Coverage Diameters of Polygons

Pawin Vongmasa
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand 10330
Email: tunaococ@msn.com

Attawith Sudsang
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand 10330
Email: attawith@cp.eng.chula.ac.th

***Abstract*— This paper formalizes and proposes an algorithm to compute *coverage diameters* of polygons in 2D. Roughly speaking, the coverage diameter of a polygon is the longest possible distance between two points through which the polygon cannot pass in between. The primary use of coverage diameter is to form a cage for transporting an object, not necessarily convex, with multiple disc-shaped robots. The main idea of the computation of coverage diameter is to convert the problem into a graph structure, then perform the search for a solution path in that graph. The proposed algorithm runs in $O(n^2 \log n)$ time for the input polygon with n vertices.**

I. Introduction

To cage an object means to limit object's configuration space to a bounded subset. Many types of caging problems have been raised, but the one that has been studied most extensively is how to form a cage by a number of point obstacles in $\mathbf{R}^2$. Solutions to this problem can be applied quite directly to transportation of an object by multiple disc-shaped robots: form a cage by robots, then move them together at the same velocity.

In practice, it is difficult to synchronize multiple robots to move together at exactly the same velocity, so if we are to transport an object by putting it in a moving cage formed by these robots, the cage should be allowed to deform a bit. This paper presents a new sufficient caging condition for this situation that is easy to maintain — it requires only that each robot can keep the distances from itself to its nearby friends under a predetermined value, which will be called "coverage diameter".

Roughly speaking, the coverage diameter of a rigid simple closed curve in $\mathbf{R}^2$ is the shortest length of gap (space between two points) that allows the curve to pass through. Therefore, the curve cannot escape from surrounding points, and is said to be *caged*, if every surrounding gap is smaller than the coverage diameter.

The notion of *caging* was first introduced in [1]. The problem of determining the *caging set* for 2-fingered gripping systems with one degree of freedom in the plane was studied in [2]. The extension to 3-fingered gripper, also with one degree of freedom, was explored in [3]. Caging in these works serves as a quick pre-process toward immobilizing grasp. Error-tolerance was added in [4].

Later in [5], a way to produce *v-grips* at concave vertices using two fingers was presented. Once two fingers form a v-grip, they can move inward or outward by some distance while preserving the caging condition. The maximum amount of distance that keeps the cage can be computed by the method presented in [6]. Note that in some cases (such as convex polygons) where no v-grips exist, *surrounding points* that satisfy our coverage diameter condition can serve as a complementary cage.

Manipulating polygonal objects using three 2-DOF robots, proposed in [7], involved caging as a transition between different form closures. The need of form closures was relaxed with the help of *MICaDs* (*maximum independent capture discs*) introduced in [8]. Then, motion planning and caging were combined in [9]. Nonetheless, these methods are applicable only to the case of three robots with high functionalities.

Cages of more than three robots were discussed in [10]. The notion of *object closure* was introduced and used in stating a sufficient and necessary condition for caging, but the test to verify object closure involves complicated operations and is extremely time-consuming. An alternative test method which takes less time was also presented in [10], but its completeness was not guaranteed.

Presented in [11] is a new sufficient condition for caging which is, though as well incomplete, much easier to check and more practical in many situations. It involves the calculation of *diameter function* of convex polygons, which was first brought up in [12]. However, the sufficient condition stated in [11] can be both tightened and simplified with the meaning of coverage diameter. The improved condition is stated in Section II after an intuitive definition of coverage diameter is discussed.

In Section III, we introduce related terms, redefine coverage diameter formally, and state lemmas that are needed in the computation of coverage diameter of polygons. The idea of the computation involves formulating the problem into a graph structure. Such formulation is explained in Section IV. Finally, the pseudocode of the algorithm and some experimental results are shown in Section V.

II. Coverage Diameter and Caging Condition

Imagine an object in a cage formed by points. If the object is about to escape the cage, it must get through a gap between some two points of the cage. Our original goal is to find the largest separation distance between points such that the object cannot escape.

But in reality, when we try to transport the object by forming a moving cage with mobile robots, distances among

them cannot easily be kept constant as they move. It is more practical to allow some distance change and maintain only the upper bound of separation distance. We call this upper bound value the *coverage diameter* and denote it by $\phi_{cov}(C)$ if the curve is C. A new sufficient condition for C to be caged by surrounding points is immediate from the notion of $\phi_{cov}(C)$.

