Mapping Polygons with Agents That Measure Angles

Yann Disser, Matúš Mihalák, and Peter Widmayer

Abstract. We study the problem of mapping an initially unknown environment with autonomous mobile robots. More precisely, we consider simplistic agents that move from vertex to vertex along the boundary of a polygon and measure angles at each vertex. We show that such agents are already capable of drawing a map of any polygon in the sense that they can infer the exact geometry up to similarity. Often, such tasks require the agent to have some prior bound on the size of the environment. In this paper, we provide an efficient reconstruction algorithm that *does not* need any a priori knowledge about the total number of vertices.

1 Introduction

From the perspective of a technology enthusiast it is fascinating to see more and more problems in our daily lives being taken over by autonomous robots. We have developed a good understanding of how to make sophisticated robots perform impressive tasks like navigating cars through traffic. On the other hand, we still do not understand how much sophistication is actually needed for even the most fundamental problems. In other words: How powerful do the sensors and movement capabilities of a robot need to be at the very least for a given problem? From a theoretical point of view, trying to answer this question leads to a better insight of both the essential difficulty of the problem at hand and of the inherent power of different sensors. From a practical point of view, investigating the question might enable us to replace sophisticated robot designs with cheap and robust counterparts that can more easily be produced in masses.

Yann Disser · Matúš Mihalák · Peter Widmayer
ETH Zurich, Institute of Theoretical Computer Science
e-mail: {ydisser,mmihalak,widmayer}@inf.ethz.ch

E. Frazzoli et al. (Eds.): *Algorithmic Foundations of Robotics X*, STAR 86, pp. 415–425.
DOI: 10.1007/978-3-642-36279-8_25

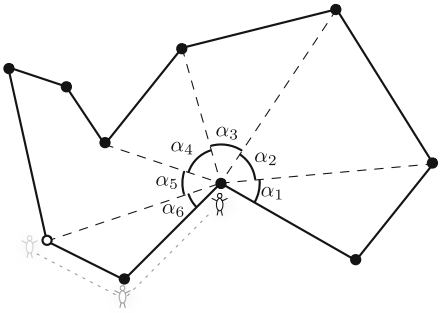

Fig. 1 The agent moves from vertex to vertex along the boundary and measures angles at each vertex

This paper aims to be a step towards the ultimate goal of developing a complete classification of robotic tasks and required capabilities. Our focus is on the problem of drawing a map, which is a core problem in almost any task that requires a robot to operate in an initially unknown environment. We approach the problem from a theoretical perspective and base our analysis on simplistic models for both environments and robots. More precisely, we take the initially unknown environment to be a simple polygon and consider the following minimalistic *agent* model instead of dealing with realistic *robots*.

Ideally, we want the agent model to be as simplistic as possible, while still allowing the agent to draw a complete map of its surrounding polygon. We make the following modeling assumptions, where $\mathcal{P}$ refers to the environment and a *line of sight* refers to a line segment lying inside $\mathcal{P}$ connecting two vertices (cf. Fig. 1):

- The agent is initially located at a vertex of $\mathcal{P}$.
- The agent can move along the boundary of $\mathcal{P}$.
- While situated at a vertex v, the agent can order the lines of sight as they appear in a counter-clockwise sweep of $\mathcal{P}$ at v.
- While situated at a vertex v, the agent can perceive the angle between any two lines of sight.

Note that we do not assume the agent to be aware of the number n of vertices of $\mathcal{P}$. Since we are interested whether the information that the agent can collect suffices to draw a map of $\mathcal{P}$ at all, we do not impose limitations on the memory or computational power available to the agent. Of course, given the choice, we prefer efficient mapping algorithms.

So, does our model empower the agent to draw a map? The answer to this question depends on what we consider to be a "map" of a polygon. The only geometrical data that is available to our agent is encoded in angles.

