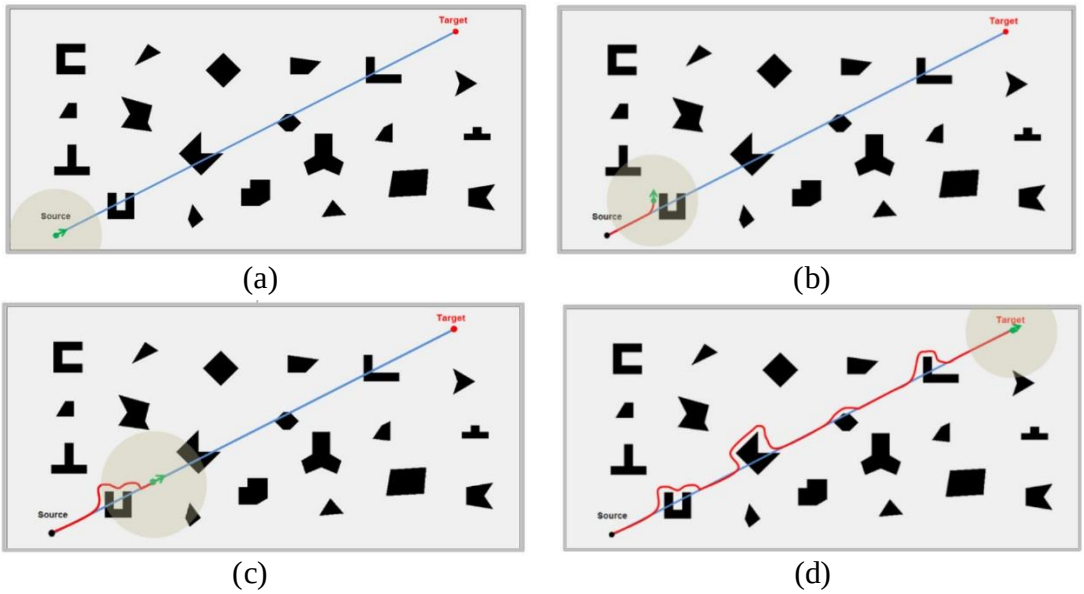


Fig.16 On-line trajectory planning using VisBug algorithm at different steps :(a) The robot initial position; (b) The robot detect an obstacle; (c)The robot return to its standard trajectory; (d) The complete path.

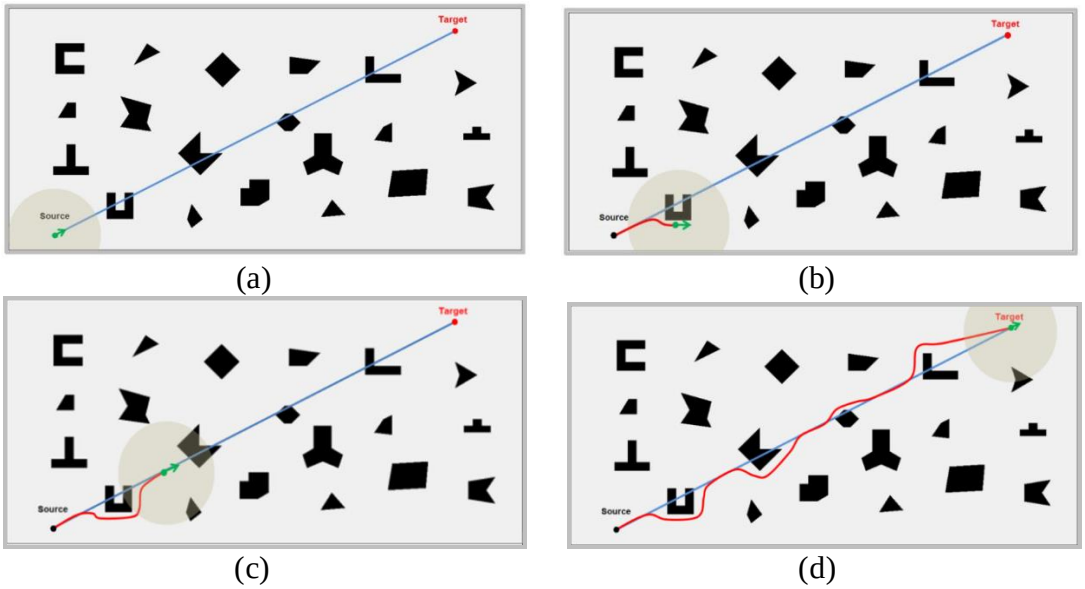


Fig.17 On-line trajectory planning using TangentBug algorithm at different steps :(a) The robot initial position; (b) The robot detect an obstacle ; (c) The robot return to its standard trajectory; (d) The complete path.

