Submitted to the *The 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*

Inner-cover of Non-convex Shapes

Daniel Cohen-Or
Tel Aviv University
dcor@post.tau.ac.il

Shuly Lev-Yehudi
Tel Aviv University
shulyl@post.tau.ac.il

Adi Karol
Tel Aviv University
krol@post.tau.ac.il

Ayellet Tal
Technion - Israel Institute of Technology
ayellet@ee.technion.ac.il

ABSTRACT

Visibility methods are often based on the existence of large convex occluders. We present an algorithm that for a given simple non-convex polygon P finds an approximate inner-cover by large convex polygons. The algorithm is based on an initial partitioning of P into a set C of disjoint convex polygons which are an exact tessellation of P. The algorithm then builds a set of large convex polygons contained in P by constructing the convex hulls of subsets of C. We discuss different strategies for selecting the subsets and we claim that in most cases our algorithm produces an effective approximation of P.

KEYWORDS

Visibility, partition, decomposition, cover, convexity

1 INTRODUCTION

Visibility methods use occlusion culling algorithms to remove hidden portions of the scene before the actual rendering [7]. Many visibility methods are based on the existence of large occluders [8, 9, 13, 19]. Some methods either preselect or even synthesize the occluders to increase their performance [4, 11, 15, 16]. Effective occluders should be convex and as large as possible. Larger occluders have larger shadow volumes and thus occlude more [8, 9]. Moreover, when the occluder is convex, it is rather easy to test whether a given object resides in its shadow volume [8, 9]. It is also desirable to have only a handful of occluders since the efficiency of occlusion culling algorithms is directly dependent on the number of occluders. This calls for algorithms that approximate a given shape by a small number of large convex shapes.

For a conservative occlusion culling the occluders have to be approximated by inner (i.e., contained) shapes. This guarantees that the occlusion cast by the approximated shape never falsely occludes a visible shape. The fact that it might misclassify an occluded object is not harmful since these visibility culling algorithms generate a *potentially visible set* (PVS) rather than an exact set. The PVS is then fed to a hidden surface removal algorithm which resolves the exact visibility.

Andújar et al. [1] have presented an algorithm that for a given object O finds a set of inner convex objects. This algorithm generates only axis-aligned bounding boxes. A similar approach is taken by Leblanc and Poulin [17] to generate a large convex occluder contained inside a given polygonal mesh. Methods which apply occluder fusion also aim at large and convex occluders which make them more effective.

Driven by visibility problems, we present a novel algorithm for covering a given polygon by a few convex, large polygons. In other words, given an input polygon P, the algorithm generates a set of polygons that satisfies the following requirements: (i) the set is small, (ii) each polygon in the set is convex, (iii) each polygon in the set is as large as possible, and (iv) each polygon in the set is contained within P. Moreover, our algorithm can generate a *partial cover* for a pre-specified percentage of the polygon. We show in the sequel that this ability of the algorithm can dramatically reduce the number of polygons in the covering set.

Our algorithm starts by simplifying the non-convex polygon to facilitate the generation of a partial covering. Then, the polygon is decomposed into a number of small pieces, which form the building blocks of the inner-cover. By combining subsets of these building blocks a covering set is found. Finally, polygons in the covering set are enlarged to form an inner-cover consisting of large convex parts.

In Section 2 we start by formally defining the problem and some necessary terminology. We give an overview of the algorithm in Section 3 and describe the details of the algorithm in Sections 4 and 5. We show some results in Section 6 and conclude in Section 7.

2 PRELIMINARIES

Let P be a simple polygon. We seek to generate a set of possibly overlapping, convex polygons that satisfy a few requirements described below. Hereafter we give some formal definitions and elaborate on our goals.

Partition: A set $\{P_1, P_2 \cdots P_k\}$ of polygons is a partition of P if $\forall i = 1 \cdots k \;\; P_i \subset P, \;\; \cup_{i=1}^{k} P_i = P$ and $P_i \cap P_j = \emptyset \;\; \forall i \neq j$.

