Ten Problems in Geobotics

Mikkel Abrahamsen
University of Copenhagen, Denmark

Dan Halperin
Tel Aviv University, Israel

Abstract

Robots sense, move and act in the physical world. It is therefore natural that algorithmic problems in robotics and automation have a geometric component, often central to the problem. Below we review ten challenging problems at the intersection of robotics and computational geometry—let's call this intersection *Geobotics*. What is common to most of these problems is that the prevalent algorithmic techniques used in robotics do not seem suitable for solving them, or at least do not suggest quality guarantees for the solution. Solving some of them, even partially, can shed light on less well-understood aspects of computation in robotics.

Keywords and phrases Computational geometry, robotics, motion planning, reconfiguration, object rearrangement, assembly planning

Digital Object Identifier 10.57717/cgt.v5i1.85

Funding *Mikkel Abrahamsen*: Supported by Independent Research Fund Denmark, grant 1054-00032B, and by the Carlsberg Foundation, grant CF24-1929.
Dan Halperin: Supported in part by the Israel Science Foundation (grant no. 3598/25) and by the Blavatnik Computer Science Research Fund.

© Mikkel Abrahamsen and Dan Halperin
licensed under Creative Commons License CC-BY 4.0

Contents

0 Introduction 3

1 Shortest coordinated paths for two robots 4

2 Can one polyhedron cage another? 7

3 Interpolate between easy and hard instances of multi-robot motion planning 10

4 Geometric reconfiguration and object rearrangement 11

5 Interlocked objects with fewer convex parts 13

6 Minimal design modification, or explainability 15

7 Assembly planning with more than two hands 18

8 Compact representation of spatial objects 19

9 Partitioning large objects for manufacturing 22

10 Convex covering of free configuration space 23

0 Introduction

Robotics has persistently raised interesting problems for computational geometry. A pioneering and exemplary case in point is the 1980s series of papers "On the Piano Movers Problem" by Schwartz and Sharir [117, 118, 116], which laid out the theoretical foundation for the study of motion planning in robotics as well as many other applications. These were followed by a sequence of results studying special instances of motion-planning problems raising fundamental questions on the structure and complexity of arrangements of curves and surfaces (see, e.g., [45, 65, 66, 79]), or special types of Voronoi diagrams [93, 103].

A large set of results in automation have followed, again using and promoting the development of computational geometry techniques, for example in part orienting and fixturing (see, e.g., [28, 107, 128, 129]) or in grasping (see, e.g., [39, 83, 99, 106]).

While this type of research moderately continued, the area of algorithms in robotics has undergone a major change in the mid 1990s, with the introduction of sampling-based (SB, for short) techniques, most notably probabilistic roadmaps (PRM) [78], ensued by rapidly exploring Random Trees (RRT) [92], and their many variants [104]. There are a bounty of methods in the algorithmic toolbox for robotics. However, we mention SB techniques in particular since many of the problems that we review below relate to motion of robots (or other objects), and this is an area that has been predominated by SB techniques for about three decades.

Sampling-based methods revolutionized the algorithmic approach in robotics. Many problems, which were previously practically insolvable, now became not only efficiently solvable, but also with only moderate programming efforts, assuming you have at your disposal good auxiliary tools for collision detection [80] and nearest-neighbor search (see, e.g., [73, 100]). However, when the setting is tight, as it is for example in assembly planning or packing, these methods often perform poorly, as their performance heavily relies on the denseness of the setting. They can run for a very long time without finding a solution, and it is not clear whether a solution does not exist or more time is needed. This phenomenon becomes more pronounced when dealing with multi-robot or multi-body systems, and when striving for high quality, optimized, solutions. This issue is exemplified and discussed in detail in Section II.C of [113]—see in particular Figures 4 and 5 there—for the case of two robots.

Machine learning (ML) techniques are becoming ubiquitous in robotics in recent years. Besides a conference, CoRL, dedicated to learning in robotics, these techniques preponderate the more traditional long-running conferences in the field; indeed in ICRA 2024, the two winners of the best paper award are focused on ML and data centered robotics. To the best of our knowledge, the open problems that we suggest here have not yet been successfully addressed with ML techniques. Therefore, we pose an overarching problem of whether any of the algorithmic questions reviewed below can be efficiently solved using ML.

For each problem, we provide the basic background including terminology and references in the respective sections. We do assume some familiarity with the basic terminology of motion planning, and in particular (the number of) the degrees of freedom of a system, and the notion of configuration space. For the fundamentals and terminology, consider the classical books by Latombe [89] or LaValle [91], or the surveys in the Handbook on Discrete and Computational Geometry (3rd edition) [63, 67].

Several problems in this review concern *assembly planning*. For the needed algorithmic basics, consider, for example, [63, Section 51.2] or the first three sections of the survey [64].

In recent years, robotics has expanded so rapidly and across such a wide range of

Workspace	A subset W of 2D or 3D physical space: $W \subset \mathbb{R}^k$, $k = 2$ or 3.
Configuration	Any mathematical specification of the position and orientation of every rigid body composing a robot, relative to a fixed coordinate system. The configuration of a single rigid body is also called a **placement** or a **pose**.
Configuration space	The set of all configurations of a robot. For almost any robot, this set is a smooth manifold. Given a robot R, we will let $R(q)$ denote the subset of the workspace occupied by R at configuration q.
# of degrees of freedom	The dimension of the configuration space.

Table 1 Basic definitions in **robot** motion planning, adapted from [63].

domains that it is no longer feasible to comprehensively enumerate all current and potential applications. Spanning industrial automation, logistics, healthcare, autonomous systems, agriculture, and domestic services, the field continues to diversify at an accelerating pace. Consequently, any survey of challenges must focus on representative areas rather than attempt exhaustive coverage. This review emphasizes problems with strong geometric components, reflecting the authors' perspective.

1 Shortest coordinated paths for two robots

Let A and B be two unit-disc robots ("roombas") moving among polygonal obstacles $\mathcal{O}$ in the plane. We specify the *configuration* (see Table 1) of each robot by the coordinates of its center: When robot R is placed with its center at the point $p \in \mathbb{R}^2$, we let $R(p)$ denote $\{x \in \mathbb{R}^2 \mid \|x - p\| < 1\}$. Our goal is to plan a collision-free motion for the two robots from given free start configurations marked A_0 and B_0, to given free target configurations A_1 and B_1. Throughout the motion the robots should not collide with the obstacles, nor with one another. More formally, a solution to the coordinated motion for A and B is a plan (π_A, π_B), where for each $R \in \{A, B\}$, we have $\pi_R : [0, 1] \to \mathbb{R}^2$ with $\pi_R(0) = R_0, \pi_R(1) = R_1$, and for all $t \in [0, 1] : R(\pi_R(t)) \cap \mathcal{O} = \emptyset$ and $A(\pi_A(t)) \cap B(\pi_B(t)) = \emptyset$. Notice that the obstacles $\mathcal{O}$ constitute a closed set and each robot is an open set, so in a valid motion plan the robots may touch, but never overlap each other or the obstacles. It is not known how to efficiently find a coordinated motion plan that minimizes the total length of the paths, or if the problem is provably hard. We formulate this as the following question.

> **Problem 2DiscsMinSum**: Given two unit discs moving in the plane among polygonal obstacles with a total of n vertices, devise an efficient algorithm to find a coordinated motion plan (π_A^*, π_B^*) for the discs from free start to free target configurations such that the total length of the paths of the two robots is the smallest among all possible solutions,
>
> $$(\pi_A^*, \pi_B^*) = \text{argmin}_{(\pi_A, \pi_B)\text{-plan}} |\pi_A| + |\pi_B|,$$
>
> if such a motion exists. Alternatively, show that the problem is NP-hard.

If we relax the requirement that the paths are optimal (MinSum) in **2DiscsMinSum**, and just seek to produce feasible solution paths (where the robots do not collide with the obstacles nor with one another along the way), then an efficient solution due to Sharir and

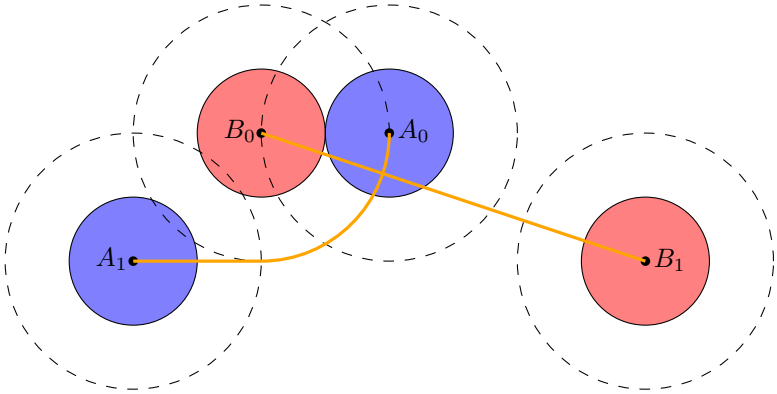


Figure 1 The optimal solution of an instance of **2DiscsMinSum** in the absence of obstacles. The shaded discs are the unit-disc robots at their start and target configurations, and the dotted circles are of radius 2 around the start and target configurations. Example following [81].

Sifrony has been known for a long time [120], running in $O(n^2)$ time, as well as an algorithm by Yap [136] for two discs (respectively, three discs) running in $O(n^2)$ (respectively, $O(n^3)$) time.

Interestingly, if we keep the optimality requirement (MinSum) and drop the obstacles, namely the two disc robots move freely in the plane and just have to avoid collision with each other, the problem is still non-trivial and was open for a long time until settled several years back by Kirkpatrick and Liu [81]. They analyze the shape of the optimal coordinated paths, and show that these paths consist of at most six line segments and circular arcs. See Figure 1 for an illustration.

The solution for the MinSum no-obstacle setting has recently been adapted to the case of two squares by Esteban, Halperin, Ruíz, Sacristán and Silveira [49]. The solution for two squares is more involved than for two discs since some symmetries (relative orientation of the two robots) that can be exploited in the latter do not exist in the former. Since then, Kirkpatrick and Liu [82] have generalized the results for disks and squares to arbitrary centrally symmetric shapes.

Back to the full-blown problem **2DiscsMinSum** seeking an optimal solution in the presence of obstacles, we call its two-square variant, where instead of a unit disc each robot is a translating-only axis-parallel unit square, problem **2SqrsMinSum**. In the case where the obstacles are orthogonal, Agarwal, de Berg, Holmgren, Steiger and Struijs recently described a polynomial-time algorithm [12] (see also [14]).

The shortest joint paths (MinSum) is one of several frequently used optimization criteria in multi-robot motion planning. Additional measures, which we state here for teams of many robots (not limited to two), include

MinMax: Minimizing the length of the longest path taken by a robot in the team.

Makespan: Minimizing the time it takes to complete the motion of all the robots in the team, namely the elapsed time from the moment the first robot leaves its start configuration to the time the last robot reaches its goal configuration. Here (and in the FlowTime version below) we assume that each robot moves with at most unit speed and that the robots can instantaneously change speed and direction. With two unit-squares among orthogonal obstacles, this version was shown to be NP-hard in [12].

FlowTime: Minimizing the sum of the individual arrival times of the robots, namely the time it takes each robot to complete its motion, from the joint starting time.

We emphasize that in the setting with no obstacle, only the MinSum variant is characterized [81] but not the other measures, although some have been considered [43].

Problem 2DiscsNoObstacles: Given two unit discs moving in the empty plane (in the absence of obstacles), characterize the joint motion of the discs that minimizes the MinMax, Makespan or FlowTime.

Often in robotics we wish to account for several optimization criteria simultaneously. A typical goal is to have short paths and at the same time keep large (or at least sufficient) *clearance*. The clearance of a robot R at a fixed configuration q is the shortest distance between $R(q)$ and the obstacles. In the multi-robot case, the clearance of a robot R_i in the team is the shortest distance of R_i (at a fixed configuration) to the obstacles or to any other robot at the same fixed configuration (which now specifies the placements of all the robots).

For a point robot moving among polygonal obstacles in the plane (which through employing the Minkowski sum operation applies also to a polygon translating among polygons in the plane), we can scalarize the two criteria into a single measure. Let π denote the path traversed by the point. Parameterize the path by its length ℓ. For every point $q \in \pi$, let $c(q)$ be the clearance at q, namely the shortest distance from q to the obstacles. Then the cost of the path is defined as follows

$$L(\pi) = \int_{q \in \pi} \frac{1}{c(q)}\, dq \ .$$

This measure has been introduced by Wein, van den Berg and Halperin [131], who also gave an approximation algorithm to find the path. The algorithm was later improved by Agarwal, Fox and Salzman [13].

