# GEOMETRIC APPROXIMATIONS AND THEIR APPLICATION TO MOTION PLANNING

A Dissertation

by

MUKULIKA GHOSH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

| | |
|---|---|
| Chair of Committee, | Nancy M. Amato |
| Committee Members, | Ergun Akleman |
| | John Keyser |
| | Scott Schaefer |
| Head of Department, | Dilma Da Silva |

May 2019

Major Subject: Computer Science

# ABSTRACT

Geometric approximation methods are a preferred solution to handle complexities (such as a large volume or complex features such as concavities) in geometric objects or environments containing them. Complexities often pose a computational bottleneck for applications such as motion planning. Exact resolution of these complexities might introduce other complexities such as unmanageable number of components. Hence, approximation methods provide a way to handle these complexities in a manageable state by trading off some accuracy.

In this dissertation, two novel geometric approximation methods are studied: aggregation hierarchy and shape primitive skeleton. The aggregation hierarchy is a hierarchical clustering of polygonal or polyhedral objects. The shape primitive skeleton provides an approximation of bounded space as a skeleton of shape primitives. These methods are further applied to improve the performance of motion planning applications. We evaluate the methods in environments with 2D and 3D objects.

The aggregation hierarchy groups nearby objects into individual objects. The hierarchy is created by varying the distance threshold that determines which objects are nearby. This creates levels of detail of the environment. The hierarchy of the obstacle space is then used to create a decomposition of the complementary space (i.e, free space) into a set of sampling regions to improve the efficiency and accuracy of the sampling operation of the sampling based motion planners. Our results show that the method can improve the efficiency ($10 - 70\%$ of planning time) of sampling based motion planning algorithms.

The shape primitive skeleton inscribes a set of shape primitives (e.g., sphere, boxes) inside a bounded space such that they represent the *skeleton* or the connectivity of the space. We apply the shape primitive skeletons of the free space and obstacle space in motion planning problems to improve the collision detection operation. Our results also show the use of shape primitive skeleton in both spaces improves the performance of collision detectors (by $20 - 70\%$ of collision detection time) used in motion planning algorithms.

In summary, this dissertation evaluates how geometric approximation methods can be applied to improve the performance of motion planning methods, especially, sampling based motion planning methods.

# DEDICATION

To my family and my teachers.

# ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Nancy M. Amato, for her encouragement, guidance and continuous support. Thank you for helping me grew as a researcher.

I would also like to thank my committee members, Dr. Ergun Akleman, Dr. John Keyser, and Dr. Scott Schaefer, for their support and constructive feedback. I appreciate they committed their time to help me grow as a researcher.

I also thank all the collaborators that I have worked with : Dr. Jyh-Ming Lien, Dr. Marco Morales, Dr. Sam Rodriguez, Dr. Shawna Thomas, Dr. Jory Denny, Dr. Hsin-Yi Yeh, Daniel Tomkins, Ashley Tharp, Diego Ruvalcaba and Kiffany Lyons. I am especially thankful to Shawna for her guidance and help with my research. Thank you to all the Parasol members for your help throughout my graduate career. I am grateful to Hsin-Yi Yeh, Chinwe Ekenna and Diane Uwacu. I will cherish their friendship and support not only in research work but also in service.

I would also like to thank all the conferences and workshops that provided travel grants during my graduate career. These include funding to attend the Grace Hopper Celebration of Women in Computing Conference, the Symposium of Computational Geometry, the IEEE/RSJ International Conference on Intelligent Robots and Systems, and the Workshop on the Algorithmic Foundations of Robotics. It helped in developing my professional experience.

Finally, I would like to thank my family whose constant support and encouragement throughout my life. I am grateful to my parents for believing in me and allowing me to pursue graduate studies outside my home country. I am indebted to my husband for being a rock and encouraging me to follow my dreams. I am thankful to my sister for being a constant role model.

CONTRIBUTORS AND FUNDING SOURCES

## Contributors

This work was supported by a dissertation committee consisting of Professor Nancy M. Amato and Professors John Keyser, Scott Schaefer of the Department of Computer Science & Engineering and Professor Ergun Akleman of the Department of Visualization.

All work conducted for the dissertation was primarily completed by the student, under the advisement of Professor Nancy M. Amato of the Department of Computer Science & Engineering.

Parts of the analyses depicted in Chapter 3 (Section 3.3) were conducted in collaboration with Shawna Thomas of the Department of Computer Science & Engineering, Marco Morales of Department of Digital Systems Instituto Tecnológico Autónomo de México and Sam Rodriguez of Texas Wesleyan University, and were published in 2016 in the proceedings of the 2016 *IEEE International Conference on Intelligent Robots and Systems* (IROS). Parts of the analyses depicted in Chapter 4 (Section 4.3) were conducted in collaboration with Shawna Thomas of the Department of Computer Science & Engineering and was accepted for publication in 2018 in the proceedings of the 2018 *International Workshop on the Algorithmic Foundations of Robotics* (WAFR).

## Funding Sources

Graduate study was supported by research funding from Professor Nancy M. Amato of the Department of Computer Science & Engineering and fellowships from Texas A&M University.

This work was made possible in part by NSF awards under CCF-1423111, IIS-0916053, IIS-0917266, EFRI-1240483.

Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the NSF.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

# LIST OF TABLES

TABLE Page

# 1. INTRODUCTION

Complexities in the geometric object(s) or the environment containing them often pose a computational challenge in applications such as graphics [1], computational biology [2] and robotics [3]. For example, in a city-like environment with a large number of buildings and trees and differences in width, length and curvature of roads and gullies take a toll on applications such as navigation. These complexities can be either in the form of a large number of components or a large volume or complex features such as concavities, branching and non-uniform distribution. This affects the overall performance of applications that use geometric objects.