Lemma 1: Let P be a polygon with vertices P_1, P_2, P_3, ..., $P_n \in \mathbf{R}^2$ arranged counterclockwise and let $P_0 = P_n$. If C is a rigid closed curve that lies inside P and $\|P_i - P_{i-1}\| < \phi_{cov}(C)$ where $i \in \{1, 2, 3, ..., n\}$, then C is caged by $\{P_1, P_2, P_3, ..., P_n\}$.

Note that the cage formed by surrounding points can contain more than one objects provided that all separation distances are smaller than the minimum coverage diameter of all objects (Fig. 2).

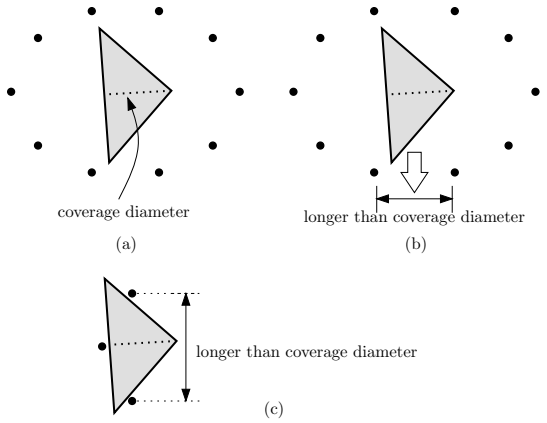

Fig. 1. The condition is sufficient but not necessary. (a) The object is caged because all gaps are smaller than the coverage diameter. (b) The object may escape when there is a gap larger than the coverage diameter. (c) The object cannot escape despite the presence of a gap larger than the coverage diameter.

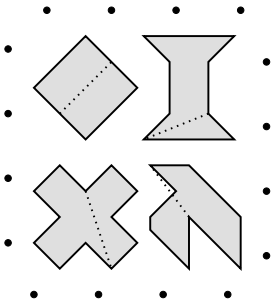

Fig. 2. Shown in dotted lines are coverage diameters. All objects are caged because all gaps are smaller than the shortest coverage diameter.

III. Fundamentals

Definition 1: An *arc* is a connected set of points homeomorphic to the closed interval $[0, 1]$. A *directed arc* is a triple

$$Z = (A, s, d)$$

where A is an arc, s is an endpoint of A, and d is the other endpoint of A. $initial(Z) = s$ is called the *initial point* of Z, $final(Z) = d$ is called the *final point* of Z, and $graph(Z) = A$ is called the *graph* of Z.

Motion of a point will be represented by a directed arc because magnitude of velocity is not relevant to our discussion. Next, we will consider the paired motion of two points together because a gap in $\mathbf{R}^2$ is defined by two points.

Definition 2: Let V be a Euclidean space.

1) If $a \in V$ and $b \in V$, the *inner distance* of $(a, b) \in V^2$ is $\|(a, b)\|_{in} = \|a - b\|$.
2) If $A \subseteq V^2$, the *maximum inner distance* of A is

$$\|\lceil A \rceil\|_{in} = \max\{\|a\|_{in} \mid a \in A\}.$$

Similarly, the *minimum inner distance* of A is

$$\|\lfloor A \rfloor\|_{in} = \min\{\|a\|_{in} \mid a \in A\}.$$

A. Coverages and the Coverage Diameter

Imagine when C is being pushed through a gap between two points a and b in $\mathbf{R}^2$. The situation will be viewed from a-b's frame of reference where a is the origin and b is on the positive-y axis. Note that b is allowed to move up and down along the positive-y axis.

In the initial set up, let C lie totally in the left half plane. Every point of C has an integer called the *coverage count* attached to it. All coverage counts are initially zero.

At any instant, the *cross section* is the part of C that intersects the straight line segment drawn from a to b. If a point of C is in the cross section and is moving into the right half plane, the coverage count of that point is increased by 1. Oppositely, the coverage count is decreased by 1 if it is moving into the left half plane.

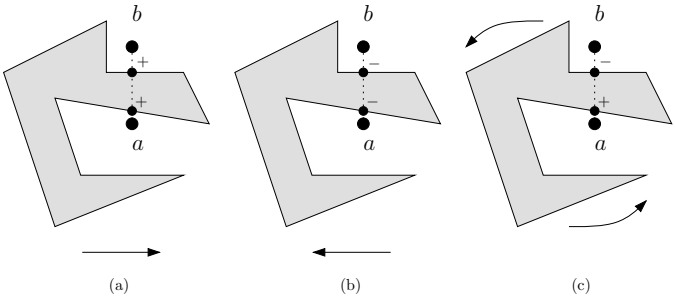

Fig. 3. (a) The object is moving to the right. Coverage counts of points in the cross section will increase by 1. (b) The object is moving to the left. Coverage counts of points in the cross section will decrease by 1. (c) The object is rotating around a point (not shown). Coverage counts of points in the cross section may decrease or increase depending on the position of the fixed point.