Cover: A set $\{P_1, P_2 \cdots P_k\}$ of polygons is a cover of P if $\forall i = 1 \cdots k \;\; P_i \subset P, \;\; \cup_{i=1}^{k} P_i = P$. Here, unlike in the case of partition, the polygons need not be disjoint.

Inner cover: Given $0 \leq p \leq 1$, a set $\{P_1, P_2 \cdots P_k\}$ of polygons is a p-inner cover of P if $\forall i = 1 \cdots k \;\; P_i \subset P$, $area(P) \geq area(\cup_{i=1}^{k} P_i) \geq p * area(P)$. In other words, not only the polygons need not be disjoint, but also their union may only partially cover P.

Coverage ratio: The ratio between the area of the union of polygons in the inner-cover of P and the area of P.

Convex partition/Cover/Inner-cover: A set $\{P_1, P_2 \cdots P_k\}$ of polygons which is a partition, cover or inner-cover, respectively, is called convex if $\forall i = 1 \cdots k, \;\; P_i$ is convex.

Reflex (concave) vertex: A vertex v of P is called reflex if its internal angle is larger than Π (see Figure 1(a)).

Convex vertex: A vertex v of P is called convex if its internal angle is less than or equal to Π.

Non-convex polygon: A polygon with at least one reflex vertex.

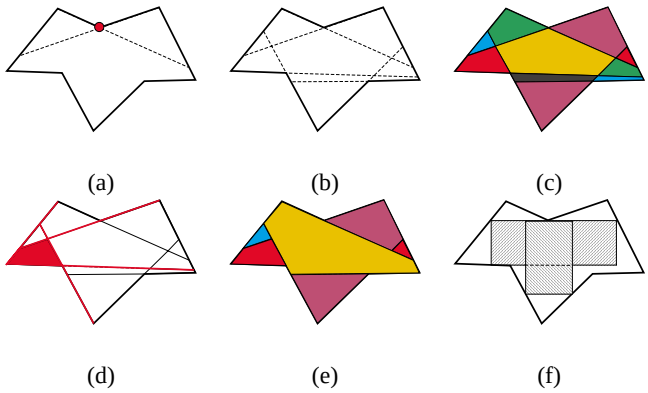


Figure 1: (a) The non-convex polygon is partitioned by two line segments around a reflex vertex. (b) Partitioning around all reflex vertices. (c) The arrangement. (d) The face with the largest potential defined by the sum of lengths of its extended edges. (e) A convex inner-cover. (f) An axis-aligned partial cover.

There are various papers dealing with convex partitions of polygons [2, 12, 14]. Others handle covers; in particular, rectilinear covers [18]. We are not aware, however, of any previous work dealing with inner-covers using general convex polygons, which is our goal.

Given a polygon P we aim at generating a convex inner cover of P (i.e., each polygon in the set is convex and entirely contained within P), having a few additional requirements: (i) the inner-cover should consist of a small number of polygons, (ii) the inner-cover should cover as much area of P as possible and (iii) each polygon P_i in the cover should be as large as possible. In other words, we strive to maximize the coverage ratio, minimize the cardinality of the covering set, and maximize the area of each polygon in the cover.

There are two alternatives to make this problem well defined. One alternative is to let the user specify a lower bound on the coverage ratio (e.g., at least 90% of the polygon's area should be covered), and aim at minimizing the cardinality of the covering set. The other alternative is to impose a limit on the number of polygons in the inner-cover (e.g., no more than four polygons should be used), and aim at maximizing the coverage ratio. In both cases, an additional requirement is to maximize the size of each polygon in the set.

An optimal solution to this problem is hard. Just finding the largest inner convex polygon in P is quite involved and has a complexity of $O(n^9 logn)$, where n is the number of vertices of P [6]. Thus, even in the special case (of the second alternative) when the given limit is 1, the problem cannot be solved optimally in any reasonable amount of time. The general cover problem that we are pursuing is harder. In fact, it was proven in [10] that finding the minimum cover is an NP–complete problem.

