2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Two-Finger Squeezing Caging of Polygonal and Polyhedral Object

Peam Pipattanasomporn, Pawin Vongmasa and Attawith Sudsang

***Abstract*— The problem of object caging is defined as a problem of designing a formation of fingers to restrict an object within a bounded space. Assuming two pointed fingers and a rigid polygonal or polyhedral object, this paper addresses the problem of two-finger *squeezing caging*, i.e., to characterize all possible formations of the fingers that are capable of caging the object via limiting their separation distance. Our study is done entirely in the object's frame allowing the object to be considered as a static obstacle so that the analysis can be performed in terms of the finger motion. Our solution is based on partitioning the configuration space of the problem into finite subsets called nodes. A graph of these nodes can then be constructed to represent all possible finger motion where a search based method can be applied to solve the caging problem. The partitioning of the configuration is based on convex decomposition of the free space. Let m be the number of convex subsets from the decomposition, our proposed algorithm reports all squeezing cage sets in $O(n^2+nm+m^2 \log m)$ for a polygonal input with n vertices and $O(nN^3 + n^2 + nm + m^2 \log m)$ for a polyhedron with n vertices and having N edges exhibiting a reflex angle. After reporting all squeezing cages, the proposed algorithm can answer whether a given finger placement can cage the object within a logarithmic time.**

I. INTRODUCTION

An object is caged when it is restricted to stay in a bounded space by a formation of fingers. The caging problem was originally posed by Kuperberg in [1] as a problem of designing an algorithm for finding a set of points that prevent a polygon from moving arbitrarily far from a position. In the past few decades, the concept of caging has been applied to a number of manipulation and related problems such as transportation using mobile robots [2], part feeding [3], and object grasping [4].

A number of works have attempted to solve the caging problem in 2D. In particular, the problem of two-finger caging of concave planar objects has received considerable attention. Rimon and Blake [5] applied the stratified Morse theory to solve the problem of caging an object in the plane with two fingers and introduced the notion of caging set (also known as capture region [6]), a set of system configurations (e.g., finger formation) that can prevent the object from escaping. Although their solution can be applied to general planar objects, it requires complex numerical computation. Sudsang and Luewirawong [7] confined their problem to polygonal objects and proposed an analytical method for computing, for any immobilizing pair of vertices, an acceptable separation distance between the two fingers that guarantees caging. Since their method takes into account only immediate edges of the immobilizing vertices without considering neighboring edges, it can report only subset of entire caging (i.e., larger separation distance could be missing). Recently, Pipattanasomporn and Sudsang [8] and Vahedi and Stappen [9] have independently developed complete $O(n^2 \log n)$ algorithms for characterizing all two-finger squeezing and stretching cage sets of a polygon with n vertices. They also provided data structures for querying whether a given finger placement forms a cage in $O(\log n)$.

To the best of our knowledge, the two-finger squeezing caging problem for polyhedral objects has never been explored. It is the main objective of this paper to present a solution to the problem. Like our previous work [8], our study is done entirely in the object's frame allowing the object to be considered as a static obstacle so that the analysis can be performed in terms of the finger motion. Our solution requires partitioning the configuration space of the problem (i.e., set of all admissible finger placements) into finite subsets called nodes. A graph of these nodes can then be constructed to represent all possible finger motion where a search based method can be applied to solve the caging problem. The partitioning of the configuration is based on convex decomposition of the free space. Let m be the number of convex subsets from the decomposition, our proposed algorithm reports all squeezing cage sets in $O(n^2+nm+m^2 \log m)$ for a polygonal input with n vertices and $O(nN^3+n^2+nm+m^2 \log m)$ for a polyhedron with n vertices and having N edges exhibiting a reflex angle. After reporting all squeezing cages, the proposed algorithm can answer whether a given finger placement can cage the object within a logarithmic time. Note that the main algorithm is designed to be independent from the dimension of the object, so it can be easily applied to the polyhedral case and the polygonal case as well. However, for readability, illustrations given in the paper are drawn based on the polygonal cases.

The remainder of the paper is organized as follows. In Section II, necessary background and assumptions are given. This section begins with an informal definition of caging and gradually defines formal notations to transform the definition into a more formal version in which the idea of critical distance given in Section III can be applied. The application of the idea leads to the main result in Section IV where the proposed algorithm for reporting all squeezing cage sets is presented. The running time analysis of the algorithm is given in Section V. Finally, in Section VI, we conclude the paper with some discussion and future works.

P. Pipattanasomporn, P. Vongmasa and A. Sudsang are with Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 10330 Bangkok, Thailand attawith@cp.eng.chula.ac.th

II. DEFINITIONS AND ASSUMPTIONS

The caging problem is defined here to be the problem of designing a placement of fingers for restricting an object to stay within a bounded space. We assume freely movable pointed fingers and a rigid polygonal or polyhedral object, possibly more than one connected components but behave like a single rigid body (i.e. the distance between any two points, either of the same component or of two distinct components, is preserved during the object's motion). The object is also assumed that it must not contain any inaccessible regions (i.e. the free space is a connected component) for simplicity. Without loss of generality, our study is conducted entirely in the object's frame. As a result, the object can be viewed as a static obstacle while the fingers are allowed to move in the free space. This viewpoint enables the caging problem to be conveniently analyzed in terms of the fingers' motion. As for our squeezing caging problem, we can state the definition of caging as follows.