The effect of changing the target location has been studied in an environment with 40 obstacles differs in shape and dimension. Five random target locations were chosen and the results are summarized in Table 1. We notice that for all the five target locations the virtual circle tangents algorithm has the best performance. The trajectories produced by the three algorithms are indicated in Fig. 20.

The third set of the simulations studies the effect of changing the sensing range on the path length. This simulation was done on the environment with 40 obstacles have different shape and dimension. 20 different realizations have been implemented for each value of the sensing radius of the robots and at each realization a different distribution of obstacles has been taken into account. An example of the paths produced by the three different algorithms in one of the realizations is shown in Fig.21. The mean value of the distance between source and goal in the 20 realizations have been calculated and indicated that the virtual circle tangents algorithm has the best performance as shown in Fig.22.

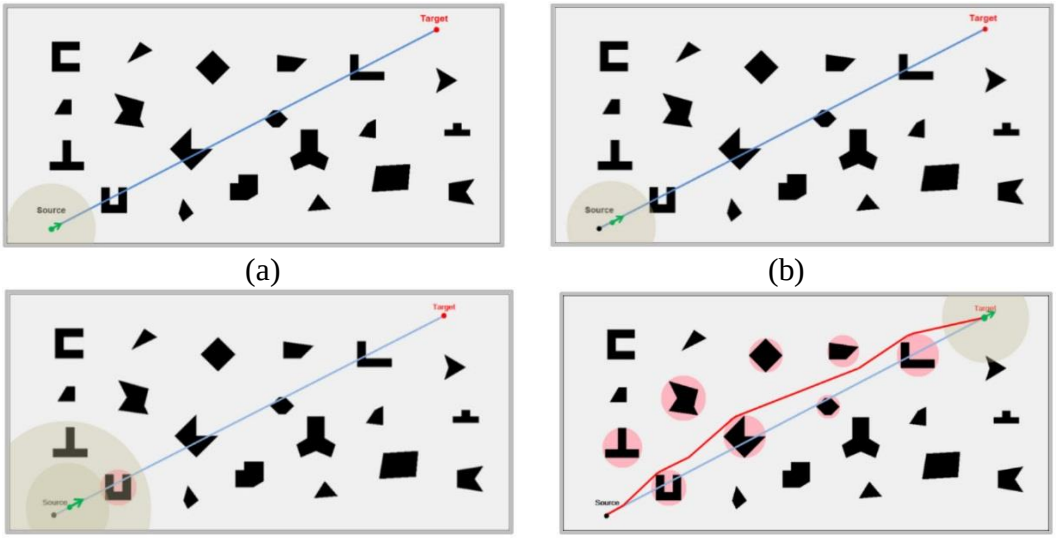


Fig.18 On-line trajectory planning using Virtual circle tangents algorithm at different times :(a) The robot initial position; (b) The robot detect an obstacle ; (c)The robot return to its standard trajectory; (d) The complete path.