When the whole motion is known, a point of C starts to be *covered* at the last moment its coverage count changes from 0 to 1. Once a point becomes covered, it remains covered for the rest of the motion (Fig. 4).

When all coverage counts are equal to 1, C is said to be *covered* by the paired motion of a and b viewed from C's frame of reference. Such paired motion can be represented by a directed arc in $\mathbf{R}^4$ that is called a *coverage* of C.

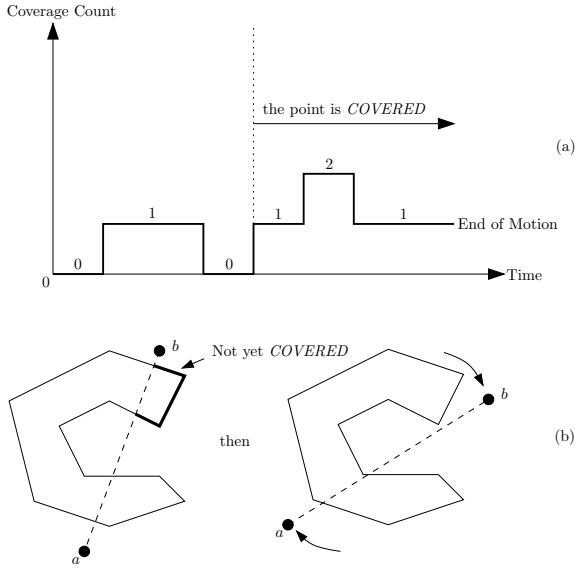

Fig. 4. (a) The point starts to be *covered* at the moment its coverage count increases from and never decreases to 0. If we know that the point is eventually covered, it is possible to view backward (from right to left) and marks the point as *uncovered* once its count reaches zero. (b) An example of not-yet-covered points in the real situation.

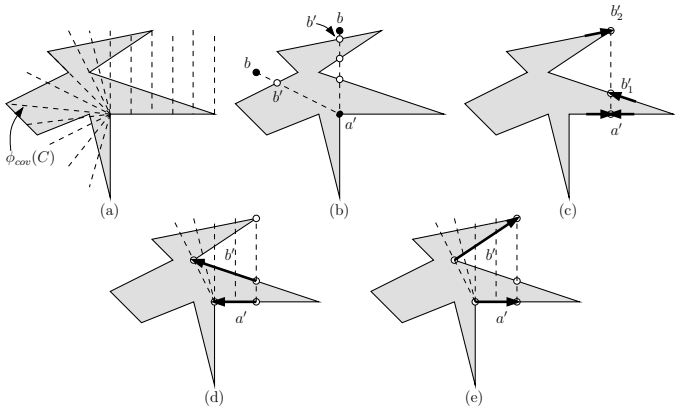

Fig. 5. (a) Some snapshots of Z. (b) Covered points in some cross sections are shown in white. a' and b' are the farthest pair in K, i.e. $\|(a', b')\|_{in} = \|\lceil K^2 \rceil\|_{in}$. (c) The point of discontinuity: (a', b'_1) and (a', b'_2) are different limits. (d-e) A directed arc in U that fixes the discontinuity.

The maximum inner distance of a coverage is equal to the maximum separation distance between a and b during the motion. It is obvious that the smallest maximum inner distance of all coverages is exactly the same as *coverage diameter* we have mentioned.

Definition 3: If C is a simple closed curve in $\mathbf{R}^2$, its coverage diameter is

$$\phi_{cov}(C) = \min\{\|\lceil graph(Z) \rceil\|_{in} \mid Z \text{ is a coverage of } C\}$$

Note that there are infinitely many coverages whose maximum inner distances are equal to $\phi_{cov}(C)$. We need to find just one of them.

B. Boundary Coverages

A coverage Z of C is called a *boundary coverage* if

1) $graph(Z) \subseteq C^2$
2) $\|initial(Z)\|_{in} = \|final(Z)\|_{in} = 0$

We claim that there always exists a boundary coverage whose maximum inner distance is equal to $\phi_{cov}(C)$. This fact allows us to limit the search to boundary coverages only.