This means that we cannot hope the agent to do better than to draw a map that describes the geometry of the polygon up to similarity. The question is whether the data available through the sensors of the agent alone already *uniquely* determines the shape of $\mathcal{P}$, or whether there can be two polygons of different shape that yield the same observations.

In this paper, we show that the agent as modeled above can infer the shape of any polygon uniquely, and hence draw a map.

1.1 Related Work

This paper is a follow-up to our earlier results in [9], where we give an algorithm for reconstructing a polygon from a list of angle measurements. The difference towards this earlier paper is that we consider an agent that does not know the number of vertices n a priori. When n is known, it is easy for the agent to collect all data initially. In that case we may as well ignore the agent and view the problem as a geometrical reconstruction problem, where a list of angle-lists is given and we look for the shape of the polygon. In this paper, we make use of a central result of [9] to design an algorithm that allows the agent to reconstruct the visibility graph incrementally while collecting angle measurements along the boundary. We also use the observations made in [7] to improve the efficiency of the resulting algorithm.

Many studies related with our setting are concerned with geometrical reconstruction problems, where geometrical objects have to be reconstructed from given measurement data, i.e., without considering agents that have to gather the data first. The main focus of work in this area, typically, is to decide whether a certain kind of data encodes enough information for a reconstruction or not. Work in this area includes [8, 20, 23]. Another variant of geometrical reconstruction problems ask for any object compatible to given measurement data without requiring the solution to be unique. The main focus for studies for this variant usually lies in finding efficient algorithms. Work in this area includes [2, 16, 21, 22].

There have been different approaches to modeling minimalistic agents for various environments and objectives [1, 11, 17, 24]. Some works have even established hierarchies of agent models and frameworks that allow to compare otherwise unrelated models [4, 10, 19]. We based our model on the one proposed in [24], which has also been studied previously in [3, 4, 6, 5, 12, 18].

The agents (called *Bitbots*) that were studied in [17] are similar to ours in that they can only move along the walls of the environment. While Bitbots have somewhat more powerful movement capabilities than our agents, they are much more limited in their sensing. As it turns out, this makes them incapable of inferring even a topological map [17].

Another similar setting was considered in [3]. This study uses the same movement capabilities as our model, but again only provides agents with

very basic, combinatorial sensors. And again, it turns out that such agents cannot even infer a topological map of their environment.

There is also an example of agents with weaker sensors than our's which *can* infer a topological map: The agents in [6] can only distinguish convex from reflex angles, but they are allowed to move through the interior of the environment and they need to be provided with knowledge of at least a bound on the total number of vertices. In addition, the reconstruction algorithm that is developed in [6] is extremely inefficient compared to the one we introduce here.

2 The Mapping Problem

Throughout this paper, we consider the exploration of a simple polygon $\mathcal{P}$ by an autonomous agent. In particular, we assume $\mathcal{P}$ not to have holes or self-intersections. We let V denote the set of vertices of $\mathcal{P}$ and let $n = |V|$ be its size. We assume the agent to initially be located at a vertex v_0 of $\mathcal{P}$, and denote all other vertices of $\mathcal{P}$ by $v_1, v_2, \ldots, v_{n-1}$ in the order in which appear along a counterclockwise tour of the boundary starting at v_0. We express the cyclic order of the vertices in notation by writing $v_{i\pm k}$ for $v_{i\pm k \bmod n}$.

In the model which we adopt the agent can (1) move from vertex to vertex along the boundary of $\mathcal{P}$ and (2) make local observations in $\mathcal{P}$ while situated at a vertex. The agent is aware that its environment is a simple polygon, but other than that it has no initial knowledge about $\mathcal{P}$. In particular, it has no knowledge about n. This means that, initially, the agent does not know how many vertices it needs to visit in order to complete an exact tour of the boundary.