One of the applications where geometric objects are used is motion planning (which has further applications in robotics, virtual prototyping and biology). It is the process of finding a *valid* path for a moveable object (or robot) from a start to a goal location. Both the moveable objects and obstacles in the environment are generally represented as geometric objects. The complexities in geometry of these objects often affect the performance of basic operations in motion planning algorithms such as collision detection. As collision detection is generally used to validate not only the placements of robots but also to validate the path they take, these primitive operations are sometimes used as a measure of performance for motion planning algorithms [4].

Approximation methods such as simplification and decomposition are commonly used approaches in efficient handling of geometric complexities. Exact resolution of these complexities might exist. Sometimes the exact resolution might introduce other types of complexities. For example, the exact convex decomposition of a complex object (with various degrees of concavity) can result in an unmanageable number of components. Hence, approximations methods are often preferred to reduce these complexities to a manageable degree. Besides improvement in performance, approximation methods provide an alternative representation of the objects which are useful in creating levels of detail.

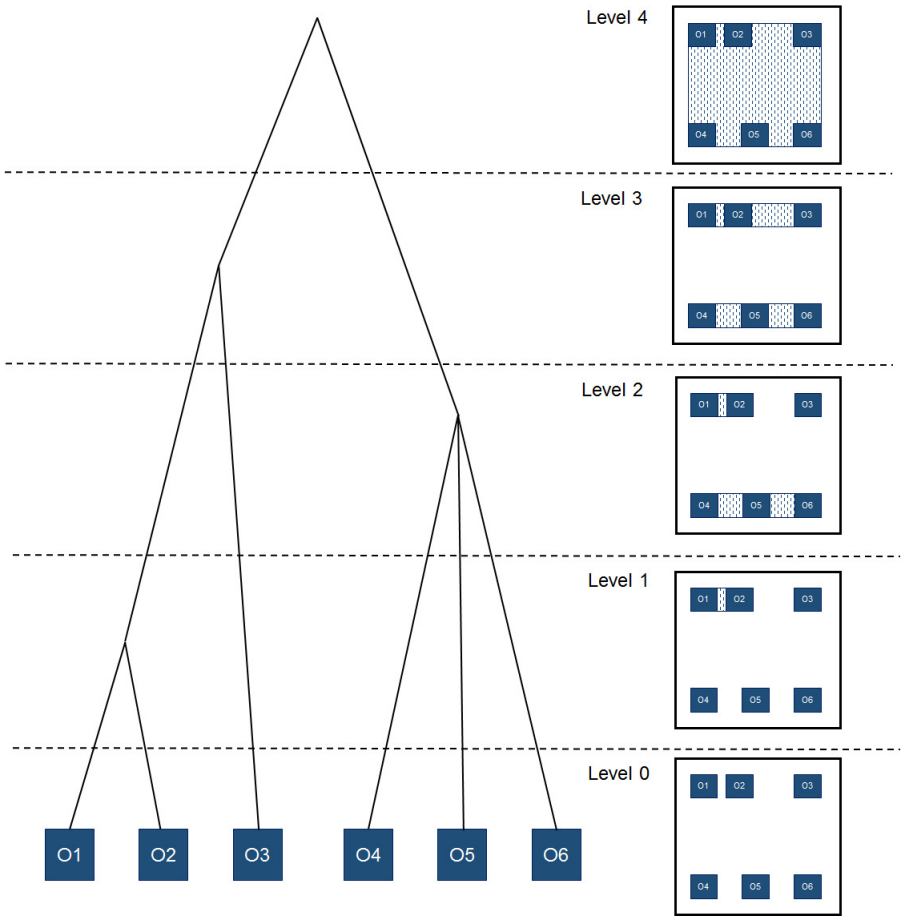
There are various geometric approximation methods in the literature addressing different types of complexities. Decomposition methods are used to handle large objects with complex properties

such as concavity, volume, etc. For example, the decomposition of the space is often used in sampling based motion planning methods to get a fair coverage of the space [5]. Clustering or aggregation methods are used to handle large number of objects in an environment. In GIS and navigation applications, clustering methods are used to ease spatial indexing. Sometimes, one type of approximation method is used to generate another type of approximation. For example, in [6] clustering is used to generate a decomposition of the object.

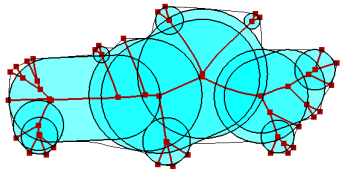
### 1.1 Research Contribution

In this dissertation, we propose two types of geometric approximations and their application in motion planning algorithms:

- **Hierarchical Distance based Aggregation:** In this method, we group or *aggregate* sets of "nearby" objects into individual objects based on the minimum distance between them. We create the levels in the hierarchy by varying the threshold distance used to define nearby objects (See Fig. 1.1a). Unlike other clustering approaches, we do not use the aggregation method for spatial indexing. We apply the aggregation hierarchy on the obstacle space to decompose the free space of the environment. The levels in the aggregation hierarchy are used in characterizing the passages (free space between the objects) based on their width (or the distance between the objects). This helps the motion planning methods to prefer generating paths through wider passages first. Generally, the decomposition of free space is used to improve the coverage of sampling based planners by distributing the sample generation in the free space volume instead of the entire volume (which includes the obstacle space volume that affects the distribution of samples in narrow passages). In most of these applications of decomposition, all the decomposed regions are simultaneously sampled or planned at each iteration. In our method, we "dig" into the solution by planning in the wider regions before narrow passages. This saves planning time (by $10 - 70\%$ of the original planning time) in unnecessary narrow regions while providing good coverage and high success in samples placement (due to low obstacle volume in the decomposed regions which are a subset of the free space).




