APPORXIMATION ALGORITHM LOCALIZING MOBILE ROBOT USING WINDOWS IN A POLYGON MAP

Dao Duy Nam\textsuperscript{1,2}, Nguyen Quoc Huy\textsuperscript{1,3*}

\textsuperscript{1}SaigonTech, SaigonTech Tower, Lot 14, Quang Trung Software City, District 12, HCMC,
\textsuperscript{2}High school for the gifted, VNUHCM, 153 Nguyen Chi Thanh, District 5, HCMC
\textsuperscript{3}Saigon University, 273 An Duong Vuong, District 5, HCMC

*Email: namdd@saigontech.edu.vn, huy.ng@saigontech.edu.vn

ABSTRACT

This paper considers an approximation algorithm for the robot localization problem with the map in the form of a simple polygon. Localization hypotheses correspond to the copied maps from assumed mark of robot location. The algorithm is based on using overlay and intersection of copied maps. Windows where the robot eliminates false hypotheses are built up at the intersection. On the basis of our implementation, we conducted experimental studies of these algorithms. The numerical results and their interpretation are given.

Key words: Computational geometry, robotics, robot localization, overlay polygon, algorithm complexity, approximation algorithm

1. INTRODUCTION

The mobile robot localization problem [1] is to determine the robot’s coordinates in the reference frame associated with the external environment. The robot is provided with an environment map as a simple planar polygon with \( n \) vertices \( P \) without holes. The mobile robot is placed at an unknown location \( p \) in \( P \) (see, Figure 1). To solve the robot localization problem, firstly, robot can view its surroundings and realize the visible region (so-called visibility polygon) \( V = V(p) \) in the map. In case of the map has only one piece, which coincides with the visibility polygon, then the problem is solved. In case of the map has several pieces, it is necessary to determine which piece corresponds to the initial robot location. For this purpose, based on the analysis of polygons \( P \) and \( V \), robot must generate a set of hypotheses \( H \) of its location \( p_i \in P \) so that the visibility polygon \( V(p_i) \) at the point \( p_i \) is congruent to \( V \). Robot moves and surveys surroundings, then it can eliminate all false hypotheses of its location, and determine its true initial location. This requires that the total length of robot movement must be minimal.

2. SOLUTIONS

The well-known mobile robot localization algorithms [2], [3], [4] and [5] comprise two phases: hypothesis generation and hypothesis elimination. The hypothesis generation phase computes the set of hypothetical locations \( p_1, p_2, \ldots, p_k \in P \) that match the observations sensed
by the robot at its initial location. The hypothesis elimination phase rules out incorrect hypotheses thereby determining the true initial location of the robot. An optimization problem of mobile robot localization is an NP-hard problem [2]. We propose an approximation (polynomial) algorithm for mobile robot localization using windows in the polygon map.

The hypothesis generation phase generates a set \( H = \{h_1, h_2, \ldots, h_k\} \) of hypothetical locations in \( P \) at which the robot might be located initially. Without loss of generality, we select an arbitrary hypothetical location \( p \) from \( H \) to serve as a reference point or origin. Next, for each hypothetical location \( p_j, 1 \leq j \leq k \), a translation vector \( t_j = p_i - p_j \) is defined that translates location \( p_j \) to \( p_i = p_j + t_j \). As result, we compute a set of copies \( P_1, P_2, \ldots, P_k \) of the environment polygon map \( P \), corresponding to the set of hypothetical location \( H \), such that \( P_j \) is congruent to \( P \) translated by the vector \( t_j \). Copy \( P_i \) is translated by the zero vector. The point in each translated polygon \( P_j \) corresponding to the hypothetical location \( p_j \) is now located at the origin \( p_i \).

**Definition 1.** The overlay polygons (polygons are considered here as a subdivision of the plane on the inner and outer faces and side polygons are treated as edges of the subdivision plane) is considered in [6], [7].

The intersection of polygons \( P_1, P_2, \ldots, P_k \) with respect to the selected instance of the map \( P = P_i \) defines the polygon as the intersection of the displaced polygons \( P_j, 1 \leq j \leq k \). This intersection is a face (or a set of faces) overlay polygons. Figure 1 shows face overlay, which is the intersection \( \text{Intersection}(P_1, P_2) \) of polygons \( P_1 \) and \( P_2 \) shaded.