Therefore, we present in this paper heuristics to solve the problem. We show that in practice our algorithm is effective and generates good inner-covers.

Finally, we do not direct our efforts at axis-aligned covers, but rather at a general convex inner-cover (as in Figure 1(e)). Figure 1(f) illustrates the reason. Trying to cover a given polygon with axis-aligned rectangles is quite limited in the general case.

3 OVERVIEW

Let P be a simple non-convex polygon. The method proposed in this paper is based on a few observations:

- The line that coincides with an edge of polygon P at a reflex vertex, cuts the corresponding reflex angle into two convex angles. This is so because supposing the reflex angle is α, the above cutting line divides α into two angles: Π and $\alpha - \Pi$. As a consequence, using edge extensions as cutting lines reduces the total number of reflex angles.

- The largest convex polygon contained in a given polygon P must be adjacent to at least one reflex vertex, if any exist [6].

- Typically, the edges of a large convex polygon contained within a given polygon P, coincide with the edges of P. We will later show that this is not always the case.

Thus, a sensible strategy would be to construct inner large convex polygons by combining smaller polygons whose edges coincide with the edges of P. In particular, we first generate a partition of P, using extensions of the edges adjacent to the reflex vertices as cutting lines. By the above observations, these are reasonable cutting lines (see Figure 1(a)).

By applying similar partitions around all reflex vertices of P, a partition of P is generated (see Figure 1(b)). We can view this decomposition of P as an *arrangement* [3] induced by the above cutting lines. An arrangement is a partition of P into *faces* – the convex regions, *segments* between line crossings, and *vertices* where lines meet (see Figure 1(c)).

The elements of the arrangement are the building blocks we use for constructing a cover of P consisting of large convex polygons. The idea is to build convex hulls of subsets of the partition. If these convex hulls are contained within P, they become candidates for participating in the inner-cover of P.

A major question is how to select good candidate subsets. Obviously an exhaustive search would be exponential. We developed various heuristics to quickly select effective subsets. In the next section we describe the various stages of the algorithm, while in Section 5 we focus on the heuristics for selecting subsets.

4 INNER-COVER GENERATION

This section describes our algorithm for generating an inner-cover for a given polygon P. The algorithm consists of five stages: simplification, generating the cutting segments, constructing an arrangement, producing the cover, and improving the cover.

Simplification

The input polygon is first simplified in order to decrease its number of vertices, and consequently, its number of reflex vertices. This results in a major speed-up in the later stages. The simplification is *conservative* in the sense that the resulting polygon is contained within the original polygon.

Each vertex is associated with a triangle formed with its two incident neighbors. The simplification is done by iteratively removing vertices associated with triangles having small areas. A vertex is removed if its associated triangle is contained within P and a prespecified threshold has not been reached. As a result of the simplification, many reflex vertices are eliminated. When the threshold is set to 100%, only collinear vertices are removed, and the polygon's outline is not affected. Typically, the simplification removes only the details of the polygon.

Generating the cutting line segments

This stage transforms the polygon into a collection of segments. The line segments are defined as the inner extensions in both directions of the edges of P, which have an endpoint at a reflex vertex of P. Each of these cutting segments induces a partition that turns the reflex vertex into convex vertices in the sub-polygons generated by this partition, as observed in Section 3.

By applying all the partitions around each reflex vertex of P we generate the convex partition of P. Figure 2(a) shows the maximal segments. The actual extraction of the convex partition is done in the next stage.

Constructing an arrangement

This stage takes as input the segments constructed during the previous stage, and generates the convex partition they induce. This is done by constructing an *arrangement* of the above line segments.