In preparation for the case of two discs, we write the measure $L(\pi)$ in an equivalent but slightly different manner. Let $\pi : [0,1] \longrightarrow \mathbb{R}^2$ be a parameterization of the path traversed by the point, and write $\pi = (x, y)$ for the coordinate functions. For every point q, let $c(q)$ be the clearance at q, namely the shortest distance from q to the obstacles. Then the cost of the path is defined as follows

$$L(\pi) = \int_0^1 \frac{\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}}{c(\pi(t))}\, dt \ .$$

When we work with more than one robot then the clearance should be kept among the moving robots as well. The following problem is an extension of **2DiscsMinSum** to account for the combined length-clearance cost measure. Let $\pi = (\pi_0, \pi_1)$ be a path in $\mathbb{R}^4$. For every point along the path (x_0, y_0, x_1, y_1), the first (respectively, last) two coordinates represent a point in the path of the first (respectively, second) robot.

Problem 2DiscsCombinedMeasure: Given two discs moving in the plane among polygonal obstacles with a total of n vertices, devise an efficient algorithm to find (or approximate) a coordinated motion plan (π_0, π_1) for the discs from free start to free target configurations such that the total combined measure is minimized,

$$L(\pi_0, \pi_1) = \int_0^1 \left(\sum_{i=0}^{1} \frac{\sqrt{\dot{x}_i(t)^2 + \dot{y}_i(t)^2}}{\min(c(\pi_i(t)), \|\pi_i(t) - \pi_{1-i}(t)\| - 2)} \right) dt,$$

where $c(q)$ measures the minimal distance between the robot at configuration q and the obstacles[1], and $\pi_i : [0,1] \longrightarrow \mathbb{R}^2$ is a parameterization of the path of disc $i \in \{0,1\}$

[1] The (minimal) distance between two sets A and B is the smallest distance between any point of A and any point of B.

from its start to target position, with coordinate functions $\pi_i = (x_i, y_i)$. Otherwise, if no such motion exists, the algorithm stops and reports so.

We subtract 2 from $\|\pi_i(t) - \pi_{1-i}(t)\|$ because the discs collide when the distance between the centers is less than 2. As a warm-up, we can also consider the problem with point-shaped robots instead of discs, which eliminates subtracting 2. That version may have a simpler behaviour. It is furthermore clear that there is always a solution if there exist paths from the start configurations to the targets.

The combined measure of path length and clearance together with several variants of it have been studied in the context of *sampling-based robot motion planning*; see, e.g., [41, 91, 112]. A method called hybridization graphs (HGraphs for short) [108] takes as input a small number of paths, each produced by a randomized planner for the same motion planning problem. It then adds collision-free bridges between pairs of configurations along the paths (either a single path or two paths). Finally, it searches for the best path according to some scalarized objective function in this augmented structure. It typically produces much higher quality paths than the initial input paths, and also higher quality paths than are produced when we give a sampling-based planner (say RRT) as much time as the entire HGraphs procedure requires. This approach seems to select the best portions from each path. However it comes with no guarantees on the quality of the final solution. It would be interesting to theoretically analyze its performance, to explain its favorable behavior.

There are many interesting combinations of optimality criteria and we conclude with pointing out a combination suitable for teams of robots. Two prevalent quality measures for the motion of teams of robots, which we have already mentioned for two discs, are MinSum and Makespan. It would be desirable to take these two criteria simultaneously into account in optimizing multi-robot motion planning.

2 Can one polyhedron cage another?

Let A and B be two rigid bodies (objects) in the plane or in 3-space. We assume that object A is static and object B can move. We say that object A *cages* object B at configuration c (like before we denote the object B at configuration c by $B(c)$) if (i) A and $B(c)$ are interior-disjoint, and (ii) B cannot be moved arbitrarily far from configuration c without colliding with A.

Caging [109] has been proposed and studied in robotics as a relaxed form of *grasping* [105, 110, 111]. Object A can be viewed as a static snapshot of a robot gripper, and it holds the work-piece B in a way that, for the task at hand, sufficiently restricts the motion of object B. Consider the survey by Makita and Wan [96] on robotic caging for, among others, the relation of caging to a variety of mathematical problems.

We will focus our discussion of the caging problem on polygonal or polyhedral objects. For full rigid-body motion in space, our caging problem can be formulated as follows.

Problem 3DCaging: Let A and B be two polyhedra in $\mathbb{R}^3$, with m and n vertices respectively. Efficiently determine whether A can cage B. If so, efficiently find such a caging configuration.

Thus **3DCaging** comprises two subproblems: A decision problem and the problem of actually determining a caging configuration, if one exists. As we shall argue below, the decision subproblem may in some cases be easier than finding a caging configuration. In general, though, we currently do not know how to solve the decision problem more efficiently

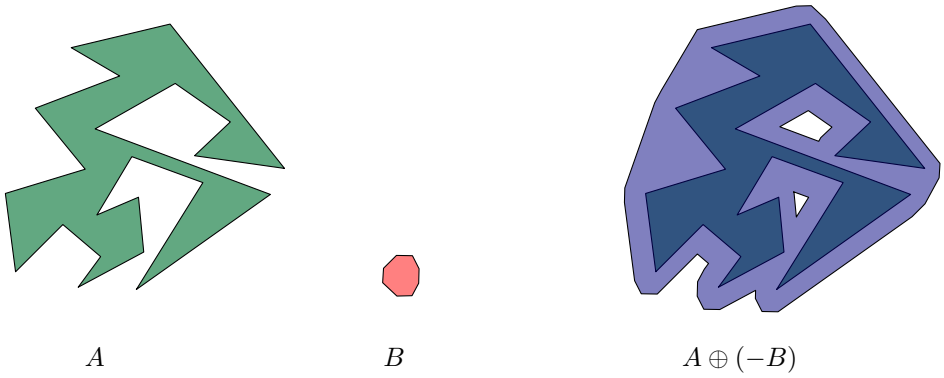


Figure 2 Polygons A and B and the Minkowski sum $A \oplus (-B)$ on top of A. As is seen, $A \oplus (-B)$ has two holes, which correspond to the two "pockets" of A that cage B.

than finding a caging configuration. The statement of the problem emphasizes efficiency, since we have a polynomial time solution for it and for all the problems that we mention in this section. However, this problem raises a curious issue that is long time unresolved and leaves a notable gap between the upper and lower bounds on the running time of the algorithms for its solution. This intriguing phenomenon already occurs when the two objects are polygons in the plane, and where B is only allowed to translate.

> **Problem 2DCagingTrans**: Let A and B be two polygons in $\mathbb{R}^2$, with m and n vertices respectively. Polygon B can only translate. Efficiently determine whether A can cage B under translation only. If so, efficiently find such a caging configuration.

We now take a small side-track and define a fundamental mathematical operation, which is ubiquitous in robotics and automation as well as in CAD/CAM, 3D modeling, layout optimization, and even economics: *Minkowski summation*. For two sets P, Q in $\mathbb{R}^d$, their Minkowski sum, denoted $P \oplus Q$, is the pairwise vector sum of the points in both sets, that is $P \oplus Q = \{p + q \mid p \in P, q \in Q\}$. The Minkowski sum of two polygons is a polygonal set. Let $-P$ denote the object P in $\mathbb{R}^d$ reflected through the origin, namely $-P = \{-p \mid p \in P\}$. We demonstrate the connection between the caging problem and Minkowski sums by the following equivalent statement of **2DCagingTrans** (see Figure 2):

> **Problem 2DHoleInMinkowskiSum (equivalent to 2DCagingTrans)**: Let A and B be two polygons in $\mathbb{R}^2$, with m and n vertices respectively. Does the Minkowski sum $A \oplus (-B)$ contain a hole? If so, find a point inside this hole.

The equivalence between the two problems stems from the following simple observation.

▶ **Observation 1.** *Let A and B be two rigid bodies in $\mathbb{R}^d$, and t a vector in $\mathbb{R}^d$. The object B translated by the vector t, namely $B + t$, intersects A if and only if $t \in A \oplus -B$.*

The vector t represents the configuration of the moving object (robot) B when only translations are allowed. Therefore the configuration space of this motion problem is $\mathbb{R}^d$ and $A \oplus -B \subseteq \mathbb{R}^d$ is the collection of forbidden configurations, namely, configurations where the translated B intersects with A. In caging, we are looking for a translation t such that $t \notin A \oplus -B$. However, this translation $t \in \mathbb{R}^d$ needs to be completely surrounded by points in $A \oplus -B$, as otherwise there will be a path from t to infinity in the complement of $A \oplus -B$ and A will not cage B. In other words t must be inside a hole in $A \oplus -B$.

Let's assume for simplicity that $m = n$. We can compute the Minkowski sum $A \oplus -B$ in $O(n^4)$ time. This algorithm in turn will reveal whether there is a hole in the sum and will also give us a caging configuration when there is a hole in the sum. However, in general we do not know to decide whether there is a hole in the Minkowski sum of two polygons with n vertices each in $o(n^4)$ time. Erickson [48] showed that the **2DHoleInMinkowskiSum** is so-called 6-*SUM hard*. In the problem k-SUM, we are given integers $x_1, \ldots, x_n$, and we want to know if there exist k distinct of these integers that sum to 0. It is folklore that this problem can be solved in $O(n^{\lceil k/2 \rceil})$ time [2]. In certain restricted models of computation, a lower bound of $\Omega(n^{\lceil k/2 \rceil})$ has also been shown [16, 47]. It has therefore become a popular conjecture that k-SUM cannot be solved in $O(n^{\lceil k/2 \rceil - \varepsilon})$ time for any $\varepsilon > 0$ in the standard word RAM model. Assuming this conjecture, the 6-SUM hardness implies that there does not exist an algorithm for **2DHoleInMinkowskiSum** with running time $O(n^{3-\varepsilon})$. This still leaves a gap in the order of magnitude of n between the lower bound and the running time of the existing algorithm.

For the first problem of this section, **3DCaging**, we need to generalize the notion of Minkowski sums, and resort to the general theory of arrangements of surfaces [66]. By standard arguments in this theory, **3DCaging**, assuming $m = n$, can be solved in $O(n^{16+\varepsilon})$ time [36, 85], for any fixed $\varepsilon > 0$. Improving the running time in this case introduces many challenges, and some of them are fundamental and hard questions in the study of arrangements of surfaces in higher dimensions and are beyond the scope of this survey. So let's get back to the two-dimensional translational case.

Finding an efficient algorithm for **2DCagingTrans**, or equivalently for **2DHoleInMinkowskiSum**, seems a hard nut to crack. What else can be done? Karasev [77] has characterized a family of non-convex polygons with the property that the Minkowski sum of any pair of them does not have holes. Note that this is trivial for convex polygons, as the Minkowski sum of a pair of convex polygons is a convex polygon. Karasev suggests a measure for how far the summand polygons are from being convex, and if they are not too far by that measure, then their Minkowski sum does not have holes, which can be decided much more efficiently than constructing the sum. Can we find other families of polygons for which the Minkowski sum of pairs of them is guaranteed not to have holes?

Being a ubiquitous operation in many applications, and since the Minkowski sum of two polygons with a total of n vertices can have very high (quartic) complexity, it is essential to be efficient in computing the sum. In particular, we are in search of an output-sensitive algorithm. Erickson's result [48] implies that we will probably not be able to find such an algorithm with less than an $\Omega(n^3)$ running-time overhead. Thus we seek alternative ways to speed up this computation. Baram, Fogel, Halperin, Hemmer and Morr [21] studied a problem on holes in the summands of the Minkowski sum: They showed that if the summands of the Minkowski sum operation are polygons possibly with holes, then there are certain easy-to-apply rules, which will always eliminate the holes from at least one of the summands, without affecting the resulting Minkowski sum. The elimination of holes makes the construction of the Minkowski sum when some holes are "filled" more efficient in practice; see experiments in [21, Section 4]. Can we find other rules that are efficient to apply and will make the computation of the Minkowski sum faster?

Minkowski sums raise a wealth of absorbing (open) problems in combinatorics and computation. We conclude this section with the issue of inverse problems in Minkowski sums or *Minkowski sum deconstruction*: Given a set $M \subseteq \mathbb{R}^d$, is M the Minkowski sum (or its

approximation) of two non-trivial[2] summands in $\mathbb{R}^d$? For motivation and examples, see [25].

3 Interpolate between easy and hard instances of multi-robot motion planning

Multi-robot motion planning (MRMP, for short) is one of the most intriguing and challenging problems, both from the robotics perspective as well as from the computational geometry perspective. The recent years have seen an increasing global trend to use fleets of robots for logistics, inspection and surveillance tasks, agriculture, maintenance and repair, and more. New and demanding applications often require sophisticated, ultra fast coordination. At the same time, multi-robot motion planning has been proved to be *computationally hard*[3] [70, 124, 125].

MRMP is also highly varied, and often means different things to different people. For example, there is a vast body of work on the discrete case of graphs—Multi-Agent Path Finding (MAPF) [53, 134], or swarm solutions, which often follow a distributed approach [119]. Here we deal with a basic continuous variant, where there is a central control that knows the environment and the exact location of each robot at all times.