To prove the claim, let Z be a coverage with $\|graph(Z)\|_{in} = \phi_{cov}(C)$ (Fig. 5.a). At each instant of the motion that constitutes Z, let a and b be the two moving points and let K be the set of points in the cross section *that are "covered" (as in Fig. 4) at that time*. It is obvious that $\|\lceil K^2 \rceil\|_{in} \leq \phi_{cov}(C)$ for all K.

Let U be the union of all K^2, written as:

$$U = \bigcup_{\text{all } K} K^2$$

It is again obvious that $\|\lceil U \rceil\|_{in} = \phi_{cov}(C)$.

Next, let a' and b' be two new moving points in K that are closest to a and b (Fig. 5.b). We know that $(a', b') \in U$ but a' and b' might not be continuous with respect to time (Fig. 5.c). However, these discontinuities can be filled by directed arcs in U whose initial and final points are the different limits (Fig. 5.d, 5.e). These directed arcs always exist because U is connected due to the fact that once a point is covered, it remains covered through the rest of the motion. Therefore, the paired motion (a', b') with all discontinuities removed composes a coverage whose graph is a subset of $U \subseteq C^2$.

Combining this result with the fact that $(p, p) \in U$ for all $p \in C$, the coverage just described can be extended such that its initial and final points have zero inner distance. If we let the initial point be any member of the first non-empty K^2 and the final point be any member of the last non-empty K^2, the result of this extension becomes a boundary coverage. Our claim is now proved and we summarize it in the following lemma.

Lemma 2: For a given simple closed curve $C \subset \mathbf{R}^2$, there always exists a boundary coverage Z such that $\|\lceil graph(Z) \rceil\|_{in} = \phi_{cov}(C)$.

Proof: The above remark constitutes a proof. ■

C. Line Segments

A line segment is a special kind of arcs whose points can be written as a linear function of one variable. Line segments will be involved when we work with polygons, a special kind of simple closed curves. The following lemma assumes that distance of a point is measured from the origin.

Lemma 3: If p and q are the two endpoints of a line segment L, $\max\{\|x\| \mid x \in L\} = \max\{\|p\|, \|q\|\}$.

Proof: If $\|p\| \geq \|q\|$, the circle with radius $\|p\|$ centered at the origin will contain L; therefore, $\max\{\|x\| \mid x \in L\} = \|p\|$. The other case where $\|p\| < \|q\|$ is proved by exchanging p and q. ■

Directed arcs whose graphs are line segments are called *directed line segments*. The following lemma shows an impor-

tant characteristic of paired directed line segments that will be used in the next section.

Lemma 4: Given two directed line segments X and Y in $\mathbf{R}^2$, if Z is a line segment in $\mathbf{R}^4$ whose endpoints are $(initial(X), initial(Y))$ and $(final(X), final(Y))$, then

$$\begin{aligned}\|\lceil Z\rceil\|_{in} &= \max\{\|initial(X) - initial(Y)\|, \\ &\quad \|final(X) - final(Y)\|\}\end{aligned}$$

Proof: See Appendix. ∎

IV. Computation of $\phi_{cov}(C)$

We are going to find the coverage diameter of the polygon $C \subseteq \mathbf{R}^2$ that has n vertices and n edges, namely V_i and E_i where $i \in \mathbf{Z}_n$ [1]. V_i are ordered counterclockwise. Every E_i is a line segment in $\mathbf{R}^2$ with endpoints V_i and V_{i+1}.

To find $\phi_{cov}(C)$, we will construct a graph G that contains enough information of C, then perform a search in G. The steps toward construction of G are outlined below:

1) We will define subsets of C^2 called *states*. Every point of C^2 will belong to exactly one state.
2) Every directed arc in C^2 will have a corresponding *state sequence*. We will show that it is adequate to consider only state sequences without looking at any corresponding directed arcs.
3) *Initial* and *final* states will be defined by examining state sequences of boundary coverages.
4) Some states will be chosen to become *nodes* of G and we will derive *edges* of G from state adjencies.

After G is completely defined, the algorithm to find $\phi_{cov}(C)$ will be presented.

A. States

The following subsets of C^2 for all $i, j \in \mathbf{Z}_n$ are called states:

- $V_iV_j = \{(V_i, V_j)\}$
- $V_iE_j = \{V_i\} \times E_j$
- $E_iV_j = E_i \times \{V_j\}$
- $E_iE_j = E_i \times E_j$

Note that the union of all states is C^2.