**Definition 2.** The inner edge of the overlay polygons is such an edge intersection (one of several) that separates the area of intersection of the other internal faces of the overlay, as opposed to those edges that belong to the intersection, but separated by the intersection of the outer edge overlay on the two-dimensional plane (see. Fig. 2, which shows the interior edges \( e_1 \) and \( e_2 \)).

Consider a hypothesis \( h_j (h_j \neq h_i) \), and let \( F_{ij} \) denote the face in the intersection \( \text{Intersection}(P_i, P_j) \), which contains the initial location \( \gamma_0 \) (see. Example, \( F_{12} \) in fig. 3). Edge \( F_{ij} \) has at most \( 2n \) edges [4].

![Figure 1. An overlay of polygons](overlay.png)  
*Figure 1. An overlay of polygons*  
*Overlay\((P_1, P_2)\).*  

![Figure 2. The interior edges e1 and e2](interior_edges.png)  
*Figure 2. The interior edges e1 and e2*
Each of the $O(n)$ edge $e \subseteq F_{ij}$ is of one of three types: (i) $e$ lies on the boundary $P_i$ but not of $P_j$; (ii) $e$ lies on the boundary $P_j$ but not of $P_i$; or (iii) $e$ lies on the boundary of both $P_i$ and $P_j$. The robot can distinguish between $h_i$ and $h_j$, if and only if the robot sees edge $e$ of type (i) or (ii).

If $\gamma_0$ sees any edge of type (i) or type (ii), then the robot can distinguish between $h_i$ and $h_j$, without moving from the origin $\gamma_0$. Thus, assume that all edges of $F_{ij}$ that are visible from $\gamma_0$, are of type (iii). Let $e$ be an edge of $F_{ij}$ is of type (i) or type (ii). The set $VP(e)$ points of $F_{ij}$ that are visible to some point of $e$ is a simple polygon (the visibility polygon of $e$) within $F_{ij}$, which we know, by assumption, does not include the point $\gamma_0$. There is a chord of $F_{ij}$, $w(e)$, that lies on the boundary of $VP(e)$, separating $e$ from $\gamma_0$. The line segment $w(e)$ is often called a window [4] (see. Figure 4).

Now we can describe the proposed robot localization algorithm using the window map. Let an input polygon map be $P$, and the robot be placed in an unknown initial location in the $P$. The algorithm consists of the following steps:

1. Compute the visibility polygon $V$ of the robot according to sensors at the initial unknown location.
2. Generate a set $H$ of $k$ hypotheses on the map $P$, corresponding to the visibility polygon $V$.

3. Choose an arbitrary hypothesis $h_i$ from $H$ and the corresponding point of a hypothetical location as a starting point for the construction of the overlay.

Operations from P.4 to P.7 are performed for all active (not yet eliminated) $h_j (j = 1, 2, ..., k')$ hypotheses.

4. Construct $Overlay(P_i, P_j)$ and $Intersection(P_i, P_j)$ of copies maps $P_i$ and $P_j$.

5. Computed a connected component $F_{ij}$ of $Intersection(P_i, P_j)$, comprising a starting point.

6. Construct all the windows of the component intersection $F_{ij}$.

7. Among the midpoints of all windows in $F_{ij}$, we find a point $r_j$, which is nearest to the current location of the robot, i.e finding the "nearest" window in $F_{ij}$.

8. Compute $Intersection(P_1, P_2, ..., P_{k'})$ for active hypotheses and coherent component $F$, containing the starting point. Among the points $r_j$ which fall in $F$, we find the nearest point $r$ to the current robot location.

9. Move the robot to a point $r$.

10. Eliminate hypothesis by comparing the current visibility polygon data of the robot at the point $r$ with data of the visibility calculated equivalent in all points corresponding to all active hypotheses.

11. Let $E$ be a set of hypotheses that have been eliminated in the previous step. Repeat steps 3-10 as the number of active hypotheses $H - E$ until there will be only one hypothesis, which will correspond to the true initial location of the robot.

Table 1 shows the complexity of algorithm in stages and summary.