(a) Aggregation Hierarchy



(b) Shape Primitive Skeleton

Figure 1.1: Examples of aggregation hierarchy and shape primitive skeleton. (a) Aggregation hierarchy tree of the environment shown in right. (b) Shape primitive skeleton approximation of a car.

- **Shape Primitive Skeleton:** In this method, we approximate a bounded space into sets of *shape primitives* (e.g., sphere and boxes) which represents the topology or *skeleton* of the space (See Fig. 1.1b). We use the shape primitive skeletons of both obstacle space and free space to improve the performance of motion planning algorithms (specifically the collision detection operation). Existing shape primitive approximation methods like bounding volume hierarchies and packing algorithms are generally applied to one of the spaces (the obstacle space) to improve the collision detection. Use of approximation in one space helps in providing either early in or early out decisions about the collision status (example, for bounding volume hierarchies it helps in providing early out, i.e., collision free decision). By using the approximation in both spaces, our application method can provide both early in or early out collision status decisions. This improves the collision detection time by $20 - 70\%$ of the traditional bounding volume hierarchy based collision detection time. Instead of biasing on a single type of primitive (as is generally done in bounding volume hierarchies and packing methods), we use a variety of shapes. This makes the method applicable to a variety of environments: from box-like city or office plans to tubular biological structures.

Portions of this research were previously published and presented. Aggregation hierarchy and its application to improve motion planning algorithms was published in the proceedings of the 2016 *IEEE International Conference on Intelligent Robots and Systems* (IROS) [7]. Shape primitive skeleton approximation and its application to improve collision detection in motion planning algorithms is accepted and will appear in the proceedings of the 2018 *International Workshop on the Algorithmic Foundations of Robotics* (WAFR) [8].

# 2. PRELIMINARIES AND RELATED WORK

This chapter provides a comparative study of various geometric approximation techniques and how they are used to improve the performance of motion planning algorithms.

## 2.1 General Approximations

We consider the object or objects in the environment are represented by either a polygon in 2D or a polyhedral mesh in 3D. A polygon or polyhedron ($P$) can be defined as a closed subset of Euclidean space ($R^2$ or $R^3$) bounded by set of facets (line segments in 2D and polygonal faces in 3D).

### 2.1.1 Aggregation or Clustering

The aggregation hierarchy method is similar to clustering methods as both are used to split a large number of items into a small number of groups. A detailed survey of clustering algorithms can be found in [9, 10].

Hierarchical clustering methods have been defined for applications to a varied set of data including polygonal objects and data points. Methods are classified as agglomerative or divisive based on the approach used in its construction [9]. Clustering methods differ in distance metric, clustering function (single linkage or complete linkage) and termination condition [10]. Our hierarchy can be classified as a divisive method which uses a clustering criteria similar to single-linkage methods. As stated in [11], the disadvantages of most of these methods include vagueness of termination condition and that the hierarchy is not revisited for improvements once clusters are constructed. We address these issues by allowing the user to tune the hierarchy easily based on its application once the basic untuned hierarchy is created. Unlike the popular usage of hierarchical clustering for spatial indexing queries such as nearest neighbor search, we focus the application of our method in modeling the environment containing a large number of geometric objects.

The hierarchical clustering methods are often used to cluster a set of data-points [12] or a set of non-disjoint polygons [13]. Most of these polygonal clustering approaches are used in geo-spatial

applications where the polygons are clustered to describe regions based on area, neighborhood, etc [13, 14, 15, 16]. Polygonal clustering is used in simplification of urban scenes [17, 18], where given a user viewpoint the buildings near the viewpoint are simplified using mesh simplification and the buildings far away from the viewpoint are aggregated by their footprint polygons and mapped back as a clustered mesh. Applications such as ArcGIS and PostGIS aggregate polygons based on distance. In this work, we extend this idea to determine the groups that will form the aggregates. Wang et. al. [19] analyses the grouping of polygons based on various criteria for spatial similarity and abstract the neighborhood relationship of the objects in form of graph. They use a graph partition technique to obtain the various levels of clusters. In our method, we use a similar notion of a neighborhood graph to abstract each level in the aggregation hierarchy (i.e., to identify the objects to aggregate in single object). Gewali and Manandhar [20] provides a method to cluster convex polygonal objects by two types of approaches: (1) using a visibility graph and covering the cluster with the convex hull cover and (2) using a refined voronoi diagram and voronoi edges around the clusters to create the cover. Most of these polygonal clustering methods either used in geospatial applications or in simplification of urban models, use or test with polygons that are similar in shape or simple where the number of vertices is low (less than 100) or the shape is orthogonal or the shape is convex. Our method does not assume similarity or simplicity of the input shapes and adapts the approximate shape of the grouped object on-the-fly based on the shape of the free space between the objects.

Aggregation of a polyhedral mesh is often used as a union operator in a mesh generator [21]. In most applications, the meshes used for union are intersecting or an intersecting boundary is generated to join the meshes along the intersecting boundary [21]. Disjoint three dimensional points are clustered to form a simplified or new polyhderal structure [22]. Alpha shapes [23], skin surfaces [24] and other methods are used to determine the polyhedral mesh cover. Disjoint 3D objects are clustered to densely pack similar structures inside simple shapes [25] or to virtually sculpt general shapes [1]. Most of these methods use homogeneous objects, and hence traditional alpha shapes [26, 23] are enough to serve the purpose. However, to handle non-uniformity in

inputs, weighted alpha shapes [27] and conformal alpha shapes [28] are defined, which are used in applications such as mesh reconstruction and feature detection. Instead of determing the weights on the point set, we determine alpha adaptively based on the properties of the space between the objects.