Even this fundamental variant is far from being completely understood. While the problem is in general hard, various scenarios are relatively easily solvable. We focus on a specific MRMP problem, where we know to distinguish between easy and hard instances, and for which we have an efficient solution tailored for easier instances. The setting is as follows. There are m unit disc robots moving in a planar workspace, which is a single simple polygon with n edges. The robots are located at their given start positions, and we are also given m target positions. The problem is *unlabeled* in the sense that we do not care which robot goes to which target position, as long as at the end of the motion each target position is occupied by one robot. We specify each start or target position by the coordinates of the center of the robot at that position. We impose one extra condition: We require that between every pair of start or target position there is a distance of at least 4 units. For this setting there is always a solution and there is an algorithm with running time $O(n \log n + mn + m^2)$ to find such a solution [10]; see Figure 3 (left) for an example of an instance. This result has later been sharpened in allowing tighter separation conditions between start and target positions, while still always having a solution [20].

If we do not assume the extra spacing condition then the problem is presumably PSPACE-hard, even though we are in the unlabeled setting. This was proved for unit squares in polygons with holes [124], and for disc robots of two different sizes [33]. More recently it has also been shown for unit squares inside a simple polygon and unit discs in a polygon with holes [4].

The question we pose is to devise an algorithm that smoothly interpolates between 'easy' instances as above and hard instances where such separation is not assumed. We introduce another parameter to the problem, ε, which measures how much above 2 is the separation between any pair of start and/or target positions; see Figure 3 (right). We assume that $\varepsilon > 0$, namely that robots placed at the pair of positions that achieve the minimum distance are not osculating.

---

[2] By non-trivial we preclude the case where one summand is a point.

[3] For a general review of the computational complexity of motion planning, see, e.g., the survey by Solovey [123].

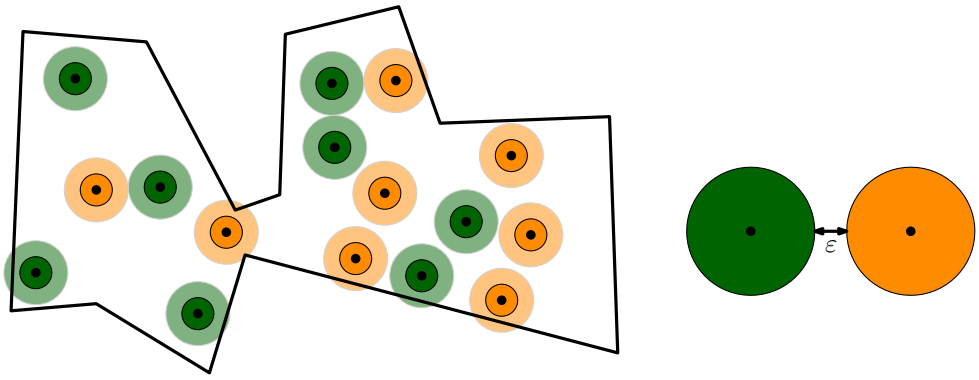


Figure 3 Left: The setting solved in [10]: The dark green and orange discs are the start and target positions, respectively. The light colored larger discs of radius 2 show that the distance between any pair of start and/or target positions is at least 4. Right: We propose to study the situation where the distance is at least $2 + \varepsilon$ for an arbitrary value $\varepsilon > 0$.

Problem UnlabeledMultiDiscInterpolate: A collection of m unlabeled unit discs are moving inside a simple polygon with n edges. We are given *valid* start and target positions for the m discs, namely the robots at their start (respectively target) positions are pairwise interior-disjoint. Let $d := 2 + \varepsilon$ denote the minimal distance between the start and target positions of the robots for a value $\varepsilon > 0$. Devise an algorithm to find a collision free motion plan for the robots from start positions to target positions with running time that is polynomial in m, n, and $1/\varepsilon$, or decide that it does not exist.

In this context we mention the results of Alt, Fleischer, Kaufmann, Mehlhorn, Näher, Schirra and Uhrig [18] on motion planning for a rectangular robot translating and rotating among polygonal obstacles in the plane. This work stands out in the motion planning literature in computational geometry in that it introduced the *tightness* of the workspace as a measure of the difficulty of the problem. It also analyses the running time of the algorithm both as a parameter of the complexity of the obstacles as well as of this tightness, which is similar in this respect to the problem we have just posed (and different from it in various other ways).

Finally, we mention several recent works by Geft et al., which aim to delineate the boundary between easy and hard problems in MAPF and MRMP, typically by considering special properties (for example, monotonicity—see Section 4) or certain extra parameters, beyond separation, that arise in these problems [5, 55, 56, 57].

4 Geometric reconfiguration and object rearrangement

In a *valid configuration* of a set of objects in two- or three-dimensional space, each object has a fixed pose (i.e., position and orientation) and the objects are pairwise interior-disjoint. In *object reconfiguration* we are given two valid configurations of the same set of objects, a start configuration and a target configuration, and the goal is to plan collision-free moves that will take the objects from start to target, or determine that no such reconfiguration is possible. Naturally, numerous tasks in robotics and automation can be cast as object reconfiguration, including packaging, product assembly, product maintenance, shelf stocking, to mention just a few.



Figure 4 The lifting model demonstrated during the festival *Copenhagen Distortion*, 2024. Picture by Hao Wu.

The family of reconfiguration problems is broad and has an abundance of variants and occurrences. We distinguish between two main categories in this family: (i) *basic* object reconfiguration, where the objects are "free-flying", and (ii) *manipulated* object reconfiguration, where the objects are moved from start to target by robots. The basic category is relevant when the objects have autonomy of motion, when they can be manipulated without direct contact (e.g., using force fields [29]), or as a preprocessing step to certain solutions of the typically much harder manipulated object reconfiguration. The problems of this basic type have been researched for decades in mathematics, computer science, and various engineering disciplines [42, 44]. Interest in problems of the manipulated type is on the rise with the increasing usage of robots in ever more complex task; see, e.g., [72, 101, 121, 126].

How is object reconfiguration different from multi-robot motion planning? These are two largely different families of problems, but they are similar when we discuss free-flying object reconfiguration with complex allowable motions. The term *reconfiguration* has been used in the context of planning the motion of many moving bodies, when the type of motion allowed is very simple [44]; these are sometimes referred to as coin-moving puzzles [42]. Reconfiguration algorithms often aim to optimize the overall effort exerted in moving the objects from one configuration to another, typically in the absence of obstacles and using simple allowable motions. It is reminiscent of shape formation problems, which have been studied in robotics [17, 75]. Other objectives have likewise been considered, such as *space-aware* reconfiguration, where the goal is to minimize the physical space needed for the reconfiguration to be possible [68], or makespan as defined in Section 1 (see, e.g., [43]).

Many restrictions on the motions can be considered. Three natural models, for objects in the plane, are (i) the *lifting* model, where each move consists of lifting an object (in the third dimension, say) and placing it at a free position (see Figure 4), (ii) the *sliding* model, where the objects can slide along arbitrary continuous curves, and (iii) the *translating* model, in which the objects can move along straight line segments. Dumitrescu and Jiang [44] showed that for unlabeled unit discs, it is NP-hard to find the minimum number of moves (with one object moving at a time) to reconfigure from the start positions to the target positions in

the sliding and translating models. Another restriction often studied in reconfiguration is *monotonicity*, where each object is allowed to move exactly once, from its start to a target position.

There is a simple greedy algorithm for finding a suitable sequence of moves if it exists in the *monotone lifting model*: We say that a target is *finished* if it coincides with a disc. A target is *blocked* if it is not finished and there is a disc overlapping it partially. A target is *available* if it is neither finished, nor blocked. In *monotone*[4] (or single-move) reconfiguration, discs are moved one after the other, and in each move a disc reaches its destination where it stays put thereafter. Note that we can only move a disc to an available target. If all targets are finished, we are done. Otherwise, we give up if there is no available target. If there is an available target, we find the blocked target that is blocked by a minimum number of discs and move one of them to an available target. This makes sure that we minimize the number of moves before we get one more available target.

The situation becomes more complicated and interesting when we consider the sliding or translating model, which leads us to the following questions.

> **Problem UnlabeledMonotoneReconfiguration:** Consider the following question: Given two sets S and T, each consisting of n pairwise interior-disjoint unit discs in the plane (but discs in S can overlap discs in T), is there a monotone reconfiguration from S to T using sliding moves? Can the problem be solved in polynomial time?

As a warm-up, one might consider the case of 2×2 squares placed on the integer grid, possibly even restricting the motions to horizontal and vertical translating moves. Even in this case, the problem does not seem to be trivial.

In *manipulated object reconfiguration*, where we also take into account the robots that move the objects, further complications arise. Already for one robot, we need to make sure that the robot does not collide with the objects that are not expected to move at a certain stage of the process. When two or more robots simultaneously reconfigure the objects, we are faced also with multi-robot motion coordination (see, e.g., [121]). Then, we may additionally have the flexibility to choose preferred placement of the bases of the manipulating robots, an optimization that could significantly simplify or speed up the task. Due to the complexity of each of these sub-problems they are often tackled in separation, which may result in far from optimal solutions. While taking them all into account simultaneously is in general prohibitively difficult, it would be interesting to discover settings where this is feasible and can lead to a complete optimized solution for the "full manipulated-reconfiguration stack."

5 Interlocked objects with fewer convex parts

Assembly planning is a large sub-area of robotics where the aim is to move a set of objects from initial positions to a desired target configuration without causing the objects to collide. It is often conceptually easier to study the equivalent reverse question, namely whether objects that start in a specific configuration, typically as a dense cluster, can be moved apart using prescribed types of motions.

Since assembly in practice is often performed by adding one object after the other, a variant of particular importance is whether we can move one object away from the rest at a time. If the objects are not convex, it is easy to come up with examples of two objects

[4] This usage of the term *monotone reconfiguration* as it appears in this section and in Section 6 is borrowed from assembly planning [64] and has been adapted to reconfiguration (see, e.g., [56, 130]).

that are *locked* into each other, so that they cannot be moved apart. This is even the case in the plane, where the objects can for instance be modelled as polygons. However, if we are dealing with *convex* objects in the plane, there always exists an object P that is not blocked from, say, above (namely in the y direction) by another object [62], so P can be moved vertically up without collisions, and this process can be repeated until all objects have been moved away from each other.

This is not the case in the three-dimensional space: Three rectangles can block each other cyclically so that none of them can be translated vertically up.[5] Snoeyink and Stolfi [122] showed that in any configuration of at most five convex objects in space, there is a direction $\vec{d}$, so that an object can be translated away from the others in direction $\vec{d}$. They also gave an example of six objects so that there is no such direction. Even stronger, it is not possible to partition the objects into two groups, where one group can be moved away from the other by translation. In other words, the objects cannot be taken apart by two *hands*, if these hands can only translate and not rotate the objects. However, if arbitrary rigid motions are allowed, that is, the objects can also be rotated during the movement, then two hands suffice. Snoeyink and Stolfi then described a configuration of 30 convex objects that cannot be moved away from each other by two hands with arbitrary rigid motions allowed. See Figure 5. The configuration is completely locked: There exists no subset that can be moved even infinitesimally without causing a collision with the remaining objects.

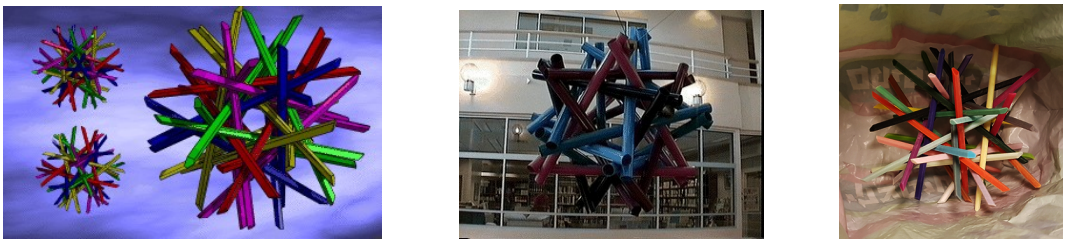


Figure 5 The Snoeyink-Stolfi construction of 30 convex parts that cannot be taken apart with two hands. Left: An illustration of the actual construction. Middle: A sculpture at UBC inspired by the construction. The left and middle figures are taken from Snoeyink's website `https://www.cs.unc.edu/~snoeyink/`. Right: The construction printed on a plaster-based 3D printer at Tel Aviv University.

Additional interesting inseparability results of this type have been shown over the years; see, e.g., the review by Toussaint[6] [127].

Using a hand for each object, it is possible to disassemble any configuration of convex objects $P_1, \ldots, P_n$: Choose a reference point p_i in each object P_i, and translate P_i by the velocity vector $p_i - p_1$. This corresponds to scaling up the initial configuration and then scaling down each object P_i to its original size, keeping the reference point p_i fixed. Hence, the movement does not cause collisions (this in fact works for any collection of star-shaped objects, where the reference points should be chosen from the respective kernels of the objects). Let us conclude this section by suggesting some interesting questions for future research.

[5] This is a classical example in computer graphics to a failure of the "painter's algorithm"; see, e.g., `https://en.wikipedia.org/wiki/Painter's_algorithm`.