Given a finite set L of lines in the plane, the *line arrangement* $A(L)$ is the decomposition of the plane into connected open cells of dimensions $0, 1, 2$ induced by L [3]. In particular, the set L induces a subdivision of the plane that consists of vertices, edges, and faces ($0, 1$ and 2-dimensional cells, respectively). The complexity of an arrangement is the total number of vertices, edges, and faces of the arrangement - $\theta(n^2)$ in the worst case - where n is the number of lines.

The faces of the arrangement are always convex, and are the building blocks for the next stages. Before being fed as input to the next stage, the faces are sorted. Various sorting criteria are possible and will be discussed in Section 5.

Producing the cover

The *faces* of the arrangement that were generated in the previous stages, are used for constructing an *inner-cover* of P. The idea is to build convex hulls around selected subsets of faces which are within P. These convex hulls are the candidates for the *inner-cover*.

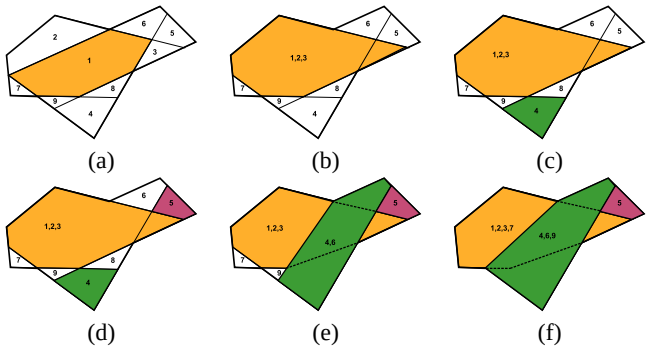


Figure 2: Illustration of the algorithm: (a) Initially, the original polygon is decomposed into nine faces sorted by area. Then Face 1 is added to the cover. (b) Faces 2 and 3 are combined with polygon 1. (c)-(d) Faces 4 and 5 are added as new polygons. (e) Face 6 is combined with polygon 4. (f) The final cover, after faces 7-9 are processed. Face 7 is combined with polygon 1-2-3, face 8 is already covered so it is skipped, and face 9 is combined with polygon 4-6.

P is a polygon with n vertices and n edges. The number of faces in the arrangement is quadratic in the size of P because it is created by at most 2n lines (and typically has over a dozen of pieces). We are looking for effective subsets and obviously an exhaustive search cannot be used. We developed different heuristics to quickly select such subsets, as will be discussed in Section 5.

The inner-cover is built incrementally in a greedy fashion. Starting with an empty cover, the algorithm examines each face in ascending order (see Section 5.2), and tries to find an existing polygon in the cover that this face can be combined with. This is done by forming the convex hull of the face and the polygon, and checking whether this convex hull is entirely contained in P. If so, the cover is updated. If no such polygon is found, a new polygon containing only this face is added to the cover. This is repeated until either the desired coverage ratio is reached or until the number of polygons in the cover exceeds a given bound.

Figure 2 illustrates the above procedure for a coverage ratio of 100%. Observe that the order of the faces influences the resulting cover. The choice depends on the strategy used. (We describe possible strategies below.)

Enlargement

This stage is required in order to get large convex polygons in the inner-cover. It tries to enlarge each polygon in the cover so that it will remains entirely contained in the original polygon.

Enlargement is done using a technique similar to that of the previous stage. For each polygon in the cover, we maintain a priority queue of faces that are candidates for being combined with the polygon. First, the queue contains the faces adjacent to the polygon, sorted by their areas in descending order. We try to combine each candidate face with the polygon. If the combined polygon is contained within the original polygon, we update the cover, and add all the faces adjacent to the combined face to the priority queue.

