Localization with Few Distance Measurements

Dan Halperin Steven M. LaValle Barak Ugav

September 13, 2022

Abstract

Given a polygon W, a *depth sensor* placed at point $p = (x, y)$ inside W and oriented in direction θ measures the distance $d = h(x, y, \theta)$ between p and the closest point on the boundary of W along a ray emanating from p in direction θ. We study the following problem: Give a polygon W, possibly with holes, with n vertices, preprocess it such that given a query real value $d \geq 0$, one can efficiently compute the preimage $h^{-1}(d)$, namely determine all the possible poses (positions and orientations) of a depth sensor placed in W that would yield the reading d. We employ a decomposition of $W \times S^1$, which is an extension of the celebrated trapezoidal decomposition, and which we call *rotational trapezoidal decomposition* and present an efficient data structure, which computes the preimage in an output-sensitive fashion relative to this decomposition: if k cells of the decomposition contribute to the final result, we will report them in $O(k+1)$ time, after $O(n^2 \log n)$ preprocessing time and using $O(n^2)$ storage space. We also analyze the shape of the projection of the preimage onto the polygon W; this projection describes the portion of W where the sensor could have been placed. Furthermore, we obtain analogous results for the more useful case (narrowing down the set of possible poses), where the sensor performs two depth measurement from the same point p, one in direction θ and the other in direction $\theta + \pi$. These problems are inspired by *localization* questions in robotics, where we are given a map of the environment, a robot is placed in an unknown pose, and we wish to determine where the robot is. While localization is often carried out by exploring the full *visibility polygon* of a sensor placed at a fixed point of the environment, the approach that we propose here opens the door to sufficing with only few depth measurements, which is advantageous as it allows for usage of inexpensive sensors and could also lead to savings in storage and communication costs.

1 Introduction

A depth sensor that provides a single distance measurement is placed inside a known polygonal workspace W with n vertices. Consider the depth mapping $d = h(x, y, \theta)$ in which $(x, y) \in W$ is the position of the sensor and $\theta \in S^1$ (namely $[0, 2\pi)$) is the direction of the ray emanating from the sensor and along which the distance to the workspace boundary is measured. Thus the configuration space (C-space for short), namely the parametric space of sensor placements, is three-dimensional.

We wish to devise a data structure such that, after preprocessing the workspace, will efficiently answer the following queries: Given a distance measurement d as query, report all the possible sensor configurations that could yield such a measurement. In other words, we aim to compute h's preimage $h^{-1}(d) = \{(x, y, \theta) \in W \times S^1 \mid d = h(x, y, \theta)\}$.

Determining a sensor's location is one of the main tasks of sensor fusion systems. Examples are ubiquitous, including Global Positioning Systems (GPS), celestial navigation, trilateration, camera pose estimation, and localization of autonomous vehicles. Our problem is motivated mostly by the last one, in which a mobile robot has a complete map of its environment and must use sensing to determine its configuration (typically, position and orientation) with respect to the map. This so-called *robot localization* problem is crucial to robotics and has been researched for several decades. There are both *incremental* versions, in which the configuration estimate was known a moment ago and must be updated based on recent motions and sensor readings, and the *kidnapped robot problem* [6], in which the robot must determine its configuration "from scratch". The second case resembles our problem and could correspond to a freshly deployed or rebooted robot, or one that experiences failures of more complex sensor systems. There are both an *active* variant [5, 10, 11], in which the robot must determine where to move next to quickly reduce uncertainty, and *passive* variant, in which the sensor fusion system works with whatever sensing and motion data are available. Passive approaches mainly address stochastic uncertainties [7] and are typically integrated with mapping to obtain SLAM (simultaneous localization and mapping) [4, 12]. Our work considers the passive localization case and stands out from previous work in that it uses much less sensor data. Most robot localization methods are based on imaging sensors, such as cameras and Lidars, that provide a dense collection of measurements [13]. From a computational geometry viewpoint, these may be considered as providing visibility polygons (see [8]), whereas our sensor analogously provides only a single visibility ray with each measurement.

Localization with only few depth measurements is advantageous over exploring the full visibility polygon in that it allows for using inexpensive sensors and requires less storage space and communication. Our current work. which focuses on one or two measurements, is a step forward in this direction. However, it still falls short of giving a full localization answer, which in general requires at least three measurements.

Contributions. Let W be a polygon in the plane, possibly with holes, and having n vertices. (i) We introduce an extension of the well-known planar trapezoidal decomposition (see, e.g., [2, Chapter 6]) to the three-dimensional space $W \times S^1$. The decomposition, which we refer to as *rotational trapezoidal decomposition* (RTD, for short), has $O(n^2)$ cells of constant complexity each, and it can be computed in $O(n^2 \log n)$ time. RTD is easier to construct than the standard three-dimensional vertical decomposition [3, 9] and it suits the problem at hand better, as the imaginary walls of the RTD are parallel to the measurement direction. We use this decomposition to build a data structure to answer the preimage queries—given a real value $d \geq 0$, determine $h^{-1}(d)$—in $O(k+1)$ time where k is the number of cells of the RTD that contain part of the answer. The preprocessing time remains $O(n^2 \log n)$ and the reuiqred storage is $O(n^2)$. (ii) We analyze the precise shape of the projection of the preimages $h^{-1}(d)$ onto the workspace W. These projections could also serve as an answer to a measurement query as they delineate the regions in the plane (positions only) where the sensor could be. (iii) We then proceed to devise a data structure for more elaborate queries: Given two antipodal depth measurements of a sensor from the same (unknown) point in W in the (unknown) directions θ and $\theta + \pi$, determine all the poses of the sensor that would yield these measurements. After $O(n^2 \log n)$ preprocessing time and using $O(n^2)$, this data structure answers the antipodal queries in $O(k+1)$ time where k is the number of cells of the RTD that contain part of the answer.

2 Data Structure for Single Distance Measurement

We describe a data structure that for a given sensor reading d, efficiently determines all the sensor placements that would result in this reading. We present the construction of the data structure in two stages. In Section 2.1 we describe a decomposition of the three-dimensional C-space of the sensor into cells of constant descriptive complexity each, such that within each cell C, the sub-region $C_d = \{(x, y, \theta) \in C \mid d = h(x, y, \theta)\}$ also has constant descriptive complexity. We then show, in Section 2.2, how to maintain the cells of the decomposition in a search structure, such that given a query d we can easily retrieve all the cells C for which $C_d \neq \emptyset$, in time proportional to their number.

2.1 Rotational Trapezoidal Decomposition

Suppose for the purpose of exposition that we know that the sensor measures its distance in the upward vertical direction, namely $\theta = \pi/2$. We apply trapezoidal (vertical) decomposition [2, Chapter 6] to the workspace. Given a measurement d, we can easily compute for each trapezoid[1] in the decomposition if there are any points in it that are at vertical distance d from the top edge of the trapezoid. These points lie along a segment parallel to the top edge, and the desired answer is the union all such segments over all trapezoids. A trapezoid is identified by its ceiling and floor (namely top and bottom edges, respectively), and the left and right vertices, which define the left and right artificial vertical edges. See Figure 1 for an illustration.

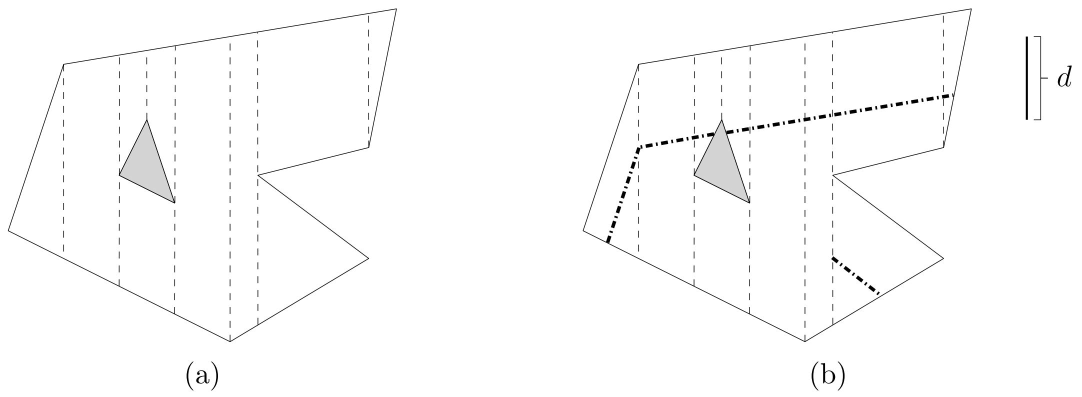


Figure 1: Example of a vertical decomposition of a polygon workspace with a single hole (a) and the line segments (bold, dotted) along which the sensor may be when it reads the distance d in the upward vertical direction (b).