[6] Godfried Toussaint, who passed away in 2019, presented a more modern review of similar problems in the International Conference on Mathematics and Mechanics in Nice in 2017. To obtain the accompanying mini-survey that he wrote for the occasion, contact the second author at `danha@tauex.tau.ac.il`.

Problem 3DInterlockedFewerThan30: Does there exist a configuration with fewer than 30 convex objects that cannot be taken apart by two hands?

How many hands may be needed to move apart n convex objects, as a function of n? All we know is that 2 do not always suffice and n are always sufficient. For a given configuration of objects, is there an algorithm for deciding how many hands are needed and describing how to move the objects apart?

To show that their construction of 30 objects cannot be taken apart with two hands, Snoeyink and Stolfi considered an exponential number of decompositions of the objects into two proper subsets, and for each such decomposition applied linear programming to test whether one subset can be moved infinitesimally from the other [122]. Later, Guibas, Halperin, Hirukawa, Latombe and Wilson [60] gave a polynomial time algorithm for the same problem, which runs in time that is roughly $O(n^{12})$, where n is the complexity of the polyhedral parts.[7] Can one devise a more efficient algorithm for the task? This seems highly plausible by employing sophisticated data structures.

We remark that in assembly planning one often has to take into consideration additional requirements, beyond avoiding collision between parts. For example, a practical requirement is that the two sub-assemblies, held by the two hands, will each constitute a connected object. Interestingly, the connectivity requirement makes the problem computationally hard even in very simple settings [11].

6 Minimal design modification, or explainability

In everyday life as well as in many industrial settings, it is often the case that it is impossible to perform some desired task because of unfavorable properties of the scenario. We then seek to change the setting just enough to be able to carry out the task. Another view of the same problem is, our algorithm fails to find a feasible plan for the task, and we would like to give the user an explanation (or an excuse, as it is called in [58]), why the algorithm failed; in such a case a compact or small explanation is desirable [69]. We use the term *minimum design modification* as an umbrella term for problems belonging to this category, where the modification is geometric such as "slightly modify the pose of an object in the input", "remove an obstacle", or "change the dimensions of an object". A minimum design modification problem should consist of (i) a desired task/plan, (ii) a set of allowable modifications, and (iii) an objective function.

As introduced in Section 5, assembly planning is concerned with designing a sequence of motions that will bring a set of separate parts into their desired juxtaposition in a final product. In general the problem is hard and we add constraints to make it tractable. One such constraint is the number of hands used to move the pieces (see Sections 5 and 7), and another one is the allowable motions of the parts. We assume here that the assembly must be carried out by two hands and by translating the parts. Let us again explain what this means in the context of *disassembling* rather than assembling the parts: We wish to partition the parts forming the final product into two groups so that one group can be moved, as a single rigid object, arbitrarily far away by translating them together along a single line segment, without colliding with the other group. This process can be repeated recursively on the two groups so that in the end, the assembly has been split into the

[7] The actual running time is governed by parameters related to the kinematic analysis of the contacts between the parts under infinitesimal rigid motion; see [60] for details.

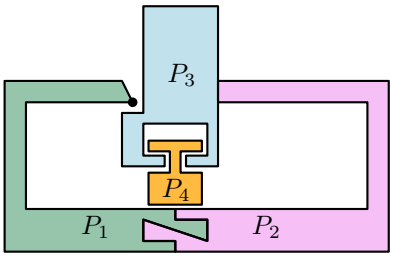


Figure 6 This configuration cannot be disassembled with two hands, but moving the marked vertex of P_1 a bit to the left, the polygons P_3 and P_4 can be moved up and away from P_1 and P_2.

individual parts. For polyhedral parts, it can be decided efficiently whether an assembly can be disassembled like this [52, 64, 133]. For simplicity of exposition, we restrict ourselves to two-dimensional assemblies of polygons. Now assume that we are given an assembly that cannot be disassembled with these constraints, as for instance is shown in Figure 6. We are also given a tolerance parameter $\varepsilon > 0$, meaning that we are allowed to move each vertex by a distance of at most ε, as long as we avoid self-intersections. For a more elaborate model of toleranced modification, see, e.g., [90].

As our first problem, we just want the assembly to be partitioned into two parts that can be disassembled from each other:

> **Problem MDM2DPartition**: We are given a two-dimensional assembly A made of polygons and a tolerance parameter $\varepsilon > 0$. What is the minimum number of vertices in the polygons that need to be moved by a distance of at most ε such that the modified assembly A' can be partitioned into two parts that can be moved arbitrarily far away from each other by two hands using a single translation? Devise an efficient algorithm to answer this question.

We also ask the following more general question, where we want to disassemble the assembly entirely.

> **Problem MDM2DAssembly**: As in the previous problem, but we want to be able to repeat the process so that all parts are moved away from each other.

An interesting variant of **Problems MDM2DPartition** and **MDM2DAssembly** is to find the smallest value of ε such that the assembly can be partitioned or assembled as above, namely by moving each vertex distance at most ε.

We now turn to a different manufacturing process. Casting is a process where liquid material is poured into a mold having the shape of a desired product. After the material solidifies, the product is removed from the mold. We want to remove our produced object from the mold without collision. The casting problem is closely related to assembly planning; indeed, we need to make sure that the mold and the cast object can be disassembled from each other.

To model the casting problem more formally, we assume that the object P to be cast is a polyhedron in $\mathbb{R}^3$ with a boundary consisting of n polygonal facets. The mold is box-shaped and has a cavity whose shape is exactly that of P, such that one of the facets of P is the top facet of the cavity. See Figure 7 for an illustration in 2D. We wish to move P away from the mold by translation in a single direction. A direction that does not cause collision is called a *valid removal direction*.

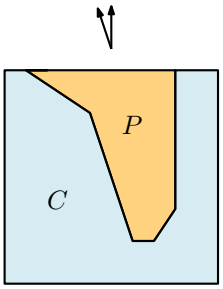


Figure 7 The object P can be moved away from the cast C by translation. The arrows show the range of valid removal directions.

Bose, Fogel, Geft, Halperin and Shamai [30, 31] showed that one can decide for each facet of P whether P is castable with that facet used as the top facet, for a total time of $O(n)$ for all the facets. They then showed how to compute the set of all valid removal directions for all top facets in $O(n \log n)$ time and that this is asymptotically optimal. Interestingly, they also showed that there can be at most six top facets for which the set of valid removal directions is non-empty. These results improved earlier work on the casting problem [15, 19].

When no facet can be used as a top facet, we may want to modify P as little as possible to another object P' that *is* castable, i.e., we can choose a top facet that has a valid removal direction. This leads to our next problem, which is a special case of **MDM2DPartition**, but in three dimensions rather than two.

> **Problem MDMCasting**: We are given a three-dimensional polyhedron P and a tolerance parameter $\varepsilon > 0$. What is the minimum number of vertices that need to be moved by a distance of at most ε such that the modified polyhedron P' is castable? Devise an efficient algorithm to answer this question.

Minimal design modification naturally arises in *reconfiguration problems* (see Section 4). Let us again consider a setting where we wish to get from one configuration of unit disc robots to another (we refer to the start and target placements collectively as *endpoint* placements). It might be impossible because any way of moving the discs to their final positions would cause the discs to collide. In the remainder of this section we focus on *monotone* reconfiguration; recall that in monotone reconfiguration discs are moved one after the other, and in one move a disc reaches its destination where it stays thereafter. Also we assume that the scenes are obstacle free. One can clearly consider generalizations of these premises. To make an instance solvable a possible modification is to move some endpoint placements by $\varepsilon > 0$ from their original setting.

> **Problem MDM2DReconfiguration**: Given an infeasible instance of unit disc reconfiguration (labeled or unlabeled), what is the minimum value $\varepsilon > 0$ so that if all endpoint placements are allowed to move by ε, we arrive at a feasible instance? Devise an efficient algorithm to answer this question.

We can also ask for the minimum *number* of endpoint placements that should be moved by a given value ε to reach a feasible configuration.

Alternatively, we can ask for the minimum number of robots whose radii should be shrunk by a value at most some fixed $\varepsilon > 0$ so that we arrive at a feasible instance. An answer to this question can be interpreted as an explanation for why the original instance is not feasible.

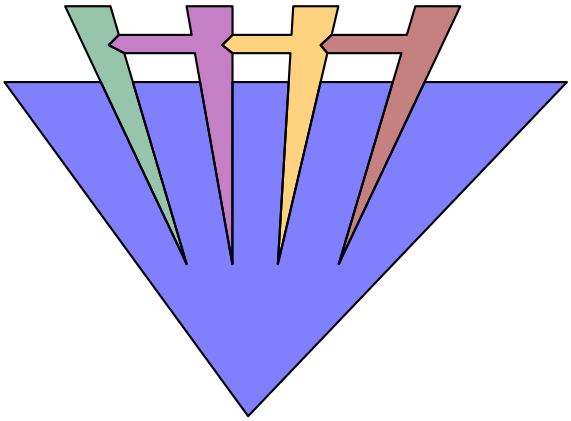


Figure 8 A 5-handed assembly, following an example from [102].

The minimum radius variant of this question could be relevant when the discs' radii had been uniformly enlarged to account for uncertainty and safety issues. This enlargement resulted in an infeasible instance, and we wish to find a smaller enlargement (but the largest possible) that would make the instance feasible. This could be phrased as a maximal enlargement problem, but to keep with the spirit of minimal design modification we phrase it as "what is the minimum value $\varepsilon > 0$ so that if all discs shrink by ε (keeping their center points), we arrive at a feasible instance?" Note that if moving discs by ε leads to a feasible configuration, then shrinking by ε also does, whereas the opposite implication is not true in general.

Another minimum design modification problem that has received much attention is the problem of removing connected obstacles in the plane, so as to establish connectivity between two points; see for instance [86, 88, 87, 46].

7 Assembly planning with more than two hands

We have already discussed the problem of *assembly partitioning* in previous sections. For completeness, let's review it one more time: We are given a collection A of pairwise interior-disjoint objects in two- or three-dimensional space, together with an allowable set of motions, and we ask whether there is a proper subset $S \subset A$ that can be moved away as a rigid body from the rest of the objects, $A \setminus S$, without colliding with the objects in $A \setminus S$. If so, we wish to output the subset S and a specific collision-avoiding motion (from the allowable set). In Section 5/problem **3DInterlockedFewerThan30**, the allowable motion are infinitesimal translations and rotations in 3-space, and in Section 6/problem **MDM2DAssembly**, the allowable motions are infinite translations in the plane.

Both these partitioning problems are instances of 2-handed partitioning, where we split the assembly into *two* subsets, and one of them needs to move away from the other. For certain partitioning problem two hands may not suffice: Consider for example Figure 8. If we only allow infinite translations in the plane then we need to partition the assembly into five subsets, one stationary and each of the other moving along its distinct individual direction, and we refer to this assembly as 5-handed.

The most prevalent industrial assembly operations are 2-handed. Furthermore, there is a general framework [64, 133] to solve assembly partitioning for constant-description allowable motion. The idea is to study the problem in the so-called *motion space*, whose dimension

equals the number of degrees of freedom of the path, where every point represents a possible path in the family of allowable motions. This space is then overlaid with an arrangement [66] of certain critical hyper-surfaces, such that inside each cell of this arrangement all the paths corresponding to points in the cell induce the same valid partitions.

We say that an assembly A is *k-handed*, for $k > 2$, if it cannot be partitioned into $k-1$ or fewer subsets that are separable as above, but it can be so partitioned with k hands.

We look for an extension of the motion-space framework to $k > 2$-handed assembly partitioning. It would already be interesting to solve this for 3 hands for infinite translations in the plane.

> **Problem 3HandedPartition**: Given an assembly A of polygons in the plane, which we assume not to be 2-handed, devise an efficient algorithm to find two proper subsets $A_1, A_2 \subset A$, such that $A \setminus (A_1 \cup A_2)$ is non-empty, together with two vectors d_1, d_2, such that A_1 and A_2 can be moved simultaneously: A_1 along the direction d_1 at speed $|d_1|$ and A_2 along direction d_2 at speed $|d_2|$, such that throughout the motion the three sets A_1, A_2 and $A \setminus (A_1 \cup A_2)$ are pairwise interior-disjoint. Otherwise, report that no such motion exists. Devise an efficient algorithm to answer this question.

Naturally, we are interested in extensions of **3HandedPartition** not only to $k > 3$ as mentioned, but also to more complex settings of objects and allowable motions.

8 Compact representation of spatial objects

Polyhedra are three-dimensional objects, and they are used in robotics to model the actual robots and obstacles in the environment. They are likewise used in several other practical domains such as computer graphics, engineering and simulation to model all kinds of spatial objects. Mathematically, we can define a polyhedron as a compact region in $\mathbb{R}^3$ bounded by a finite number of (two-dimensional) polygons. There are two main types of representations used to model a polyhedron P. The perhaps simplest and most frequently used is to just specify the surface of P as a collection of triangles. This is known as a *surface* model. Knowing the surface, we also implicitly know the volume inside P.