Before the enlargement, the cover consists of overlapping parts consisting of disjoint sets of faces, while after they are not disjoint. It should be noted that we had to build the cover first and then enlarge the polygons within it. It might seem that it could be done in one stage, where each face, in turn, is combined with every polygon in the cover that can be combined with it (and not just the first

one). Though in this case the algorithm would yield larger polygons earlier, these polygons would lose their flexibility to combine with more faces in the sequence. Furthermore, the algorithm would have tried to enlarge a polygon by combining it only with faces that considered after the polygon is created, and not with all the possible faces. Given that larger faces are examined first, this would be a major drawback. Therefore, one stage would not result in a cover whose polygons are as large as possible.

5 HEURISTICS STRATEGIES

Since a face can often be combined with more than one polygon (as shown in Figure 2(f)), the choice depends on the strategy used. Possible strategies are described in Section 5.1. In addition, the order in which the faces are visited also affects the resulting cover. Possible orders are discussed in Section 5.2.

5.1 Cover Strategies

We have experimented with different strategies for choosing a polygon. These strategies include:

- The largest polygon.
- The smallest polygon.
- The largest adjacent polygon to the face, and as a secondary preference, the largest.
- As above, but as a secondary preference, the smallest.
- Random selection.

Choosing the largest polygon may be advantageous since the combined polygon is likely to be large and consequently a larger portion of the original polygon will be covered. Furthermore, the final cover will consist of large polygons, satisfying our requirement. On the other hand, smaller polygons may be good candidates, since it is more likely that they can be combined with new faces. This stems from the fact that the resulting convex hulls are smaller, and thus more likely to be contained in the original polygon.

The motivation for preferring polygons that are adjacent to the face is similar to that of choosing smaller polygons, namely the size of the convex hulls increases slowly.

Choosing the polygons randomly did not demonstrate good results. This strategy was, however, helpful for comparisons.

5.2 Face order

There are several possibilities for ordering the faces to be examined by the algorithm. It should be clear by now that it is important that the faces are ordered by dominance. In most occlusion culling applications a coverage ratio of 100% is not necessary. Moreover, our first requirement demands that the number of polygons in the cover is relatively small. Thus, there is no need to examine all the faces but only some of them, until the desired cover is achieved. Therefore, the following two properties are desirable:

- The coverage ratio increases rapidly with each iteration, thereby reaching the threshold quickly.
- The polygons in the cover can be combined (in future iterations) with more faces, thereby minimizing the number of polygons in the final cover.

	Cover Strategies				
Face Orders	largest	smallest	random	adjacent & large	adjacent & small
size	3 (66%)	2 (60%)	3 (66%)	3 (66%)	2 (60%)
potential	3 (57%)	3 (57%)	3 (57%)	3 (57%)	3 (57%)
random	3 (57%)	3 (57%)	3 (57%)	3 (57%)	3 (57%)

Table 1: Various covers for the polygon in Figure 3(a). Each entry contains the number of polygons in the cover and the average coverage ratio of a polygon in the cover after the enlargement.

These requirements motivated us to experiment with several approaches for sorting the faces. Sorting by area satisfies the first requirement since larger faces are examined first. Sorting by *potential* tries to meet the second requirement. The *potential* of a face is defined as a function of the lengths of the cutting segments, incident to the face such as the maximal length, the average length and the sum of lengths. The potential reflects the face's possibility of being combined with other faces. In our experiments we used the sum of length as the potential function (See Figure 1(d)). We have tried to use several functions, and the experiments have shown that none of these functions outperforms a random order.

Therefore, our default strategy is to sort by area. As a consequence, the smallest faces which have the least impact on the coverage ratio, appear last and are seldom examined by the algorithm. This results in a major speed-up of the algorithm.

6 IMPLEMENTATION AND RESULTS

We implemented the inner-cover algorithm presented in this paper and developed an interactive application to test it. The user can set the coverage ratio and the limit on the number of polygons, as well as choosing the cover strategy and face order. Our implementation was developed using the CGAL library of geometric algorithm [5]. In particular we used the planar maps and arrangements package.