Definition 1: An object is squeezing caged by a finger placement with separation distance δ when there is no collision-free finger motion to bring the fingers arbitrarily far from the object given that the separation distance between the fingers never exceeds δ during the entire motion.

It is important to notice that if the fingers can go arbitrarily far while limiting their separation distance not to exceed δ (equivalent to that the object can escape from the finger placement), it is true that the fingers can thereafter move to the same location (both fingers are at the same point) without increasing their separation distance and once at the same location, they can move together anywhere in the free space. With this observation, the definition of caging above can be modified by replacing the condition on nonexistence of finger motion to bring the fingers arbitrarily far from the object by the condition on nonexistence of finger motion to bring both fingers to the same location. In Section IV, it will be shown that this modified definition translates nicely to the proposed algorithm for computing all caging sets. For convenience, let us refer to a finger placement having both fingers at the same location as an escapable placement. In the rest of this section, a more formal version of Definition 1 will be derived.

Let us be more precise about the configuration of the problem. Our system consists of two pointed fingers and a rigid object (either polygonal or polyhedral). Each finger can be placed anywhere outside the interior of the object. Let us refer to the set of all locations where a finger can be placed by $\mathbb{W}$. Any formation of two fingers for which each finger is in the workspace $\mathbb{W}$ is called a finger placement (in the configuration space of the problem, i.e., $\mathbb{C} = \mathbb{W} \times \mathbb{W}$). A finger placement $\boldsymbol{x} = (\boldsymbol{a} \in \mathbb{W}, \boldsymbol{b} \in \mathbb{W})$ in $\mathbb{C}$ is said to have a *separation distance* $[\boldsymbol{x}] = \|\boldsymbol{a} - \boldsymbol{b}\|$ where $\|\boldsymbol{a} - \boldsymbol{b}\|$ denotes the Euclidean distance between $\boldsymbol{a}$ and $\boldsymbol{b}$.

Clearly, the fingers in one placement can be brought to another by executing a synchronized collision-free motion. This motion is essentially a trajectory in $\mathbb{C}$. However, the trajectory's velocity can be ignored in our caging analysis, so it suffices to consider only the underlying configuration curve path of the trajectory. We say that placements $\boldsymbol{x}$ and $\boldsymbol{y}$ are connected by a configuration curve Γ when $\boldsymbol{x} \in \Gamma$ and $\boldsymbol{y} \in \Gamma$. Let us also define the *upper bound separation distance* of a configuration curve Γ to be

$$\lceil \Gamma \rceil = \max_{\boldsymbol{x} \in \Gamma} [\boldsymbol{x}].$$

With the newly introduced notations, our squeezing cage's definition can be restated.

Definition 2: An object is squeezing caged by a finger placement $\boldsymbol{x}$ with a separation distance $[\boldsymbol{x}] = \delta$ when there is no configuration curve Γ from $\boldsymbol{x}$ to some escapable placement such that $\lceil \Gamma \rceil \leq \delta$.

If the placement $\boldsymbol{x}$ with $[\boldsymbol{x}] = \delta$ forms a squeezing cage, it then follows from Definition 2 that any placement $\boldsymbol{y}$ connected with $\boldsymbol{x}$ by any configuration curve Γ such that $\lceil \Gamma \rceil \leq \delta$ will also form a squeezing cage (because if $\boldsymbol{y}$ was not a squeezing cage, there would exist a configuration curve Γ' with $\lceil \Gamma \rceil \leq [\boldsymbol{y}]$ from $\boldsymbol{y}$ to an escapable placement and combining Γ with Γ' would lead $\boldsymbol{x}$ to an escapable placement as well). What this means is that if we start out from the squeezing cage placement $\boldsymbol{x}$ and move the fingers by keeping their separation distance not to exceed $[\boldsymbol{x}] = \delta$, it is guaranteed that any finger placement reachable by the motion must also form a squeezing cage (Fig. 1)

At a squeezing cage placement $\boldsymbol{x}$, what will happen if we gradually move the fingers away from each other? Of course, at some point the finger placement will lose ability to cage the object. Let us call the separation between the fingers when this event occurs as the *critical distance* of $\boldsymbol{x}$, denoted $\delta^*(\boldsymbol{x})$. We can define the notion of critical distance in our terms as follows.