In some cases, however, it is convenient to have an explicit representation of the interior of P and not only the surface. To this end, we can use a *tetrahedralization* $\mathcal{T}$ of P. Here, $\mathcal{T}$ is a collection of tetrahedra whose union is P, and any two tetrahedra in $\mathcal{T}$ are either disjoint or intersect in a face of both of dimension at most 2 (i.e., the intersection is a vertex, an edge or a (triangular) facet of both tetrahedra). This is a so-called *volumetric* model, since the tetrahedra together constitute the entire volume of P.

Tetrahedralizations are also called *triangulations*, and they are indeed three-dimensional analogous to triangulations of two-dimensional polygons. A surface model of a polyhedron corresponds to representing a polygon just by its boundary edges, whereas a volumetric model (using a tetrahedralization) corresponds to representing a polygon by a triangulation. However, whereas triangulations of polygons are very well understood, many mysteries surround their three-dimensional counterparts.

A fundamental task is to convert a model of one type into the other. Obtaining a surface model from a volumetric model is quite straight-forward: We just need to identify the facets of tetrahedra where there is no tetrahedron on the other side. Such facets must necessarily belong to the surface of P, so they together constitute a surface model of P. However, obtaining a volumetric model from a surface model is far more challenging and some prominent research questions remain to be answered in this direction.

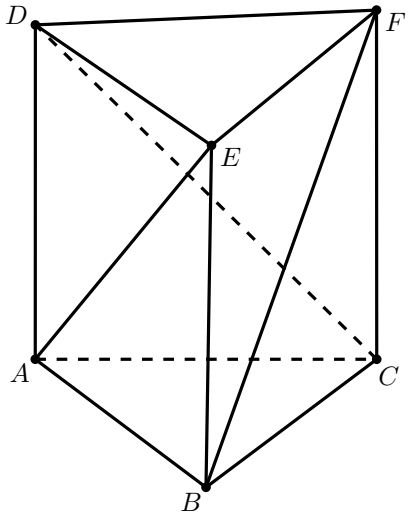


Figure 9 The Schönhardt polyhedron P: Almost a prism, except that the triangle DEF is slightly rotated compared to ABC, forming concave edges AE, BF and CD. No tetrahedron spanned by four of the corners is contained in P, so P cannot be tetrahedralized without adding a Steiner point.

An important property of a representation is whether or not it uses so-called Steiner points. A *Steiner point* is a vertex of a triangle/tetrahedron that is not a vertex of the object P. Any simple polygon with n vertices can be triangulated without Steiner points, and all such triangulations consist of $n-2$ triangles. Any *convex* polyhedron with n vertices can be tetrahedralized without Steiner points, but different tetrahedralizations may have different numbers of tetrahedra, and the number can vary from $\Theta(n)$ to $\Theta(n^2)$ [26].

A *non-convex* polyhedron may not have a tetrahedralization without adding Steiner points. A well-known example where a Steiner point is necessary is the *Schönhardt polyhedron* [115]; see Figure 9. It is known that $\Theta(n^2)$ Steiner points and tetrahedra are sufficient and sometimes necessary [35]. Deciding whether or not Steiner points are needed is NP-hard [24].

In order to obtain a compact volumetric model of a polyhedron, we would like to minimize the number of tetrahedra. Somewhat surprisingly, adding a Steiner point may allow for fewer tetrahedra even when tetrahedralizing a convex polyhedron [22]. Finding the minimum number of tetrahedra without using Steiner points is NP-hard for convex polyhedra [23]. Chin, Fung, and Wang [40] found a $2-\Omega(1/\sqrt{n})$-approximation algorithm for this problem, which is optimal unless P=NP. However, no algorithm is known of non-convex polyhedra. This leads to our first question, which is also raised in [26, 27].

> **Problem CompactTetrahedralization**: Given a polyhedron with n vertices, can a tetrahedralization with Steiner points while using at most $O(\mathrm{OPT})$ tetrahedra be found in time polynomial in n? Here OPT is the minimum number of tetrahedra in any tetrahedralization.

In *collision detection*, it is crucial to have a compact representation of the involved objects. A common method to check whether three-dimensional objects intersect is to use *bounding volume hierarchies* [84, 80]. A bounding volume hierarchy of an object P is a rooted tree T where each node u corresponds to a subset $u(P)$ of P. Here we will consider binary trees only, but trees of higher degrees are also often used [114]. The root r of T corresponds to all of P, i.e., $r(P)=P$. For a vertex u of T with children v_1, v_2, the subsets $v_1(P), v_2(P)$ is a partition of $u(P)$. The leaves of T correspond to the geometric primitives of P, such

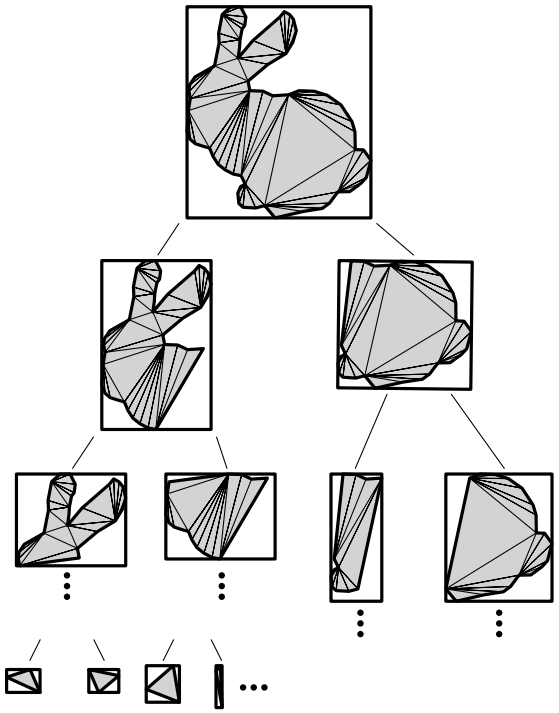


Figure 10 Part of a bounding volume hierarchy of a triangulation of the silhouette of the Stanford Bunny. The two-dimensional variant of problem **BoundingVolumeHierarchy** is to minimize the total area of the bounding boxes.

as the triangles or tetrahedra defining P using surface or volumetric models, respectively. Several heuristics have been proposed for generating bounding volume hierarchies, including grouping the primitives according to when they are hit by a Morton space-filling curve [114] or splitting the nodes recursively according to some heuristic [59, 84]. However, to the best of our knowledge, no methods have been described that find a hierarchy that is provably (close to) optimal according to any natural measure of optimality. It is not clear what objective we should aim to optimize, but a good candidate is the total volume of the axis-parallel bounding boxes of all nodes. The experiments of [84] suggest that this measure gives fast collision detection queries in practice: In the paper, the splitting heuristic leading to the fastest queries is the one where we minimize the sum of the volumes of the two children. This results in a hierarchy that is *locally* optimal, but there may be other hierarchies with a smaller total sum over all nodes. This leads to the next problem.

Problem BoundingVolumeHierarchy: Given a polyhedron P and a tetrahedralization $T = T(P)$, can an optimal bounding volume hierarchy, whose leaves are the tetrahedra of T, be found in polynomial time, minimizing the sum of the bounding box volumes over all nodes?

The problem is also interesting and apparently unexplored in the two-dimensional case; see Figure 10 for an example.

9 Partitioning large objects for manufacturing

Consider a manufacturing process where we want to produce a large object P, but we are only able to produce parts of bounded size. We then want to partition P into parts that are sufficiently small to be fabricated, and in the end these can be assembled to form the desired object P. To keep both fabrication and assembly as simple as possible, we naturally want as few parts as possible. This problem for instance appears in 3D printing, where we can only print parts that fit in the printing volume, which typically has the shape of a box. (For simplicity, we don't care whether the parts can in fact be assembled to form P without colliding—considerations of the type discussed above in Sections 5 and 6.) Heuristics have been proposed to solve this problem [38, 76, 95], and several videos can be found on YouTube showing how it can be done manually using various software (for instance, search for "splitting large part for 3D printing"), but no algorithm with provable guarantees on the number of resulting parts is known. Abrahamsen and Stade [9] studied a two-dimensional version of this problem where the input is a simple polygon P with n corners. They proved that it is NP-hard to partition P into a minimum number of pieces each of which is contained in an axis-parallel unit square. Abrahamsen and Rasmussen [8] gave constant-factor approximation algorithms for this and several related problems (for alternative definitions of smallness). This was improved by Aamand, Abrahamsen, Browne, Goswami, Kasthurirangan, Kleist, Mitchell, Polishchuk and Stade [1], who gave a PTAS that also works if the polygon has holes. They likewise showed that there is no $o(\log n)$-approximation algorithm (unless P=NP) for the three-dimensional analogue of the problem: partitioning a simple polyhedron with n corners into pieces each of which is contained in an axis-parallel unit cube. There is, however, no matching algorithm known. This leads us to the next problem.

> **Problem PartitioningPolyhedra**: Given a simple polyhedron P in $\mathbb{R}^3$ with n corners, is there an algorithm with running time polynomial in n and OPT for partitioning P into $O(\mathrm{OPT} \log n)$ pieces, each of which is contained in an axis-parallel unit cube? Here, OPT denotes the minimum number of pieces in any such partition.

In 3D printing, it is often advantageous to print several parts simultaneously, if the parts can be packed into the printing volume. Compared to printing the parts individually, in central industrial 3D printing methodologies, doing many at once is much more economical [135]. Let us again consider a two-dimension setting where we wish to "print" a polygon P, and we can produce any set of polygons that can be packed into the unit square. If P has area exactly 1, then the Wallace-Bolyai-Gerwien theorem[8] says that it is possible to partition P into finitely many triangles that can be packed into the unit square. Here, the triangles are allowed to be rotated. It follows that if the area of P is at most 1, it is always possible to partition P into finitely many polygons that can be packed into the unit square. Again, we naturally want to minimize the number of pieces, and we arrive at the following problem; see Figure 11 for an example.

> **Problem PartitioningForPacking**: Given a polygon P with n corners and area at most 1, is there an algorithm with running time polynomial in n and OPT for partitioning P into $O(\mathrm{OPT})$ pieces, so that these pieces can be simultaneously packed into the unit square? Here, OPT denotes the minimum number of pieces in any such partition. We can allow the pieces to be rotated arbitrarily or allow translations only.

[8] See, e.g., the Wikipedia page "Wallace-Bolyai-Gerwien theorem."

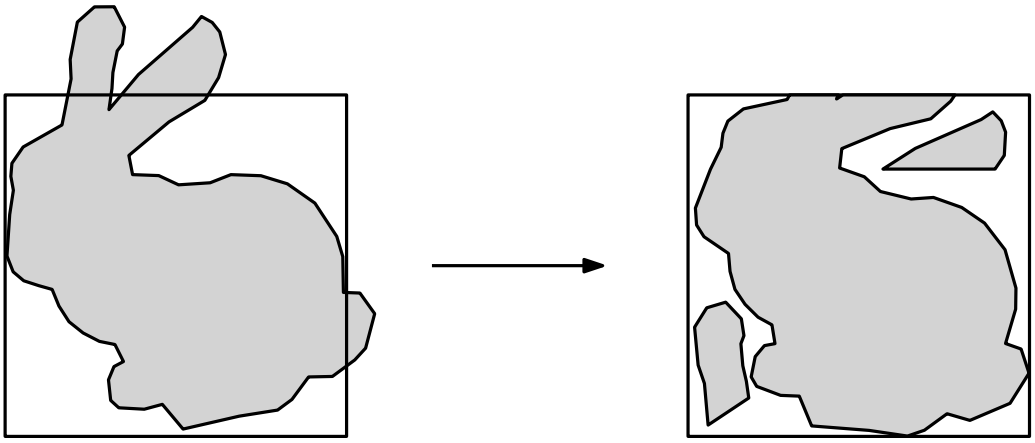


Figure 11 The silhouette of the Stanford Bunny can be partitioned into three pieces that can be packed in the unit square after suitable rotations.

The problem has been studied from a practical perspective [37]. Abrahamsen, Miltzow and Seiferth [7] proved that many packing problems are $\exists\mathbb{R}$-complete and thus likely not in NP. This for instance holds for the problem where we are given a set of convex polygons and want to know if they fit in a unit square. We expect that the problem **PartitioningForPacking** is also $\exists\mathbb{R}$-hard to solve optimally for similar reasons, but will not attempt to prove it here.