Let $\lfloor X\rfloor_{in}$ denote the point of a state X with smallest inner distance. If there are more than one points with equal inner distance, we can choose any one of them. Note that $\|\lfloor X\rfloor\|_{in}$ (from Definition 2) is equal to the inner distance of $\lfloor X\rfloor_{in}$.

We need to know $\lfloor X\rfloor_{in}$ of all states X. For states of the form V_iV_j, it is obvious that $\lfloor V_iV_j\rfloor_{in} = (V_i, V_j)$. Next, if we let $P_{i,j}$ be *projections* [2] of V_i on E_j, then $\lfloor V_iE_j\rfloor_{in} = (V_i, P_{i,j})$ and $\lfloor E_iV_j\rfloor_{in} = (P_{j,i}, V_i)$.

In order to find $\lfloor E_iE_j\rfloor_{in}$, it is easy to show that $\lfloor E_iE_j\rfloor_{in}$ must be equal to at least one of these four points: $\lfloor V_iE_j\rfloor_{in}$, $\lfloor V_{i+1}E_j\rfloor_{in}$, $\lfloor E_iV_j\rfloor_{in}$ and $\lfloor E_iV_{j+1}\rfloor_{in}$. Three comparisons will give the correct value of each $\lfloor E_iE_j\rfloor_{in}$.

[1] $\mathbf{Z}_n$ is a group of non-negative integers less than n. Additions and subtractions are calculated modulo n.

[2] In this context, the *projection* of a point p on a set S is the point of S closest to p. With this definition, the projection must exist and be unique given that S is a line segment.

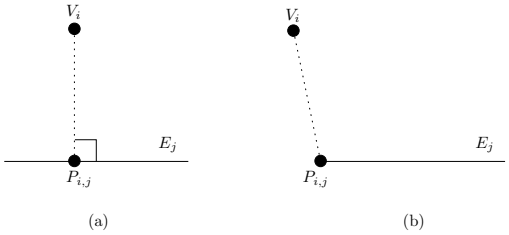

Fig. 6. The line segment $\overline{V_iP_{i,j}}$ may or may not be perpendicular to E_j. (a) $\overline{V_iP_{i,j}}$ is perpendicular to E_j. (b) $\overline{V_iP_{i,j}}$ is not perpendicular to E_j and $P_{i,j}$ coincides with V_j or V_{j+1}.

B. State Sequences

For a given directed arc Z in C^2, let $\sigma(Z)$ be the sequence of states that points of Z belong to. Here arises one problem: a point of C^2 may be a member of more than one states. We will try to choose states of the form V_iV_j first, V_iE_j and E_iV_j next, then E_iE_j. As "belong to" is defined this way (different from "$\in$"), the function $\sigma(Z)$ is now clearly defined.

Let S be a finite state sequence of length m:

$$S = (S_1, S_2, S_3, ..., S_m)$$

The *maxi-min inner distance* of S is

$$\|\lceil\lfloor S\rfloor\rceil\|_{in} = \max\{\|\lfloor S_i\rfloor\|_{in} \mid i = 1, 2, 3, ..., m\}$$

It is easy to show that $\|\lceil graph(Z)\rceil\|_{in} \geq \|\lceil\lfloor S\rfloor\rceil\|_{in}$ whenever $\sigma(Z) = S$.

Two states X and Y are *adjacent* if and only if the state sequence (X, Y) exists. All state adjacencies are listed below:

- States adjacent to V_iV_j:
  E_iE_j, E_iE_{j-1}, $E_{i-1}E_j$, $E_{i-1}E_{j-1}$,
  V_iE_j, V_iE_{j-1}, E_iV_j and $E_{i-1}V_j$
- States adjacent to V_iE_j:
  E_iE_j, $E_{i-1}E_j$, V_iV_j and V_iV_{j+1}
- States adjacent to E_iV_j:
  E_iE_j, E_iE_{j-1}, V_iV_j and $V_{i+1}V_j$
- States adjacent to E_iE_j:
  V_iV_j, V_iV_{j+1}, $V_{i+1}V_j$, $V_{i+1}V_{j+1}$,
  V_iE_j, $V_{i+1}E_j$, E_iV_j and E_iV_{j+1}

We are going to show that if X and Y are adjacent states, then $\lfloor X\rfloor_{in} \in X \cap Y$ or $\lfloor Y\rfloor_{in} \in X \cap Y$. Since V_iE_j are not adjacent to any V_kE_l or E_kV_l, it suffices to consider only these cases:

- If $X = V_iV_j$: $X \subseteq Y$, so $\lfloor X\rfloor_{in} \in X \cap Y$.
- If $X = E_iE_j$: $Y \subseteq X$, so $\lfloor Y\rfloor_{in} \in X \cap Y$.