To support different orientations of measurement, we need to apply a similar procedure to the scene for every orientation. For each orientation we decompose the workspace such that the imaginary walls are drawn in the respective orientation, and return the union of all the results. As we let the orientation θ of the measuring ray vary a little, and accordingly decompose the scene in the direction of the measurement, we notice that the

[1]Some of the two-dimensional cells in the decomposition may be triangles; for brevity we will also refer to them in this context as trapezoids.

trapezoids remain similar (in a sense to be made precise shortly) unless they undergo some combinatorial change. As the direction of measurement changes, the trapezoid changes continuously, and we refer to the union of these trapezoids as a three-dimensional cell, or *cell* for short. Since throughout the change of orientation the ceiling and floor edges, as well as the vertices determining the left and right boundaries of the trapezoid remain the same we use the following notation: A cell C is identified by its ceiling and floor (namely top and bottom edges, respectively), e_t^C, e_b^C, and the left and right vertices, v_l^C, v_r^C, which define the left and right artificial edges (parallel to the direction of measurement). The two limiting vertices, v_l^C, v_r^C, may be endpoints of e_t^C, e_b^C, or some other vertices. See Figure 2 for an illustration.

Now, in addition to describing a cell by its floor and ceiling, and the two vertices that define the two artificial side edges, we also add the first and last orientations where the trapezoid exists (in the combinatorial sense, namely having the same edges and vertices defining it). These cells constitute a decomposition of the C-space of the sensor. With this description, there are only $O(n^2)$ unique cells, and we can compute each cell's exact boundary as a function of the orientation θ of the sensor.

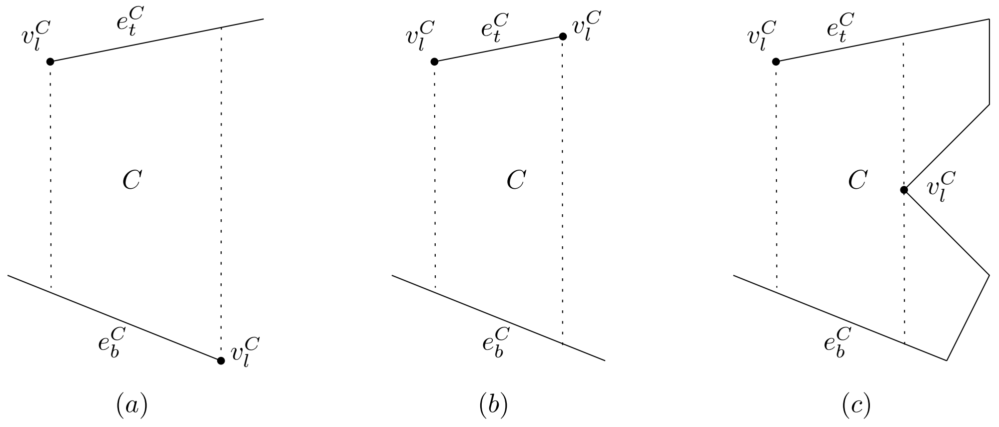


Figure 2: Examples of limiting left and right vertices of a trapezoidal cell C.

We construct these cells by simultaneously performing a radial sweep around all the vertices of the workspace, where each vertex is the origin of a sweeping ray, all rays point at the same direction and are rotated together. For each of these sweeps, we maintain a balanced binary search tree, which contains all the edges of the workspace that the sweeping ray intersects (similar to standard radial plane sweep). An event occurs when one of the rays hits a vertex. We compute all the $O(n^2)$ events in advance and radially sort all of them together (by the angle θ). In addition, each ray stores at all times (angles) the cells on both its sides, allowing us to keep track of all the existing trapezoids of the decomposition at the current orientation. During the handling of an event, where two vertices align along the sweeping ray (its origin and another vertex), if they are visible from one another, namely the open line segment connecting them lies in the interior of the workspace, cells should be created or terminated.

There are two types of events along the radial sweep:

- Two vertices of the same edge are incident to a rotating ray. See Figure 3 for an illustration. The triangular cell that contained both vertices on its boundary is "squeezed" and terminated, and a new triangular cell is created. The cell sharing an artificial edge with the triangular cell also terminates and a new one is created due to change of one of its limiting vertices. In total, two cells are terminated and two are created.

- Two vertices of two different edges are incident to a rotating ray. See Figure 4 for an illustration. The middle cell is "squeezed" and terminated, and a new middle cell is created. The cells from both sides of the connecting ray terminate and new ones are created due to change in the limiting vertices. In total, three cells are terminated and three are newly created.

In both event types, the top and bottom edges and left and right vertices are known locally from the information maintained by the rays or the terminated cells. During handling of an event we terminate or create a constant number of cells. As there are $O(n^2)$ events, there are also $O(n^2)$ cells in total.

To start the process, trapezoidal decomposition in the horizontal direction (namely, with $\theta = 0$), is performed, requiring $O(n \log n)$ time and starting $\Theta(n)$ cells. We let θ vary in the range $[0, 2\pi)$. Notice that the trapezoidal decomposition at $\theta = 0$ artificially cuts cells into two; this however does not affect the analysis that follows or the asymptotic resources required by the algorithm. In total, we construct these $O(n^2)$ cells and their angle intervals in $O(n^2 \log n)$ time, while using $O(n^2)$ space. We will refer to such decomposition as *rotational trapezoidal decomposition*, or RTD for short. It is easy to see that $\Omega(n^2)$ can be created on so the bound on the maximum number of cells is tight.

Proposition 1. *Given a polygonal workspace W with a total of n vertices, the three-dimensional configuration space $W \times S^1$ is partitioned by the Rotational Trapezoidal Decomposition (RTD) described above into $O(n^2)$ cells, each of constant descriptive complexity. The construction of the partitioning takes $O(n^2 \log n)$ time and requires $O(n^2)$ storage space. The size of the decomposition is worst-case optimal.*

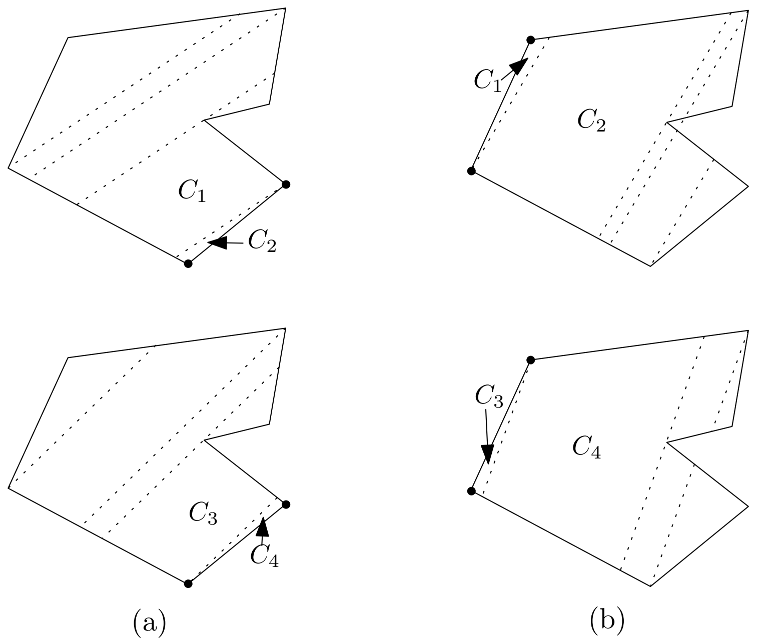


Figure 3: Type I events: two vertices of the same edge align along the rotating ray. Rays are rotating counterclockwise. The event vertices are drawn as small black discs. Each column depicts one example, before (top) and after (bottom) the event.