We now define precisely what local observations the agent can make at a vertex v_i. All vertices v_j which can be connected to v_i via straight line segments in $\mathcal{P}$ (including the boundary) are considered to be *visible* to v_i and hence to be visible to the agent. We refer to the corresponding line segments as *lines of sight*, and we order the vertices visible to v_i *locally* according to the angles formed inside $\mathcal{P}$ by the $\overline{v_i v_{i+1}}$ and the corresponding line of sight. The local observation of the agent at v_i is encoded in the vector of angles $\boldsymbol{\alpha}(v_i) = (\alpha_1, \alpha_2, \ldots)$, where α_l is the angle inside $\mathcal{P}$ between the l-th and the $(l+1)$-th line segment. We refer to $\boldsymbol{\alpha}(v_i)$ as the *angle measurement at* v_i. From $\boldsymbol{\alpha}(v_i)$ it is easy to infer the angle spanned by any two non-consecutive vertices visible to v_i, simply by summing up all enclosed angles. On the other hand, the angle measurement does not provide the identity of the visible vertices, i.e., the agent cannot tell which vertices of V are visible to v_i just by inspecting $\boldsymbol{\alpha}(v_i)$. Of course, the agent always knows that the first visible vertex is v_{i+1} and the last one is v_{i-1}.

In order to argue that an agent can always draw a map, we need to provide an algorithm that governs how the agent moves and reasons about the

knowledge it has gained so far. We use the term *exploration strategy* to refer to such an algorithm. An exploration strategy consists of a succession of three types of operations: (1) moving to a neighboring vertex and collecting the angle measurement at the new location, (2) making computations that may involve any of the data collected so far, and (3) terminating. The efficiency of an exploration strategy is measured upon termination by the number of physical moves and sensing operations, the number of atomic computations, and the required memory. We say that an exploration strategy computes a certain quantity if it always terminates after a finite number of moves and atomic computations, in a state where the agent knows the desired quantity.

The *shape* of $\mathcal{P}$ is given by its geometry disregarding scale, rotation and translation. We say that an exploration strategy *reconstructs* a polygon $\mathcal{P}$ if an agent executing it in $\mathcal{P}$ knows the shape of $\mathcal{P}$ upon termination. We say the agent can solve the *mapping problem* if there is an exploration strategy that reconstructs every polygon $\mathcal{P}$. As long as the agent only measures angles, computing the shape is the best it can hope for. Any scaled, rotated or translated version of a polygon $\mathcal{P}$ yields the exact same observations and can therefore not be distinguished from $\mathcal{P}$ by any exploration strategy for the agent.

Our main result relies on the fact that the agent can efficiently compute the visibility graph G_{vis} of $\mathcal{P}$, which is defined as follows. Every vertex of $\mathcal{P}$ is a node of G_{vis}, and there is an edge in G_{vis} for every two vertices of $\mathcal{P}$ that see each other. The number of vertices and edges of G_{vis} is denoted by n, m, respectively, throughout the paper. We will argue that from G_{vis} together with the angle measurement for every vertex, the agent can efficiently compute the shape of $\mathcal{P}$. Note that the characterization of visibility graph is a long-standing open problem [13, 15, 14].

3 Reconstructing the Shape of $\mathcal{P}$

We first show that once the agent knows the visibility graph G_{vis} of $\mathcal{P}$ and the angle measurements at each vertex, it can compute the shape of $\mathcal{P}$. We show this by considering a subgraph of G_{vis} that corresponds to a triangulation of $\mathcal{P}$, and by using this subgraph to construct the shape of $\mathcal{P}$. After we established that knowing G_{vis} allows the agent to solve the mapping problem we move on to our main result, namely the design of an exploration strategy that computes the visibility graph.

3.1 Inferring the Shape from G_{vis}

In this section we argue that the visibility graph $G_{\text{vis}} = (V, E)$ of $\mathcal{P}$ together with the angle measurements $\boldsymbol{\alpha}(v_i)$ for every vertex $v_i \in V$ uniquely

determine the shape of $\mathcal{P}$. We forget the agent for the moment and establish how to efficiently reconstruct the shape of $\mathcal{P}$ from this data. Once the agent has acquired knowledge of G_{vis} and the angle measurements, it can obtain the shape by using the computation described below.

