An Optimal Algorithm for the 1-Searchability of Polygonal Rooms

Xuehou Tan

Tokai University, 317 Nishino, Numazu 410-0395, Japan
tan@wing.ncc.u-tokai.ac.jp

Abstract. The *1-searcher* is a mobile guard who can see only along a ray emanating from his position and can continuously change the direction of the ray with bounded speed. A polygonal region P with a specified point d on its boundary is called a *room*, and denoted by (P, d). The room (P, d) is said to be *1-searchable* if the searcher, starting at the point d, can eventually see a mobile intruder who moves arbitrarily fast inside P, without allowing the intruder to touch d. We present an optimal $O(n)$ time algorithm to determine whether there is a point x on the boundary of P such that the room (P, x) is 1-searchable. This improves upon the previous $O(n \log n)$ time bound, which was established for determining whether or not a room (P, d) is 1-searchable, where d is a given point on the boundary of P.

1 Introduction

Recently, much attention has been devoted to the problem of searching for a mobile intruder in a polygonal region P by a mobile searcher [6,8,9,10,11,12,13,14,15]. Both the searcher and the intruder are modeled by points that can continuously move in P. The *1-searcher* is a mobile guard who can see only along a ray emanating from his position and can change the direction of the ray with bounded speed. A polygonal region P with a specified point d (called the *door*) on its boundary is called a *room*, and denoted by (P, d). The room (P, d) is said to be *1-searchable* if the searcher, starting at the point d, can eventually see a mobile intruder who moves arbitrarily fast inside P, without allowing the intruder to touch d.

The problem of searching a polygonal room by a single 1-searcher was first studied by Lee et al. [10]. By characterizing the class of 1-searchable rooms, they described an $O(n \log n)$ time algorithm to determine if a specifed room is 1-searchable. An optimal algorithm for generating a search schedule was later given in [14]. In this paper, we present an optimal $O(n)$ time and space algorithm to determine whether there is a point x on the boundary of P such that the room (P, x) is 1-searchable. Combining with result of [14], we thus obtain an optimal solution to the problem of searching a polygonal room by a 1-searcher. Moreover, our algorithm is simple and does not require a triangulation of P. This simplicity is important as many linear-time geometric algorithms depend on the triangulation algorithm of Chazelle [3], which is too complicated to be suitable in practice.

J. Akiyama et al. (Eds.): JCDCG 2004, LNCS 3742, pp. 174–183, 2005.

2 Preliminary

Let P denote a simple polygon, i.e., it has neither self-intersections nor holes. Two points $x, y \in P$ are said to be mutually *visible* if the line segment connecting them, denoted by $\overline{xy}$, is entirely contained in P. For two regions Q_1, $Q_2 \subseteq P$, we say that Q_1 is *weakly visible* from Q_2 if every point in Q_1 is visible from some point in Q_2. For a vertex x of the polygon P, let $Succ(x)$ denote the vertex immediately succeeding x clockwise, and $Pred(x)$ the vertex immediately preceding x clockwise. A vertex of P is *reflex* if its interior angle is strictly greater than 180°; otherwise, it is *convex*. An important definition for reflex vertices is that of *ray shots*: the backward ray shot from a reflex vertex r, denoted by *Backw(r)*, is the first point of P hit by a "bullet" shot at r in the direction from $Succ(r)$ to r, and the forward ray shot *Forw(r)* is the first point hit by the bullet shot at r in the direction from $Pred(r)$ to r. See Fig. 1.

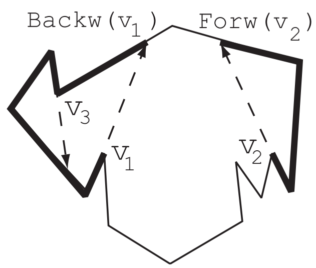

Fig. 1. Forward, backward ray shots and components

Let u, v denote two boundary points of P, and let $P[u, v]$ (resp. $P(u, v)$) denote the closed (resp. open) *clockwise* chain of P from u to v. We define the chain $P[r, Backw(r)]$ (resp. $P[Forw(r), r]$) as the *backward component* (resp. *forward component*) of the reflex vertex r. The point r is referred to as the *defining vertex* of the component. See Fig. 1 for an example, where two different components of v_1 and v_2 are shown in bold line. A backward (resp. forward) component is said to be *non-redundant* if it does not contain any other backward (resp. forward) component. A reflex vertex is *critical* if its backward or forward component is non-redundant. For example, the vertices v_1, v_2 and v_3 in Fig. 1 are critical.