This means if X and Y are adjacent states, there always exists a directed arc Z such that $\sigma(Z) = (X, Y)$ and $\|\lceil Z\rceil\|_{in} = \max\{\|\lfloor X\rfloor\|_{in}, \|\lfloor Y\rfloor\|_{in}\}$. We also know that all states are convex subsets of $\mathbf{R}^4$, which means once a point enters a state S_i, it can move to $\lfloor S_i\rfloor_{in}$ in a straight line motion. Therefore, when the state sequence S is given, it is always possible to find Z such that $\sigma(Z) = S$ and, with the help of Lemma 4, $\|\lceil Z\rceil\|_{in} = \|\lceil\lfloor S\rfloor\rceil\|_{in}$.

C. Initial and Final States

Imagine the paired motion of two points (a, b) that constitutes a boundary coverage Z again, the constraint "$\|initial(Z)\|_{in} = \|final(Z)\|_{in} = 0$" requires that the two points start from one initial point, split in opposite directions relative to each other, then meet again at one final point. Without loss of generality, we can assume two things:

- a and b do not cross during the whole time except at the initial and final points.
- When a and b split and meet, a is moving in the *clockwise* direction relative to b.

Every state that contains a point of the form (p, p) can act as both an *initial state* and a *final state*. All states that has (p, p) are: V_iV_i, V_iE_i, V_iE_{i-1}, E_iV_i, $E_{i-1}V_i$, E_iE_i, E_iE_{i+1} and E_iE_{i-1}. We will look more closely at this matter after states become nodes of G.

D. The Graph G

From all the above discussions, it seems rather clear how G should be constructed: states become nodes, transitions become edges, and state sequences become paths. The idea of the algorithm is also simple: find a state sequence that starts from one initial state, ends at one final state, and has the smallest maxi-min inner distance. This simple idea actually works, but there are numerous redundancies that should be removed for efficiency.

Start by looking at E_iE_j. It is obvious that all minimum inner distances of states adjacent to E_iE_j are never smaller than $\|\lfloor E_iE_j \rfloor\|_{in}$. This means if X and Y are states adjacent to E_iE_j, $\|\lfloor E_iE_j \rfloor\|_{in}$ can be excluded from the calculation of maxi-min inner distance of (X, E_iE_j, Y). With another observation that adjacent states of E_iE_j are NOT of the form E_kE_l, all E_iE_j can be removed from every sequence as they are subsumed by adjacent states.

Next, consider states of the form V_iV_j. Before E_iE_j are removed, all states adjacent to V_iV_j are not of the form V_kV_l, but once we bypass E_iE_j, V_iV_j can jump to some V_kV_l. Still, we can manage to insert some states of the form V_iE_j or E_iV_j in between and reduce some adjacencies. Look at the state sequence

$$(V_iV_j, E_iE_j, V_{i+1}V_{j+1}).$$

The corresponding path with E_iE_j removed is

$$V_iV_j \to V_{i+1}V_{j+1}$$

We know that intermediate states (adjacent to V_iV_j and $V_{i+1}V_{j+1}$) are V_iE_j, E_iV_j, $V_{i+1}E_j$ and E_iV_{j+1}, all of which have minimum inner distances not exceeding $\max\{\|V_i - V_j\|, \|V_{i+1} - V_{j+1}\|\}$. Inserting any of them in the middle does not increase the maxi-min inner distance of the path, so we do so

$$V_iV_j \to V_iE_j \to V_{i+1}V_{j+1}$$

All the other cases, such as (V_iV_j, X, V_iV_{j+1}), can be handled by similar arguments. Edges of G, then, do not have to include $V_iV_j \to V_kV_l$.

We now revisit the problem of specifying *initial nodes* and *final nodes*. From the previous discussion, nodes that may act as initial and final nodes are: V_iV_i, V_iE_i, V_iE_{i-1}, E_iV_i and $E_{i-1}V_i$. We want to assign each of them as initial or final only, but not both.

- V_iE_i should be declared *initial* because (p, q) belongs to V_iE_i implies that $p = V_i$ and q lies in the counter-clockwise direction from p (due to our restriction of the counterclockwise arrangement of V_i).
- V_iE_{i-1} should be declared *final* because (p, q) belongs to V_iE_{i-1} implies that $p = V_i$ and q lies in the clockwise direction from p.
- E_iV_i should be declared *final* because V_iE_i are *initial*.
- $E_{i-1}V_i$ should be declared *initial* because V_iE_{i-1} are *final*.
- V_iV_i can be removed because all nodes adjacent to V_iV_i have already been declared initial or final.