Definition 3: The critical distance of a finger placement $\boldsymbol{x}$, denoted $\delta^*(\boldsymbol{x})$, is the least upper bound separation distance among all configuration curves from $\boldsymbol{x}$ to an escapable placement. That is,

$$\delta^*(\boldsymbol{x}) = \min_{\Gamma \in \mathcal{C}(\boldsymbol{x})} \lceil \Gamma \rceil,$$

where $\mathcal{C}(\boldsymbol{x})$ denotes the set of all configuration curves from $\boldsymbol{x}$ to any escapable placements.

If a placement $\boldsymbol{x}$ forms a squeezing cage, it then follows from Definition 2 and Definition 3 that any placement $\boldsymbol{y}$ connected with $\boldsymbol{x}$ by any configuration curve Γ such that $\lceil \Gamma \rceil < \delta^*(\boldsymbol{x})$ will also form a squeezing cage. we also automatically have $\delta^*(\boldsymbol{y}) = \delta^*(\boldsymbol{x})$. This allows us to define the notion of squeezing cage set.

Definition 4: A connected set of finger placements with the same critical distance forms a squeezing cage set.

Squeezing cage sets form disjoint open connected subsets of the configuration space $\mathbb{C}$. Subtracting all squeezing cage sets from $\mathbb{C}$ will result in a single connected component containing all escapable placements (and all placements that

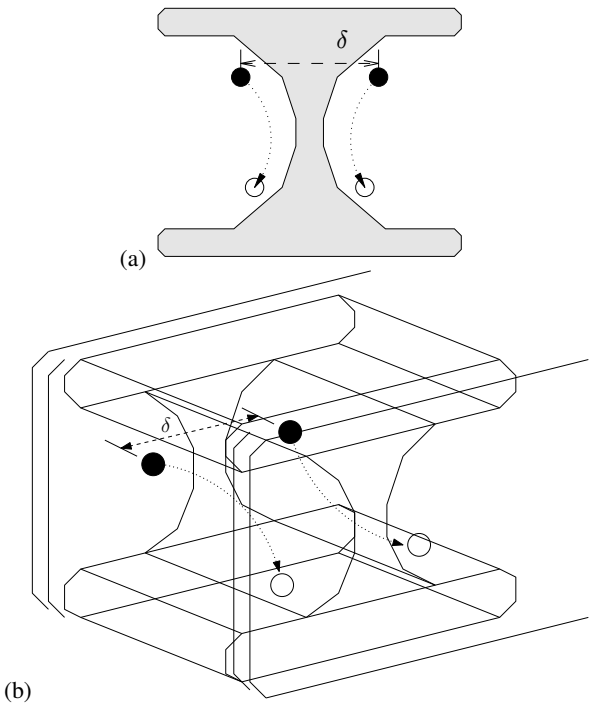

Fig. 1. From a squeezing cage placement $\boldsymbol{x}$ shown as a pair of black dots, the object is squeezing caged (even if the fingers move from $\boldsymbol{x}$ to a pair of white dots) as long as their separation distance does not exceed $[\boldsymbol{x}] = \delta$. Squeezing caging are generally formed at two opposite concave sections of the object (a) a squeezing cage of a polygon (b) a squeezing cage of a polyhedron (the I-shaped object is sandwiched with two solid plates creating two concave sections).

can be brought to an escapable placement without increasing the separation between the fingers). In words, a squeezing cage set is essentially the set of all squeezing cages for which each cage can be transformed to another cage in the set by executing a finger motion that keeps the separation distance between the fingers lower than the critical distance associated with the set during the entire motion.

Given an input object, our goal is to characterize all different squeezing cage sets along with their critical distances. In the following section, let us first explain how a critical distance of a finger placement is computed.

III. CRITICAL DISTANCE COMPUTATION

Let us consider two finger placements $\boldsymbol{u}$ and $\boldsymbol{v}$ shown in Fig. 2(a) and Fig. 2(b) respectively. It is obvious that when the fingers are at the placement $\boldsymbol{v}$, the object can escape. Since the fingers at $\boldsymbol{v}$ can be brought to an escapable placement without increasing the separation distance, we can therefore derive from Definition 3 that the critical distance of $\boldsymbol{v}$ must be equal to $[\boldsymbol{v}]$. On the contrary, when the fingers are at placement $\boldsymbol{u}$, the object is in a squeezing cage. This means that for the object to be able to escape, the separation distance between the fingers has to increase. Again, following Definition 3, we can conclude that the critical distance of $\boldsymbol{u}$ must be greater than $[\boldsymbol{u}]$.

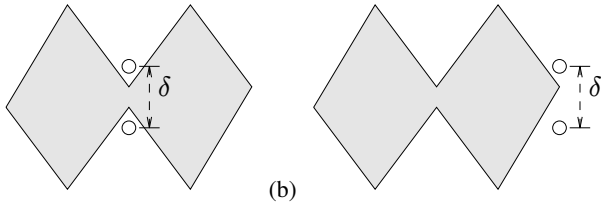

Fig. 2. (a) the object is squeezing caged by a finger placement $\boldsymbol{u}$. (b) the object is not squeezing caged when the fingers are placed at $\boldsymbol{v}$.