A polygon P is said to be *LR-visible* if there is a pair of boundary points u and v such that $P[u, v]$ and $P[v, u]$ are weakly visible from each other. Clearly, P is LR-visible with respect to the point pair (u, v) if and only if each non-redundant component of P contains either u or v. Das et al. have developed a linear-time algorithm to determine whether a polygon P is LR-visible or not [4]. Later, Bhattacharya and Ghosh [1] simplified the algorithm such that it uses only simple data structures and does not require a triangulation of the polygon. The algorithm also allows one to compute the shortest paths from an arbitrary vertex to all other vertices of P. If P is LR-visible, then all of its non-redundant components can be computed in linear time [1,4]. (Actually, the containment relation between forward components and backward components is further considered in the definition of non-redundant components given by Das et al. [4]. But, the main part of their algorithm is to compute the set of non-redundant forward or backward components.)

Lemma 1. *[1,4] It takes $O(n)$ time to determine whether or not P is LR-visible. Also, all non-redundant forward (resp. backward) components of an LR-visible polygon can be computed in $O(n)$ time.*

A pair of reflex vertices x, y is said to give a *d-deadlock*, where d is a boundary point of P, if both components $P(x, Backw(x)]$ and $P[Forw(y), y)$ do not contain d, and the points v_1, $Forw(v_2)$, $Backw(v_1)$ and v_2 are in clockwise order. (Note that the point d may be identical to x or y.) See an example in Fig. 2(a). In the case that P is LR-visible with respect to some point pairs (x, y), all the x-deadlocks and y-deadlocks in P can be reported in linear time.

Lemma 2. *[2] Suppose that P is LR-visible with respect to some point pairs (x, y). It takes $O(n)$ time to report all the x-deadlocks and y-deadlocks in P.*[1]

3 The Main Result

The characterization of 1-searchable rooms was originally given by Lee et al. [10]. To obtain the optimality of the algorithm, we make use of the following alternate characterization, which is given in terms of components and deadlocks (see also [14]).

Lemma 3. *[10,14] A polygonal room (P, d) is not 1-searchable if and only if one of the following conditions is true.*

*(**A1**) A d-deadlock occurs (Fig. 2(a)), or there are two disjoint components such that both of them do not contain d (Figs. 2(b)-(e)).*

*(**A2**) There are three reflex vertices v_1, v_2 and v_3, which are in clockwise order, such that the pair (v_1, v_3) gives both the v_2-deadlock and the $Forw(v_2)$-deadlock or $Backw(v_2)$-deadlock (Fig. 2(f)).*

*(**A3**) There are two vertices a_2 and b_2 such that both components $P[a_2, Backw(a_2)]$ and $P[Forw(b_2), b_2]$ do not contain d, and all vertices of the chain $P[a_2, b_2]$ have their deadlocks (Fig. 2(g)).*

Notice first that the condition **A2** is independent of d, which implies that if **A2** is true, then P is not 1-searchable for any room (P, d), where d is an arbitrary point on the boundary of P. Actually, if **A2** is true, then the condition **A1** is true for all the rooms (P, x), $x \in P[Forw(v_1), Backw(v_3)]$. Note also that

[1] This result, together with Lemma 1, gives an optimal algorithm for the *two-guard walkability* of simple polygons [2]. In the appendix, we give a polygon that has a 1-searchable room, but is not walkable by two guards [7]. Thus, our result is stronger than the result obtained in [2].

