Constructability of Trip-lets

Jeroen Keiren[*] Freek van Walderveen[†] Alexander Wolff[*]

Abstract

A trip-let is an object as shown on the cover of Hofstadter's book *Gödel, Escher, Bach*: a solid, three-dimensional object that, when viewed from three orthogonal directions, shows three different letters. In this paper we consider two problems related to the construction of such objects for a given set of three letters. First, we want to know whether the silhouettes of the object correspond to the letters we used to make the object. Second, we are interested in the connectedness of the final object: does it fall apart during construction? We obtain results on the combinatorial complexity of objects and silhouettes for letters given as general or rectilinear polygons with holes, and give algorithms to solve the problems efficiently for the rectilinear case.

1 Introduction

The process of making a trip-let can be described mathematically as intersecting a set of infinite cones, where each cone has its apex at the position of the viewer and the shape as its base. In this paper, we only consider orthogonal projections, thus cones become infinitely long prisms.

Definition 1 *The z-*prism *induced by a polygonal shape (with holes) $S_{xy} \subset \mathbb{R}^2$ is the volume*

$$P_z(S_{xy}) := \{(x, y, z) \mid (x, y) \in S_{xy}, z \in \mathbb{R}\} \subset \mathbb{R}^3.$$

We define $P_y(S_{xz})$ and $P_x(S_{yz})$ analogously.

Definition 2 *The* trip-let *obtained from three given shapes S_{xy}, S_{xz}, and S_{yz} is the intersection of their prisms: $P_z(S_{xy}) \cap P_y(S_{xz}) \cap P_x(S_{yz}) \subset \mathbb{R}^3$.*

In this paper we consider two properties of trip-lets; *validity* and *connectedness*. A connected trip-let is simply a trip-let consisting of one solid, connected piece. To define the validity of a trip-let, we need the notion of silhouette.

Definition 3 *The z-*silhouette *of a volume V is*

$$\pi_{xy}(V) := \{(x, y) \mid \exists z \in \mathbb{R} : (x, y, z) \in V\}.$$

Again, we define x- and y-silhouette similarly.

We say that a trip-let is valid if each of its silhouettes matches exactly the corresponding input shape S_{xy}, S_{xz}, and S_{yz}. The example shown in Figure 1(a) illustrates this. Although the intersection of any combination of two of the polygons has correct silhouettes, adding the third polygon yields an invalid trip-let. Hence, any algorithm solving this problem has to consider all three shapes together. Figure 1(b) illustrates the connectedness problem: in the centre of the trip-let there is a small cube that is not connected to the rest of the object.

At the 1998 *ACM Symposium on Computational Geometry*, O'Rourke [1] presented the problem of finding "simple conditions" to determine whether three letters can form a valid and connected trip-let. We present a first step into this direction. We analyze the combinatorial complexity of triplets and silhouettes, and we give algorithms for testing the validity and connectedness of triplets induced by *rectilinear* polygons, that is, polygons whose edges are parallel to one of the primary axes. We measure the complexity of our algorithms with respect to the total number n

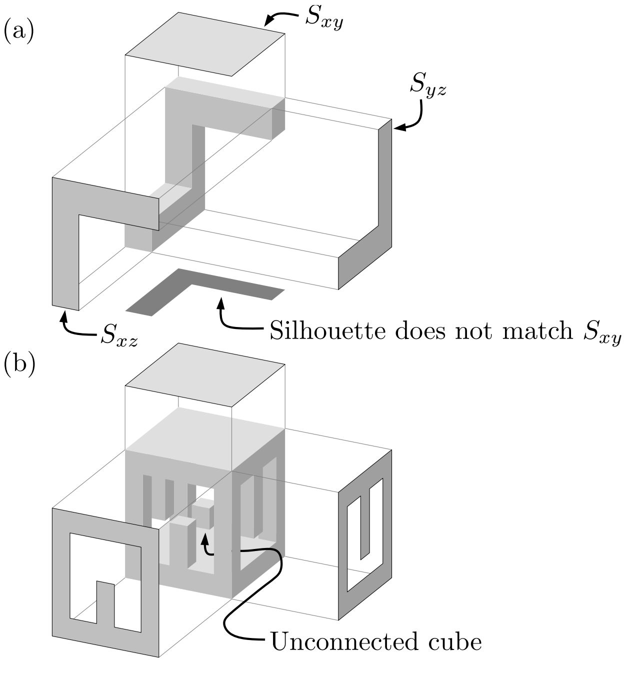


Figure 1: (a) An invalid trip-let: the z-silhouette does not match the top shape. (b) A non-connected trip-let: it consists of two components.