As mentioned above, whether a given placement can form a squeezing cage depends on its critical distance. however, computing a critical distance for an arbitrary finger placement $\boldsymbol{x}$ need some workaround since it is not feasible to find the least upper bound separation distance by examining all configuration curves from $\boldsymbol{x}$ to an escapable placement. Fortunately, the least upper bound separation distance can be analytically determined for some classes of configuration curves.

Proposition 1: Let $\boldsymbol{W}_a$ and $\boldsymbol{W}_b$ be convex subsets of the workspace $\mathbb{W}$. For any $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{W}_a \times \boldsymbol{W}_b$, there exists a configuration curve Γ from $\boldsymbol{x}$ to $\boldsymbol{y}$ such that $\lceil \Gamma \rceil \leq \max\{[\boldsymbol{x}], [\boldsymbol{y}]\}$.

Proof: Let us consider a configuration curve Γ defined by the linear interpolation between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e., $\Gamma = \{\boldsymbol{l}(t) = (1-t)\boldsymbol{x} + t\boldsymbol{y} \mid t \in [0,1]\}$. Since both $\boldsymbol{W}_a$ and $\boldsymbol{W}_b$ are convex, $\boldsymbol{W}_a \times \boldsymbol{W}_b$ is also convex set and contains Γ, a straight line between $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\boldsymbol{W}_a \times \boldsymbol{W}_b$. Since $[\boldsymbol{l}(t)]^2$ is a non-negative quadratic polynomial in t, we obtain that $\lceil \Gamma \rceil = \max\{[\boldsymbol{x}], [\boldsymbol{y}]\}$. ■

Since $\max\{[\boldsymbol{x}], [\boldsymbol{y}]\}$ is obviously the least possible upper bound separation distance for any configuration curves from $\boldsymbol{x}$ to $\boldsymbol{y}$, the following proposition can be stated.

Proposition 2: For any $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{W}_a \times \boldsymbol{W}_b$, there is no configuration curve from $\boldsymbol{x}$ to $\boldsymbol{y}$ with lower upper bound separation distance than that of Γ defined in the proof of Proposition 1.

Propositions 1 and 2 already enable us to determine the least upper bound separation distance for configuration curves having both ends in the same $\boldsymbol{W}_a \times \boldsymbol{W}_b$. The rest of this section explains how this idea can be extended to the case of arbitrary configuration curves.

Let us suppose that $\mathbb{C}$ is divided into several subsets each of which is the cartesian product of two convex subsets of $\mathbb{W}$. Let us refer to such subset of $\mathbb{C}$ as a node. We have already shown how to find the least upper bound separation distance for configuration curves with both ends in the same node. Let us now consider configuration curves traversing only in two adjacent nodes (a pair of nodes with non-empty intersection) $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ by starting at $\boldsymbol{x}_0 \in \boldsymbol{\nu}_0$ and ending at a placement in $\boldsymbol{\nu}_0 \cap \boldsymbol{\nu}_1 \neq \varnothing$. Using Proposition 1, the least upper bound separation distance of such curves can be written as $\max\{[\boldsymbol{x}_0], \min\{[\boldsymbol{x}] | \boldsymbol{x} \in \boldsymbol{\nu}_0 \cap \boldsymbol{\nu}_1\}\}$. With this reasoning, computing the least separation distance of curves

traversing through a longer sequence of nodes can be done in the same manner as the fingers need to hop from one node to the next, passing through the intersection of every two consecutive nodes in the sequence.

Proposition 3: Among the curves that traverse through a sequence of nodes: $\boldsymbol{\nu}_0, \boldsymbol{\nu}_1, ..., \boldsymbol{\nu}_m$, the least upper bound separation is $\max\{\delta_1, \delta_2, ..., \delta_m\}$ where $\delta_i = \min\{[\boldsymbol{x}] \mid \boldsymbol{x} \in \boldsymbol{\nu}_{i-1} \cap \boldsymbol{\nu}_i\}$

With the above proposition, we can now compute the critical distance for an arbitrary placement $\boldsymbol{x}$ in some node $\boldsymbol{\nu}$. Since we can have a finite partition of $\mathbb{C}$, this is done by enumerating and examining all sequences of nodes from $\boldsymbol{\nu}$ to a node containing an escapable placement. This idea will be described in detail in the algorithm presented in the next section.

IV. REPORTING ALL SQUEEZING CAGE SETS

In this section, we shall establish the algorithm for reporting all squeezing cage sets from the fundamental methodology developed in the previous section.