Clustering of features such as the vertices and faces of a polyhedral mesh are often used to define a decomposition [6, 29] or tiling of the surface [30] of the polygonal mesh. Clustering features in a polyhderal mesh are also used to simplify the mesh [31, 32, 33]. Most of these clustering methods are applied to a single mesh to decompose or simplify.

A mesh simplification strategy using carving of a Constrained Delaunay Tetrahedralization is described in [34]. Although their objective is to simplify the mesh, the algorithm can be applied to disjoint meshes to create an aggregated structure. In this work, we use the carving technique from [34] to define the aggregated structure where the distance between them determines whether they will be aggregated or not. However, our work differs by using the passage width properties between the disjoint meshes in approximating the clustered meshes.

### 2.1.2 Shape Primitive Approximations

There are primarily two basic types of approximations that use shape primitives: bounding volume hierarchies and packing algorithms.

#### *2.1.2.1 Bounding Volume Approximations*

Bounding volume hierarchies are used in approximating polygonal or polyhedral objects which are further used in collision detection algorithms such as: Spheres [4], Axis Aligned Bounding Box (AABB) [35], Oriented Bounding Box (OBB) [36, 35], and Discrete Orientation Polytopes (k-DOPs) [37]. These bounding volume hierarchies follow two major steps: decomposition of the object into regions and bounding the regions with geometric shape primitives. The choice of bounding volume depends on various criteria: tightness, simplicity in intersection check, and transformation invariance.

In all of these bounding volume hierarchy methods, the shape primitives overhang beyond the

boundary of the object. The collision detection method uses the hierarchy to filter out potential candidates that are not in collision, i.e., if an object is outside the bounding volume of another object, the objects are not in collision. In contrast, our method places the geometric shape primitives that are completely inside the object. Hence we used it to check if the object is completely inside the shape primitive inscribed in another object.

Bounding volumes are useful to determine objects not in collision whereas to determine collision one needs to traverse the depths of the hierarchy. Schneider et. al. improves the performance of bounding volume hierarchy collision detection for objects in narrow passages by early detection of collision using a distance field and oct-tree on a modified swap algorithm [38]. In our method, we apply the approximation to the obstacle space and the free space to avoid hierarchy depth traversal and expensive penetration distance computations.

#### *2.1.2.2 Packing Algorithms*

Packing algorithms "pack" the geometric shape primitives inside a given object(s). Similar to bounding volume hierarchies, various packing algorithms exist for various shape primitives [39, 40, 41, 42]. Their objective is to provide a good coverage of the space. Generally, packing algorithms use same sized shape primitives.

Weller and Zachmann [43] proposed an irregular sphere packing method. Their approach uses a discretized distance map of the inner volume which is updated after placement of each sphere. Stolpner et. al. [44] generate an inner sphere tree by approximating the medial axis. The spheres are internal to the object and are "well distributed". The distribution is achieved by placing the spheres on the intersection of the grid cells and the medial axis. However, the distribution of the spheres is dependent on the grid cell size.

Most of these approximation methods (including the bounding volume hierarchies) use a single type of geometric shape primitive. Our method of approximation uses a variety of geometric shape primitives to provide good coverage of the space with a small number of primitives. Also, all of these methods are used in approximating one space (mainly the obstacle space). We use the approximation method to approximate both obstacle space and free space to improve the performance

of collision detection methods.

## 2.2 Geometric Approximations in Motion Planning

Motion planning is the problem of finding a valid path for a moveable object(s) from a start to a goal location. The physical tangible environment in which the moveable object(s) and obstacles reside is called the *workspace*. The state or the *configuration* of the moveable object or robot is represented as a tuple of parameters (such as location, orientation, joint angles) called the *degrees of freedom (DOFs)* of the robot. The space containing all possible configurations (both valid and invalid) of the robot is called configuration space ($\mathcal{C}_{space}$) [45]. The configuration space creates an abstraction such that the motion planning problem transforms into finding a valid path for a point in the configuration space.

Sampling-based planners, such as Probabilistic Roadmaps (PRMs) [46] and Rapidly-exploring Random Trees (RRT) [47], are very popular since they are able to solve a wide variety of challenging problems. They work by sampling and connecting a set of valid configurations to produce a graph or a tree that represents the feasible motions. Since it is computationally infeasible to explicitly represent the obstacles in $\mathcal{C}_{space}$ [48], the validity of a configuration is determined through collision detection tests in the workspace. Variants of these basic planners have been developed to address issues such as the difficulty of sampling in narrow passages [49, 50, 51] or to have additional benefits such as optimal paths [52] or high clearance paths [53, 54].

As our geometric approximation methods can be implemented in the workspace their intended application is for those motion planning problems where workspace obstacles play an important role (such as to determine the validity of the path). Hence our methods are applicable to workspace heavy planning problems like navigation but not for $\mathcal{C}_{space}$ heavy problems like assembly or disassembly planning.

### 2.2.1 Clustering and Decomposition in Motion Planning

Workspace decomposition, similar to our proposed application of aggregation hierarchy, has shown benefit in various motion planning algorithms [55, 56, 57, 5]. Some of them use triangula-

tions of the free space in the environment [57, 5]. Similar to [5], Denny et. al. uses skeletonization (instead of decomposition of the free workspace) as guide for sampling and hence path planning [3]. However, most use the decomposed regions to improve sampling distribution whereas our method aims at improving sampling efficiency. Also, the aggregation hierarchy characterizes the regions in the free space based on the distance between the obstacles.