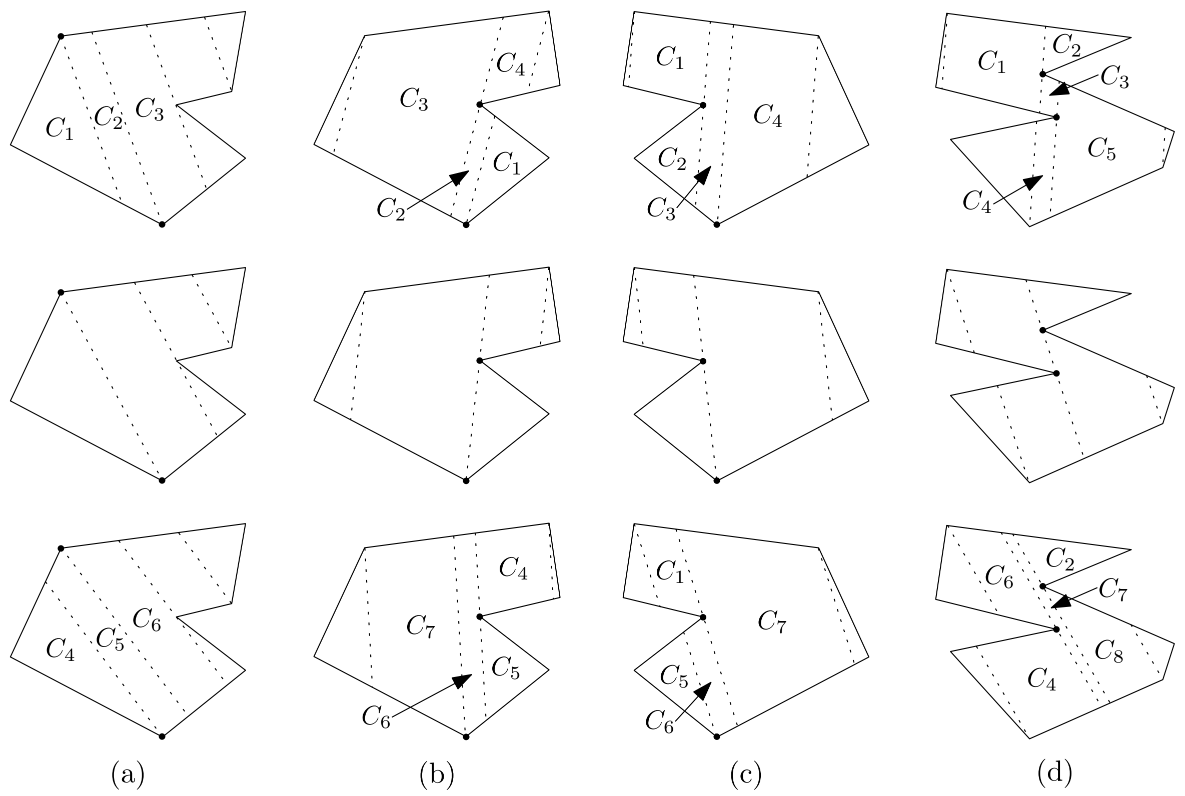


Figure 4: Type II events: two vertices of two distinct edges align along the rotating ray. Rays are rotating counterclockwise. The event vertices are drawn as small black discs. Each column depicts one example, before (top), during (middle), and after (bottom) the event.

2.2 Storing the Cells for Efficient Placement Retrieval

We wish to store the cells of the decomposition of the C-space of the sensor in a data structure such that given a distance measurement d, we can efficiently retrieve all the cells that have non-empty set of candidate placements for this measurement. To this end we define the maximal opening O^C_{max} of a cell C, which is the maximal-length θ-oriented segment that can be placed in a θ-cross-section of a cell C, over all possible valid values of θ of C.

Each cell C prevails through an interval of θ values, calculated during the sweep—we denote this interval by $\Theta^C = [\theta^C_{begin}, \theta^C_{end})$. Given a cell C and an angle $\theta \in \Theta^C$, we can easily construct the trapezoid that is the cross-section of C at θ. The top left and top right vertices (which are not necessarily vertices of the workspace) of the trapezoid both lie on e^C_t. We can compute their x values as a function of θ, denoted $x^C_{tl}(\theta), x^C_{tr}(\theta)$. We define the maximal opening of a cell formally as follows; see Figure 5 for an illustration.

Definition 1. *The* opening of a cell C *at angle* θ *and at a point* $e^C_t(x)$ *on the top edge of the cell (whose* x*-coordinate is* x*), is the length of the intersection of the line with slope* θ *through the point* $e^C_t(x)$ *on the top edge of the cell, with the trapezoidal cross-section of* C *at angle* θ*, denoted* $O^C(\theta, x)$*. Let* the maximum opening of the cell *be the maximum value of the opening function for any valid value of* θ, x*:*

$$O^C_{max} = \max_{\theta \in \Theta^C} \max_{x \in [x^C_{tl}(\theta), x^C_{tr}(\theta)]} O^C(\theta, x)$$

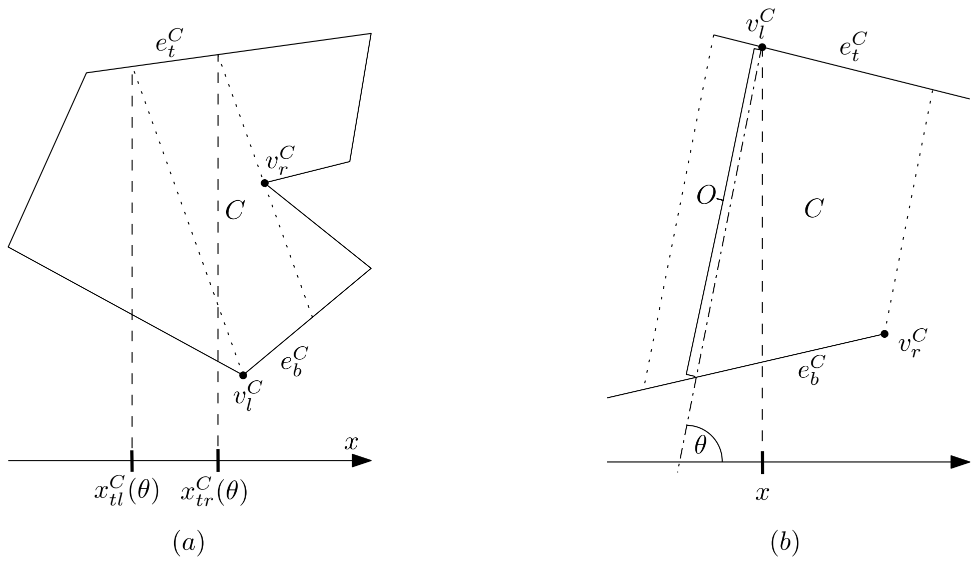


Figure 5: (a) The x values of the left and the right vertices of a cell's ceiling. (b) The opening size O as a function of x and θ.

In a preprocessing phase, we calculate O_{max}^C for each cell, and keep the cells in an array, where they are sorted in decreasing order by this value. Given a query measurement d, the cells that contain potential placements for this reading are exactly the cells for which $d \leq O_{max}^C$. We can find these cells by traversing the sorted cells in descending maximum opening order, resulting in $O(k+1)$ query time,[2] where k is the number of cells that contain workspace points of the desired answer. The calculation of the maximum opening takes constant time per cell. Since there are $O(n^2)$ cells overall, the entire preprocessing requires $O(n^2 \log n)$ time and $O(n^2)$ space, which is asymptotically subsumed by the construction of the decomposition (Section 2.1).