Max Dehn proved in 1910 that the Wallace-Bolyai-Gerwien theorem does not hold in three dimensions [137]: He proved that the regular tetrahedron with volume 1 cannot be partitioned into finitely many tetrahedra that fit in the unit cube. However, if a polyhedron P has volume at most $1-\varepsilon$ for a constant $\varepsilon > 0$, then it is not hard to see that P can be partitioned into finitely many pieces that can be packed into the unit cube: We can partition P using a regular grid of voxels of size $\delta \times \delta \times \delta$. This results in some *complete* pieces of P that are cubes of size $\delta \times \delta \times \delta$ and some *incomplete* pieces that are only subsets of such cubes. Let us observe that there are at most $O(1/\delta^2)$ incomplete pieces: Each facet of P intersects $O(1/\delta^2)$ voxels and each of these intersections corresponds to a facet of an incomplete piece. As there are $O(1)$ facets of P, it follows that there are $O(1/\delta^2)$ incomplete pieces. Choosing $\delta = 1/k$ for a sufficiently large integer k, all the complete pieces fit in a box of size $1 \times 1 \times (1-\varepsilon)$ in the bottom of the unit cube, since the volume of P is also $1-\varepsilon$. The remaining available space of the unit cube is a box of size $1 \times 1 \times \varepsilon$ with volume $\varepsilon = \Omega(1)$, so it can accommodate $\Omega(1/\delta^3)$ incomplete pieces and there is room for our $O(1/\delta^2)$ incomplete pieces when δ is small enough. We can thus also consider **PartitioningForPacking** in this three-dimensional setting with a bit of extra space in the unit cube, but it seems very difficult.

See also the surveys [32, 74] on algorithmic problems in manufacturing and the paper [51] on another decomposition problem related to manufacturing.

10 Convex covering of free configuration space

In many path planning scenarios, the free portion P of the configuration space is difficult to handle directly. Describing P as a union of convex sets has shown to be a versatile technique, simplifying many such problems. That is, we want to find a family $\mathcal{C}$ of convex sets such

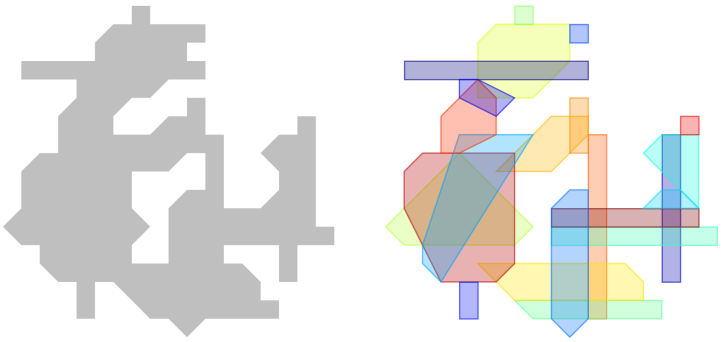


Figure 12 A polygon (left) and a convex cover of the polygon (right), found with the method from [6].

that[9] $P = \bigcup_{C \in \mathcal{C}} C$. As a simple example, consider the intersection graph G induced by $\mathcal{C}$, namely, the vertices of G are the sets $\mathcal{C}$ and there is an edge in G between two sets C_1 and C_2 if $C_1 \cap C_2 \neq \emptyset$. In order to find a path between two points $p_1, p_2 \in P$, we first find sets $u_1, u_2 \in \mathcal{C}$ where $p_1 \in u_1$ and $p_2 \in u_2$. We then find a path $\pi = v_1 v_2 \cdots v_k$ in G, where $v_1 = u_1$ and $v_k = u_2$. Letting $q_i \in u_i \cap u_{i+1}$, we then get a path $p_1 q_1 q_2 \cdots q_{k-1} p_2$ from p_1 to p_2 in P, using the convexity of our sets.

Ideally, we want $\mathcal{C}$ to contain as few sets as possible, in which case finding $\mathcal{C}$ is known as the problem *Minimum Convex Cover*. This problem has a long history in Computational Geometry and is motivated by several other practical settings as well. For instance, Garey and Johnson [54, p. 232] mentioned a very restricted version of the same problem: describing an orthogonal polygon as a union of a minimum number of axis-parallel rectangles. They state that the problem is NP-hard and call the problem *Rectilinear Picture Compression*, because such a set of rectangles is often a compact way to represent a set of pixels in an image.

Marcucci et al. [97, 98] demonstrated that convex covers of the configuration space can be used to accomplish complex tasks in motion planning. Motivated by this work, a two-dimensional version, where P is a polygon with holes, was chosen for the 2023 edition of the challenge *Computational Geometry: Solving Hard Optimization Problems* [50]. The challenge was won by Abrahamsen, Meyling and Nusser [6]; see Figure 12. They took a three-step approach, where they first triangulated P appropriately. Then they computed a visibility graph on the triangulation, where there is an edge between two triangles when any point in one triangle can see any point in the other. Finally, a small clique cover of the visibility graph was found, and each clique was turned into a convex piece of the resulting cover. Inspired by this, Werner et al. [132] used a similar approach for configuration spaces P of higher dimension. Instead of triangulating P (which may not be explicitly known in more complicated path finding settings), they sampled sufficiently many points from P. They then computed the visibility graph of the sampled points, found a clique cover, and then grew ellipsoids around each clique.

We will formulate a question about the two-dimensional version of the problem, that

[9] In general we may need infinitely many convex sets to cover P; we further refer to this issue below.

is, P is a connected region in the plane, possibly with holes. However, the question is also relevant in higher dimensions. Likewise, versions of our problem can be formulated when the configuration space is not known explicitly, but we can sample points from it. It is very difficult to find minimum convex covers: the problem of writing a simple polygon as a union of a minimum number of convex polygons has even been shown to be $\exists\mathbb{R}$-complete [3], and it is thus likely not in NP. A constant-factor approximation algorithm is known for this case [34]. In practice, it is often not feasible or necessary to cover all of P. For instance, if a concave curved arc appears on the boundary of P, then infinitely many convex sets are needed to cover P. However, we will insist that $\bigcup_{C\in\mathcal{C}} C$ is connected and is sufficiently similar to P. As a measure of similarity, we will here use the *Hausdorff distance* d_H. For two sets of points A, B, we define

$$d_H(A,B) = \max\{\sup_{a\in A}\inf_{b\in B}\|ab\|, \sup_{b\in B}\inf_{a\in A}\|ab\|\},$$

i.e., $d_H(A,B)$ is the largest distance from a point in A or B to the other set. We can now formulate our first problem related to convex covers.

> **Problem ApproximateConvexCover**: Given a tolerance $\varepsilon > 0$ and a polygon P with n vertices (and possibly with holes), devise an algorithm to find a family $\mathcal{C}$ of convex sets with the following properties, where $D = \bigcup_{C\in\mathcal{C}} C$:
> - D is connected,
> - $D \subseteq P$, and
> - $d_H(D,P) \leq \varepsilon$.
>
> The running time should be polynomial in n and $1/\varepsilon$. We want to minimize the number of sets $|\mathcal{C}|$; can we for instance find a set $\mathcal{C}$ that is only a constant factor larger than the optimal such set?

Let us note that in the literature, *approximate convex decomposition* may refer to allowing pieces that are slightly non-convex [94, 71], or to approximating the minimum number of (fully) convex pieces [34], rather than covering an approximation of the entire polygon P, which is what we mean here. One might consider other similarity measures as well, instead of just the ordinary Hausdorff distance. An interesting variation is to consider

$$d_{H^*}(A,B) = \max\{d_H(A,B), d_H(A^c,B^c)\},$$

where we also consider the Hausdorff distance between the complements. The optimal solution to the problem **ApproximateConvexCover** with d_H sometimes changes the topology of P in an undesirable way, giving up too much on connectivity, that does not happen when using d_{H^*}. On the other hand, d_{H^*} also has the downside of sometimes being too strict, thus requiring an excessive number of sets in the cover; see Figure 13.

As another version of the problem, we can also aim to minimize the total number of vertices of the sets $\mathcal{C}$, instead of the number of sets. This can be seen as the problem of finding a simpler representation of P, and is related to the study of *minimum link paths*, see for instance [61].

References

1 Anders Aamand, Mikkel Abrahamsen, Reilly Browne, Mayank Goswami, Prahlad Narasimhan Kasthurirangan, Linda Kleist, Joseph S. B. Mitchell, Valentin Polishchuk, and Jack Stade. Covering and partitioning complex objects with small pieces. In *42nd International Symposium on Computational Geometry (SoCG 2026)*, volume 367 of *LIPIcs*, pages 1:1–1:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.SoCG.2026.1.

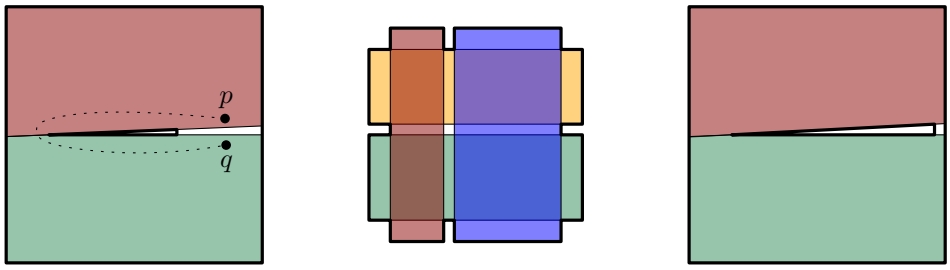


Figure 13 Left: Here, P is a square with a thin triangular hole. When using the standard Hausdorff distance d_H as similarity measure, the optimal solution results in a long uncovered corridor, leading to a long detour between the points p and q. Middle: A small hole in the cover as shown here may be acceptable as it does not significantly impair the connectivity of the region when using the cover for path planning tasks, but the stronger variant d_{H^*} will prevent that, which may result in more convex sets than are actually needed. Right: Even with d_{H^*}, narrow passages in P may be lost in the convex cover, but this is often acceptable in practice since a path using a narrow passage will have small clearance, which is undesirable (see also the last problem of Section 1: **2DiscsCombinedMeasure**).

2 Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *Automata, Languages, and Programming - 40th International Colloquium (ICALP 2013)*, pages 1–12, 2013. doi:10.1007/978-3-642-39206-1_1.

3 Mikkel Abrahamsen. Covering polygons is even harder. In *Foundations of Computer Science, (FOCS 2021)*, 2021. doi:10.1109/FOCS52979.2021.00045.

4 Mikkel Abrahamsen, Kevin Buchin, Maike Buchin, Linda Kleist, Maarten Löffler, Lena Schlipf, André Schulz, and Jack Stade. Reconfiguration of unit squares and disks: PSPACE-hardness in simple settings. In *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *LIPIcs*, pages 1:1–1:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.1.

5 Mikkel Abrahamsen, Tzvika Geft, Dan Halperin, and Barak Ugav. Coordination of multiple robots along given paths with bounded junction complexity. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 932–940, 2023. doi:10.5555/3545946.3598731.

6 Mikkel Abrahamsen, William Bille Meyling, and André Nusser. Constructing concise convex covers via clique covers (CG challenge). In *Symposium on Computational Geometry (SoCG 2023)*, 2023. doi:10.4230/LIPICS.SOCG.2023.66.

7 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for $\exists\mathbb{R}$-Completeness of Two-Dimensional Packing Problems. *TheoretiCS*, 3, 2024. doi:10.46298/theoretics.24.11.

8 Mikkel Abrahamsen and Nichlas Langhoff Rasmussen. Partitioning a polygon into small pieces. In *Symposium on Discrete Algorithms (SODA 2025)*, pages 3562–3589, 2025. doi:10.1137/1.9781611978322.118.

9 Mikkel Abrahamsen and Jack Stade. Hardness of packing, covering and partitioning simple polygons with unit squares. *SIAM Journal on Computing*, 55(5):FOCS24-29–FOCS24-84, 2026. doi:10.1137/24M171824X.

10 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Trans Autom. Sci. Eng.*, 12(4):1309–1317, 2015. doi:10.1109/TASE.2015.2470096.

11 Pankaj K. Agarwal, Boris Aronov, Tzvika Geft, and Dan Halperin. On two-handed planar assembly partitioning with connectivity constraints. In *Symposium on Discrete Algorithms (SODA 2021)*, pages 1740–1756, 2021. doi:10.1137/1.9781611976465.105.

12 Pankaj K. Agarwal, Mark de Berg, Benjamin Holmgren, Alex Steiger, and Martijn Struijs. Optimal motion planning for two square robots in a rectilinear environment. In *41st International*

Symposium on Computational Geometry (SoCG 2025), volume 332 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.SoCG.2025.5.

13 Pankaj K. Agarwal, Kyle Fox, and Oren Salzman. An efficient algorithm for computing high-quality paths amid polygonal obstacles. *ACM Trans. Algorithms*, 14(4):46:1–46:21, 2018. doi:10.1145/3230650.

14 Pankaj K. Agarwal, Dan Halperin, Micha Sharir, and Alex Steiger. Near-optimal min-sum motion planning for two square robots in a polygonal environment. In *Symposium on Discrete Algorithms (SODA 2024)*, pages 4942–4962, 2024. doi:10.1137/1.9781611977912.176.

15 Hee-Kap Ahn, Mark de Berg, Prosenjit Bose, Siu-Wing Cheng, Dan Halperin, Jirí Matousek, and Otfried Schwarzkopf. Separating an object from its cast. *Comput. Aided Des.*, 34(8):547–559, 2002. doi:10.1016/S0010-4485(01)00119-1.

16 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005. doi:10.1145/1059513.1059515.

17 Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *Int. J. Robotics Res.*, 36(9):1000–1021, 2017. doi:10.1177/0278364917719333.