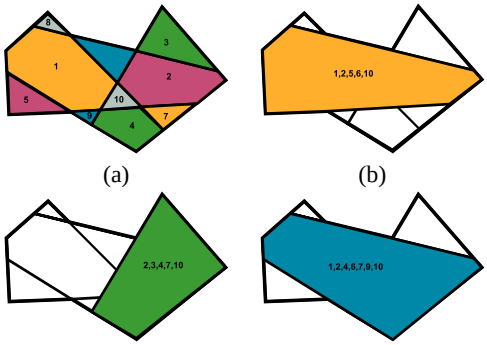
We carried out many experiments testing the efficiency of various strategies on a variety of polygons. Surprisingly, we found that all the cover strategies, with the exception of random selection, produced comparable results. Even when there were differences in the initial covers, after performing the enlargement stage, these differences diminished. This was apparent even when the random selection strategy was used.

Though the results usually do not differ, the various strategies have a minor implication on the running time of the algorithm. Giving priority to smaller polygons or to neighbors of the face generally accelerates the execution time.

To evaluate the quality of the cover, we counted the number of polygons in the cover and calculated the size of each polygon with respect to the size of the original polygon.

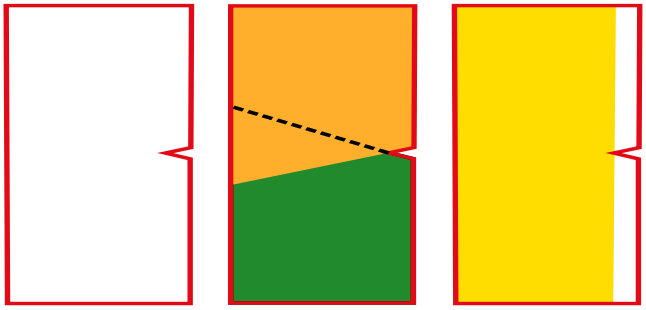
Table 1 illustrates the results obtained using different cover strategies and different face orders for the polygon in Figure 3(a). As can be seen, in most cases, the resulting cover consists of three polygons whose average coverage ratio is 59%. Two strategies give better results when the faces are ordered by size: preferring the smallest and preferring adjacent and then the smallest. These covers consist of two polygons whose average coverage ratio is 60%.

We generate randomly few hundreds of general polygons and we measure the distribution of the input polygons according to the number of polygons in the resultant inner cover. The input polygons



(a) (b) (c) (d)

Figure 3: Two inner-covers with coverage ratio of 95%. (a) The original polygon. (b)-(c) The two polygons that compose the inner-cover when the smallest first strategy is used. (b)-(d) The three polygons that compose the inner-cover when the largest first strategy is used. In both cases the faces are ordered by size.



(a) The polygon (b) Our cover (c) Optimal cover

Figure 4: Non-optimal cover. Our algorithm does not find the largest inner convex polygon. However, the largest polygon does not cover the entire input polygon.

had size (number of vertices) of n=10 and n=20. The most (75%) of the polygons with n=10 had 2-3 relfex vertices. Others had up to 5 reflex vertices. Among the polygons with n=20, 80% had 5-7 reflex vertices. Table 2 details the results of these executions for different required coverage ratio. Figure 5 and Figure 6 illustrate some of these resultant inner covers.

Usually, our algorithm yields effective inner-covers. There are, however, special cases where our strategy does not result in an optimal cover in terms of the number of the polygons in the cover. One such example is shown in Figure 4, where a cover with 86% is sought. Our algorithm generates two polygons (Figure 4(b)) while the optimal inner-cover contains only one polygon (Figure 4(c)). It should be noted, however, that our algorithm yields a coverage ratio of 100%, and for that it is optimal.

	n=10			n=20	
	Coverage ratio				
Number of polygons in the inner cover	90%	95%	100%	90%	95%
1	5%	2%			
2	55%	31%	5%		
3	38%	63%	45%	6%	
4	2%	3%	41%	28%	6%
5		1%	9%	40%	28%
6				19%	43%
7				5%	15%
8				2%	8%

Table 2: The distribution of the input polygons (percent) according to the number of the polygons in the inner cover.