### 2.2.2 Shape Primitive Approximations in Motion Planning

The use of shape primitive skeleton approximation to improve collision detection in motion planning closely relates to the concept of workspace safety certificates [58, 59]. Safety certificates are regions in free space (mainly convex) that are guaranteed to be collision free and hence help in certifying the validity of robot configurations or path segments. In [60], Lacevic et. al. provide a way to generate an approximation of C-free certificates (called burs) and use them for efficient path planning. These safety certificates methods bear resemblance to kinetic data structures which are constructed or evolve with time, whereas the safety certificates (i.e., shape primitives) in our method are statically placed as a pre-processing step. Hence the shape and size of the certificates do not depend on the shape and size of the robot but on that of the obstacles in the environment.

Deits and Tedrake [61] provide a method of computing the largest collision free convex region (polytope or ellipsoid) around an input seed point. Getting a good coverage of the obstacle-free space requires a good distribution of the seed points. Brock and Kavraki [62] use a set of spheres in the free-space as a guide for high-dimensional problems. Yang and Brock [63] use a set of spheres in the free-space to improve the sampling distribution (near to the medial axis and hence with high clearance). Both of these methods use only one type of primitive, i.e., spheres to cover the free-space. All of these methods create the primitives in obstacle free space.

Exploitation of information from both free and obstacle space is motivated from works such as Denny and Amato [64] where planning is done in both spaces and Bialkowski et. al. [65] where spatial indexing is augmented with information about obstacle density.

# 3. HIERARCHICAL AGGREGATION[1]

Object aggregation is often used to approximate a set of objects that are likely to be processed together as a single entity. This reduces the number of objects to a manageable degree.

The aggregation hierarchy represents the hierarchical clustering of the objects in the environment based on the distance between the objects. The distance threshold, $\delta$, is used to determine which objects are nearby. In other words, two objects are near to each other if the distance between them is less than $\delta$.

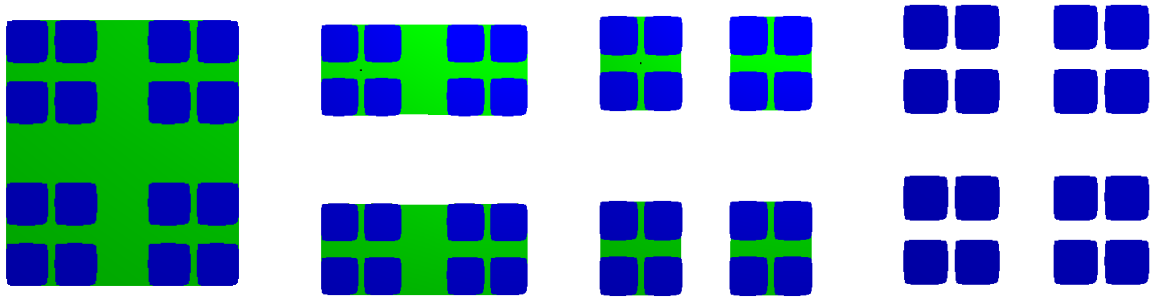
In this chapter, we first describe the hierarchical aggregation approach. We show how the method can be applied to environments with 2D and 3D objects and evaluate them based on their construction time. Next, we describe how the hierarchy can be utilized in sampling-based planning algorithms. We demonstrate the importance of using the aggregation hierarchy by their impact on the planning and sampling time and the percentage of successful sampling attempts.

## 3.1 Approach

In this section, we describe how the aggregation hierarchy is constructed for an environment with a set of objects. We assume the objects to be disjoint (or objects which have zero distance between them are already merged as a single object).

We develop a hierarchy of aggregation levels using a top-down approach such that the highest level is the coarsest level ($l_0$) with all objects aggregated into a single model (specifically the convex hull of the objects Fig. 3.1a). Then we decrease the distance threshold, $\delta$, that determines which objects are nearby to create a new level. At each level the aggregated environment is computed using a distance based aggregation method (Fig. 3.1b and Fig. 3.1c show the aggregated environment at levels N and N+1). We iterate this process until the lowest level is created which is the original input problem where all objects retain their original geometries (Fig. 3.1d).

[1]The description of the method and some experimental results are reprinted with permission from "Motion Planning using Hierarchical Aggregation of Workspace Obstacles" by M. Ghosh, S. Thomas, M. Morales, S. Rodriguez, N. M. Amato, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, [7] ©2016 IEEE.



(a) Highest Level Aggregation (b) N-Level Aggregation (c) N+1-Level Aggregation (d) Lowest-Level Aggregation/Original Environment

Figure 3.1: Different levels in the hierarchy. The highest level contains all object aggregated as one(a) and the lowest level is the original environment(d). Original objects are shown in blue whereas the aggregated space is shown in green.

Aggregation at each level clusters nearby objects into a single group while reducing the approximation error. The hierarchy tuning method is used to determine whether to store the newly created level in the hierarchy or not (in other words, it ensures the levels in the hierarchy are significantly different). Algorithm 1 provides a sketch of the algorithm.

**Algorithm 1** Hierarchical Distance Aggregation($E$)

**Require:** Environment $E$ that contains set of objects $O$.
**Ensure:** Hierarchy, $H = \{l_0, l_1, ..., l_n\}$.
1: Create initial level, $l_0 \leftarrow CH(O)$ and add it in $H$.
2: $i = 0$, $AE = l_0.Env$.
3: **while** $AE \neq E$ **do**
4: Get next $\delta$.
5: $AO \leftarrow$ Distance Aggregation($O$,$\delta$). $AE$ = Environment with aggregated objects $AO$.
6: **if** Hierarchy Tuning($AE$,$l_i.Env$) **then**
7: Add a new level, $l_{i+1}$ to $H$ with $l_{i+1}.Env \leftarrow AE$.
8: $i = i + 1$.
9: return $H$