We are now ready to list all nodes and edges of G as follows:

1) From V_iE_i:
   $V_{i-1}E_i$, V_iV_{i+1} and $E_{i-1}V_{i+1}$
2) From $E_{i-1}V_i$:
   $V_{i-1}V_i$, $V_{i-1}E_i$ and $E_{i-1}V_{i+1}$
3) From E_iV_i:
   E_iV_{i-1}, $V_{i+1}V_i$ and $V_{i+1}E_{i-1}$
4) From V_iE_{i-1}:
   V_iV_{i-1}, E_iV_{i-1} and $V_{i+1}E_{i-1}$
5) From V_iV_j, $i \neq j$:
   V_iE_j, V_iE_{j-1}, E_iV_j and $E_{i-1}V_j$
6) From V_iE_j, $i \neq j$ and $i \neq j + 1$:
   E_iV_j, E_iV_{j+1}, $E_{i-1}V_j$, $E_{i-1}V_{j+1}$, V_iV_j and V_iV_{j+1}
7) From E_iV_j, $i \neq j$ and $i \neq j + 1$:
   V_iE_j, $V_{i+1}E_j$, V_iE_{j-1}, $V_{i+1}E_{j-1}$, V_iV_j and $V_{i+1}V_j$

(1) and (2) are from initial nodes. (3) and (4) are from final nodes. (5), (6) and (7) are from internal (neither initial nor final) nodes. The total number of nodes is $3n^2 - n$ and the total number of edges is $8n^2 - 8n$.

To find $\phi_{cov}(C)$, we need to search for a path S in G that starts from an initial node, ends at a final node, and has the smallest maxi-min inner distance. Once S is found, a boundary coverage Z of C such that $\sigma(Z) = S$ and $\|\lceil graph(Z) \rceil\|_{in} = \|\lceil \lfloor S \rfloor \rceil\|_{in}$ always exists.

V. Algorithm

Our goal is to find a path S in G that *covers* C, i.e. starts from an initial node and ends at a final node. The characteristic of maxi-min inner distance of paths allows us to apply the idea from Dijkstra's shortest path algorithm.

The pseudocode of the algorithm is shown in Fig. 7. Note that if $\max\{d, \|\lfloor t \rfloor\|_{in}\}$ in line (11) is replaced by $d +$ [distance of $s \to t$], the pseudocode becomes exactly the shortest path algorithm. The running time of this algorithm is accordingly $O(n^2 \log n)$.

The algorithm has been implemented in C++ and tested on 1.5 GHz CPU with 512 MB RAM. It could process the input polygon with 1000 vertices within 3 seconds. Some visual experimental results are shown in Fig. 8.

```
(1)  Let Visited be a set, initially empty, for storing visited
     nodes.
(2)  Let H be a min-heap, initially empty, for storing
     a couple (d, s) where d is a real number used in
     comparison and s is a node of G.
(3)  For each initial node s, add (0, s) to H.
(4)  Repeat the following until φ_cov(C) is found:
(5)  begin
(6)     Retrieve and remove the minimum couple (d, s)
        from H.
(7)     Add s to Visited.
(8)     If s is a final node, report that φ_cov(C) = d and
        terminate.
(9)     For each t ∉ Visited such that edge s → t exists,
(10)    begin
(11)       Let m = max{d, ‖⌊t⌋‖_in}.
(12)       If there exists a couple (e, t) in H for some e,
           then
(13)       begin
(14)          If m < e, replace (e, t) in H by (m, t) and
              adjust H.
(15)          Else, do nothing.
(16)       end.
(17)       Else, add (m, t) to H.
(18)    end.
(19) end.
```

Fig. 7. The pseudocode of our searching algorithm.

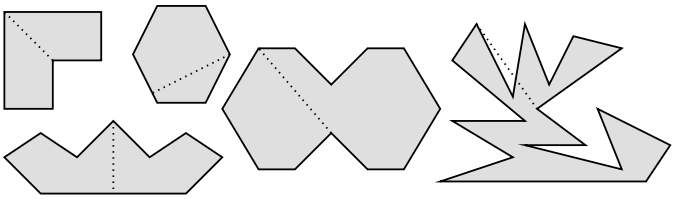

Fig. 8. Some experimental results. Coverage diameters are shown in dotted lines.