[*]Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, The Netherlands. j.j.a.keiren@student.tue.nl, www.win.tue.nl/~awolff

[†]MADALGO, Department of Computer Science, University of Aarhus, Denmark. freek@vanwal.dk

of vertices of the input polygons.

Apart from their nice visual appearance, understanding the class of objects that can be represented by trip-lets may yield useful insights in our ability to understand three-dimensional scenes from silhouettes obtained from different viewpoints [7].

Earlier work Most work related to the construction of 3D objects from shapes considers the reconstruction of actual objects from silhouettes obtained from different viewpoints. Our orthogonal case can be seen as a variant of this problem with viewpoints at infinity where we are not sure whether the object to be reconstructed actually exists.

Bottino and Laurentini [4] ask whether it is possible to find the relative positions of viewpoints parallel to a plane, given their corresponding silhouettes. If such positions can be found, they call the silhouettes *compatible.* They give sets of inequalities for the relative positions of the viewpoints for viewing directions parallel to the plane, but do not provide efficient algorithms to find these positions.

Ohgami and Sugihara [8] propose an algorithm for finding relative viewing points in the plane for three given silhouettes. Their algorithm does not guarantee to find feasible positions even if they exist.

Hachenberger and Kettner [5] describe the implementation of Boolean operations on 3D Nef polyhedra in CGAL. As a prism can be described by a 3D Nef polyhedron, trip-lets can be constructed solely using Boolean operations on Nef polyhedra. Then, the number of components of the resulting trip-let is known, solving our first problem. The silhouettes can be found by projecting the resulting polyhedron back to planes orthogonal to any of the three axes. By comparing the silhouettes with the input shapes, our second problem is also solved.

The expected running time of the intersection implementation described by Hachenberger and Kettner is dominated by the term $O(k\sqrt[3]{k}\log k)$, where k is the size of the resulting polyhedron. Hence, the above solution runs in $O(n^4 \log n)$ expected time if the object consists of $\Theta(n^3)$ cubes (see Fig. 2 for an example of such an object).

Our results First, we investigate (in Section 2) the combinatorial complexity of our objects: given three shapes of total complexity n, we show that the resulting trip-let and its silhouettes have complexity $\Theta(n^3)$ and $\Theta(n^4)$, respectively. Given rectilinear shapes, we show that in the worst case a silhouette has complexity $\Theta(n^2)$.

Second, we give algorithms for testing validity and connectedness of trip-lets constructed from rectilinear shapes, see Sections 3 and 4. Within the class of algorithms that actually construct the silhouettes and trip-lets, our algorithms are near-optimal in the worst case, testing validity and connectedness in time $O(n^2 \log n)$ and $O((n^2 + k)\log n)$, respectively. Note that k, the complexity of the given trip-let, is $\Omega(n^2) \cap O(n^3)$ in the worst case.

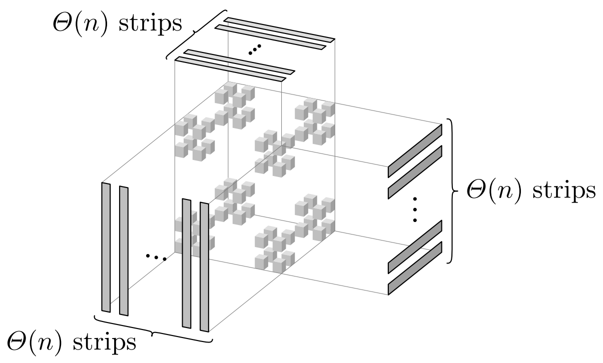


Figure 2: Three shapes generating $\Theta(n^3)$ cubes.

2 Complexity of objects and silhouettes

Determining the worst-case complexity of objects and silhouettes by bounding the respective number of vertices allows us to prove lower bounds on the running time of any algorithm that, in answering one of the problems, internally constructs the actual object or silhouette. Due to lack of space, we only state the results and give some of the more important examples.

We start with a lemma concerning the complexity of objects obtained from two or three shapes.

Lemma 1 *Given $m \in \{2, 3\}$ shapes with n vertices in total, the complexity of the intersection of their orthogonally oriented prisms is $\Theta(n^m)$ in the worst case.*

An example illustrating the lower bound is given in Figure 2. The set of three shapes yields an object consisting of $\Theta(n^3)$ vertices. Dropping one of the shapes results in an object with $\Theta(n^2)$ vertices.