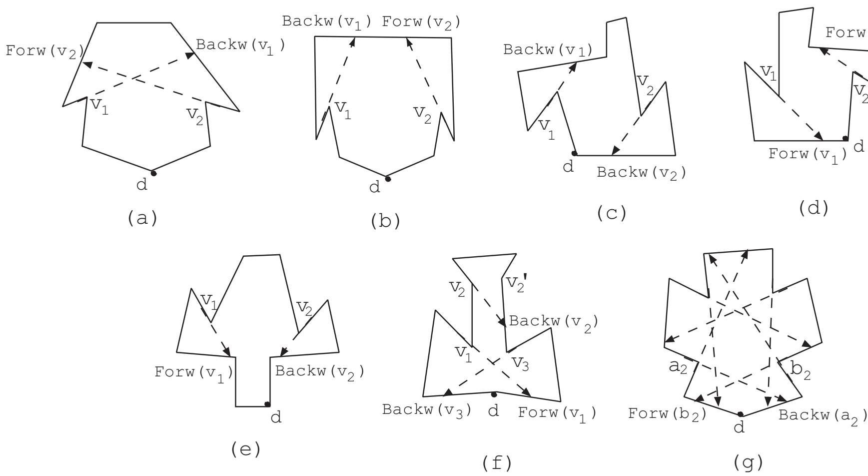

Fig. 2. The conditions A1, A2 and A3

if P is not LR-visible, then **A1** is true for every point d on the boundary of P, and thus no rooms in P are 1-searchable.

We will present an $O(n)$ time algorithm to determine whether there is a 1-searchable room in a simple polygon. Our algorithm is based on the following observations, which immediately follow from the definition of critical vertices.

Observation 1. *If there are two disjoint components such that* **A1** *is true, then we can assume that these two components are non-redundant, or equally, two defining vertices of these components are critical.*

Observation 2. *If* **A2** *is true, then we can assume that the vertex* v_2 *for* **A2** *is critical.*

Observation 3. *If* **A3** *is true, then we can assume that two vertices* a_2 *and* b_2 *for* **A3** *are critical.*

For simplicity, we consider below the ray shot from a critical vertex as two different vertices of P; one slightly preceding it and one slightly succeeding it. Following from Lemma 3 and the observations made above, it suffices to verify **A1**, **A2** and **A3** for all *vertex-door rooms* (P, d), where d denotes a vertex of P. Our algorithm can be summarized as follows.

Algorithm *searchability*

1. Run the linear-time algorithm of [1,4] to determine whether the given polygon P is LR-visible. If P is not LR-visible, report "no rooms in P are 1-searchable". (It means that no room (P, d), where d is an arbitrary point on the boundary of P, is 1-searchable.) Otherwise, compute all non-redundant components (i.e., critical vertices) of P, and then mark the ray shots from critical vertices as the vertices of P.

2. Verify the condition **A1** for all vertex-door rooms of P. If **A1** is true for all vertex-door rooms, report "no rooms in P are 1-searchable".
3. Check whether the condition **A2** is true or not. If *yes*, report "no rooms in P are 1-searchable".
4. For the vertex-door rooms (P, d) for which the condition **A1** is not true, we further verify whether the condition **A3** is true for them. If **A1** or **A3** holds for every vertex-door room, report "no rooms in P are 1-searchable". Otherwise, a 1-searchable room exists and we report it.

Theorem 1. *The algorithm* **searchability** *takes $O(n)$ time to determine whether there is a point x on the boundary of P such that the room (P, x) is 1-searchable.*

Proof. First, run the linear-time algorithm of Das et al. [1,4] to check if the polygon P is LR-visible. If P is not LR-visible, report "no rooms in P are 1-searchable", and we are done. Otherwise, all non-redundant components as well as their corresponding ray shots are computed. An order of the polygon vertices, including the ray shots from critical vertices, on the boundary of P is then obtained.