<table>
<thead>
<tr>
<th>Step</th>
<th>Actions</th>
<th>The complexity of the algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>Generating hypotheses</td>
<td>$O(mn^2)$</td>
</tr>
<tr>
<td>4-7, 8</td>
<td>Construction of intersection overlay relative to the selected hypothesis</td>
<td>$O(k'n\log n)$, where $k'$ - number of active hypotheses</td>
</tr>
<tr>
<td>6, 8</td>
<td>Construction of &quot;windows&quot; associated overlay component of the intersection</td>
<td>$O(2k'n^2)$</td>
</tr>
<tr>
<td>7, 8</td>
<td>Examine points at the midpoints of the windows. Calculation of the shortest paths to determine the next window</td>
<td>$k'O(n\log^* n) + 2k'nO(n) = k'O(n^2)$</td>
</tr>
<tr>
<td>10</td>
<td>Comparison of data on visibility polygons for active hypotheses and current robot location</td>
<td>$2k'nO(n)$</td>
</tr>
</tbody>
</table>

The full complexity |

$$O(mn^2) + \sum_{k'=1}^{k-1} \left[ O(k'n\log n) + O(2k'n^2) \right] = O(n^4)$$
3. STUDY RESULTS AND COMMENTS:

Experimental study of the algorithm is based on our program implementation (Visual C + + 2010), as well as implementations of algorithms [3], [5] and we compare characteristics such as the total value of the travel path of the robot and the running time of localization algorithm together. Maps of various model types used during the experiment are generated. An example of such work and generating the localization algorithm using windows in the polygon map shown in Fig. 5. Here, the map size $n = 746$ and the number of hypotheses $k = 7$.

![Figure 5](image.png)

*Figure 5. The map size $n = 746$ and the number of hypotheses $k = 7$."

Table 2 shows the numerical results for these experiments parameter values $n = 746$ and $k = 7$. In this table, there are three approximate mobile robot localization algorithms (algorithm 1 – robot localization algorithm using the window map, algorithm 2 [3] and the algorithm 3 - localization of the robot using triangulation map [5]). Listed $n_k$ - number of hypotheses (according to the numbering in Figure 5), $d$ - length of the path traveled by the robot at the location, and $t$ – running time of algorithm. In the randomized algorithm 2 [3] two variants were used for the number of random points for $X = 100$ and $X = 500$.

To compare the experimental results, we determine the average of the fixed values of $d$ and $t$. For example, $\bar{d} = \frac{1}{7} \sum_{i=1}^{7} d_i$, $\bar{t} = \frac{1}{7} \sum_{i=1}^{7} t_i$, $s_i(2,1) = \frac{d_i^{(2)}}{d_i^{(1)}}$, $s_i(2,2) = \frac{d_i^{(2)}}{d_i^{(1)}}$, $s_i(3,1) = \frac{d_i^{(3)}}{d_i^{(1)}}$. 

$$
\begin{array}{c|c|c|c}
\text{Initial location} & p_1 & p_2 & p_3 \\
\text{Final robot’s location} & p_4 & p_5 & p_6 \\
\end{array}
$$
\[ s_i^{(2,1)} = \frac{t_i^{(2)}}{t_i^{(1)}}, \quad s_i^{(2',1)} = \frac{t_i^{(2')}}{t_i^{(1)}}, \quad \text{and} \quad s_i^{(3,1)} = \frac{t_i^{(3)}}{t_i^{(1)}}, \]  

where \( i \) - number of hypotheses, and superscripts in parentheses are the number of algorithms.

Table 3 shows relationships for different pairs of algorithms and their average values.

### Table 2. The numerical results for these experiments parameter values \( n = 746 \) and \( k = 7 \).

<table>
<thead>
<tr>
<th>№</th>
<th>d</th>
<th>t, c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algorithm 1</td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>X = 100</td>
<td>X = 500</td>
</tr>
<tr>
<td>1</td>
<td>500,5</td>
<td>529,2</td>
</tr>
<tr>
<td>2</td>
<td>220,2</td>
<td>266,0</td>
</tr>
<tr>
<td>3</td>
<td>69,3</td>
<td>113,5</td>
</tr>
<tr>
<td>4</td>
<td>499,9</td>
<td>520,1</td>
</tr>
<tr>
<td>5</td>
<td>70,3</td>
<td>117,0</td>
</tr>
<tr>
<td>6</td>
<td>175,5</td>
<td>235,9</td>
</tr>
<tr>
<td>7</td>
<td>175,5</td>
<td>245,5</td>
</tr>
</tbody>
</table>