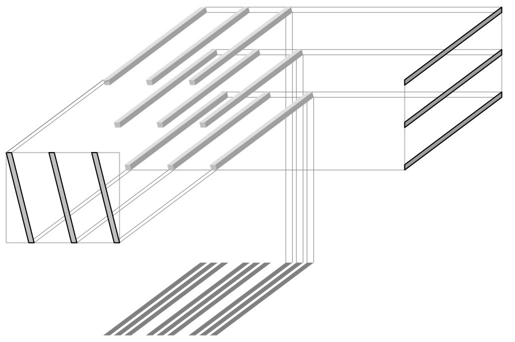
Note that the trip-let in Figure 2 is not connected. We conjecture the following.

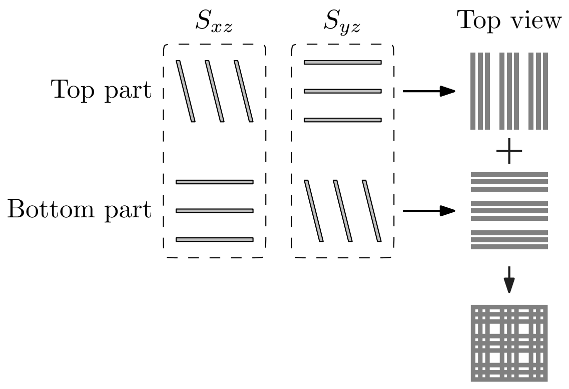
Conjecture 1 *Connected trip-lets have $O(n^2)$ vertices.*

It is easy to construct a connected trip-let with $\Theta(n^2)$ vertices.

The following lemma shows that silhouettes can have higher complexity if we allow general polygonal shapes.

Lemma 2 *A silhouette constructed from general polygonal shapes of total complexity n has complexity $\Theta(n^4)$ in the worst case.*


(a)



(b)

Figure 3: (a) Construction with $3 \times 3 = 9$ bars that do not overlap in the z-silhouette. (b) Construction with $(9-1) \times (9-1) = 64$ holes in the z-silhouette.

For proving the lower bound, consider one shape with $\Theta(n)$ parallel strips that are slanted but do not overlap when projected vertically, and one with $\Theta(n)$ parallel, horizontal strips such as in Figure 3(a). The z-silhouette obtained from these two shapes consists of $\Theta(n^2)$ parallel strips. By swapping the two shapes the silhouette turns 90 degrees. If we stack the original and rotated version, the combined silhouette is a grid pattern with $\Theta(n^4)$ vertices (see Figure 3(b)).

Hence, any algorithm that internally constructs the silhouettes to be verified can never be faster than $\Omega(n^4)$. Fortunately, if we restrict ourselves to rectilinear shapes, things cannot get that bad.

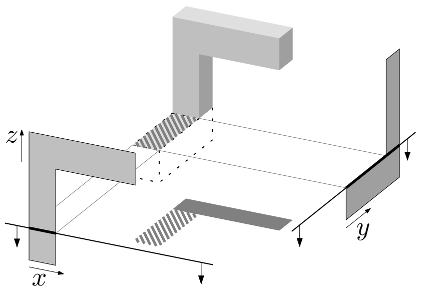


Figure 4: Constructing the bottom silhouette; the dashed area is a new rectangular facet.

Lemma 3 *A silhouette constructed from rectilinear shapes of total complexity n has complexity $\Theta(n^2)$ in the worst case.*

3 Determining validity

The correctness of one of the silhouettes of a trip-let, say the one for shape S_{xy}, can be stated by saying that the intersection of the prisms induced by the other shapes "overshadows" S_{xy}:

$$S_{xy} \subseteq \pi_{xy}(P_y(S_{xz}) \cap P_x(S_{yz})).$$

We can use this characterization as a general idea for an algorithm to determine the validity of a trip-let. If the silhouette of the intersection of two prisms is known, a simple line-sweep suffices to determine if this silhouette covers the third shape. As we saw in Section 2, general polygonal shapes may yield silhouettes of complexity $\Theta(n^4)$. We expect we cannot easily get better running times than we saw for the approach outlined in the introduction (at least for the validity problem). Therefore, from now on, we will only consider shapes described by rectilinear polygons.

We compute the orthogonal projection of the intersection of the prisms formed by two silhouettes using a sweep-plane algorithm. The orientation of the sweep plane (effectively consisting of two sweep lines) is illustrated in Figure 4. For each event—that is, a horizontal edge—the new facets of the trip-let introduced by that event that are parallel to the sweep plane are computed. This results in a set of rectangles whose union is exactly the orthogonal projection we were looking for. As this is a fairly standard procedure, we will not go into further detail.