18 Helmut Alt, Rudolf Fleischer, Michael Kaufmann, Kurt Mehlhorn, Stefan Näher, Stefan Schirra, and Christian Uhrig. Approximate motion planning and the complexity of the boundary of the union of simple geometric figures. *Algorithmica*, 8(5&6):391–406, 1992. doi:10.1007/BF01758853.

19 Boudewijn Asberg, Gregoria Blanco, Prosenjit Bose, Jesus Garcia-Lopez, Mark H. Overmars, Godfried T. Toussaint, Gordon T. Wilfong, and Binhai Zhu. Feasibility of design in stereolithography. *Algorithmica*, 19(1/2):61–83, 1997. doi:10.1007/PL00014421.

20 Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot. Unlabeled multi-robot motion planning with tighter separation bounds. In *Symposium on Computational Geometry (SoCG 2022)*, 2022. doi:10.4230/LIPICS.SOCG.2022.12.

21 Alon Baram, Efi Fogel, Dan Halperin, Michael Hemmer, and Sebastian Morr. Exact Minkowski sums of polygons with holes. *Comput. Geom.*, 73:46–56, 2018. doi:10.1016/j.comgeo.2018.06.005.

22 Alexander Below, Ulrich Brehm, Jesús A. De Loera, and Jürgen Richter-Gebert. Minimal simplicial dissections and triangulations of convex 3-polytopes. *Discret. Comput. Geom.*, 24(1):35–48, 2000. doi:10.1007/S004540010058.

23 Alexander Below, Jesús A. De Loera, and Jürgen Richter-Gebert. Finding minimal triangulations of convex 3-polytopes is NP-hard. In *Symposium on Discrete Algorithms (SODA 2000)*, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338235.

24 Alexander Below, Jesús A. De Loera, and Jürgen Richter-Gebert. The complexity of finding small triangulations of convex 3-polytopes. *J. Algorithms*, 50(2):134–167, 2004. doi:10.1016/S0196-6774(03)00092-0.

25 Eric Berberich, Dan Halperin, Michael Kerber, and Roza Pogalnikova. Deconstructing approximate offsets. *Discret. Comput. Geom.*, 48(4):964–989, 2012. doi:10.1007/S00454-012-9441-5.

26 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean Geometry*, pages 47–123. World Scientific, 1995. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812831699_0003, doi:10.1142/9789812831699_0003.

27 Marshall Bern, Jonathan R. Shewchuk, and Nina Amenta. Triangulations and mesh generation. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 29, pages 763–785. Chapman and Hall/CRC, 3rd edition, 2018. URL: http://www.csun.edu/~ctoth/Handbook/chap29.pdf, doi:10.1201/9781315119601.

28 Robert-Paul Berretty, Kenneth Y. Goldberg, Mark H. Overmars, and A. Frank van der Stappen. Computing fence designs for orienting parts. *Comput. Geom.*, 10(4):249–262, 1998. doi:10.1016/S0925-7721(98)00010-8.

29 Karl-Friedrich Böhringer. *Programmable Force Fields for Distributed Manipulation, and Their Implementation Using Microfabricated Actuator Arrays*. PhD thesis, Cornell University, 1997. URL: https://www2.ee.washington.edu/research/mems/publications/1997/theses/phd-bohringer-97.pdf.

30 Prosenjit Bose, Efi Fogel, Tzvika Geft, Dan Halperin, and Shahar Shamai. Optimal algorithms for separating a polyhedron from its single-part mold. *Computing in Geometry and Topology*, 3(1):7:1–7:19, 2024. doi:10.57717/cgt.v3i1.15.

31 Prosenjit Bose, Dan Halperin, and Shahar Shamai. On the separation of a polyhedron from its single-part mold. In *Conference on Automation Science and Engineering (CASE 2017)*, 2017. doi:10.1109/COASE.2017.8256076.

32 Prosenjit Bose and Godfried T. Toussaint. Geometric and computational aspects of manufacturing processes. *Comput. Graph.*, 18(4):487–497, 1994. doi:10.1016/0097-8493(94)90061-2.

33 Thomas Brocken, G. Wessel van der Heijden, Irina Kostitsyna, Lloyd E. Lo-Wong, and Remco J. A. Surtel. Multi-robot motion planning of k-colored discs is PSPACE-hard. In *Fun with Algorithms (FUN 2021)*, 2021. doi:10.4230/LIPICS.FUN.2021.15.

34 Reilly Browne, Prahlad Narasimhan Kasthurirangan, Joseph S. B. Mitchell, and Valentin Polishchuk. Constant-factor approximation algorithms for convex cover and hidden set in a simple polygon. In *Foundations of Computer Science (FOCS 2023)*, 2023. doi:10.1109/FOCS57990.2023.00083.

35 Bernard Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM J. Comput.*, 13(3):488–507, 1984. doi:10.1137/0213031.

36 Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theor. Comput. Sci.*, 84(1):77–105, 1991. doi:10.1016/0304-3975(91)90261-Y.

37 Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qi-Xing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. Dapper: decompose-and-pack for 3D printing. *ACM Trans. Graph.*, 34(6):213:1–213:12, 2015. doi:10.1145/2816795.2818087.

38 Xun Chen and Mingyong Chen. Skeleton partition models for 3D printing. In *International Conference on Medical Imaging and Additive Manufacturing (ICMIAM 2022)*, 2022. doi:10.1117/12.2636369.

39 Jae-Sook Cheong, Herman J. Haverkort, and A. Frank van der Stappen. On computing all immobilizing grasps of a simple polygon with few contacts. In *Algorithms and Computation, International Symposium (ISAAC 2003)*, 2003. doi:10.1007/978-3-540-24587-2_28.

40 Francis Y. L. Chin, Stanley P. Y. Fung, and Cao An Wang. Approximation for minimum triangulations of simplicial convex 3-polytopes. *Discret. Comput. Geom.*, 26(4):499–511, 2001. doi:10.1007/S00454-001-0045-8.

41 Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

42 Erik D. Demaine, Martin L. Demaine, and Helena A. Verrill. Coin-moving puzzles. In *More Games of No Chance*, pages 405–432. Cambridge University Press, 2002. doi:10.1017/9781316135167.030.

43 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019. doi:10.1137/18M1194341.

44 Adrian Dumitrescu and Minghui Jiang. On reconfiguration of disks in the plane and related problems. *Comput. Geom.*, 46(3):191–202, 2013. doi:10.1016/j.comgeo.2012.06.001.

45 Herbert Edelsbrunner, Leonidas J. Guibas, János Pach, Richard Pollack, Raimund Seidel, and Micha Sharir. Arrangements of curves in the plane - topology, combinatorics and algorithms. *Theor. Comput. Sci.*, 92(2):319–336, 1992. doi:10.1016/0304-3975(92)90319-B.

46 Eduard Eiben and Daniel Lokshtanov. Removing connected obstacles in the plane is FPT. In *Symposium on Computational Geometry (SoCG 2020)*, pages 39:1–39:14, 2020. doi:10.4230/LIPICS.SOCG.2020.39.

47 Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago Journal of Theoretical Computer Science*, 1999(8), 1999. doi:10.4086/cjtcs.1999.008.

48 Jeff Erickson. Finding holes in Minkowski sums is hard. Manuscript, 2018.

49 Guillermo Esteban, Dan Halperin, Víctor Ruíz, Vera Sacristán, and Rodrigo I. Silveira. Shortest coordinated motion for square robots. In *Algorithms and Data Structures - 18th International Symposium (WADS 2023)*, pages 430–443, 2023. doi:10.1007/978-3-031-38906-1_28.

50 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Stefan Schirra. Minimum coverage by convex polygons: The CG: SHOP challenge 2023. *Computing in Geometry and Topology*, 5(4):2:1–2:12, 2026. doi:10.57717/cgt.v5i4.115.

51 Sándor P. Fekete and Joseph S. B. Mitchell. Terrain decomposition and layered manufacturing. *Int. J. Comput. Geom. Appl.*, 11(6):647–668, 2001. doi:10.1142/S0218195901000687.

52 Efi Fogel and Dan Halperin. Polyhedral assembly partitioning with infinite translations or the importance of being exact. *IEEE Trans Autom. Sci. Eng.*, 10(2):227–241, 2013. doi:10.1109/TASE.2013.2242327.

53 Jianqi Gao, Yanjie Li, Xinyi Li, Kejian Yan, Ke Lin, and Xinyu Wu. A review of graph-based multi-agent pathfinding solvers: From classical to beyond classical. *Knowledge-Based Systems*, 283:111121, 2024. URL: https://www.sciencedirect.com/science/article/pii/S0950705123008717, doi:10.1016/j.knosys.2023.111121.

54 Michael R. Garey and David S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

55 Tzvika Geft. Fine-grained complexity analysis of multi-agent path finding on 2d grids. In *International Symposium on Combinatorial Search (SOCS)*, pages 20–28, 2023. doi:10.1609/SOCS.V16I1.27279.

56 Tzvika Geft and Dan Halperin. Refined hardness of distance-optimal multi-agent path finding. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 481–488, 2022. doi:10.5555/3535850.3535905.

57 Tzvika Geft, Dan Halperin, and Yonatan Nakar. Tractability frontiers in multi-robot coordination and geometric reconfiguration. In *Algorithmic Foundations of Robotics XVI (WAFR 2024)*, Springer Proceedings in Advanced Robotics, pages 247–267. Springer, 2025. doi:10.1007/978-3-032-09967-9_13.

58 Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 2010. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/view/1453.

59 Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1996)*, 1996. doi:10.1145/237170.237244.

60 Leonidas J. Guibas, Dan Halperin, Hirohisa Hirukawa, Jean-Claude Latombe, and Randall H. Wilson. Polyhedral assembly partitioning using maximally covered cells in arrangements of convex polytopes. *Int. J. Comput. Geom. Appl.*, 8(2):179–200, 1998. doi:10.1142/S0218195998000102.

61 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Int. J. Comput. Geom. Appl.*, 3(4):383–415, 1993. doi:10.1142/S0218195993000257.

62 Leonidas J. Guibas and F. Frances Yao. On translating a set of rectangles. In *Symposium on Theory of Computing (STOC 1980)*, 1980. doi:10.1145/800141.804663.

63 Dan Halperin, Lydia Kavraki, and Kiril Solovey. Robotics. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 51, pages 1343–1376. Chapman & Hall/CRC, 3rd edition, 2018. URL: http://www.csun.edu/~ctoth/Handbook/chap51.pdf, doi:10.1201/9781315119601.

64 Dan Halperin, Jean-Claude Latombe, and Randall H. Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26(3-4):577–601, 2000. doi:10.1007/S004539910025.

65 Dan Halperin and Micha Sharir. Almost tight upper bounds for the single cell and zone problems in three dimensions. *Discret. Comput. Geom.*, 14(4):385–410, 1995. doi:10.1007/BF02570714.

66 Dan Halperin and Micha Sharir. Arrangements. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 28, pages 723–762. Chapman & Hall/CRC, 3rd edition, 2018. URL: http://www.csun.edu/~ctoth/Handbook/chap28.pdf, doi:10.1201/9781315119601.

67 Dan Halperin, Micha Sharir, and Oren Salzman. Algorithmic motion planning. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50, pages 1311–1342. Chapman & Hall/CRC, 3rd edition, 2018. URL: http://www.csun.edu/~ctoth/Handbook/chap50.pdf, doi:10.1201/9781315119601.

68 Dan Halperin, Marc J. van Kreveld, Golan Miglioli-Levy, and Micha Sharir. Space-aware reconfiguration. *Discret. Comput. Geom.*, 69(4):1157–1194, 2023. doi:10.1007/S00454-022-00407-7.

69 Kris K. Hauser. The minimum constraint removal problem with three robotics applications. *Int. J. Robotics Res.*, 33(1):5–17, 2014. doi:10.1177/0278364913507795.

70 John E. Hopcroft, Jacob T. Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "Warehouseman's problem". *Int. J. Robotics Res.*, 3(4):76–88, 1984. doi:10.1177/027836498400300405.

71 Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213:1–213:12, 2014. doi:10.1145/2661229.2661244.

72 Baichuan Huang, Xujia Zhang, and Jingjin Yu. Toward optimal tabletop rearrangement with multiple manipulation primitives. In *International Conference on Robotics and Automation (ICRA 2024)*, pages 10860–10866, 2024. doi:10.1109/ICRA57147.2024.10610565.

73 Jeffrey Ichnowski and Ron Alterovitz. Concurrent nearest-neighbor searching for parallel sampling-based motion planning in so(3), se(3), and euclidean spaces. In *Workshop on the Algorithmic Foundations of Robotics (WAFR 2018)*, 2018. doi:10.1007/978-3-030-44051-0_5.

74 Ravi Janardan and Tony C. Woo. Manufacturing processes. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, chapter 55, pages 1241–1256. Chapman & Hall/CRC, 2004. doi:10.1201/9781420035315.ch55.

75 Jae D. Jeon and Beomhee Lee. Multi-robot formation shape control using convex optimization and bottleneck assignment. *Journal of Industrial and Intelligent Information*, 2:26–30, 2014. URL: https://api.semanticscholar.org/CorpusID:39104817.