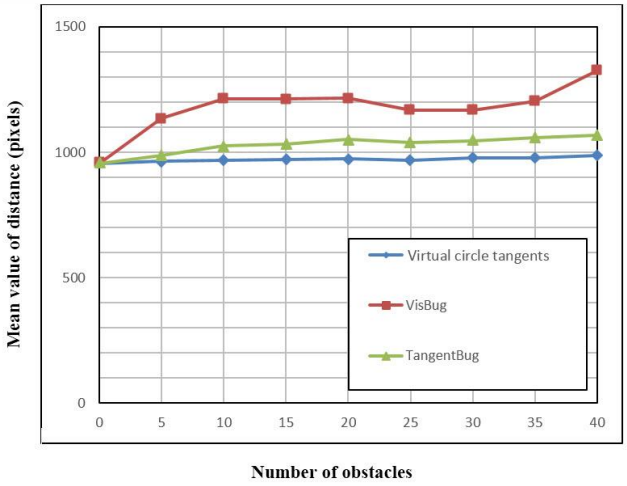


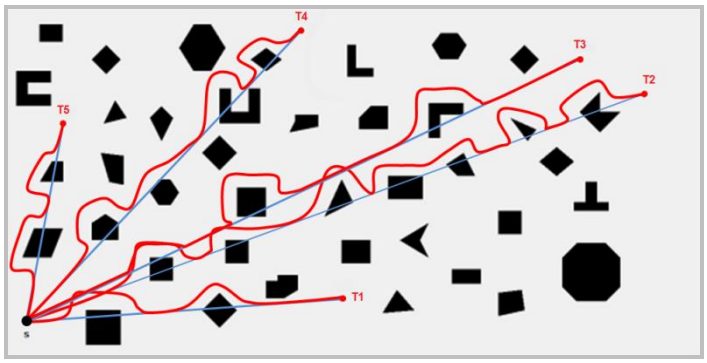
Fig.19 The performance comparison of the virtual circle tangents algorithm with two other algorithms for different number of obstacles in on-line scenario.

Table 1 Performance comparison of on-line scenario in environment has 40 obstacles with different sizes and repeated with five different target locations.

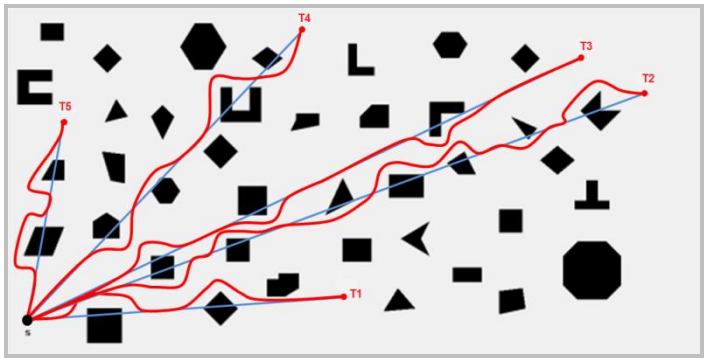
Source location	Target location	Visbug algorithm	TangentBug algorithm	Virtual circle tangents algorithm
(190,590)	(730,550)	617	585	563
(190,590)	(1250,200)	1342	1218	1163
(190,590)	(1140,140)	1215	1150	1103
(190,590)	(660,90)	1183	743	714
(190,590)	(250,250)	464	436	387

V. CONCLUSION

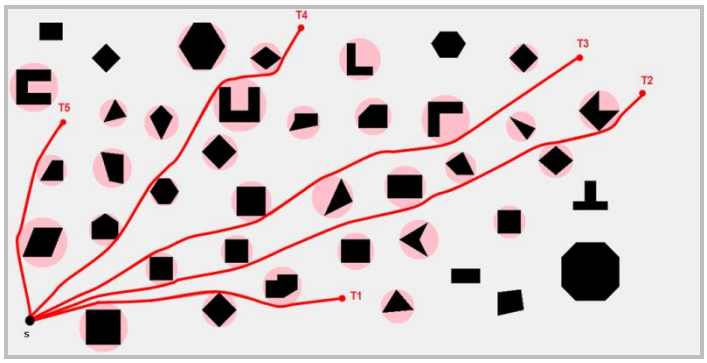
In this paper, an algorithm called virtual circle tangents for path planning of mobile robot in an environment with different number and size of polygonal shape obstacles was introduced. The introduced algorithm was applied in on-line and off-line scenarios. The paper explains the low-level and high-level path planning in details. The first part give a comprehensive view on the low-level path planning represented by the digital differential analyzer for line and arc drawing. The digital differential analyzer algorithm have an advantage of simplicity and uses only integer calculation, therefore, reduce the required resources. The other part focus on the graph building and searching for the shortest path. The illustration of both levels of the introduced algorithm shows that it requires simple resources for implementation in the hardware system. The introduced algorithm described in details and the simulation result compared with two other graph based path planning algorithms. The three algorithms compared in on-line and off-line scenarios. The main simulation result shows that the introduced algorithm has the best performance in all the simulation results that include on-line and off-line scenarios, different values of the robot sensing radius, different number and different sizes of obstacles. Also, the virtual circle tangents algorithm has a good performance with respect to the other algorithms in term of computation time.