### 3.1.1 Distance based Aggregation

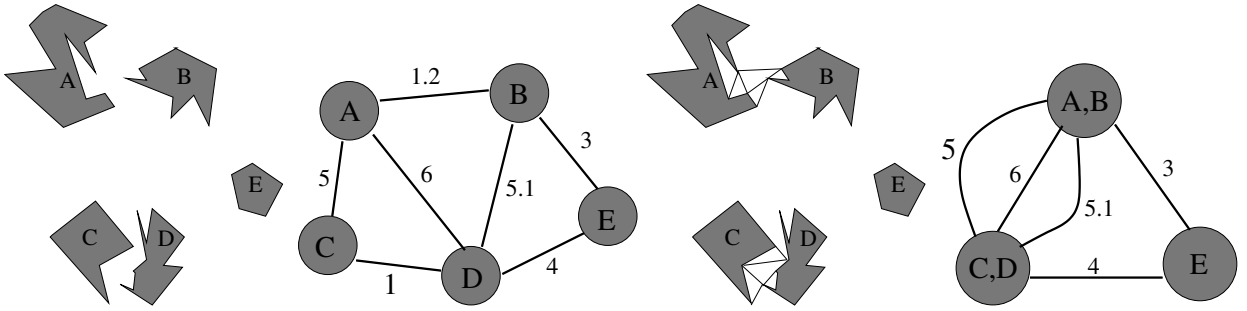
One simple way to determine which objects should be grouped together at level $i$ of the hierarchy is with a distance threshold $\delta_i$. Objects are then grouped together if the distance between them is less than $\delta_i$.

Aggregating objects by their distance involves two major steps: (1) identifying the groups of nearby objects and (2) creating approximate shapes that cover each group of objects. To find the group of nearby objects, we abstract the environment at each level with a graph called the *neighborhood graph*. The vertices in the neighborhood graph correspond to the objects in the environment and weighted edges represent the neighborhood relationship between the objects with minimum distance as weights (Fig. 3.2d is the neighborhood graph of the aggregated environment in Fig. 3.2c). Hence the vertices of the neighborhood graph at each level denote the objects that need to be grouped together as a single object. An approximate cover for the groups is then computed. Algorithm 2 provides a sketch of the algorithm.

The neighborhood graph at each level is created by collapsing the edges of the neighborhood graph of the original input environment (Fig. 3.2b) with weight less than $\delta_i$. The construction of the initial neighborhood graph (or the graph for the original environment) is detailed in Section 3.1.1.1.

**Algorithm 2** Distance Aggregation($O$,$\delta$)

**Require:** A set of objects $O$ and threshold distance $\delta$.
**Ensure:** An aggregated set of objects $AO$.
1: Initialize the Neighborhood Graph $NG(V, E)$ with each $o \in O$ as $v$ in $V$.
2: Generate the aggregated neighborhood graph, $AG(V', E')$ from $NG(V, E)$ using $\delta$.
3: **for** each $v \in V'$ **do**
4: Create the approximate shape, $ao$ of the group of objects which $v$ represents.
5: $AO \leftarrow AO \cup ao$.
6: return $AO$




(a) Original Environment (b) Original Environment Neighborhood Graph (c) Aggregated Environment (d) Aggregated Environment Neighborhood Graph

Figure 3.2: Aggregated state of environment and corresponding neighborhood graph. Given an environment in (a) and its neighborhood graph in (b) and given distance threshold value $\delta$, the corresponding aggregated view of the environment is shown in (c) and its neighborhood graph in (d). Original objects are shaded gray.

#### *3.1.1.1 Neighborhood Graph*

We represent the environment with a graph called a neighborhood graph in which the vertices represent objects and weighted edges represent the neighborhood relationship between the objects with weights indicating the distance between the objects. For example, Fig. 3.2b and Fig. 3.2d show the neighborhood graph of the aggregated environment in Fig. 3.2c and Fig. 3.2a. Given a $\delta_i$, we generate the graph for the aggregated environment by collapsing the edges with weight less than $\delta_i$ in the neighborhood graph of the original input environment (graph in Fig 3.2b). The (super) vertices in the graph of the aggregated environment represent the groups whose approximated cover needs to be computed.

Computing neighborhood relationship between every pair of objects in the environment based on visibility is computationally expensive. So to find the neighborhood relationship of objects in the original environment, we identify approximate Voronoi regions where objects are sites instead of points. The graph can be extracted from the Voronoi regions with the regions encoded as vertices and the neighboring regions (ridges in Voronoi diagram) as edges in the graph. The minimum distance between the objects is then computed if there exists an edge between the representative vertices of the objects in the neighborhood graph. The neighborhood graph can contain multiple

edges if there exists non-adjacent spaces (or passages) between a pair of objects (i.e., each non-adjacent Voronoi ridge between a pair of objects are mapped as a separate edge in the neighborhood graph).

The most computationally expensive operation in the neighborhood graph construction is the determination of the weights of the edges which are the minimum distance between the objects. The distance can be computed using a brute-force method of calculating the minimum distance for every pair of vertices in the objects that have a corresponding edge in the neighborhood graph. To improve the computation further, instead of computing the distance for every pair of vertices in the objects, we can compute the distance for the pairs of subsets of vertices of the objects that are visible to the Voronoi ridge which corresponds to the edge in the neighborhood graph. The distance transform method [66] can also be used to approximate the distance between the objects where the pixel or voxel size or the distance between the contours need to be parametrized. For 2D environments, we can also consider Frechet distance computation methods [67] between the polyline boundaries of the objects.