Theorem 1. *Given a polygonal workspace with a total of n vertices, the RTD cells of the configuration space $W \times S^1$ can be computed and sorted in decreasing order of their maximum opening in time $O(n^2 \log n)$ and $O(n^2)$ space. Then, given a query real parameter d, we can report all the cells that contain a possible configuration yielding depth measurement d in time $O(k+1)$, where k is the number of the relevant cells.*

This structure has a couple of advantages: (i) it is trivial to construct and implement, and (ii) it is worst-case optimal in storage space and query time *assuming the Rotational Trapezoidal Decomposition.* However, it should be asked what is the relationship between k to the output complexity of an optimal algorithm, as k is an outcome of the algorithm. A workspace can be constructed in which the number of RTD cells with non-empty result is greater by a factor of $O(n)$ than the complexity of an optimal algorithm, even if the workspace is simple (without holes). This question is for further research.

3 Sensor Positions for a Fixed Reading

Given a sensor reading d, we concern ourselves in the section with the shape and complexity of the locus of sensor *positions* that result in such a reading, namely ignoring the

[2]For $k = 0$ we still need to check the cell at the top of the list to find out that no relevant cells exist.

orientation of the sensor's ray. We believe that reporting the possible positions only is, although incomplete, a useful feedback that is more intuitive and easy to grasp by the user of a robot system. This in itself can serve as an answer to a query, in particular, when we wish to further process the regions using additional information about the potential placement of the sensor.

We first show in Section 3.1 how to compute for each three-dimensional cell C of the decomposition, the region $\Psi_d(C)$ of positions where the sensor could be, namely $\Psi_d(C)$ is the projection onto the xy-plane of C_d. Then, in Section 3.2 we discuss the presentation of the union of the regions $\Psi_d(C)$ over all cells C in the decomposition.

3.1 Sensor Positions for a Single Cell

Fix a cell C with top and bottom edges e_t^C, e_b^C, limiting vertices v_l^C, v_r^C which exist in an angle interval Θ^C. Recall that $x_{tl}^C(\theta), x_{tr}^C(\theta)$ denote the x value of the left and right top vertices of trapezoidal cross-section of C (which are not necessarily vertices of the workspace), respectively, at angle θ.

Let $p_l^C(\theta)$ denote the position of the sensor, where a distance measure d at angle θ will hit e_t^C at x value $x_{tl}^C(\theta)$, while ignoring other obstacles in the workspace. Specifically, the point which its x-coordinate is $x_{tl}^C(\theta) - d\cos\theta$ and y-coordinate is $e_t^C(x_{tl}^C(\theta)) - d\sin\theta$, where $e_t^C(\cdot)$ is the line equation of the top edge. Similarly denote by $p_r^C(\theta)$ the placement of the sensor that will result in hitting e_t^C at $x_{tr}^C(\theta)$. Recall that $\Psi_d(C)$ denotes the region $\{(x,y)|(x,y,\theta) \in C \text{ and } h(x,y,\theta) = d\}$, namely the positions of the sensor in the workspace, where the sensor configuration is in C and there is a sensor reading d. We wish to trace the curves drawn by $p_l^C(\theta)$ and $p_r^C(\theta)$ as θ varies in the range Θ^C. These curves are portions of the boundary of the region $\Psi_d(C)$. The type of the curve that the function $p_l^C(\theta)$ (respectively, $p_r^C(\theta)$) draws, depends on v_l^C (respectively, v_r^C), the vertex defining the left (respectively, right) wall of the cell. We describe it here for v_l^C. The description for v_r^C is analogous.

- The vertex v_l^C defining the left wall of the trapezoid/cell lies on e_t^C. See Figure 6(a) for an illustration. The top left vertex, and therefore $x_{tl}^C(\theta)$, are fixed (i.e., independent of θ), and the curve is a circular arc of radius d around v_l^C: $v_l^C - (d\cos\theta, d\sin\theta)$.
- The vertex v_l^C defining the left wall of the trapezoid/cell does *not* lie on e_t^C. See Figure 6(b,c) for illustrations. The curve traced by $p_l^C(\theta)$ is a conchoid of Nicomedes; see Appendix D for details.

For any angle $\theta \in \Theta^C$, all points on the line segment $(p_l^C(\theta), p_r^C(\theta))$ are points the robot can be at and measure d at e_t^C (ignoring other obstacles). Together with the types of the curves traced by $p_l^C(\theta), p_r^C(\theta)$, we have an exact description of the region of the workspace containing all the points the robot might be at, while its configuration is in the cell C.

If the interval Θ^C contains the angle perpendicular to the top edge, denoted θ_p^C, we divide it into two sub-intervals, one with angles that are greater than the perpendicular angle and the other with smaller angles, $[\theta_{begin}^C, \theta_p^C), [\theta_p^C, \theta_{end}^C)$. A single shape per sub-interval will be created. If θ_p^C is not included in Θ^C, a single interval $[\theta_{begin}^C, \theta_{end}^C)$ and a single corresponding shape is used.

We handle each of the two sub-intervals separately. For each sub-interval (or for the entire interval Θ^C when no splitting is required) $[\theta_1, \theta_2)$, we describe all the points

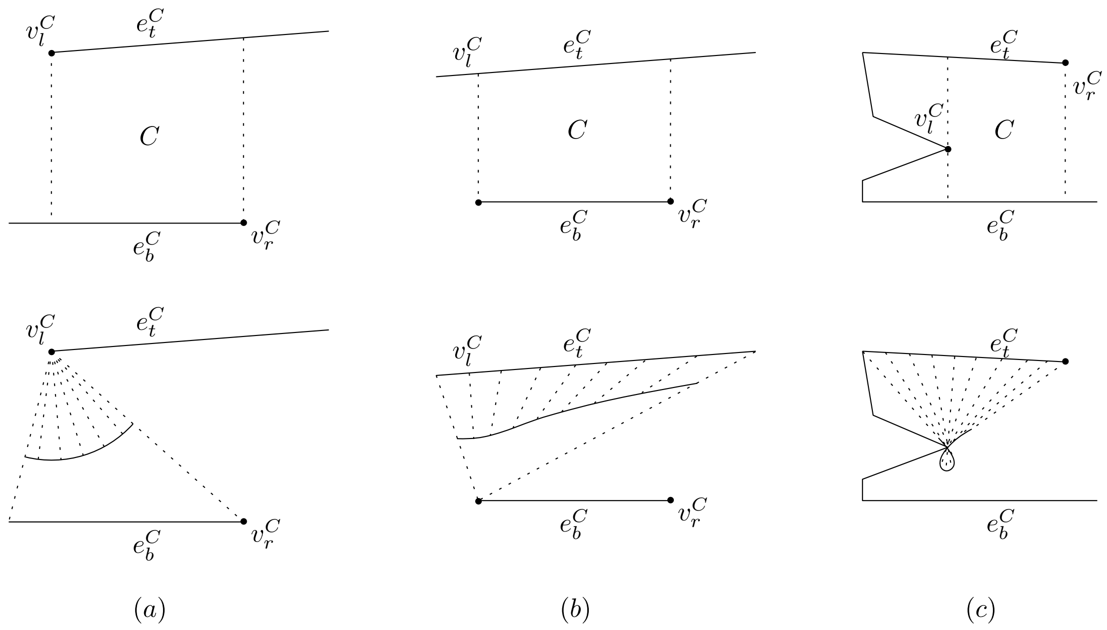


Figure 6: (a) Type I curve: when the limiting vertex lies on the cell's ceiling, the curve is a circular arc. (b)(c) Type II curve: when the limiting vertex does not lie on the cell's ceiling, curve is a conchoid of Nicomedes.

on the line segments $(p_l^C(\theta), p_r^C(\theta))$ for any $\theta \in [\theta_1, \theta_2)$ by a shape defined by the vertices $(v_1, v_2, v_3, v_4) = (p_l^C(\theta_1), p_l^C(\theta_2), p_r^C(\theta_2), p_r^C(\theta_1))$ and the edges p_l^C, p_r^C between $(v_1, v_2), (v_3, v_4)$ respectively and straight line edges between $(v_2, v_3), (v_4, v_1)$. The union of the two shapes is the desired two-dimensional region in the workspace the sensor might be in. The only obstacle we need to consider is the bottom edge e_b^C of the trapezoid, and we can simply intersect the result with the upper half-plane defined by the line through e_b^C to get the final result of the cell. See Figure 7 for illustrations.