7 CONCLUSIONS

We have presented in this paper an algorithm for generating the convex inner-cover of a given polygon. This problem is motivated by occlusion culling algorithms which require that effective occluders be few, convex and large.

Our algorithm is based on first generating an arrangement by eliminating the reflex vertices of the given polygon. Obviously, the number of (convex) faces in the arrangement depends on the number of its reflex vertices. These faces are then used to construct an initial cover that is later optimized to yield a small number of large convex pieces that cover the polygon well.

We have implemented our algorithm and run it on a large number of polygons. We have shown that although our algorithm does not guarantee to produce the optimal cover it generates an effective inner-covers in the sense that typically, the non-convex polygon is covered by less than k convex pieces, where k is the number of reflex vertices in the non-convex polygon.

REFERENCES

[1] C. Andújar, C. Saona-Vásquez, I. Navazo. *LOD Visibility Culling and Occluder Synthesis* . Computer-Aided Design, Volume 32(13), 773–783.

[2] B. Chazelle and D. P. Dobkin. *Optimal convex decompositions*. Computational Geometry, editor G. T. Toussaint, publisher North-Holland, Amsterdam, Netherlands, 1985, 63–133.

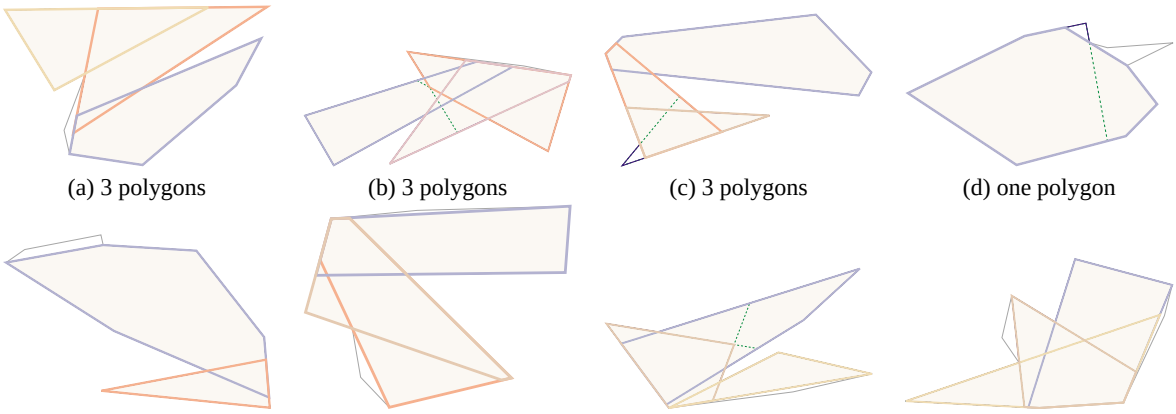
[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[4] P. Brunet, I. Navazo, J. Rossignac, and C. Saona-Vásquez. *Hoops: 3d curves as conservative occluders for cell-visibility*. In A.Chalmers and T.-M.Rhyne, editors, Computer Graphics Forum (Proc.of Eurographics '01), volume 20(3), Manchester, UK, September 2001.

[5] Computational Geometry Algorithms Library. http://www.cgal.org/

[6] J. S. Chang and C. K. Yap. *A polynomial solution for potato-peeling and other polygon inclusion and enclosure problems*. In 25th Annual Symposium on Foundations of Computer Science, pages 408-416, Singer Island, Florida, 24-26 October 1984.