The step 2 of the algorithm *searchability* is to check if **A1** is true for every vertex-door room (P, d). The condition **A1** for (P, d), except for the d-deadlock case, can be verified as follows. Let v_1 denote the critical vertex of P such that it is closest to d counterclockwise and the component $P[v_1, Backw(v_1)]$ does not contain d, and v_2 the critical vertex such that it is closest to d clockwise and the component $P[Forw(v_2), v_2]$ does not contain d. If the points v_1, $Backw(v_1)$, $Forw(v_2)$ and v_2 are in clockwise order, the configuration shown in Fig. 2(b) occurs, and thus **A1** is true for (P, d). Otherwise, the configuration shown in Fig. 2(b) never occurs for (P, d). This is because $P[Backw(v_1), Forw(v_2)]$ contains all chains $P[Backw(v_1'), Forw(v_2')]$, where v_1', v_2' are critical and the points v_1', $Backw(v_1')$, $Forw(v_2')$ and v_2' are in clockwise order. For each vertex d, the corresponding vertices v_1 and v_2 as well as the order of v_1, $Backw(v_1)$, $Forw(v_2)$ and v_2 can be found in (amortized) constant time. Thus, we can determine in $O(1)$ amortized time if the configuration shown in Fig. 2(b) occurs. Other situations shown in Figs. 2(c)-2(e) can be dealt with analogously.

Consider now the deadlock case for the condition **A1**. Suppose that there are no two disjoint components in P which make the condition **A1** be true for (P, d), but there are two vertices u_1 and u_2 which give the d-deadlock. (Note that u_1 or u_2 may not be critical.) Then, $P(Backw(u_1), d]$ (resp. $P[d, Forw(u_2)))$ does not contain any other component; otherwise, the defining vertex of the contained component and u_1 (resp. u_2) give some configuration of **A1** shown in Figs. 2(b)-2(e), a contradiction. For the same reason, there are no two disjoint components in $P[u_1, u_2]$. Hence, there is at least one point $d' \in P[Forw(u_2), Backw(u_1)]$ such that P is LR-visible with respect to the point pair (d, d'). We can then use Bhattacharya et al.'s algorithm [2] to determine if a d-deadlock occurs. It follows from Lemma 2 that all the v-deadlocks can be reported in $O(n)$ time, provided that the configurations of **A1** shown in Figs. 2(b)-2(e) do not occur for the rooms (P, v).

Turn to the step 3 of the algorithm *searchability*. Suppose that (P, d) is a vertex-door room, for which **A1** is not true. Let v_2 be the critical vertex such that it is closest to d counetrclockwise and the component $P[v_2, Backw(v_2]$ does not contain d (if it exists). Let P' denote the portion of P obtained by cutting off the region bounded by $P[v_2, Backw(v_2)]$ and the line segment $\overline{v_2 Backw(v_2)}$. None of the configurations shown in Figs. 2(b)-2(e) occurs for two rooms (P', v_2) and $(P', Backw(v_2))$ simultaneously; otherwise, there are three disjoint components in P and thus P is not LR-visible [4], a contradiction. As discussed above, we can then determine in $O(n)$ time if there is a v_2-deadlock or a $Backw(v_2)$-deadlock in the polygon P'. If *yes*, two vertices giving the deadlock and v_2 make the condition **A2** be true, and thus no rooms in P are 1-searchable. Otherwise, we further find the critical vertex v_2' such that it is closest to d clockwise and the component $P[Forw(v_2'), v_2']$ does not contain d, and perform the same procedure for v_2' (if it exists). If **A2** is not ever satisfied, it can never be true for the polygon P, as we have assumed that the condition **A1** is not true for the room (P, d).

Finally, consider the step 4 of *searchability*. Again, let (P, d) denote a vertex-door room, for which **A1** is not true. Let $l_1, \ldots, l_i$ be the sequence of critical vertices on P such that l_1 is closest to d counterclockwise and all the components $P[l_k, Backw(l_k)]$ $(1 \le k \le i)$ do not contain d. The points $Backw(l_1)$, $Backw(l_2)$, $\ldots$, $Backw(l_i)$ are then in clockwise order. See Fig. 3. Similarly, let $r_1, \ldots, r_j$ be the sequence of critical vertices on the boundary of P such that r_j is closest to d clockwise and all the components $P[Forw(r_k), r_k]$ $(1 \le k \le j)$ do not contain d. Also, the points $Forw(r_1)$, $Forw(r_2)$, $\ldots$, $Forw(r_j)$ are in clockwise order. Assume that both l_i and r_1 exist (otherwise, the room (P, d) is 1-searchable and we are done), and that the points d, l_i and r_1 are in clockwise order (otherwise, the d-deadlock occurs, a contradiction). To verify the condition **A3** for (P, d), we first determine if P is LR-visible with respect to both point pairs (d, l_i) and (d, r_1) [1,4]. If *yes*, then P is LR-visible with respect to any point pair (d, d'), $d' \in P[l_i, r_1]$. So we can verify whether all vertices of $P[l_i, r_1]$ have their deadlocks (Lemma 2). If there is a vertex in $P[l_i, r_1]$ that does not have the deadlock, then the room (P, d) is 1-searchable and we are done. Otherwise, **A3** is true for (P, d) as well as the rooms (P, v), $v \in P(Backw(l_i), Forw(r_1))$.