Let us partition the workspace $\mathbb{W}$ into closed convex subsets $\{\boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_m\}$ such that any two convex subsets near each other, say $\boldsymbol{W}_i$ and $\boldsymbol{W}_j$ must have a non-empty intersection ($\boldsymbol{W}_i \cap \boldsymbol{W}_j \neq \varnothing$). Such intersection is called a partition between $\boldsymbol{W}_i$ and $\boldsymbol{W}_j$. An example of a workspace partitioning satisfying the above properties is shown in Fig. 3 – the convex subsets from the partitioning are the blank areas divided by partitions shown in dashed lines. It follows from this approach of partitioning that these convex subsets can be easily grouped into nodes covering the configuration space $\mathbb{C}$. Such nodes are $\boldsymbol{\nu}_{11}, \boldsymbol{\nu}_{12}, ..., \boldsymbol{\nu}_{mm}$ where $\boldsymbol{\nu}_{ij} = \boldsymbol{\nu}_{ji} = \boldsymbol{W}_i \times \boldsymbol{W}_j$. A pair of nodes $\boldsymbol{\nu}_{ij}$, $\boldsymbol{\nu}_{kl}$ are said to be *adjacent* to each other when their intersection is not empty, $\boldsymbol{\nu}_{ij} \cap \boldsymbol{\nu}_{kl} \neq \varnothing$. Note that $\boldsymbol{\nu}_{ij} \cap \boldsymbol{\nu}_{kl} \neq \varnothing$, if and only if, $\boldsymbol{W}_i \cap \boldsymbol{W}_k \neq \varnothing$, and $\boldsymbol{W}_j \cap \boldsymbol{W}_l \neq \varnothing$. In further analysis, let us refer to this structure as an undirected graph i.e. nodes of this graph are the previously defined nodes and any two nodes are linked with an *edge* when they are adjacent to each other. Each edge, say $E = \{\boldsymbol{\nu}_{ij}, \boldsymbol{\nu}_{kl}\}$, has a *distance* δ_E which is equal to the the least upper bound separation distance among the curves that traverse between $\boldsymbol{\nu}_{ij}$ and $\boldsymbol{\nu}_{kl}$. This *distance* notation is also applied to all paths in the graph. Distance of a path is defined as the *maximum* of δ_E for every E being an edge in such path. It can also be interpreted as the least upper bound separation distance among curves that traverse through a sequence of nodes defined by the path. We address the path that has the least distance and routes from a node containing an escapable placement (e.g. $\boldsymbol{\nu}_{kk}$ for some k) to a node $\boldsymbol{\nu}$ as *critical path* to $\boldsymbol{\nu}$. In this light, the problem of finding a critical distance of a placement $\boldsymbol{x}$ inside a node $\boldsymbol{\nu}$ is transformed into the problem of finding a critical path to $\boldsymbol{\nu}$. It follows from Proposition 1 that, once the critical path is known, the critical distance of $\boldsymbol{x}$ is either $[\boldsymbol{x}]$ or the critical path's distance depending on which one is greater. Though the concept of distance in this context is slightly different from that in the Dijkstra's shortest path problem, we shall show that it is possible to solve the problem of finding a critical path to a node, say the *critical path problem*, with the shortest path algorithm. To begin with, let us compare the how the two problems determine their optimal distances.

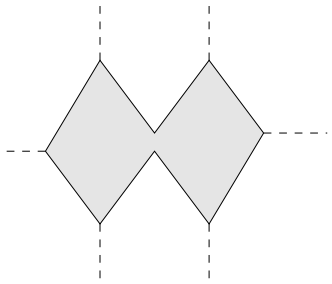

Fig. 3. An example of a valid workspace partitioning (see text).

In case of the shortest path problem, the shortest distance from an initial node to a node $\boldsymbol{\nu}$, denotes $d^*_{\boldsymbol{\nu}}$, is the minimum distance of the sum of shortest distance to some adjacent node $\boldsymbol{\nu}'$ ($d^*_{\boldsymbol{\nu}'}$) and the distance between $\boldsymbol{\nu}$ and $\boldsymbol{\nu}'$ ($d_{\{M,N\}}$). Let $\mathbb{E}$ be the set of all edges in the graph, we have:

$$d^*_{\boldsymbol{\nu}} = \min\left\{d^*_{\boldsymbol{\nu}'} + d_{\{\boldsymbol{\nu},\boldsymbol{\nu}'\}} \mid \{\boldsymbol{\nu},\boldsymbol{\nu}'\} \in \mathbb{E}\right\} \tag{1}$$

For the critical path problem, the distance of a critical path to a node $\boldsymbol{\nu}$ ($\delta^*_{\boldsymbol{\nu}}$) is the minimum distance of the maximum between the distance of a critical path to some adjacent node $\boldsymbol{\nu}'$ ($\delta^*_{\boldsymbol{\nu}'}$) and the distance between $\boldsymbol{\nu}$ and $\boldsymbol{\nu}'$ ($\delta_{\{\boldsymbol{\nu},\boldsymbol{\nu}'\}}$). That is:

$$\delta^*_{\boldsymbol{\nu}} = \min\left\{\max\left\{\delta^*_{\boldsymbol{\nu}'}, \delta_{\{\boldsymbol{\nu},\boldsymbol{\nu}'\}}\right\} \mid \{\boldsymbol{\nu},\boldsymbol{\nu}'\} \in \mathbb{E}\right\} \tag{2}$$