Consider a vertex $v_i \in V$ of degree d in G_{vis}. Let $\boldsymbol{\alpha}(v_i) = (\alpha_1, \alpha_2, \ldots, \alpha_{d-1})$ be the angle measurement at v_i, and let $N(v_i) = \{v_{i+\delta_1}, v_{i+\delta_2}, \ldots, v_{i+\delta_d}\}$ be its neighborhood in G_{vis}, with $1 = \delta_1 < \delta_2 < \ldots < \delta_d = n-1$. For geometrical reasons, it turns out that the *global* vertex $v_{i+\delta_l}$ is exactly the l-th vertex *locally* visible to v_i. This means in particular that α_l is the angle inside $\mathcal{P}$ between the line segments $\overline{v_i v_{i+\delta_l}}$ and $\overline{v_i v_{i+\delta_{l+1}}}$. Knowing the angle measurements hence implies knowing the angle between $\overline{v_i v_{i+\delta_l}}$ and $\overline{v_i v_{i+\delta_k}}$ for any $1 \leq l < k \leq d$. This, in turn, means that we can enhance G_{vis} by assigning an angle to every pair of edges at each vertex of G_{vis}.

We give the following fact without its simple but technical proof. Intuitively, if two neighbors of a vertex v_i do not see each other, there needs to be something obstructing their line of sight. This in turns implies that v_i sees another vertex in-between. A proof can be found in [13, 14] (Lemma 3, resp. Lemma 6.2.3).

Proposition 1. *Let $v_i \in V$ be a vertex of degree d and let $N(v_i) = \{v_{i+\delta_1}, v_{i+\delta_2}, \ldots, v_{i+\delta_d}\}$ be its neighborhood in G_{vis}, with $1 = \delta_1 < \delta_2 < \ldots < \delta_d = n-1$. Then $v_{i+\delta_l}$ and $v_{i+\delta_{l+1}}$ see each other for every $1 \leq l < d$.*

We proceed with the proof of the main theorem of this section.

Theorem 1. *The shape of a polygon $\mathcal{P}$ can be inferred from its visibility graph together with its angle measurements in time and space $\mathcal{O}(m)$.*

Proof. We show how to inductively obtain the shape of a polygon $\mathcal{P}$ from its enhanced visibility graph G_{vis}. If $|V| = 3$, the polygon is a triangle and, since v_0, v_1, v_2 need to appear in counter-clockwise order, the shape is uniquely determined.

Consider a polygon $\mathcal{P}$ with $|V| > 3$ vertices with its enhanced visibility graph G_{vis}. We distinguish two cases concerning the neighborhood of v_1 (cf. Fig. 2). If v_1 has degree two, then v_0, v_2 see each other by Proposition 1. By induction, we can compute both the shape of the subpolygon induced by $v_2, v_3, \ldots, v_0$ as well as the shape of the triangle v_0, v_1, v_2. We obtain the shape of $\mathcal{P}$ by combining both these shapes. If v_1 has degree at least 3, we determine the second visible vertex v_i of v_1. By induction, we can compute the shape of the subpolygons induced by $v_1, v_2, \ldots, v_i$ and $v_i, v_{i+1}, \ldots, v_0, v_1$. We can again combine both shapes to obtain the shape of $\mathcal{P}$.