Suppose that **A3** is true for the rooms (P, v), $v \in P(Backw(l_i), Forw(r_1))$. We need to further check whether the condition **A3** is true for the vertex-door

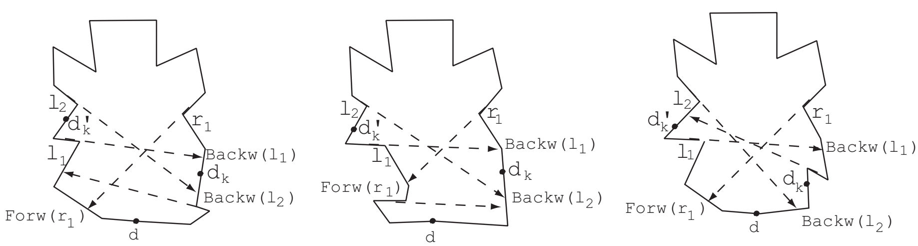

Fig. 3. The polygon P is LR-visible with respect to both point pairs (d, l_i) and (d, r_1)

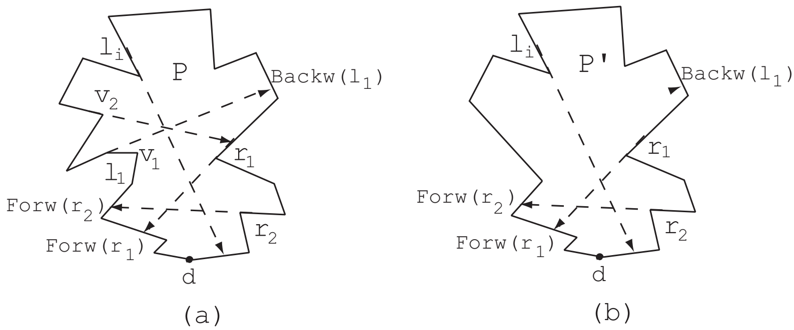

Fig. 4. The polygon P is LR-visible only with respect to the point pair (d, l_i)

rooms (P, d), $d \in P[Backw(l_1), Backw(l_i)] \cup P[Forw(r_1), Forw(r_j)]$. Since the condition **A1** has previously been verified, by a scan of the polygon boundary, we can find all the vertex-door rooms (P, d_k), $d_k \in P[Backw(l_{i-k}), Backw(l_{i-k-1})]$ and $0 \le k \le i-2$, for which **A1** is not true. Assume that **A1** is not true for a room (P, d_k), $d_k \in P[Backw(l_{i-k}), Backw(l_{i-k-1})]$, and d_k is contained in $P[r_1, d]$ (it can easily be verified, too). In this case, two chains $P[d_k, d'_k]$ and $P[d'_k, d_k]$, for any point $d'_k \in P[l_{i-k-1}, l_{i-k}]$, are mutually weakly visible; otherwise, **A1** is true for (P, d_k) or some vertices of $l_1, \ldots, l_i$ are not critical, a contradiction in either case. See Fig. 3 for some examples, where the vertex l_{i-k-1} and the vertex destroying the weak visibility (the component of that vertex does not contain d_k nor d'_k) make **A1** be true for (P, d_k). Thus, we can determine if all vertices of $P[l_{i-k-1}, l_{i-k}]$ have their deadlocks (Lemma 2), and if so the condition **A3** is true for the room (P, d_k). If **A3** is not true for some room (P, d_k), then it is 1-searchable and we are done. Otherwise, we perform a symmetric procedure for the sequence of vertices $r_1, \ldots, r_j$. In this way, we can determine in $O(n)$ time if there is a 1-searchable room in P, and if so report such a room. (Note that the algorithm of Bhattacharya et al. [2] needs to run only once for the polygon P, although its outputs (i.e., the deadlocks reported) are used several times in our algorithm.)