To reduce the critical path problem to the shortest path problem, we equate (1) with (2):

$$\begin{aligned}\min\left\{\delta^*_{\boldsymbol{\nu}'} + d_{\{\boldsymbol{\nu},\boldsymbol{\nu}'\}} \mid \{\boldsymbol{\nu},\boldsymbol{\nu}'\} \in \mathbb{E}\right\} = \\ \min\left\{\max\left\{\delta^*_{\boldsymbol{\nu}'}, \delta_{\{\boldsymbol{\nu},\boldsymbol{\nu}'\}}\right\} \mid \{\boldsymbol{\nu},\boldsymbol{\nu}'\} \in \mathbb{E}\right\}\end{aligned} \tag{3}$$

Without loss of generality, we specify that the best path to $\boldsymbol{\nu}$ need to visit a node $\boldsymbol{\nu}^*$ adjacent to $\boldsymbol{\nu}$. Therefore:

$$d_{\{\boldsymbol{\nu},\boldsymbol{\nu}^*\}} = \max\left\{\delta^*_{\boldsymbol{\nu}^*}, \delta_{\{\boldsymbol{\nu},\boldsymbol{\nu}^*\}}\right\} - \delta^*_{\boldsymbol{\nu}^*} \tag{4}$$

Equation (4) indicates that the distance between any two adjacent nodes ($d_{\{\boldsymbol{\nu},\boldsymbol{\nu}^*\}}$) is non-negative. This allows us to apply the shortest path algorithm to solve this problem.

After the execution, we obtain distances of critical paths to all nodes in the graph which can be used in reporting all squeezing cage sets or performing queries for critical distance of any finger placement. A mean to reporting all squeezing cage sets is to list all groups of nodes with the same critical distance. Note that each group is a connected component of the graph and there exist a path to any pair of nodes in the group with distance less the group's critical distance. For critical distance query, the critical distance of a finger placement $\boldsymbol{x}$ in a node $\boldsymbol{\nu}$ is simply $\max\{[\boldsymbol{x}], \delta^*_{\boldsymbol{\nu}}\}$.

In the following section, we shall proceed to running time analysis, and fill in implementation details of the algorithm.

V. RUNNING TIME ANALYSIS

The running time of the algorithm for reporting all squeezing cage sets involves the time spent in the following sequential tasks:

A. partitioning $\mathbb{W}$ into convex subsets,
B. preprocessing and
C. computing critical distance of all nodes

A. Convex Partitioning

Let n, m be the number of vertices of the input object (either polygons or polyhedra) and the number of convex subsets (resp.). For polygons, several $O(n \log n)$ algorithms are readily available for an approximated minimal convex decomposition [10]. The number of convex subsets is in the order of input vertex, $m = O(n)$, and is guaranteed to be less than four times of the minimum number of convex subsets. For polyhedra, partitioning takes $O(nN^3)$ where N is the number of edges with a reflex angle [11]. The algorithm produces $m = O(N^2)$ convex subsets. Note that chosen partitioning algorithms for this analysis decompose the workspace into disjoint convex subsets. In order to satisfy the partitioning requirement stated in the beginning of the previous section, the partitioned convex subsets need to be grown (infinitesimally) into closed subsets. Let p be the number of partitions, p always has a linear relationship with m, $p = O(m)$, according to Euler Characteristic [12].

B. Preprocessing

The preprocessing task pre-computes δ_E for an edge E of the graph in the form $\{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_k\}$. δ_E depends on the shape of the convex subset $\boldsymbol{W}_k$ and the partition $\boldsymbol{W}_i \cap \boldsymbol{W}_j$, in that:

$$\begin{aligned}\delta_E &= \min\{[\boldsymbol{x}] \mid \boldsymbol{x} \in (\boldsymbol{W}_i \times \boldsymbol{W}_k) \cap (\boldsymbol{W}_j \times \boldsymbol{W}_k)\} \\ &= \min\{\|\boldsymbol{a} - \boldsymbol{b}\| \mid \boldsymbol{a} \in \boldsymbol{W}_i \cap \boldsymbol{W}_j \wedge \boldsymbol{b} \in \boldsymbol{W}_k\}\end{aligned}$$

It follows from the chosen convex partitioning approach that a partition $\boldsymbol{W}_i \cap \boldsymbol{W}_j$ is merely a face (or a line segment, in case of polygonal object). δ_E is therefore equal to the *distance*[1] between some pair of a vertex and a face (or a line segment) such that:

(a) the vertex is in $\boldsymbol{W}_i \cap \boldsymbol{W}_j$ and the face (or the line segment) is in $\boldsymbol{W}_k$ (Fig. 4(a)), or
(b) the vertex is in $\boldsymbol{W}_k$ and the face (or the line segment) is in $\boldsymbol{W}_i \cap \boldsymbol{W}_j$ (Fig. 4(b)).