A careful recursive implementation of this algorithm runs in time $\mathcal{O}(m)$ and essentially needs the space required for storing G_{vis}. □

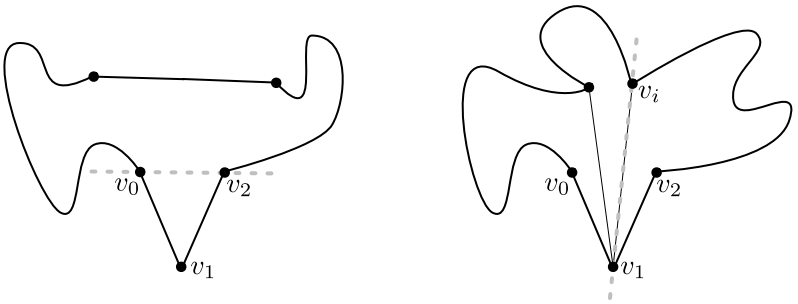

Fig. 2 The two different cases in the induction for deducing the shape of $\mathcal{P}$ from G_{vis}

3.2 An Exploration Strategy to Compute G_{vis}

In this section, we introduce an exploration strategy that builds G_{vis} incrementally. The following definitions will allow us to describe increasing portions of G_{vis} that have already been inferred by our strategy. The graph $G_i^j = (V_i^j, E_i^j)$ is defined to be the subgraph of G_{vis} induced by the vertices $v_i, v_{i+1}, \ldots, v_j$. Observe that, in this notation, $G_{\text{vis}} = G_0^{n-1}$. The degree of a vertex v_k in G_i^j is denoted by $d_i^j(v_k)$, and we write $d_k := d_0^{n-1}(v_k)$ for the degree of v_k in the visibility graph G_{vis} of $\mathcal{P}$.

The key of our exploration strategy is to maintain a growing subgraph of G_{vis} that has already been determined. After each new measurement, the agent incorporates the new data to obtain a bigger part of G_{vis}. More precisely, in step t, the agent moves from vertex v_{t-1} to vertex v_t and computes G_0^t from G_0^{t-1} and the new angle measurement at v_t. The main ingredient to our exploration strategy originates from [7, 9] and can be formalized as follows.

Lemma 1 ([7, 9]). *The graph G_i^j, $, 0 \leq i < j$, can be computed from G_i^{j-1}, G_{i+1}^j, and $\boldsymbol{\alpha}(v_i), \boldsymbol{\alpha}(v_{i+1}), \ldots, \boldsymbol{\alpha}(v_j)$ in constant time.*

The main problem that has to be solved for the computation in Lemma 1 can be reformulated as follows: Given the edges $E_i^j \setminus \{v_i, v_j\}$ and the angles given by $\boldsymbol{\alpha}(v_i), \boldsymbol{\alpha}(v_{i+1}), \ldots, \boldsymbol{\alpha}(v_j)$, decide whether $\{v_i, v_j\} \in E_i^j$, i.e., decide whether v_i sees v_j or not. In [9], we provided a necessary and sufficient condition for making this decision. Later, Chen and Wang [7] showed how to test the condition in constant time. The notation that we use here is motivated by the incremental nature of the strategy that the agent has to employ, and therefore different from the one used in the original papers. For convenience, we describe the inner workings of Lemma 1 in Sect. 3.3.

Theorem 2. *There is an exploration strategy that computes the shape of any polygon $\mathcal{P}$ using $\mathcal{O}(n)$ moves and sensing operations, $\mathcal{O}(n^2)$ atomic computations, and $\mathcal{O}(m)$ bits of memory.*

Proof. Lemma 1 provides the means for our incremental exploration strategy. We give a listing of the strategy in Algorithm 1.1. It uses the operations SENSE and MOVE to govern the physical actions of the agent. When located at a vertex v_k, the operation SENSE returns $\boldsymbol{\alpha}(v_k)$, while the operation MOVE moves the agent one step along the boundary to v_{k+1}.