In our implementation, we use the Constrained Delaunay Triangulation/Tetrahedralization (CDT) of the free space between the objects to identify the Voronoi regions (obtained from the dual of the CDT) and hence construct the neighborhood graph. We approximate the distance computation through the CDT edge lengths. The minimum distance between two objects is approximated as the minimum length of CDT edges connecting the two objects. As we consider the approximation of the objects, approximation of the minimum distance measure saves computation time with minimal loss in accuracy. Large facets in the objects i.e., boundary edges or faces are re-sampled and divided to keep the distance computation accuracy within reasonable bounds.

#### *3.1.1.2 Creation of Aggregated Model*

There are many ways to define the shape cover of the aggregated objects. The convex hull of the aggregated objects may be too conservative in many cases, particularly as the concavity of the aggregated objects increases. As shown in Fig 3.3, the convex hull covers the free region shown in pink which might be necessary in finding a path in motion planning problems. The objects can

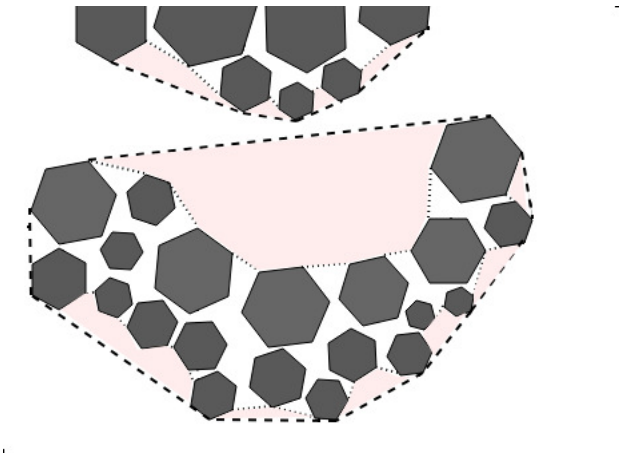
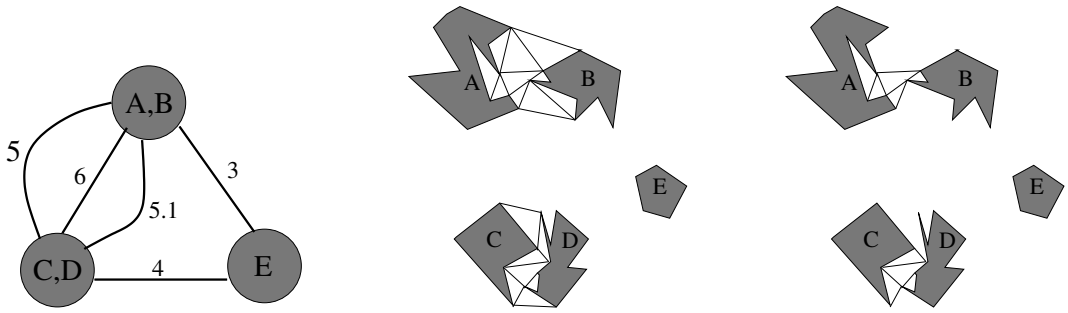


Figure 3.3: Disadvantage of convex hull shape cover. Convex hull cover (shown in dotted lines) is too conservative for aggregates which occludes large amounts of free space between objects (shown in pink).

be aggregated using their minimal area enclosure (that connects two objects by a minimum distant line segment connecting the object). However, this would increase the number of features instead of reducing it and not cover unnecessary regions (such as unnecessary narrow passages).

Alpha shape is one of the state-of-the-art solutions to form a surface representation of a set of vertices. However, the value of alpha needs to be adaptively determined based on the applications such as in weighted alpha shapes [27], density or anisotropic alpha shapes [68]. As our objective is to utilize the aggregation hierarchy to improve the performance of motion planning algorithms, we use a shape cover similar to weighted alpha shape cover where the weight is adapted based on the variation of the passage widths.

In this work, we use a shape cover based on the iterative removal of triangles/tetrahedra from a Constrained Delaunay Triangulation/Tetrahedralization (CDT) of the free space between the objects (see Fig. 3.4). First, we remove the triangle or tetrahedra from the CDT if it can be mapped to an edge in the neighborhood graph of the aggregated environment. This generates a set of initial approximations of the aggregated objects (Fig. 3.4b). Next, for level $i$ of the aggregation, we remove a triangle/tetrahedron based on the distance threshold $\delta_i$ to get the final aggregated environment (Fig. 3.4c). We further adapt $\delta_i$ to ${\delta_i}^{sd}$ for shape cover use based on the variation of passage




(a) Neighborhood Graph of the Aggregated Environment

(b) Corresponding Aggregated Environment

(c) Final Aggregated Environment

Figure 3.4: Extraction of final aggregated model. The final model (c) is obtained by iterative removal of CDT triangles from the aggregated model at (b) corresponding to the neighborhood graph of the aggregated environment in (a) for a given $\delta = 2.5$. Original objects are shaded gray and CDT triangles in black wireframe. Reprinted, with permission from [7], ©2016 IEEE.

width and the distance between the objects as given by:

$$\delta_i{}^{sd} = \alpha * (d_{min} + \sigma_d) + (1 - \alpha)(\delta_i)$$

where $d_{min}$ is the minimum distance between the grouped objects, $\sigma_d$ is the variation of the passage width between the objects and $\alpha$ is a linear weighting ($0 \leq \alpha \leq 1$). A triangle/tetrahedron in the CDT is removed if it contains an edge whose length is greater than $\delta_i{}^{sd}$.

The equation balances $\delta_i$ with minimum distance connection and topological variations. Including the minimum distance creates connections closer to minimum edge connection and hence avoid wide connections (that can cover necessary free spaces). The standard deviation incorporates the topological variations of the passages between the objects to create the connection. It allows to aggregate objects having large variations in passage width with wider connections than those having small variations in passage width. For example, the passage between two nearly rectangular objects will have a single level of connection in the hierarchy. In our experiments, we use $\alpha = 0.5$ ($\alpha = 0$ produces shape cover similar to alpha shapes).