3.2 The Overall Region of Potential Sensor Positions for a Fixed Reading

Given a sensor reading d, we retrieve all the cells with non-empty potential placements using the sorted list of Section 2.2. For each such cell C we compute the region $\Psi_d(C)$.

Alternatively, we may wish to return the (two-dimensional) union of the regions $\Psi_d(C)$, for all the cells that we retrieved. See Figure 8 for an illustration. Each region $\Psi_d(C)$ has constant descriptive complexity, and if we retrieved k such regions, then their union, can be easily computed in time $O((k + m) \log(k + m))$, where m is the complexity of the union. In the worst case $m = O(k^2)$. It might be the case that special properties of the regions $\Psi_d(C)$ and their juxtaposition could be used to show that the complexity of the union is $o(k^2)$. We leave this as an open problem for further research.

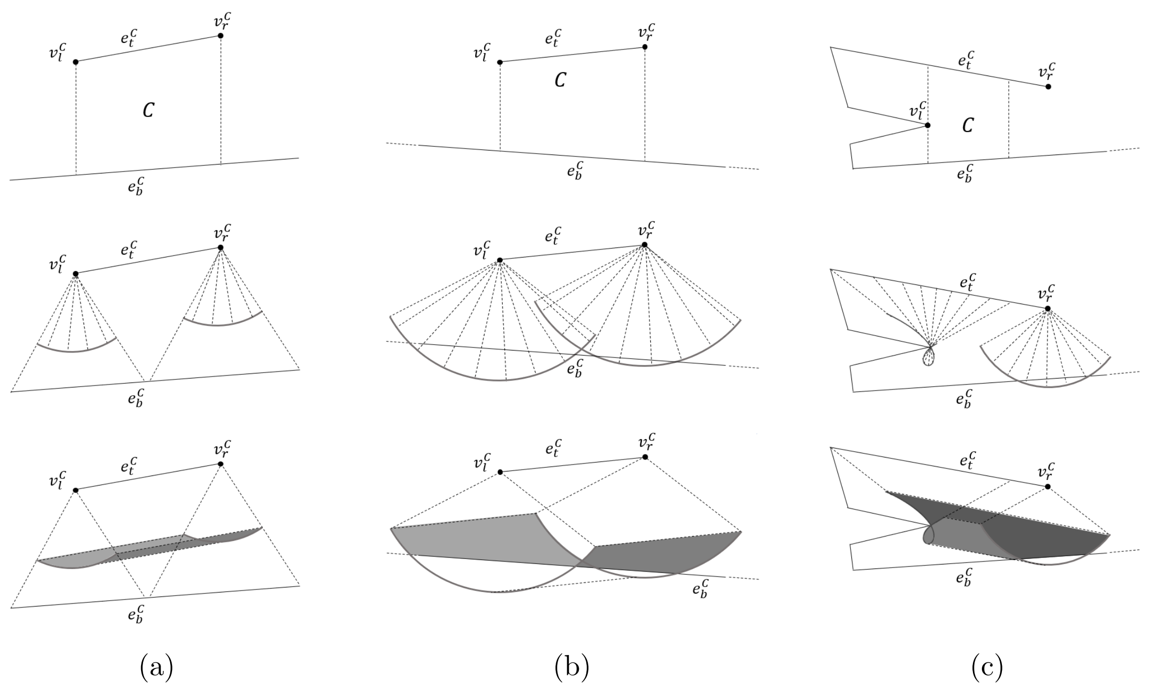


Figure 7: Examples of the planar projection of possible configurations of a single cell with a fixed reading d: (a) both limiting vertices lie on the top edge, the resulting curves are arcs and do not intersect the bottom edge (b) the limiting vertices and arc curves are similar to (a), but the result intersects the bottom edge (c) one limiting vertex lies on the top edge and one is not, resulting in an arc and a conchoid.

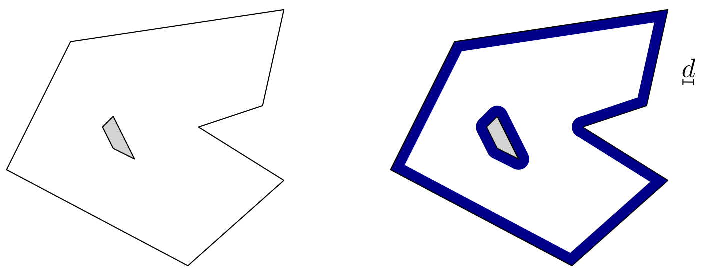


Figure 8: Given the workspace of the left and a distance measurement d, the union of the results of all cells is shown on the right.

4 Two Antipodal Distance Measurements

Another variant of the problem that might be considered, is one which the sensor takes two different distance measurement from the same location in opposing directions. We obtain one measurement d_1 in (unknown) direction θ_1 and a second measurement d_2 in direction $\theta_1 + \pi$. A nice, mitigating, property of this setting is that the two distance measurements are taken to two *distinct* edges of the workspace.

We apply the same decomposition of the sensor C-space as described in Section 2, where the orientation θ is the orientation of the first measurement. For a fixed orientation θ, the cross-section of a cell C contains one potential sensor placement (assuming the top and bottom edges of the cell are not parallel), and we can calculate it by solving $O^C(\theta, x) = d_1 + d_2$, and extracting the point along the segment obtaining this opening that is at distance d_1 from the top edge (and hence at distance d_2 from the bottom edge).

For each such cell C we compute the locus $\Gamma(C)$ of potential sensor placements, which is a curve parameterized by the orientation of the first measurement.

In order to store the cells of the decomposition so as to be able to efficiently retrieve the cells C with non-empty loci $\Gamma(C)$, we now need to also compute the minimum opening of a cell O^C_{min}, which is defined analogously to O^C_{max}. Given a pair of antipodal measurements (d_1, d_2), a cell C has a non-empty set of loci if and only if $O^C_{min} \leq d_1 + d_2 \leq O^C_{max}$. Each cell is associated with an interval $[O^C_{min}, O^C_{max}]$, and we construct an interval tree [2, Chapter 10] over these $O(n^2)$ intervals. Given the measurements (d_1, d_2), we extract all the intervals that contain the value $d_1 + d_2$, and their corresponding cells contribute potential placements to the final answer.

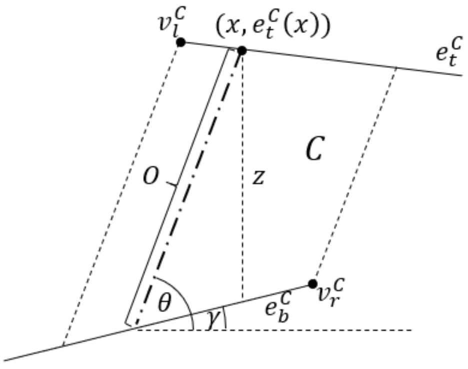
The loci $\Gamma(C)$ constitute an arc of an ellipse with its center at the (possibly imaginary) intersection of the lines supporting the top and bottom edges of the cell; see Appendix E for details. After prepossessing in $O(n^2 \log n)$ time and using $O(n^2)$ space, a query can be answered in $O(\log n + k)$ time, where k is the number of cells C with non-empty $\Gamma(C)$.