Algorithm 1.1. Algorithm for computing the visibility graph G_{vis}.

function computeG_{vis}
$G_0^0 \leftarrow (v_0, \emptyset)$;
$\boldsymbol{\alpha}(v_0) \leftarrow$ SENSE;
$j \leftarrow 0$
while $d_0^j(v_0) < |\boldsymbol{\alpha}(v_0)| + 1$**:**
  $j \leftarrow j + 1$;
  MOVE;
  $\boldsymbol{\alpha}(v_j) \leftarrow$ SENSE;
  $G_j^j = (v_j, \emptyset)$;
  for $l \leftarrow j-1, j-2, \ldots 1, 0$
    compute G_l^j from $G_{l+1}^j, G_l^{j-1}, \boldsymbol{\alpha}(v_l), \boldsymbol{\alpha}(v_{l+1}), \ldots, \boldsymbol{\alpha}(v_j)$; (see Lemma 1)
return G_0^j;

The algorithm maintains the invariant that, after each iteration of the outer loop, G_l^r has been computed for all $0 \le l \le r < j$. In particular, $G_{\text{vis}} = G_0^{n-1}$ has been computed after $n-1$ iterations. Because $d_0^{n-1}(v_0) = |\boldsymbol{\alpha}(v_0)| + 1$, this means that the algorithm terminates after $n-1$ iterations and returns $G_{\text{vis}} = G_0^{n-1}$. The agent performs exactly $n-1$ moves and n sensing operations. The total computational running time is $\mathcal{O}(n^2)$, due to Lemma 1. Together with Theorem 1, we have proven the claim. □

3.3 Deciding Whether v_i Sees v_j

In this section, we describe in more detail how to construct G_i^j from G_i^{j-1} and G_{i+1}^j when the angle measurements $\boldsymbol{\alpha}(v_i), \boldsymbol{\alpha}(v_{i+1}), \ldots, \boldsymbol{\alpha}(v_j)$ are known (Lemma 1). Since all the edges in $E_i^j \setminus \{v_i, v_j\}$ are already present in either G_i^{j-1} or G_{i+1}^j, the problem boils down to deciding whether $\{v_i, v_j\} \in E_i^j$, i.e., whether v_i sees v_j. Of course, this problem is non-trivial only when $v_j \neq v_{i+1}$. Consider Fig. 3 alongside the discussion below.

Recall that, in our notation, v_i sees d_i vertices in total. Of those vertices, we can identify the first $d_i^{j-1}(v_i)$, simply by looking at G_i^{j-1}. Now, if v_i actually sees v_j, then v_j is the first unidentified vertex of v_i, i.e., the $(d_i^{j-1}(v_i) + 1)$-th vertex among the vertices visible to v_i. Similarly, for vertex v_j, the last

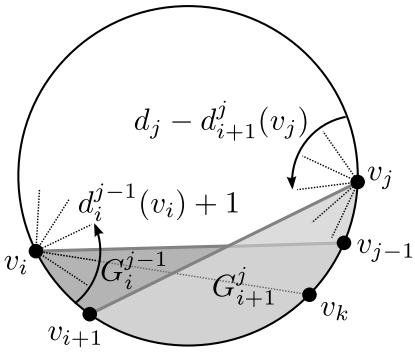

Fig. 3 Illustration of the construction of G^j_i from G^{j-1}_i and G^j_{i+1}. The difficulty is to decide whether v_i sees v_j.

$d^j_{i+1}(v_j)$ vertices visible to v_j are identified. If v_j sees v_i, then v_i is the last unidentified vertex of v_j, i.e., the $(d_j - d^j_{i+1}(v_j))$-th visible vertex of v_j.