A chosen strategy to compute δ_E for all E is as follows:

1) Initially, δ_E, for all edge E of the graph, is set to a sufficiently large value.
2) Enumerate a vertex $\boldsymbol{v}$ and a face $\boldsymbol{F}$ such that $\boldsymbol{v}$ and $\boldsymbol{F}$ is a vertex of $\boldsymbol{W}_a$ and a face of $\boldsymbol{W}_b$ (resp.), for some integer a, b which take value from 1 to m.
3) Compute the distance δ between $\boldsymbol{v}$ and $\boldsymbol{F}$.

[1]Let $\boldsymbol{v}$ and $\boldsymbol{P}$ be a vertex and a set of points, the *least distance* from a vertex to a set of points is $\min\{|\boldsymbol{v} - \boldsymbol{u}| \mid \boldsymbol{u} \in \boldsymbol{P}\}$.

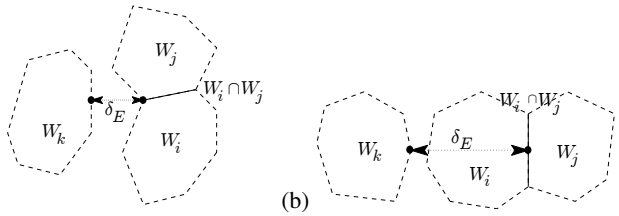

Fig. 4. δ_E of $E : \{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_k\}$. $\boldsymbol{W}_i$, $\boldsymbol{W}_j$, and $\boldsymbol{W}_k$ are shown as dashed lines surrounding the sets. $\boldsymbol{W}_i \cap \boldsymbol{W}_j$ is shown as a solid line.

4) For every partition $\boldsymbol{W}_i \cap \boldsymbol{W}_j$ and a convex subset $\boldsymbol{W}_k$ such that $\boldsymbol{v} \in (\boldsymbol{W}_i \cap \boldsymbol{W}_j) \wedge \boldsymbol{F} \in \boldsymbol{W}_k$ or $\boldsymbol{F} \in (\boldsymbol{W}_i \cap \boldsymbol{W}_j) \wedge \boldsymbol{v} \in \boldsymbol{W}_k$, the distance of an edge $E = \{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_k\}$: δ_E is updated to δ if it is greater than δ.

In step (4), $\boldsymbol{F}$ is contained within at most two convex subsets; while the number of convex subsets containing $\boldsymbol{v}$ ranges from one to m convex subsets. This results in at most $2m$ updates for a pair of a vertex and a face. Nevertheless, the overall preprocessing time is governed by $O(n^2 + nm + m^2)$. This fact, again, can be proven based on Euler Characteristic, see Appendix A.

C. Computing Critical Distances

Running time for the shortest path algorithm is $O(e \log v)$ [13] where e and v is the number of edges and nodes (resp.). We have $v = m^2$ as the number of nodes is m^2. Since e depends on the number of partitions p, a partition represents a possibility to move from one convex subset to another, say $\boldsymbol{W}_i$ to $\boldsymbol{W}_j$. For each of such possibility, an arbitrarily chosen convex subset, say $\boldsymbol{W}_k$, can be paired with $\boldsymbol{W}_i$ and $\boldsymbol{W}_j$ to form an edge (e.g. $\{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_k\}$). Though it is possible for an edge to involve four different convex subsets of $\mathbb{W}$ (i.e. $\{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_l\}$), these edges may be dropped without effecting the critical distances since they always can always be replaced by a two successive hops on existing edges: either $\{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_k\} \rightarrow \{\boldsymbol{W}_j \times \boldsymbol{W}_k, \boldsymbol{W}_j \times \boldsymbol{W}_l\}$ or $\{\boldsymbol{W}_i \times \boldsymbol{W}_k, \boldsymbol{W}_i \times \boldsymbol{W}_l\} \rightarrow \{\boldsymbol{W}_i \times \boldsymbol{W}_l, \boldsymbol{W}_j \times \boldsymbol{W}_l\}$, depending on which path has lower upper bound distance. It follows that $e = O(pm) = O(m^2)$ and, consequently, the running time is $O(m^2 \log m)$.

D. Summary

The time required for constructing the graph and computing $\delta^*_{\boldsymbol{\nu}}$ for all possible $\boldsymbol{\nu}$ is:

$$t(n) + O(n^2 + nm + m^2) + O(m^2 \log m)$$

where $t(n)$ denotes the time spent in partitioning the workspace $\mathbb{W}$ into convex subsets. With rough estimation, this becomes $O(n^2 \log n)$ for polygons, and $O(n^4 \log n)$ for polyhedra.

After the all nodes are assigned with their critical distance, the pre-computation prior to point-location query requires $O(n \log n)$ for both convex polygons and polyhedra [14].