The main problem with actually computing the union of this set of rectangles is that there may for example be $\Theta(n^2)$ rectangles whose union again has a complexity of $\Theta(n^2)$ (consider for example a rectilinear version of Figure 3), so adding one rectangle at a time will yield a running time of $\Theta(n^3)$ or worse. Instead, we determine the union in one go using a sweep-line algorithm. We maintain the status of the sweep line in a data structure that can efficiently (i) add a segment when a new rectangle is encountered, (ii) report the intervals covered by this new segment that are not currently covered (as they introduce new edges in the silhouette polygon), and (iii) remove a segment when the sweep line hits the bottom of a rectangle. Segment trees [3, Section 10.3] come close but do not provide an efficient implementation of the second operation: reporting k empty intervals within a given range in $O((k+1)\log n)$ time. We solve this by augmenting the segment tree with a field for every node ν that indicates whether the interval corresponding to ν is completely covered by segments that end somewhere inside this interval. Combining this with information about the segments completely covering this interval that is already present in the segment tree, we can implement the empty-interval query in the given time complexity.

Theorem 4 *The validity of a trip-let obtained from three rectilinear shapes of total complexity n can be determined in $O(n^2 \log n)$ time.*

A formal proof is omitted due to lack of space.

4 Determining connectedness

To check whether the intersection of three orthogonal prisms is connected, we again use a sweep-plane algorithm. In short, it maintains the intersection of the sweep plane with the object, which we call a *section*, and keeps a data structure representing the connectedness of the components of the object already passed by the sweep plane. Each time a new part of the object hits the sweep plane, the algorithm determines whether this part is connected to any other known part and perhaps even connects two parts that were not connected before. Then, using a union-find data structure, we can determine whether the final object is connected. Our objective is to make the running time of the algorithm linearly dependent on the complexity of the object, up to a polylogarithmic factor.

Because our algorithm effectively calculates the intersection of the three prisms, it may be compared to the algorithm for Boolean operations on orthogonal polyhedra by Aguilera and Ayala [2][1]. The key difference with this algorithm is in the way new sections are computed. Instead of doing the same Boolean operation (intersection in our case) on two objects of lower dimension (the rectilinear polygon forming the intersection of the sweep plane and two of the three prism on the one hand, and the third shape on the other hand), we update our frame incrementally so that we do not spend time on parts of a section that will not cause new vertices of the final object.

This way, we can determine whether a trip-let from rectilinear shapes is connected in $O((n^2+k)\log n)$ time. Combining this result with that from the previous section, we obtain our last theorem.

Theorem 5 *Given three rectilinear shapes with n vertices in total, we can determine whether the intersection of the prisms induced by these shapes forms a valid and connected trip-let in $O((n^2+k)\log n)$ time, where k is the complexity of the trip-let.*

[1]We do not understand Aguilera and Ayala's claim that their algorithm runs in time linear in the number of vertices of the input polyhedra.

Note that by Lemma 1, $k = O(n^3)$ in the worst case; if Conjecture 1 holds, the algorithms run in $O(n^2 \log n)$ time.

5 Open problems

The running times of our algorithms for rectilinear input shapes are almost optimal within the class of algorithms that actually construct the objects or silhouettes. Finding the conditions O'Rourke is looking for, or the problem of whether it is possible to do better by somehow finding the answers without constructing the silhouettes or objects first, is still open: we do not know of any non-trivial lower bounds for this case yet. Furthermore, does our conjecture hold, that is, does any connected trip-let have at most quadratic complexity?

Acknowledgments

We thank Christ van Willegen for bringing these problems to our attention.

References

[1] P. K. Agarwal and J. O'Rourke. Computational Geometry Column 34. *SIGACT News* 29(3):27–32, 1998. `http://www.cs.duke.edu/~pankaj/scg98-openprobs/`.

[2] A. Aguilera and D. Ayala. Orthogonal polyhedra as geometric bounds in constructive solid geometry. *Proc. 4th ACM Symp. Solid Modeling and Applications*, pp. 56–67, 1997.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, third edition, 2008.

[4] A. Bottino and A. Laurentini. Introducing a new problem: Shape-from-silhouette when the relative positions of the viewpoints is unknown. *IEEE Trans. Pattern Analysis and Machine Intel.*, 25(11):1484–1493, 2003.

[5] P. Hachenberger, L. Kettner, and K. Mehlhorn. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. *Comput. Geom. Theory Appl.*, 38(1–2):64–99, 2007.

[6] D. R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, 1979.

[7] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Analysis and Machine Intel.*, 16(2):150–162, 1994.

[8] T. Ohgami and K. Sugihara. Realizability of solids from three silhouettes. In *Abstracts 24th European Workshop Comput. Geom.*, pp. 233–236, 2008.