(a)

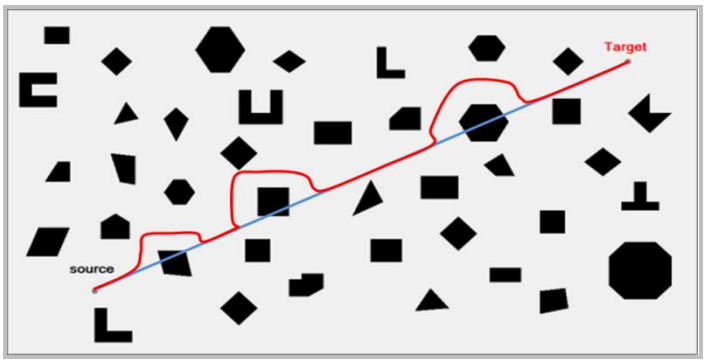


(b)

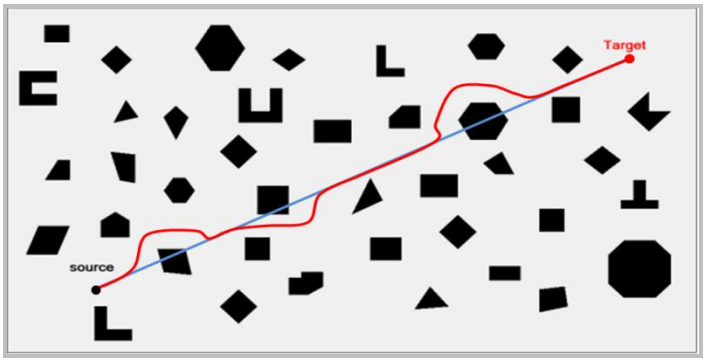


(c)

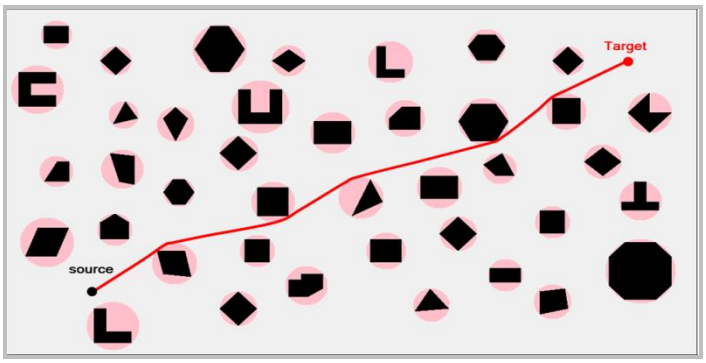
Fig.20 The performance comparison for five different target locations in on-line scenario with 40 obstacles have different sizes: (a) VisBug algorithm (b) TangentBug algorithm; (c) Virtual circle tangents algorithm.



(a)



(b)



(c)

Fig.21 Study the effect of changing the robot sensing radius on the trajectory planning performances: (a) VisBug algorithm (b) TangentBug algorithm (c) Virtual circle tangents algorithm.

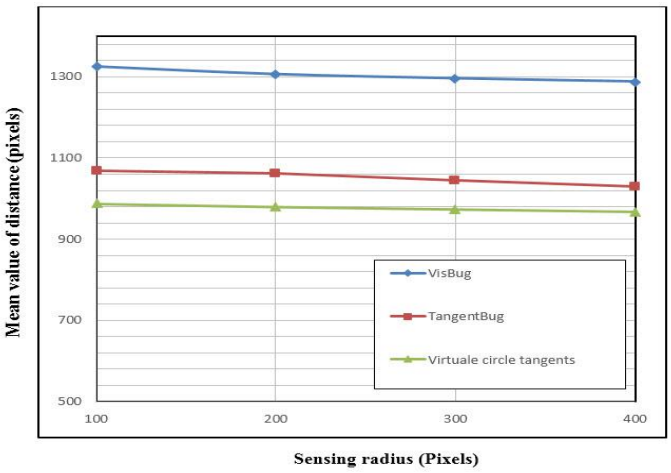


Fig.22 The performance comparison of the virtual circle tangents algorithm with two other path planning algorithms for different values of the sensing radius.