TABLE I
RUNNING TIME OF THE ALGORITHMS

Input Object	Partitioning	Preprocessing	Critical Distance	All Squeezing Cage Sets	Query Preprocessing	Query
Polygons	$O(n \log n)$	$O(n^2 + nm + m^2)$	$O(m^2 \log m)$	$O(n^2 + nm + m^2 \log m)$	$O(n \log n)$	$O(\log n)$
Polyhedra	$O(nN^3)$	$O(n^2 + nm + m^2)$	$O(m^2 \log m)$	$O(nN^3 + n^2 + nm + m^2 \log m)$	$O(n \log n)$	$O(\log^2 n)$

Once the preparation is complete, a node containing a finger placement can be queried within $O(\log n)$ and $O(\log^2 n)$ in case of polygons and polyhedra, respectively.

The results of running time analysis are summarized in Table I.

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented a more efficient algorithm for reporting all squeezing cage sets for polygons and polyhedra based on convex decomposition of the workspace. Still, the algorithm is limited to point fingers. For round fingers, this algorithm would give an approximation of caging sets (small caging sets may not be reported) as the object need to be expanded by the radius of the finger prior to polygons or polyhedra approximation. Whether a finger placement can perform squeezing caging can also be queried within logarithmic time after the all-caging-set report is completed. This approach is much more efficient than our previously proposed ray-shooting based approach, [8]. However, the new approach may not handle the case of stretching caging conveniently because more than one stretching cage set may reside in a node – a more sophisticated graph formulation is required. It is remained open for further researches on more efficient way to report all stretching caging sets for polygons and polyhedra and for systems with more fingers.

APPENDIX

A. Preprocessing Running Time

From Euler Characteristic, the relationship between the number of convex subsets w_i surrounding a vertex $\boldsymbol{v}_i$ can be formulated as:

$$w_i = f_i - \epsilon_i + 2 \tag{5}$$

where f_i and ϵ_i are the number of faces and line segments (of the object or a partition) (resp.) containing $\boldsymbol{v}_i$ (i.e. ϵ_i is equal to the degree of $\boldsymbol{v}_i$). For each vertex $\boldsymbol{v}_i$, $\boldsymbol{v}_i$ have to be paired with all F faces where F is equal to the total number of faces of the object including the partitions. The distance between such pair is computed once and the distance updated is performed for $2Fw_i$ edges. It follows that the total number of distance updates is:

$$\sum_{i=1}^{n} 2Fw_i = 2F \cdot (\sum_{i=1}^{n} f_i + \sum_{i=1}^{n} \epsilon_i + 2n) \tag{6}$$

We can further simplify this using the fact that:

1) a face is connected to three vertices, and
2) a line segment is connected to two vertices.

Let E is the total number of all line segments composing the object and the partitions, we have that $\sum_{i=1}^{n} \epsilon_i = 2E$, $\sum_{i=1}^{n} f_i = 3F$; therefore, (6) becomes:

$$2F \cdot (3F + 2E + 2n) = O(n^2 + nm + m^2) \tag{7}$$

since $F = O(n + m)$ and $E = O(n + m)$.

REFERENCES

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

[2] A. Sudsang, F. Rothganger, and J. Ponce, "An implemented planner for manipulating a polygonal object in the plane with three disc-shaped mobile robots," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, October 2001, pp. 1499–1506.

[3] S. J. Blind, C. C. McCullough, S. Akella, and J. Ponce, "Manipulating parts with an array of pins: A method and a machine," *The International Journal of Robotics Research*, vol. 20, no. 10, pp. 808–818, 2001.

[4] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Preecedings of IEEE International Conference on Robotics and Automation*, vol. 1, April 2000, pp. 348–353.

[5] 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*, vol. 2, April 1996, pp. 1459–1464.

[6] A. Sudsang, J. Ponce, and N. Srinivasa, "Algorithms for constructing immobilizing fixtures and graps of three dimensional objects," *Algorithmic Foundations of Robotics II*, pp. 363–380, 1997.

[7] A. Sudsang and T. Luewirawong, "Capturing a concave polygon with two disc-shaped fingers," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, September 2003, pp. 1121–1126.

[8] P. Pipattanasomporn and A. Sudsang, "Two-finger caging of concave polygon," in *Proceeding of IEEE International Conference on Robotics and Automation*, May 2006.

[9] M. Vahedi and A. F. van der Stappen, "Caging polygons with two and three fingers," in *Workshop on the Algorithmic Foundations of Robotics (WAFR) 2006*, July 2006.

[10] J. M. Keil, *Handbook of Computational Geometry*. Elsevier Science/North-Holland: North-Holland Publishing Co. Amsterdam, 2000, ch. 11. Polygon Decomposition, pp. 491–518.

[11] B. M. Chazelle, "Convex decompositions of polyhedra," in *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1981, pp. 70–79.

[12] J. L. Gross and J. Yellen, Eds., *Handbook of Graph Theory*, 2003, p. 614.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001, ch. Graph Algorithms, pp. 595–599.

[14] J. Snoeyink, *Handbook of Discrete and Computational Geometry*, 2nd ed. New York: Chapman & Hall/CRC, 2004, ch. 34. Point Location, pp. 767–785.