By Proposition 1, the $d^{j-1}_i(v_i)$-th and the $(d^{j-1}_i(v_i)+1)$-th visible vertex of v_i see each other. If v_i would see v_j, then v_j would be this $(d^{j-1}_i(v_i)+1)$-th visible vertex of v_i, and hence v_i, the $d^{j-1}_i(v_i)$-th visible vertex of v_i, and v_j would form a triangle in $\mathcal{P}$. The three enclosed angles of this triangle would sum up to 180 degrees. Whether this is the case can be determined by inspecting G^{j-1}_i and G^j_{i+1}: From G^{j-1}_i we know the global identity v_k of the $d^{j-1}_i(v_i)$-th visible vertex of v_i, and from G^j_{i+1} we can check whether v_k sees v_j. If v_k does not see v_j, we can conclude that v_i does not see v_j. Otherwise, let α denote the angle at v_i between v_k and the $(d^{j-1}_i(v_i)+1)$-th visible vertex, let β denote the angle at v_k between v_j and v_i, and let γ denote the angle at v_j between the $(d_j - d^j_{i+1}(v_j))$-th visible vertex and vertex v_k. Again, if $\alpha+\beta+\gamma \neq 180°$, we can conclude that v_i does not see v_j. What is more surprising is that the opposite holds as well, i.e., if $\alpha+\beta+\gamma = 180°$, then v_i and v_j *must* see each other. This has been proved formally in [7, 9] – we illustrate the idea behind the proof in Fig. 4. Intuitively, if $\alpha+\beta+\gamma = 180°$, then the triangle v_i, v_k, v_j is empty, and hence nothing can obstruct the line of sight between v_i and v_j.

Overall, this necessary and sufficient criterion gives us a simple and computationally efficient means to check whether v_i and v_j see each other, and thus whether or not $\{v_i, v_j\} \in E^j_i$.

Lemma 2 ([9]). *Let $v_i, v_j \in V$ with $v_j \neq v_{i+1}$ and let $v_k, \alpha, \beta, \gamma$ be defined as before, then*

$$v_i \text{ sees } v_j \text{ if and only if } \alpha+\beta+\gamma = 180°.$$

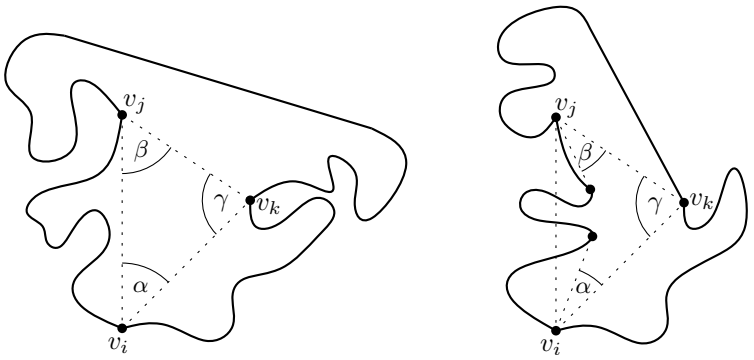

Fig. 4 An illustration that v_i sees v_j if and only if $\alpha + \beta + \gamma = 180°$

4 Conclusion

We have presented an exploration strategy for an agent that moves along the boundary of a polygon and observes the angles formed by the lines of sight at each vertex. We have shown that the agent can infer the shape of the polygon in $O(n)$ moves using $O(n^2)$ atomic computations and $O(m)$ memory. The exploration strategy we presented is based on an incremental construction of the visibility graph, and does not need any knowledge about n, which is an improvement over previous results. This, in fact, is a rare property when considering problems for autonomous agents. We usually need some means to "break the symmetry", such as pebbles that can be used to mark vertices.

A natural open problem is to achieve a better running time than $\mathcal{O}(n^2)$ if the underlying visibility graph is sparse. We can of course not do better than $\mathcal{O}(m)$, but the gap remains. Maybe $\mathcal{O}(m)$ atomic computations are enough?

References

1. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Transactions on Robotics and Automation 15(5), 818–828 (1999)
2. Biedl, T., Durocher, S., Snoeyink, J.: Reconstructing Polygons from Scanner Data. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 862–871. Springer, Heidelberg (2009)
3. Bilò, D., Disser, Y., Mihalák, M., Suri, S., Vicari, E., Widmayer, P.: Reconstructing visibility graphs with simple robots. Theoretical Computer Science 444, 52–59 (2012)
4. Brunner, J., Mihalák, M., Suri, S., Vicari, E., Widmayer, P.: Simple Robots in Polygonal Environments: A Hierarchy. In: Fekete, S.P. (ed.) ALGOSENSORS 2008. LNCS, vol. 5389, pp. 111–124. Springer, Heidelberg (2008)
5. Chalopin, J., Das, S., Disser, Y., Mihalák, M., Widmayer, P.: How Simple Robots Benefit from Looking Back. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 229–239. Springer, Heidelberg (2010)