Theorem 2. *A query of two antipodal distance measurements can be answered in time $O(\log n + k)$ where k is the number of RTD cells C with non-empty result, using a data structured constructed in time $O(n^2 \log n)$ and $O(n^2)$ space.*

The natural question to ask is how does this approach performs compared to the output of an optimal algorithm, rather than k which is an outcome of the algorithm. A workspace can be constructed in which the number of RTD cells with non-empty result is greater by a factor of $O(n)$ than the complexity of an optimal algorithm, even if the workspace is simple (without holes). This question is for further research.

A Calculating $O^C(\theta, x)$

For each RTD cell C, the opening function $O(\theta, x)$ defined at Definition 1, can be analytically described. Given a cell $C(e_t^C, e_b^C, v_l^C, v_r^C, \theta^C)$, assume w.l.o.g that the top and bottom edges are not vertical, and denote by $e_t^C(x) = xm_{e_t} + b_{e_t}$ the equation for the top edge and equivalently for the bottom edge $e_b^C(x) = xm_{e_b} + b_{e_b}$.



We can look at the triangle constructed from O, z, e_b^C. The angle opposite of e_b^C is $\frac{\pi}{2} - \theta$, the angle opposite of z is $\theta - \gamma$ and the angle opposite of O is $\frac{\pi}{2} + \gamma$. Using the law of sines:

$$O(\theta, x) = \frac{z \sin(\frac{\pi}{2} + \gamma)}{\sin(\theta - \gamma)} = \frac{z \cos\gamma}{\sin(\theta - \gamma)} = \frac{z\cos\gamma}{\sin\theta\cos\gamma - \cos\theta\sin\gamma}$$

$$= \frac{z\frac{1}{\sqrt{1+m_{e_b}^2}}}{\sin\theta\frac{1}{\sqrt{1+m_{e_b}^2}} - \frac{m_{e_b}}{\sqrt{1+m_{e_b}^2}}\cos\theta} = \frac{z}{\sin\theta - m_{e_b}\cos\theta} = \frac{e_t^C(x) - e_b^C(x)}{\sin\theta - m_{e_b}\cos\theta}$$

$$= \frac{x(m_{e_t} - m_{e_b}) + b_{e_t} - b_{e_b}}{\sin\theta - m_{e_b}\cos\theta}$$

B The Endpoints of a Trapezoid: $x_{tl}^C(\theta), x_{tr}^C(\theta)$

For each RTD cell C, which exists in an angle interval Θ^C, an exact trapezoid description can be calculated analytically for any fixed and θ. The top and bottom edge are fixed, and the two artificial edges are at angle θ. It remains to express the vertices created by the intersections of the artificial edges and the top edge, denoted v_{tl}, v_{tr} with x values $x_{tl}^C(\theta), x_{tr}^C(\theta)$, as defined at *Section* 2.2. Given a cell $C(e_t^C, e_b^C, v_l^C, v_r^C, \theta^C)$, assume w.l.o.g that the top and bottom edges are not vertical, and denote by $e_t^C(x) = xm_{e_t} + b_{e_t}$ the equation for the top edge and equivalently for the bottom edge $e_b^C(x) = xm_{e_b} + b_{e_b}$. Any point (x, y) on the left artificial edge can be expressed as:

$$y = \tan\theta(x - x_{v_{tl}}) + y_{v_{tl}}$$

To calculate the intersection with the top edge, we substitute the top edge equation:

$$m_{e_t}x + b_{e_t} = x\tan\theta - x_{v_{tl}}\tan\theta + y_{v_{tl}}$$

$$x(m_{e_t} - \tan\theta) = y_{v_{tl}} - x_{v_{tl}}\tan\theta - b_{e_t}$$

$$x = \frac{y_{v_{tl}} - x_{v_{tl}} \tan\theta - b_{e_t}}{m_{e_t} - \tan\theta}$$

Similar calculation can be done to the right top vertex. In conclusion:

$$x^C_{tl}(\theta) = \frac{y_{v_l} - x_{v_l} \tan\theta - b_{e_t}}{m_{e_t} - \tan\theta} \quad x^C_{tr}(\theta) = \frac{y_{v_r} - x_{v_r} \tan\theta - b_{e_t}}{m_{e_t} - \tan\theta}$$

C Solving $O^C(\theta, x) = z$

For each RTD cell C, solving $O(\theta, x) = z$ for a specific value z is required by the Section 4 with $z = d_1 + d_2$. Given a cell $C(e^C_t, e^C_b, v^C_l, v^C_r, \theta^C)$, assume w.l.o.g that the top and bottom edges are not vertical, and denote by $e^C_t(x) = xm_{e_t} + b_{e_t}$ the equation for the top edge and equivalently for the bottom edge $e^C_b(x) = xm_{e_b} + b_{e_b}$. Using the opening equation:

$$O^C(\theta, x) = z$$

$$\frac{e^C_t(x) - e^C_b(x)}{\sin\theta - m_{e_b}\cos\theta} = z$$

$$e^C_t(x) - e^C_b(x)) = z(\sin\theta - m_{e_b}\cos\theta)$$

$$m_{e_t}x + b_{e_t} - m_{e_b}x - b_{e_b} = z(\sin\theta - m_{e_b}\cos\theta)$$

$$x = \frac{z(\sin\theta - m_{e_b}\cos\theta) + b_{e_b} - b_{e_t}}{m_{e_t} - m_{e_b}}$$

This result is the x value of the possible measurement on the top edge, to get the possible robot location one should subtract $(d_1 \cos\theta, d_1 \sin\theta)$.

D Conchoid of Nicomedes representing p^C_l, p^C_r

For each RTD cell C, when computing the result of possible configuration within the cell, the possible curves of p^C_l, p^C_r defined in Section 3.1 are arcs or conchoids. Analytical description of the arcs is trivial. Analytical description of the conchoids is described in this section. Given a cell $C(e^C_t, e^C_b, v^C_l, v^C_r, \theta^C)$, assume w.l.o.g that the top and bottom edges are not vertical, and denote by $e^C_t(x) = xm_{e_t} + b_{e_t}$ the equation for the top edge and equivalently for the bottom edge $e^C_b(x) = xm_{e_b} + b_{e_b}$.

The conchoid of Nicomedes is driven from a fixed point q, a straight line l and a length d, where for every line through q that intersects l, the two points on the line which are d from the intersection are on the conchoid. In our case, q is the limiting vertex, the straight line is the top edge of the cell and d is the query measurements.

Equation for classic conchoid, $q = (0, 0)$ $l : x = a$

$$(x - a)^2(x^2 + y^2) = d^2x^2$$

Equation for general conchoid, $q = (x_0, y_0)$ l : at angle t, distance a from q:

$$s = \sin t, \quad c = \cos t$$

$$((x-x_0)c-(y-y_0)s-a)^2(((x-x_0)c-(y-y_0)s)^2+((x-x_0)s+(y-y_0)c)^2) = d^2((x-x_0)c-(y-y_0)s)^2$$

In our case, l is the top edge, and its angle is defined by its slope:

$$t = \tan^{-1}(m_{e_t}), \quad \sin\tan^{-1}(m_{e_t}) = \frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}}, \quad \cos\tan^{-1}(m_{e_t}) = \frac{1}{\sqrt{1+m_{e_t}^2}}$$