Let us turn to the situation in which the polygon P is LR-visible with respect to only one point pair, say, (d, l_i). In this case, r_1 (as well as d) is not contained in the component $P[l_1, Bcakw(l_1)]$. See Fig. 4(a). Following from the discussion made above, the work of verifying the condition **A3** is to compute the deadlocks for the vertices of $P[l_1, r_j]$. Since P is LR-visible with respect to both point pairs (d, l_i) and $(d, Backw(l_1))$ in this case, we can simply determine if all the vertices of $P[l_1, Backw(l_1)]$ have their deadlocks (Lemma 2). But, a new method for reporting the vertices of $P[Backw(l_1), r_j]$ having their deadlocks has to be developed. Let v_1 and v_2 denote two vertices such that their backward components $(P[v_l, Backw(v_l)]$, $l = 1, 2)$ do not contain r_1 and all such vertices are contained in $P[v_1, v_2]$. See Fig. 4(a). Clearly, $P[v_1, v_2] \subset P[d, l_i]$ holds. For any vertex $v \in P[v_1, v_2]$, no backward shot $Backw(v)$ can contribute to an x-deadlock, $x \in P[Backw(l_1), r_j]$; otherwise, the d-deadlock occurs, a contradiction. However, the shot $Forw(v)$ may contribute to an x-deadlock, $x \in P[Backw(l_1), r_j]$.

Let v' denote the vertex such that two shots $Backw(v')$ and $Forw(v)$ give the x-deadlock. Then, the vertex v' is contained in any component $P(Backw(v''), v'')$, $v'' \in P(v, v_2]$; otherwise, three vertices v, v' and v'' make the condition **A2** be true, a contradiction. This implies that the vertex l_i is contained in $P[v, v']$. Since the polygon P is weakly visible with respect to the point pair (d, l_i), these x-deadlocks with one defining vertex belonging to $P[v_1, v_2]$ can thus be found using Lemma 2. Clearly, when we compute other deadlocks, all vertices of $P[v_1, v_2]$ can be ignored. Note that the vertices v_1 and v_2 can be found by computing the shortest paths from r_1 to all vertices of $P[d, l_i]$ [5], and marking the vertices v such that the shortest path from r_1 to $Succ(v)$ turns left at v (as viewed from r_1). Let P' denote the polygon obtained after the chain $P[v_1, v_2]$ is deleted (i.e., connnecting $Pred(v_1)$ and $Succ(v_2)$ by a line segment). See Fig. 4(b) for an example. The polygon P' is now LR-visible with respect to both point pairs (d, r_j) and (d, l_i) (or $(d, Backw(l_1)$ if l_i is deleted). As discussed above, we can find the vertices of $P'[Backw(l_1), d]$, which have their deadlocks in the polygon P'. Since any pair of reflex vertices giving a deadlock in P' corresponds to a unique pair of reflex vertices of P, the same deadlock also occurs in P. In conclusion, we can determine in $O(n)$ time whether there is a 1-searchable room in P.

The situation in which the polygon P is LR-visible with respect to only the point pair (d, r_1) can be dealt with analogously. Note that the polygon P is LR-visible with respect to at least one pair of $(d, l_i))$ and (d, r_1); otherwise, the condition **A1** is true for (P, d), contradicting our assumption. This completes the proof. □

4 Conclusion

We have proposed an optimal $O(n)$ time algorithm to determine whether there is a point d on the boundary of P such that the room (P, d) is 1-searchable. Our result improves upon the previous $O(n \log n)$ time bound, which was established for determining whether a specified room is 1-searchable. A further work is to give a linear time algorithm to determine whether a simple polygon is 1-searchable, without considering any door [8,13,15].

Acknowledgements

This research is partially supported by the Grant-in-Aid of the Ministry of Education, Science, Sports and Culture of Japan.

References

1. B.K.Bhattacharya and S.K Ghosh, Characterizing LR-visibility polygons and related problems, *Comput. Geom. The. Appl.* **18** (2001) 19-36.
2. B.K.Bhattacharya, A. Mukhopadhyay and G.Narasimhan, Optimal algorithms for two-guard walkability of simple polygons, *Lect. Notes Comput. Sci.* **2125** (2001) 438-449.