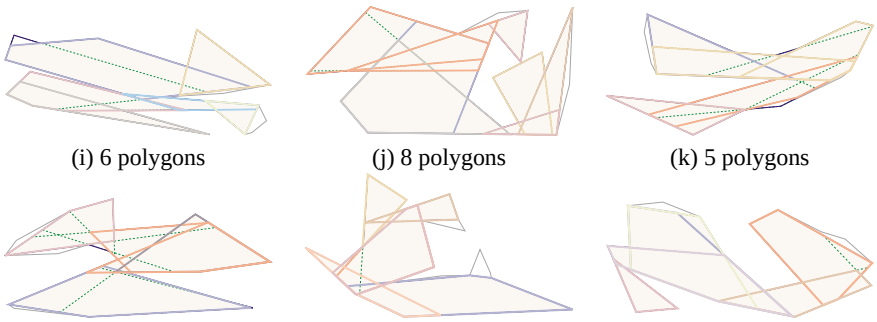
[7] D. Cohen-Or, Y. Chrysanthou, C. Silva and F. Durand. *A Survey of Visibility for Walkthrough Applications*. Siggraph 2001 course notes on visibility.



(a) 3 polygons (b) 3 polygons (c) 3 polygons (d) one polygon

(e) 2 polygons (f) 3 polygons (g) 3 polygons (h) 3 polygons

Figure 5: Examples of the results of the algorithm where coverage ratio of 95% was required. The size (number of vertices) of the input polygons is n=10. The number of polygons in the resultant inner cover is written below each example.



(i) 6 polygons (j) 8 polygons (k) 5 polygons

(l) 5 polygons (m) 6 polygons (n) 6 polygons

Figure 6: Examples of the results of the algorithm where coverage ratio of 95% was required. The size (number of vertices) of the input polygons is n=20. The number of polygons in the resultant inner cover is written below each example.

[8] D. Cohen-Or, G. Fibich, D. Halperin, and E. Zadicario. *Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes*. Computer Graphics Forum, 17(3): 243-254, 1998.

[9] S. Coorg and S. Teller. *Real-time occlusion culling for models with large occluders*. Symposium on Interactive 3D Graphics, pages 83-90, April 1997.

[10] J.C. Culberson and R.A. Reckhow. *Covering polygons is hard*. Journal of Algorithms, 17(1):2-44, July 1994.

[11] R. Germs and F.W. Jansen. *Geometric Simplification for Efficient Occlusion Culling in Urban Scenes*. WSCG 2001 Conference Proceedings, pages 291-298.

[12] D.H. Greene. *The Decomposition of Polygons into Convex Parts*. Computational Geometry, editor Franco P. Preparata 1983, 235–259.

[13] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff and H. Zhang. *Accelerated occlusion culling using shadow frustra*. In Proc.13th Annu. ACM Sympos. Comput. Geom., pages 1-10, 1997.

[14] J. M. Keil *Decomposing a polygon into simpler components*. SIAM J. Comput. 14, 1985, 799–817.

[15] V. Koltun, Y. Chrysanthou and D. Cohen-Or. *Virtual occluders: An efficient intermediate pvs representation*. Rendering Techniques 2000: 11th Eurographics Workshop on Rendering, pages 59-70, June 2000. ISBN 3-211-83535-0.

[16] F-A. Law and T-S. Tan. *Preprocessing occlusion for real-time selective refinement*. In Stephen N. Spencer, editor, Proceedings of the Conference on the 1999 Sym-posium on interactive 3D Graphics, pages 47-54, New York, April 26-28 1999. ACM Press.

[17] L. Leblanc and P. Poulin. *Guaranteed occlusion and visibility in cluster hierarchical radiosity*. In Eurographics Workshop on Rendering, pages 89-100, June 2000.

[18] C. Levcopoulos, J. Gudmundsson. *Approximation Algorithms for Covering Polygons with Squares and Similar Problems*. RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science, 1997.

[19] H. Zhang, D. Manocha, T. Hudson and K-n.E. Hoff III. *Visibility culling using hierarchical occlusion maps*. In Turner Whitted, editor, SIGGRAPH 97 Conference Proceedings, Annual Conference Series, pages 77-88. ACM SIGGRAPH, Addison Wesley, August 1997.