By using the above identities, we get:

$$\left((x-x_0)\frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}} - (y-y_0)\frac{1}{\sqrt{1+m_{e_t}^2}} - a\right)^2 \left(\left((x-x_0)\frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}} - (y-y_0)\frac{1}{\sqrt{1+m_{e_t}^2}}\right)^2 + \left((x-x_0)\frac{1}{\sqrt{1+m_{e_t}^2}} + (y-y_0)\frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}}\right)^2\right) = d^2\left((x-x_0)\frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}} - (y-y_0)\frac{1}{\sqrt{1+m_{e_t}^2}}\right)^2$$

$$\frac{1}{1+m_{e_t}^2}\left((x-x_0)\frac{m_{e_t}}{\sqrt{1+m_{e_t}^2}} - (y-y_0)\frac{1}{\sqrt{1+m_{e_t}^2}} - a\right)^2 \left((m_{e_t}(x-x_0) - (y-y_0))^2 + ((x-x_0) + m_{e_t}(y-y_0))^2\right) = \frac{1}{1+m_{e_t}^2} d^2 (m_{e_t}(x-x_0) - (y-y_0))^2$$

$$\left(\frac{m_{e_t}(x-x_0) - (y-y_0)}{\sqrt{1+m_{e_t}^2}} - a\right)^2 \left(m_{e_t}^2(x-x_0)^2 - 2m_{e_t}(x-x_0)(y-y_0) + (y-y_0)^2 + (x-x_0)^2 + 2m_{e_t}(x-x_0)(y-y_0) + m_{e_t}^2(y-y_0)^2\right) = d^2(m_{e_t}(x-x_0) - (y-y_0))^2$$

$$\left(\frac{m_{e_t}(x-x_0) - (y-y_0)}{\sqrt{1+m_{e_t}^2}} - a\right)^2 \left((x-x_0)^2 + (y-y_0)^2\right)(1+m_{e_t}^2) = d^2(m_{e_t}(x-x_0) - (y-y_0))^2$$

$$\left(m_{e_t}(x-x_0) - (y-y_0) \pm a\sqrt{1+m_{e_t}^2}\right)^2 \left((x-x_0)^2 + (y-y_0)^2\right) = d^2(m_{e_t}(x-x_0) - (y-y_0))^2$$

The term with $\pm$ sign is determined by relation between q and l. If l is above q, we use the plus sign, if l is below q, we use the minus sign. In total the equation that represent the curve p_l^C within a cell is (can be defined similarly for p_r^C):

$$\left(m_{e_t}(x-x_{v_l}) - (y-y_{v_l}) \pm a\sqrt{1+m_{e_t}^2}\right)^2 \left((x-x_{v_l})^2 + (y-y_{v_l})^2\right) = d^2(m_{e_t}(x-x_{v_l}) - (y-y_{v_l}))^2$$

E Ellipse representing the loci $\Gamma(C)$

In Section 4 a variant of the problem with two antipodal measurements was discussed. Given two measurements d_1, d_2, and a fixed RTD cell, all positions of a sensor that measured d_1 at some angle θ_1 and d_2 at angle $\theta_2 = \theta_1 + \pi$ can be calculated analytically. The possible sensor position points are the points in which a line segment of length d_1, d_2 intersect at its endpoints the top and bottom edges. These points form a curve, a Glissette, which is an ellipse.

Glissettes are the curves trances out by a point carried by a curve, which is made to slide between given points or curves. In our case specifically, there is a fixed line which slides between two fixed lines (which are not necessary at right angles), and a carried point in the line (no necessary at the center). A solution for the case where the two fixed lines are at right angles was given at [1, p. 51]. We solve it for the general case.

For simplicity, assume one of the fixed lines is the x-axis, and it intersect the other fixed line at the origin, and the angle between them is α. If the moving line has a length $d1 + d2$, and the carried point as at distance $d1, d2$ from the line endpoints, the Glissette points satisfy the following for any angle ϕ in the **oblique** co-ordinates:

$$x \sin\alpha = f(\phi), \quad y \sin\alpha = f(\phi + \alpha)$$

Where $f(\cdot)$ is the **tangential polar equation**. In our case:

$$f(\phi) = \begin{cases} d_1 \cos\phi, & \text{if } \phi \in [0, \pi/2], \\ d_2 \cos\phi, & \text{else} \end{cases}$$

Again, these expressions represent the oblique co-ordinates, when transforming into our right co-ordinates, we get:

$$y = d_2 \cos(\phi + \alpha), \quad x = \frac{d_1 \cos\phi}{\sin\alpha} + \frac{y}{\tan\alpha}$$

The above expressions are correct for any angle ϕ, and we would like to express all the result points in a single equation. Next, we perform angle elimination:

$$\theta = \arccos(\frac{y}{d_2}) - \alpha$$

$$x = \frac{d_1 \cos(\arccos(\frac{y}{d_2}) - \alpha)}{\sin\alpha} + \frac{y}{\tan\alpha}$$

$$x = \frac{d_1(\cos\arccos\frac{y}{d_2}\cos\alpha + \sin\arccos\frac{y}{d_2}\sin\alpha)}{\sin\alpha} + \frac{y}{\tan\alpha}$$

$$x = \frac{d_1(\frac{y}{d_2}\cos\alpha \pm \sqrt{1 - \frac{y^2}{d_2^2}}\sin\alpha)}{\sin\alpha} + \frac{y}{\tan\alpha}$$

$$x = d_1(\frac{y}{d_2}\frac{\cos\alpha}{\sin\alpha} \pm \sqrt{1 - \frac{y^2}{d_2^2}}) + \frac{y}{\tan\alpha}$$

$$x = \frac{d_1}{d_2}\frac{y}{\tan\alpha} \pm d_1\sqrt{1 - \frac{y^2}{d_2^2}} + \frac{y}{\tan\alpha}$$

$$x - (1 + \frac{d_1}{d_2})\frac{y}{\tan\alpha} = \pm d_1\sqrt{1 - \frac{y^2}{d_2^2}}$$

$$x^2 - 2x(1 + \frac{d_1}{d_2})\frac{y}{\tan\alpha} + (1 + \frac{d_1}{d_2})^2\frac{y^2}{\tan^2\alpha} = d_1^2(1 - \frac{y^2}{d_2^2})$$

$$x^2 - 2\frac{1+\frac{d_1}{d_2}}{\tan\alpha}xy + (\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + \frac{d_1^2}{d_2^2})y^2 = d_1^2$$

$$\frac{x^2}{d_1^2} - 2\frac{1+\frac{d_1}{d_2}}{d_1^2\tan\alpha}xy + (\frac{(1+\frac{d_1}{d_2})^2}{d_1^2\tan^2\alpha} + \frac{1}{d_2^2})y^2 = 1 \tag{1}$$

The above equation already express all the points in the ellipse, but we would like to work with standard representation of rotated ellipse. Rotated ellipse equation centered at (x_0, y_0) at angle θ:

$$\frac{((x-x_0)\cos\theta + (y-y_0)\sin\theta)^2}{a^2} + \frac{((x-x_0)\sin\theta - (y-y_0)\cos\theta)^2}{b^2} = 1$$

We assumed the intersection is at the origin, therefore $x_0 = y_0 = 0$. Denote $\sin\theta = s, \cos\theta = c, \tan\theta = t$

$$\frac{(xc+ys)^2}{a^2} + \frac{(xs-yc)^2}{b^2} = 1$$

$$(\frac{c^2}{a^2} + \frac{s^2}{b^2})x^2 + 2cs(\frac{1}{a^2} - \frac{1}{b^2})xy + (\frac{s^2}{a^2} + \frac{c^2}{b^2})y^2 = 1$$

From Equation (1) and the above we can derive three equations with three unknowns:

$$\frac{1}{d_1^2} = \frac{c^2}{a^2} + \frac{s^2}{b^2} \tag{2}$$

$$-2\frac{1+\frac{d_1}{d_2}}{d_1^2\tan\alpha} = 2cs(\frac{1}{a^2} - \frac{1}{b^2}) \tag{3}$$

$$\frac{(1+\frac{d_1}{d_2})^2}{d_1^2\tan^2\alpha} + \frac{1}{d_2^2} = \frac{s^2}{a^2} + \frac{c^2}{b^2} \tag{4}$$

We can solve these equations to calculate a, b, θ.