3. B.Chazelle, Triangulating a simple polygon in linear time, *Discrte Comput. Geometry* **6** (1991) 485-524.
4. G.Das, P.J.Heffernan and G.Narasimhan, LR-visibility in polygons, *Comput. Geom. Theory Appl.* **7** (1997) 37-57.
5. L.J.Guibas, J.Hershberger, D.Leven, M.Sharir and R.E.Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, *Algorithmica*, **2** (1987) 209-233.
6. L.J.Guibas, J.C.Latombe, S.M.Lavalle, D.Lin and R.Motwani, Visibility-based pursuit-evasion in a polygonal environment, *Int. J. Comput. Geom. & Appl.* **9**, (1999) 471-493.
7. C. Icking and R. Klein, The two guards problem, *Int. J. Comput. Geom. & Appl.* **2** (1992) 257-285.
8. S.M.LaValle, B.Simov and G.Slutzki, An algorithm for searching a polygonal region with a flashlight, in *Int. J. Comput. Geom. & Appl.* **12** (2002) 87-113.
9. J.H. Lee, S.Y.Shin and K.Y.Chwa, Visibility-based pursuit-evasion in a polygonal room with a door, *Proc. 15th Annu. ACM Symp. Comput. Geom.* (1999) 281-290.
10. J.H.Lee, S.M.Park and K.Y.Chwa, Searching a polygonal room with one door by a 1-searcher, *Int. J. Comput. Geom. & Appl.* **10** (2000) 201-220.
11. J.H.Lee, S.M.Park and K.Y.Chwa, Simple algorithms for searchng a polygon with flashlights, *Inform. Process. Lett.* **81** (2002) 265-270.
12. I.Suzuki and M.Yamashita, Searching for mobile intruders in a polygonal region, *SIAM J. Comp.* **21** (1992) 863-888.
13. I.Suzuki, Y.Tazoe, M.Yamashita and T.Kameda, Searching a polygonal region from the boundary, *Int. J. Comput. Geom. & Appl.* **11** (2001) 529-553.
14. X.Tan, Efficient algorithms for searching a polygonal room with a door, *Lect. Notes Comput. Sci.* **2098** (2001) 339-350.
15. X.Tan, A characterization of polygonal regions searchable from the boundary, *Lect. Notes Comput. Sci.* **3330** (*Proc. of IJCCGGT 2003*) 200-215.

Appendix

A simple polygon P with two marked points s, t on its boundary is called a *corridor*, and denoted by (P, s, t). The 1-searcher is termed as *two guards* [7], if we require that the movement of the endpoint of the ray (as well as the 1-searcher) be continuous on the polygon boundary. The corridor (P, s, t) is said to be *walkable* by two guards if two guards starting at s can force the mobile intruder out of P through t, without allowing the intruder to touch s [7]. It has been shown that (P, s, t) is walkable by two guards if and only if $P[s, t]$ and $P[t, s]$ are weakly visible from each other and neither s-deadlocks nor t-deadlocks occur [7]. An $O(n)$ time algorithm has been given to determine if there is a point pair (s, t) on the boundary of P such that (P, s, t) is walkable by two guards [2].

It is clear that if the polygon P is walkable by two guards, then there is a 1-searchable room in P. However, the converse is not true. The room (P, d) shown in Fig. 5 is 1-searchable, but P is not walkable by two guards. This is because all points of $P[a, b]$ have their deadlocks, and any two chains $P[u, v]$ and $P[v, u]$, $u, v \in P[b, a)$, are not weakly visible from each other.

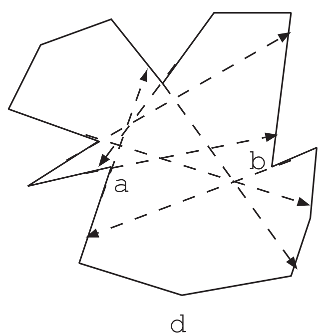

Fig. 5. A polygon has a 1-searchable room, but it is not walkable by two guards