REFRENCES

[1] S.M. LaValle, PLANNING ALGORITHMS, University of Illinois, 2006.
[2]B.coppin, Artificial Intelligence Illuminated, 2004.
[3] J.Kaur, V. K. Banga and G.Singh, "Robotic Path Planning Using the Intelligent Control", International Conference on Advances in Electrical and Electronics Engineering (ICAEE'2011)
[4] P. Raja, S. Pugazhenthi, "Optimal path planning of obile robots: A review", International Journal of Physical Sciences,Vol. 7, No. 9, pp.1314 – 1320, 2012.
[5] Y. Koren and J. Borenstein,"Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation" Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, California, pp. 1398-1404, 1991.
[6] M. HAMANI, A. HASSAM, "Mobile Robot Navigation in Unknown Environment Using Improved APFMethod", the 13th International Arab Conference on Information Technology ACIT'2012.
[7] S. Weijun, M. Rui and Y. Chongchong, "A Study on Soccer Robot Path Planning with Fuzzy Artificial Po-tential Field", Proceedings of the 2010 International Conference on Computing, Control and Industrial En-gineering (CCIE), vol.1 , pp.386-390, 2010.
[8] J. Gue, Y. Gao and G. Cui, "Path planning of mobile Robot base on Improved Potential Field", Information Technology Journal vol. 12, No. 11, 2013.
[9] X.Yun,K.Tan, "A Wall-Following Met hod for Escaping Local Minima in Potential Field Based Motion Planning",ICAR, Monterey, CA, 1997
[10] G.Li, Y.Tamura and A.Yamash,H.Asama, "Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning", Int. J. Mechatronics and Automation, Vol. 3, No. 3, 2013
[11] P.Bhattacharya,M. L. Gavrilova, "Voronoi diagram in optimal path planning", 4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)
[12] S.Mohammadi,N.Hazar, "A Voronoi-Based Reactive Approach for Mobile Robot Navigation", H. Sarbazi-Azad et al. (Eds.): CSICC 2008, CCIS 6, pp. 901–904, 2008
[13] A.Sahraei1, M.T.Manzuri, M.Tajfard and S.Khoshbakht, "Obstacle Avoidance using a Near Optimal Trajectory for Omni-directional Mobile Robots", 12th International CSI Computer Conference (CSICC'07) Shahid Beheshti University, Tehran, Iran, 2007.
[14] J.W. Choi, R.E. Curry and G.H. Elkaim, "Real-time obstacle avoiding path planning for mobile robots", in: Proceedings of the AIAA Guidance, Navigation and Control Conference, AIAA GNC 2010, Toronto, Ontario, Canada, 2010.
[15] T.Lozano-perez, "Automatic Planning of Manipulator Transfer Movements", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-1 1, NO. 10, 1981.
[16] T. Lizano-Perez, M.A. Wesley," An algorithm for planning collision-free paths among polyhedral obstacles", Communications of the ACM , Vol. 2, .No. 10,1979.
[17] K.Nawade,V. Aditya,A.Shrivastava and B. K. Rout, "Geometrical approach to On line Trajectory generation, Obstacle avoidance and Footstep planning for a Humanoid Robot", 15th IEEE-RAS International Conference on Humanoid Robots, Seoul, Korea, 2015.
[18] Y.H. Liu, S. Arimoto, "Proposal of tangent graph and extended tangent graph for path planning of mobile robots", Proceedings of the 1991 IEEE,Intemational Conference *on* Robotics and Automation Sacramento, Califomia , 1991.
[19] C.Alexopoulos and P. M. Griffin," Path Planning for a Mobile Robot", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 22, NO. 2, MARCWAPRIL 1992.
[20] D.Hearn, M.P.Baker,Computer Graphics C (1996).
[21] L.Moroz, J.L. CieVliNski, M.Stakhiv and V.Maksymovych, "A Comparison of Standard One-Step DDA Circular Interpolators with a New Cheap Two-Step Algorithm", Hindawi Publishing Corporation Modelling and Simulation in Engineering, 2014.
[22] J.L. Cie´sli´nski and L.V.Moroz, Fast exact digital differential analyzer for circle generation, 2013.
[23]E.Welzl, "Smallest inclosing disks (balls and ellipsoids)",APPERED IN"New Results And New Trends in Computer Science 555, p.p. 359-370, 1991.
[24] V.J. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function", IEEE Transactions on Systems, Man, and Cybernetics Vol.20, No.5, pp.1058–1068, 1990.
[25] I. Kamon and E. Rivlin, "A new range-sensor based globally algorithm for mobile", in: Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, 1996.