76 Xiaotong Jiang, Xiaosheng Cheng, Qingjin Peng, Luming Liang, Ning Dai, Mingqiang Wei, and Cheng Cheng. Models partition for 3D printing objects using skeleton. *Rapid Prototyping Journal*, 23:54–64, 2017. doi:10.1108/RPJ-07-2015-0091.

77 Roman N. Karasev. A measure of non-convexity in the plane and the Minkowski sum. *Discret. Comput. Geom.*, 44(3):608–621, 2010. doi:10.1007/s00454-010-9258-z.

78 Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics Autom.*, 12(4):566–580, 1996. doi:10.1109/70.508439.

79 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discret. Comput. Geom.*, 1:59–70, 1986. doi:10.1007/BF02187683.

80 Young J. Kim, Ming C. Lin, and Dinesh Manocha. Collision and proximity queries. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 39, pages 1029–1056. Chapman & Hall/CRC, 3rd edition, 2018. URL: http://www.csun.edu/~ctoth/Handbook/chap39.pdf, doi:10.1201/9781315119601.

81 David G. Kirkpatrick and Paul Liu. Characterizing minimum-length coordinated motions for two discs. In *Canadian Conference on Computational Geometry (CCCG 2016)*, pages 252–259, 2016. URL: http://arxiv.org/abs/1607.04005.

82 David G. Kirkpatrick and Paul Liu. Minimum-length coordinated motions for two convex centrally-symmetric robots. *CoRR*, 2025. doi:10.48550/arXiv.2503.02010.

83 David G. Kirkpatrick, Bhubaneswar Mishra, and Chee-Keng Yap. Quantitative steinitz's theorems applications to multifingered grasping. *Discret. Comput. Geom.*, 7:295–318, 1992. doi:10.1007/BF02187843.

84 James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Vis. Comput. Graph.*, 4(1):21–36, 1998. doi:10.1109/2945.675649.

85 Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51(5):699–730, 2004. doi:10.1145/1017460.1017461.

86 Matias Korman, Maarten Löffler, Rodrigo I. Silveira, and Darren Strash. On the complexity of barrier resilience for fat regions and bounded ply. *Comput. Geom.*, 72:34–51, 2018. doi:10.1016/J.COMGEO.2018.02.006.

87 Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, and Subhash Suri. A constant factor approximation for navigating through connected obstacles in the plane. In *Symposium on Discrete Algorithms (SODA 2021)*, pages 822–839, 2021. doi:10.1137/1.9781611976465.52.

88 Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, Subhash Suri, and Jie Xue. Point separation and obstacle removal by finding and hitting odd cycles. In *Symposium on Computational Geometry (SoCG 2022)*, pages 52:1–52:14, 2022. doi:10.4230/LIPICS.SOCG.2022.52.

89 Jean-Claude Latombe. *Robot motion planning*, volume 124 of *The Kluwer international series in engineering and computer science*. Kluwer, 1991. doi:10.1007/978-1-4615-4022-9.

90 Jean-Claude Latombe, Randall H. Wilson, and Frédéric Cazals. Assembly sequencing with toleranced parts. *Comput. Aided Des.*, 29(2):159–174, 1997. doi:10.1016/S0010-4485(96)00044-9.

91 Steven M LaValle. *Planning algorithms*. Cambridge University Press, 2006. doi:10.1017/CBO9780511546877.

92 Steven M. LaValle and James J. Kuffner Jr. Randomized kinodynamic planning. *Int. J. Robotics Res.*, 20(5):378–400, 2001. doi:10.1177/02783640122067453.

93 Daniel Leven and Micha Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. *Discret. Comput. Geom.*, 2:9–31, 1987. doi:10.1007/BF02187867.

94 Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Comput. Geom.*, 35(1-2):100–123, 2006. doi:10.1016/J.COMGEO.2005.10.005.

95 Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph.*, 31(6):129:1–129:9, 2012. doi:10.1145/2366145.2366148.

96 Satoshi Makita and Weiwei Wan. A survey of robotic caging and its applications. *Advanced Robotics*, 31(19-20):1071–1085, 2017. doi:10.1080/01691864.2017.1371075.

97 Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. *Sci. Robotics*, 8(84), 2023. doi:10.1126/SCIROBOTICS.ADF7843.

98 Tobia Marcucci, Jack Umenberger, Pablo A. Parrilo, and Russ Tedrake. Shortest paths in graphs of convex sets. *SIAM J. Optim.*, 34(1):507–532, 2024. doi:10.1137/22M1523790.

99 Xanthippi Markenscoff, Luqun Ni, and Christos H. Papadimitriou. The geometry of grasping. *Int. J. Robotics Res.*, 9(1):61–74, 1990. doi:10.1177/027836499000900102.

100 Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Computer Vision Theory and Applications (VISAPP 2009)*, pages 331–340, 2009. doi:10.5220/0001787803310340.

101 Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Adam Fishman, and Dieter Fox. Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation. In *International Conference on Robotics and Automation (ICRA 2023)*, pages 1866–1874, 2023. doi:10.1109/ICRA48891.2023.10161528.

102 B. K. Natarajan. On planning assemblies. In *Symposium on Computational Geometry (SoCG 1988)*, 1988. doi:10.1145/73393.73424.

103 Colm Ó'Dúnlaing, Micha Sharir, and Chee-Keng Yap. Retraction: A new approach to motion-planning (extended abstract). In *Symposium on Theory of Computing (STOC 1983)*, 1983. doi:10.1145/800061.808750.

104 Andreas Orthey, Constantinos Chamzas, and Lydia E. Kavraki. Sampling-based motion planning: A comparative review. *Annu. Rev. Control. Robotics Auton. Syst.*, 7(1), 2024. doi:10.1146/ANNUREV-CONTROL-061623-094742.

105 Dario Prattichizzo and James C. Trinkle. Grasping. In Bruno Siciliano, Oussama Khatib, and Vincenzo Villani, editors, *Springer Handbook of Robotics*, pages 955–988. Springer Verlag, 2016. doi:10.1007/978-3-319-32552-1_38.

106 Anil S. Rao and Kenneth Y. Goldberg. Computing grasp functions. *Comput. Geom.*, 6:145–168, 1996. doi:10.1016/0925-7721(95)00022-4.

107 Anil S. Rao, David J. Kriegman, and Kenneth Y. Goldberg. Complete algorithms for reorienting polyhedral parts using a pivoting gripper. In *Symposium on Computational Geometry (SoCG 1995)*, 1995. doi:10.1145/220279.220332.

108 Barak Raveh, Angela Enosh, and Dan Halperin. A little more, a lot better: Improving path quality by a path-merging algorithm. *IEEE Trans. Robotics*, 27(2):365–371, 2011. doi:10.1109/TRO.2010.2098622.

109 Elon Rimon and Andrew Blake. Caging 2D bodies by 1-parameter two-fingered gripping systems. In *International Conference on Robotics and Automation (ICRA 1996)*, 1996. doi:10.1109/ROBOT.1996.506911.

110 Elon Rimon and Joel Burdick. *The Mechanics of Robot Grasping*. Cambridge University Press, 2019. doi:10.1017/9781108552011.

111 Alberto Rodriguez, Matthew T. Mason, and Steve Ferry. From caging to grasping. *Int. J. Robotics Res.*, 31(7):886–900, 2012. doi:10.1177/0278364912442972.

112 Oren Salzman. Sampling-based robot motion planning. *Commun. ACM*, 62(10):54–63, 2019. doi:10.1145/3318164.

113 Oren Salzman, Michael Hemmer, and Dan Halperin. On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. *IEEE Trans Autom. Sci. Eng.*, 12(2):529–538, 2015. doi:10.1109/TASE.2014.2331983.

114 Robert Schmidtke and Kenny Erleben. Chunked bounding volume hierarchies for fast digital prototyping using volumetric meshes. *IEEE Trans. Vis. Comput. Graph.*, 24(12):3044–3057, 2018. doi:10.1109/TVCG.2017.2784441.

115 Erich Schönhardt. Über die zerlegung von dreieckspolyedern in tetraeder. *Mathematische Annalen*, 98(1):309–312, 1928. doi:10.1007/BF01451597.

116 Jacob T Schwartz and Micha Sharir. On the piano movers' problem: Iii. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2(3):46–75, 1983. doi:10.1177/0278364983002003.

117 Jacob T Schwartz and Micha Sharir. On the "piano movers'" problem I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983. doi:10.1002/cpa.3160360305.

118 Jacob T Schwartz and Micha Sharir. On the "piano movers" problem II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298–351, 1983. doi:10.1016/0196-8858(83)90014-3.

119 Muhammad Muzamal Shahzad, Zubair Saeed, Asima Akhtar, Hammad Munawar, Muhammad Haroon Yousaf, Naveed Khan Baloach, and Fawad Hussain. A review of swarm robotics in a nutshell. *Drones*, 7(4), 2023. doi:10.3390/drones7040269.

120 Micha Sharir and Shmuel Sifrony. Coordinated motion planning for two independent robots. *Ann. Math. Artif. Intell.*, 3(1):107–130, 1991. doi:10.1007/BF01530889.

121 Rahul Shome, Kiril Solovey, Jingjin Yu, Kostas E. Bekris, and Dan Halperin. Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups. *IEEE Trans Autom. Sci. Eng.*, 18(3):888–901, 2021. doi:10.1109/TASE.2021.3055144.

122 Jack Snoeyink and Jorge Stolfi. Objects that cannot be taken apart with two hands. *Discret. Comput. Geom.*, 12:367–384, 1994. doi:10.1007/BF02574386.

123 Kiril Solovey. Complexity of planning. In Marcelo H. Ang, Oussama Khatib, and Bruno Siciliano, editors, *Encyclopedia of Robotics*, pages 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg, 2020. doi:10.1007/978-3-642-41610-1_166-1.

124 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *International Journal of Robotics Research*, 35(14):1750–1759, 2016. doi:10.1177/0278364916672311.

125 Paul G. Spirakis and Chee-Keng Yap. Strong NP-hardness of moving many discs. *Information Processing Letters*, 19(1):55–59, 1984. doi:10.1016/0020-0190(84)90130-3.

126 Bingjie Tang and Gaurav S. Sukhatme. Selective object rearrangement in clutter. In *Conference on Robot Learning (CoRL 2022)*, 2022. URL: https://proceedings.mlr.press/v205/tang23a.html.

127 Godfried Toussaint. Movable separability of sets. In *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 335–375. North-Holland, 1985. doi:10.1016/B978-0-444-87806-9.50018-9.

128 A. Frank van der Stappen, Robert-Paul Berretty, Kenneth Y. Goldberg, and Mark H. Overmars. Geometry and part feeding. In *Sensor Based Intelligent Robots, International Workshop*, 2000. doi:10.1007/3-540-45993-6_15.

129 A. Frank van der Stappen and Kenneth Y. Goldberg. Geometric eccentricity and the complexity of manipulation plans. *Algorithmica*, 26(3-4):494–514, 2000. doi:10.1007/S004539910022.

130 Rui Wang, Kai Gao, Daniel Nakhimovich, Jingjin Yu, and Kostas E. Bekris. Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 6621–6627. IEEE, 2021. doi:10.1109/ICRA48506.2021.9561716.

131 Ron Wein, Jur P. van den Berg, and Dan Halperin. Planning high-quality paths and corridors amidst obstacles. *Int. J. Robotics Res.*, 27(11-12):1213–1231, 2008. doi:10.1177/0278364908097213.

132 Peter Werner, Alexandre Amice, Tobia Marcucci, Daniela Rus, and Russ Tedrake. Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs. In *International Conference on Robotics and Automation (ICRA 2024)*, pages 10359–10365, 2024. doi:10.1109/ICRA57147.2024.10610005.

133 Randall H. Wilson and Jean-Claude Latombe. Geometric reasoning about mechanical assembly. *Artif. Intell.*, 71(2):371–396, 1994. doi:10.1016/0004-3702(94)90048-5.

134 Mengdie Wu, Wenyao Yan, Haslin Hasan, and Robiatul A'Dawiah Jamaluddin. A review of multi-agent path finding algorithms. In *International Conference on Information Systems and Computing Technology (ISCTech 2023)*, pages 69–73, 2023. doi:10.1109/ISCTech60480.2023.00020.

135 Yizhe Yang, Haochen Li, Kexin Zhang, Xinjian Jia, Gong Wang, and Bingshan Liu. A 3D nesting method based on the convex-concave coding similarity of the voxelized model for additive

manufacturing. *Additive Manufacturing*, 64:103429, 2023. URL: https://www.sciencedirect.com/science/article/pii/S2214860423000428, doi:10.1016/j.addma.2023.103429.

136 Chee K. Yap. Coordinating the motion of several discs. Technical Report Technical Report No. 105, Courant Institute of Mathematical Sciences, New York University, February 1984. Computer Science Department, Robotics Report No. 16.

137 E. C. Zeeman. On Hilbert's third problem. *The Mathematical Gazette*, 86(506):241–247, 2002. doi:10.2307/3621846.