from Equation (2):

$$\frac{1}{d_1^2} = \frac{c^2}{a^2} + \frac{s^2}{b^2} = \frac{c^2b^2 + s^2a^2}{a^2b^2}$$

$$a^2b^2 = d_1^2(c^2b^2 + s^2a^2)$$

$$a^2 = \frac{d_1^2c^2b^2}{b^2 - d_1^2s^2} \tag{5}$$

from Equation (3)

$$-2\frac{1+\frac{d_1}{d_2}}{d_1^2\tan\alpha} = 2cs(\frac{1}{a^2} - \frac{1}{b^2})$$

$$\frac{1+\frac{d_1}{d_2}}{csd_1^2\tan\alpha} = \frac{1}{b^2} - \frac{1}{a^2}$$

assigning a^2 using Equation (5)

$$\frac{1+\frac{d_1}{d_2}}{csd_1^2\tan\alpha} = \frac{1}{b^2} - \frac{b^2 - d_1^2s^2}{d_1^2c^2b^2}$$

$$\frac{1+\frac{d_1}{d_2}}{csd_1^2\tan\alpha} = \frac{d_1^2c^2 - b^2 + d_1^2s^2}{d_1^2c^2b^2}$$

$$\frac{1+\frac{d_1}{d_2}}{s\tan\alpha} = \frac{d_1^2 - b^2}{cb^2}$$

$$\frac{c(1+\frac{d_1}{d_2})}{s\tan\alpha} + 1 = \frac{d_1^2}{b^2}$$

$$b^2 = \frac{d_1^2}{\frac{c(1+\frac{d_1}{d_2})}{s\tan\alpha}+1} = \frac{d_1^2}{1+\frac{c}{s\tan\alpha}(1+\frac{d_1}{d_2})} \tag{6}$$

assigning b^2 using Equation (6) into Equation (5)

$$a^2 = \frac{d_1^2c^2b^2}{b^2 - d_1^2s^2} = \frac{d_1^2c^2}{1-\frac{d_1^2s^2}{b^2}} = \frac{d_1^2c^2}{1-\frac{d_1^2s^2}{\frac{d_1^2}{1+\frac{c}{s\tan\alpha}(1+\frac{d_1}{d_2})}}}$$

$$= \frac{d_1^2c^2}{1-s^2-\frac{cs}{\tan\alpha}(1+\frac{d_1}{d_2})} = \frac{d_1^2c^2}{c^2-\frac{cs}{\tan\alpha}(1+\frac{d_1}{d_2})} = \frac{d_1^2}{1-\frac{s}{c\tan\alpha}(1+\frac{d_1}{d_2})}$$

$$a^2 = \frac{d_1^2}{1-\frac{s}{c\tan\alpha}(1+\frac{d_1}{d_2})} \tag{7}$$

assigning a^2, b^2 using Equation (7) Equation (6) into Equation (3)

$$\frac{(1+\frac{d_1}{d_2})^2}{d_1^2\tan^2\alpha} + \frac{1}{d_2^2} = \frac{s^2}{a^2} + \frac{c^2}{b^2}$$

$$\frac{(1+\frac{d_1}{d_2})^2}{d_1^2\tan^2\alpha} + \frac{1}{d_2^2} = \frac{s^2}{\frac{d_1^2}{1-\frac{s}{c\tan\alpha}(1+\frac{d_1}{d_2})}} + \frac{c^2}{\frac{d_1^2}{1+\frac{c}{s\tan\alpha}(1+\frac{d_1}{d_2})}}$$

$$\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + \frac{d_1^2}{d_2^2} = s^2(1-\frac{s}{c\tan\alpha}(1+\frac{d_1}{d_2})) + c^2(1+\frac{c}{s\tan\alpha}(1+\frac{d_1}{d_2}))$$

$$\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + \frac{d_1^2}{d_2^2} = s^2 - \frac{ts^2}{\tan\alpha}(1+\frac{d_1}{d_2}) + c^2 + \frac{c^2}{t\tan\alpha}(1+\frac{d_1}{d_2})$$

$$\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + \frac{d_1^2}{d_2^2} = 1 + \frac{c^2-t^2s^2}{t\tan\alpha}(1+\frac{d_1}{d_2})$$

$$\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + \frac{d_1^2}{d_2^2} - 1 = \frac{1-2s^2}{tc^2\tan\alpha}(1+\frac{d_1}{d_2})$$

$$\frac{(1+\frac{d_1}{d_2})^2}{\tan^2\alpha} + (\frac{d_1}{d_2}-1)(\frac{d_1}{d_2}+1) = \frac{1-2s^2}{tc^2\tan\alpha}(1+\frac{d_1}{d_2})$$

$$\frac{(1+\frac{d_1}{d_2})}{\tan^2\alpha} + \frac{d_1}{d_2} - 1 = \frac{1-2s^2}{tc^2\tan\alpha}$$

$$\frac{(1+\frac{d_1}{d_2})}{\tan\alpha} + \frac{d_1\tan\alpha}{d_2} - \tan\alpha = \frac{1-2s^2}{tc^2}$$

$$\frac{d_1 + d_2 + d_1 \tan^2 \alpha - d_2 \tan^2 \alpha}{d_2 \tan \alpha} = \frac{1 - 2s^2}{tc^2}$$

$$\frac{d_1(1 + \tan^2 \alpha) + d_2(1 - \tan^2 \alpha)}{d_2 \tan \alpha} = \frac{1 - 2s^2}{tc^2}$$

$$\frac{1 - 2s^2}{c^2 t} = \frac{cos(2\theta)}{c^2 t} = 2\cot(2\theta)$$

$$1 + \tan^2 \alpha = \frac{1}{\cos^2 \alpha}, \quad 1 - \tan^2 \alpha = \frac{1 - 2\sin^2 \alpha}{\cos^2 \alpha} = \frac{\cos(2\alpha)}{\cos^2 \alpha}$$

$$\frac{\frac{d_1}{\cos^2 \alpha} + \frac{d_2 \cos(2\alpha)}{\cos^2 \alpha}}{d_2 \tan \alpha} = 2\cot(2\theta)$$

$$\frac{d_1 + d_2 \cos(2\alpha)}{d_2 \sin \alpha \cos \alpha} = 2\cot(2\theta)$$

$$\frac{\frac{d_1}{d_2} + \cos(2\alpha)}{\sin(2\alpha)} = \cot(2\theta)$$

$$\theta = \frac{1}{2}\cot^{-1}\left(\frac{\frac{d_1}{d_2} + \cos(2\alpha)}{\sin(2\alpha)}\right) \tag{8}$$

With θ calculated, we can go back to Equation (7) and Equation (6) and calculate their exact value easily. We have calculate the fixed values a, b, θ with the assumption the bottom edge is the x-axis and it intersect the top edge at the origin. The ellipse is expressed by the equation:

$$\frac{(x\cos\theta + y\sin\theta)^2}{a^2} + \frac{(x\sin\theta - y\cos\theta)^2}{b^2} = 1$$

To fully generalize, to any cell, we use $\theta' = \theta + \tan^{-1}(m_b^C)$, and offset the ellipse center to the top and bottom edge intersection (x_0, y_0):

$$\frac{((x - x_0)\cos\theta' + (y - y_0)\sin\theta')^2}{a^2} + \frac{((x - x_0)\sin\theta' - (y - y_0)\cos\theta')^2}{b^2} = 1$$

References

[1] W. H. Besant. *Notes on Roulettes and Glissettes.* Deighton, Bell, Cambridge, 1870.

[2] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition.* Springer, 2008.

[3] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. *Discret. Comput. Geom.*, 15(1):35–61, 1996.

[4] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. 17(3):229–241, 2001.

[5] G. Dudek, K. Romanik, and S. Whitesides. Global localization: Localizing a robot with minimal travel. 27(2):583–604, Apr. 1998.

[6] S. P. Engelson and D. V. McDermott. Error correction in mobile robot map learning. pages 2555–2560, 1992.

[7] D. Fox, S. Thrun, W. Burgard, and F. Dallaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 401–428. Springer-Verlag, Berlin, 2001.

[8] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[9] D. Halperin and M. Sharir. Arrangements. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 28, pages 723–762. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2018.

[10] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. 17(5):748–760, Oct. 2001.

[11] J. M. O'Kane and S. M. LaValle. Localization with limited sensing. *IEEE Transactions on Robotics*, 23(4):704–716, Aug. 2007.

[12] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(5):1–25, Apr. 1998.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.