6. Chalopin, J., Das, S., Disser, Y., Mihalák, M., Widmayer, P.: Telling convex from reflex allows to map a polygon. In: Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, pp. 153–164 (2011)
7. Chen, D., Wang, H.: An improved algorithm for reconstructing a simple polygon from the visibility angles. Computational Geometry: Theory and Applications 45, 254–257 (2012)
8. Coullard, C., Lubiw, A.: Distance visibility graphs. In: Proceedings of the 7th Annual Symposium on Computational Geometry, pp. 289–296 (1991)
9. Disser, Y., Mihalák, M., Widmayer, P.: A polygon is determined by its angles. Computational Geometry: Theory and Applications 44, 418–426 (2011)
10. Donald, B.R.: On information invariants in robotics. Artificial Intelligence 72(1-2), 217–304 (1995)
11. Ganguli, A., Cortés, J., Bullo, F.: Distributed deployment of asynchronous guards in art galleries. In: Proceedings of the 2006 American Control Conference, pp. 1416–1421 (2006)
12. Gfeller, B., Mihalák, M., Suri, S., Vicari, E., Widmayer, P.: Counting Targets with Mobile Sensors in an Unknown Environment. In: Kutyłowski, M., Cichoń, J., Kubiak, P. (eds.) ALGOSENSORS 2007. LNCS, vol. 4837, pp. 32–45. Springer, Heidelberg (2008)
13. Ghosh, S.K.: On recognizing and characterizing visibility graphs of simple polygons. Discrete and Computational Geometry 17(2), 143–162 (1997)
14. Ghosh, S.K.: Visibility Algorithms in the Plane. Cambridge University Press (2007)
15. Ghosh, S.K., Goswami, P.P.: Unsolved problems in visibility graph theory. In: Proceedings of the India-Taiwan Conference on Discrete Mathematics, pp. 44–54 (2009)
16. Jackson, L., Wismath, S.K.: Orthogonal polygon reconstruction from stabbing information. Computational Geometry 23(1), 69–83 (2002)
17. Katsev, M., Yershova, A., Tovar, B., Ghrist, R., LaValle, S.M.: Mapping and pursuit-evasion strategies for a simple wall-following robot. IEEE Transactions on Robotics 27(1), 113–128 (2011)
18. Komuravelli, A., Mihalák, M.: Exploring Polygonal Environments by Simple Robots with Faulty Combinatorial Vision. In: Guerraoui, R., Petit, F. (eds.) SSS 2009. LNCS, vol. 5873, pp. 458–471. Springer, Heidelberg (2009)
19. O'Kane, J.M., LaValle, S.M.: On comparing the power of robots. International Journal of Robotics Research 27(1), 5–23 (2008)
20. O'Rourke, J.: Uniqueness of orthogonal connect-the-dots. In: Toussaint, G.T. (ed.) Computational Morphology, pp. 97–104. North-Holland (1988)
21. Rappaport, D.: On the complexity of computing orthogonal polygons from a set of points. Technical Report SOCS-86.9, McGill University, Montreal, Canada (1986)
22. Sidlesky, A., Barequet, G., Gotsman, C.: Polygon reconstruction from line cross-sections. In: Proceedings of the 18th Annual Canadian Conference on Computational Geometry, pp. 81–84 (2006)
23. Snoeyink, J.: Cross-ratios and angles determine a polygon. Discrete and Computational Geometry 22(4), 619–631 (1999)
24. Suri, S., Vicari, E., Widmayer, P.: Simple robots with minimal sensing: From local visibility to global geometry. International Journal of Robotics Research 27(9), 1055–1067 (2008)