### Table 3. The relationship \( s_i \) for different pairs of algorithms and their average values.

<table>
<thead>
<tr>
<th>№</th>
<th>d</th>
<th>t, c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Algorithm 1</td>
<td>Algorithm 2</td>
</tr>
<tr>
<td></td>
<td>X = 100</td>
<td>X = 500</td>
</tr>
<tr>
<td>1</td>
<td>1,06</td>
<td>1,01</td>
</tr>
<tr>
<td>2</td>
<td>1,21</td>
<td>1,02</td>
</tr>
<tr>
<td>3</td>
<td>1,64</td>
<td>1,10</td>
</tr>
<tr>
<td>4</td>
<td>1,04</td>
<td>0,92</td>
</tr>
<tr>
<td>5</td>
<td>1,66</td>
<td>1,38</td>
</tr>
<tr>
<td>6</td>
<td>1,34</td>
<td>0,98</td>
</tr>
<tr>
<td>7</td>
<td>1,40</td>
<td>1,37</td>
</tr>
</tbody>
</table>

\[ \bar{s}^{(l,i)} = \frac{1}{7} \sum_{j=1}^{7} s_{ij}^{(l,i)} \]

Various types of line graphs in Fig. 6 and 7 correspond to the relationship: line 1 – respect \( \bar{s}^{(2,1)} \); line 2 – respect \( \bar{s}^{(2',1)} \); line 3 – respect \( \bar{s}^{(3,1)} \). Fig. 6 is a graph of the average ratio of lengths of the paths as a function of size map \( n \). Fig. 7 shows a graph of the average ratio of running time as a function of the size map \( n \). Comparative analysis of the data corresponding to Fig. 6-7 shows the following:
1. The values of the average path for the algorithms 1 and 2’ at $X = 500$ and 3 are very close, and the algorithm 2 at $X = 100$ is bigger. For example, for the case of $n = 746$ corresponding to table 2, we have:

$$d_1 = 244.46; \frac{d_2}{d_1} = 1.34; \frac{d_2'}{d_1} = 1.11; \frac{d_3}{d_1} = 1.15;$$

where the subscripts correspond to the numbers of algorithms, and for the algorithm 2 there are two indexes: 2 for $X = 100$ and 2’ for $X = 500$.

![Figure 6. Average ratio of lengths paths](image1)

![Figure 7. Average ratio of running time](image2)

2. The biggest average time algorithm indicates 2’, and the smallest time is algorithm 1.

In case of $n = 746$, for example,

$$t_1 = 216.39 \text{ c}; \frac{t_2}{t_1} = 2.79; \frac{t_2'}{t_1} = 10.82; \frac{t_3}{t_1} = 6.62;$$

4. CONCLUSIONS

The experiments with other model configurations maps showed same results and suggest that all considered algorithms provide comparable accuracy, but the running time 1 (result of algorithm 1) is smaller than that of other algorithms, and it is more preferable in combination of two characteristics: the operating time and accuracy.

However, the asymptotic complexity $O(n^4)$ of the proposed algorithm is still big. Our reduction can be expected by optimizing the most time-consuming steps of the algorithm, as well as through their implementation on parallel structures. For example, graphics accelerators and heterogeneous computing structures [8], [9] should be used.

REFERENCES


7. Скворцов А.В. Построение объединения, пересечения и разности произвольных многоугольников в среднем за линейное время с помощью триангуляции // Вычислительные методы и программирование, 2002, Т. 3. С. 116-123.


THUẬT TOÁN GẦN ĐƯỢC ĐỊNH VỊ RÔ BỐT DI ĐỘNG SỬ DỤNG CỬA SỔ TRONG BẢN ĐỒ ĐA GIÁC

Dao Duy Nam1,2, Nguyen Quoc Huy1,3*

1SaigonTech, SaigonTech Tower, Lot 14, Quang Trung Software City, District 12, HCMC,
2High school for the gifted, VNUHCM, 153 Nguyen Chi Thanh, District 5, HCMC
3Saigon University, 273 An Duong Vuong, District 5, HCMC

*Email: namdd@saigontech.edu.vn, huy.nq@saigontech.edu.vn

TÓM TÁT


Từ khóa: Hình học máy tính, kỹ thuật rõbot; định vị rõbot, phủ đa giác, độ phức tạp thuật toán, thuật toán gần đúng