VI. Discussion and Conclusion

We have formalized a new property of simple closed curves in $\mathbf{R}^2$ called "coverage diameter" and shown how it can be used in the problem of caging an object with point obstacles. A new sufficient condition for caging which is tighter and simpler than the condition stated in [11] is also proposed. Though our condition is sufficient but not necessary, it is suitable for forming a moving cage with robots that have low functionalities.

The idea of the algorithm to compute coverage diameter of a polygon is based on the notion of paired motion of points and the property of directed line segments shown in Lemma 4. The algorithm runs in $O(n^2 \log n)$ time provided that the polygon has n vertices. Coverage diameters of all figures shown in this paper have been verified by the algorithm we implemented.

The following are probable extensions we have foreseen:

- Extracting some more information, such as possible two-fingered contracting grips and cages, from the graph constructed in Section IV.
- Allowing curved edges in the input shape. (Lemma 4 will require a substitute.)

Appendix

Proof of Lemma 4

Proof: The existence and uniqueness of Z follow at once as its two endpoints are specified. It is possible to let points in X, Y and Z be formulated as linear functions of the same variable $t \in [0, 1]$ as follows:

$$\begin{aligned} X(t) &= (1-t) \cdot initial(X) + t \cdot final(X) \\ Y(t) &= (1-t) \cdot initial(Y) + t \cdot final(Y) \\ Z(t) &= (X(t), Y(t)) \end{aligned}$$

Let $W(t) = X(t) - Y(t)$; it follows that the graph of $W(t)$ where $0 \le t \le 1$ is a line segment in $\mathbf{R}^2$ with endpoints $X(0) - Y(0)$ and $X(1) - Y(1)$. From Lemma 3, we have that

$$\begin{aligned} &\max\{\|W(t)\| \mid 0 \le t \le 1\} \\ &\quad = \max\{\|X(0) - Y(0)\|, \|X(1) - Y(1)\|\} \end{aligned}$$

Also, from Definition 2,

$$\|Z(t)\|_{in} = \|X(t) - Y(t)\| = \|W(t)\|,$$

and finally,

$$\begin{aligned} &\|\lceil Z \rceil\|_{in} \\ &\quad = \max\{\|Z(t)\|_{in} \mid 0 \le t \le 1\} \\ &\quad = \max\{\|W(t)\| \mid 0 \le t \le 1\} \\ &\quad = \max\{\|X(0) - Y(0)\|, \|X(1) - Y(1)\|\} \\ &\quad = \max\{\|initial(X) - initial(Y)\|, \\ &\qquad \|final(X) - final(Y)\|\} \end{aligned}$$

The proof is now finished. ∎

References

[1] W. Kuperberg, "Problems on polytopes and convex sets," *DIMACS Workshop on Polytopes*, pp. 584–589, January 1990.

[2] E. Rimon and A. Blake, "Caging 2d bodies by 1-parameter two-fingered gripping systems," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 1996, pp. 1459–1464.

[3] C. Davidson and A. Blake, "Caging planar object with a three-finger one-parameter gripper," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1998, pp. 2722–2727.

[4] ——, "Error-tolerant visual planning of planar grasp," in *Proceedings of IEEE International Conference on Conference on Computer Vision*, 1998, pp. 911–916.

[5] K. G. Gopalakrishnan and K. Goldberg, "Gripping parts at concave vertices," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 2002, pp. 1590–1596.

[6] A. Sudsang and T. Luewirawong, "Capturing a concave polygon with two disc-shaped fingers," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2003.

[7] A. Sudsang, J. Ponce, M. Hyman, and D. J. Kriegman, "On manipulating polygonal objects with three 2-dof robots in the plane," in *Proceedings of IEEE International Conference on Robotics and Automation*, May 1999, pp. 2227–2234.

[8] A. Sudsang and J. Ponce, "A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane," in *Proceedings of IEEE International Conference on Robotics and Automation*, April 2000, pp. 1068–1075.

[9] A. Sudsang, F. Rothganger, and J. Ponce, "Motion planning for disc-shaped robots and pushing a polygonal object in the plane," *IEEE Transactions on Robotics and Automation*, 2002.

[10] Z. WANG and V. Kumar, "Object closure and manipulation by multiple cooperating mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2002.

[11] A. Sudsang, "A sufficient condition for capturing an object in the plane with disc-shaped robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2002, pp. 682–687.

[12] K. Y. Goldberg, "Orienting polygonal parts without sensors," in *Algorithmica (Historical Archive)*, vol. 10, Aug 1993, pp. 201–225.