The iterative removal of triangles/tetrahedra to define the cover of the aggregated objects is specific to our implementation. However, other shape covers such as skin surfaces can be used. Any

single connections (such as trailing edges with no adjacent faces) are removed in post processing.

### 3.1.2 Levels in Hierarchy

We vary $\delta_i$ to create different levels in the aggregation hierarchy. When creating the levels in the hierarchy, one must ensure that the levels are distinct so that the aggregated environment at one level is not same as in other levels in the hierarchy. This depends on the choice of starting $\delta_i$, how to decrease its value to create a new level and removal of unnecessary levels.

#### *3.1.2.1 Finding $\delta_i$ for Each Level*

The weights of the edges in the neighborhood graph of the original environment can be used as different $\delta_i$ values. This would ensure that the number of connections are distinct at each level. For large environments with simple objects such as city like environment with nearly orthogonal shapes, levels determined by the distinct edge lengths in the initial neighborhood graph will create distinct and manageable number of levels. However, the hierarchy might not consider some levels especially in environments with heterogeneous complex objects as only the minimum distance between the objects determines the levels in the hierarchy.

So we consider all possible distance between the objects as $\delta_i$ values to be used to create the hierarchy. In our implementation, we approximate the minimum distance between objects from the length of the CDT edges connecting the objects. So we consider all the distinct CDT edge lengths as different $\delta_i$ values. To avoid creation of unmanageable number of levels, we consider the edge lengths which are different by at least $\epsilon$ (where $\epsilon$ is a user-defined threshold based on the size of the environment).

#### *3.1.2.2 Adapting the Basic Hierarchy*

We use the levels in the hierarchy of the obstacle space to decompose the free space. The basic hierarchy may construct unnecessary levels such that the space freed in the level is not large enough for processing causing over-segmentation. These unnecessary levels can hinder performance and create unmanageable number of levels which are not significantly different. The basic hierarchy can be filtered such that levels with insufficient free space for a given volume are removed. While

this addresses the performance issue for a single run, the hierarchy is specific to a single threshold volume and cannot be reused if the threshold volume changes. Frequently, different robots (of varying volume) navigate the same environment and hence utilize the same hierarchy. Thus this solution is inefficient long term.

Here we dynamically adapt the hierarchy using volumetric tuning to reuse the same hierarchy for different robot sizes. The basic or untuned hierarchy is created independent of any volumetric tuning as described above. Each level in the untuned hierarchy is augmented to store information about the volume of the space freed at that level from the previous level. Note that this information is readily available during hierarchy construction and does not incur significant overhead verses the original method.

Tuning occurs as requested. The threshold volume is passed as an input parameter to the hierarchy. Given this volume, levels with insufficient free volume in the un-tuned hierarchy are combined with their children into a single level until the freed volume is greater than the threshold volume. This combined level is then returned to requesting application. The volume based tuning also avoids creation of small connections.

This dynamic tuning creates different specialized views of the same hierarchy for different versions of the problem using the same environment (e.g., robots of different sizes working in the same environment). Thus, the hierarchy may be reused for different volume inputs at a significant savings over reconstruction.

#### *3.1.2.3 Local Refinements in Aggregation*

The basic hierarchy can block essential narrow passages that might be required to find the solution path. In such cases, the aggregation hierarchy will have minimal to no improvement in planning time. Local refinements in the hierarchy will allow detailing the environment in necessary regions and blocking the unnecessary regions (e.g., those farther away from the solution path). This would retain the performance improvement observed by using the aggregation hierarchy in problems where the solution path lies through wide regions.

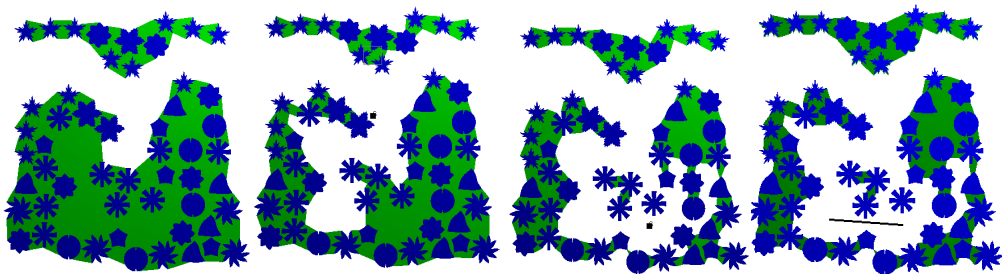
Often, while visualizing a virtual environment, objects that are closer to the point of vision

are more detailed than the objects farther away. We extend the aggregation hierarchy to emulate a similar effect through *focus-points* and *focus-edges*. These are places of interest in the environment where objects closer to them are more refined than objects farther away. We use points and edges for adapting the level of aggregation as they can express both particular places of interest (focus-points) and trajectories of interest (focus-edges) which are useful in path planning.

This adaptive refinement is achieved by scaling the distance threshold for each object based on its proximity to a focus-element. Given a $\delta_i$ for level $i$ of the hierarchy, we define a scaled threshold $\delta'_i$ for object $o$ as follows:

$$\delta'_i(o) = \delta_i * \frac{d(o)}{d_{max}}$$

where $\delta_i$ is the distance threshold for level $i$, $d(o)$ is the distance of object $o$ to the nearest focus-element and $d_{max}$ is the distance of the farthest object to the focus-element witness of $d(o)$. Thus, the distance threshold used to aggregate objects is no longer constant throughout the environment at a particular level in hierarchy.



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

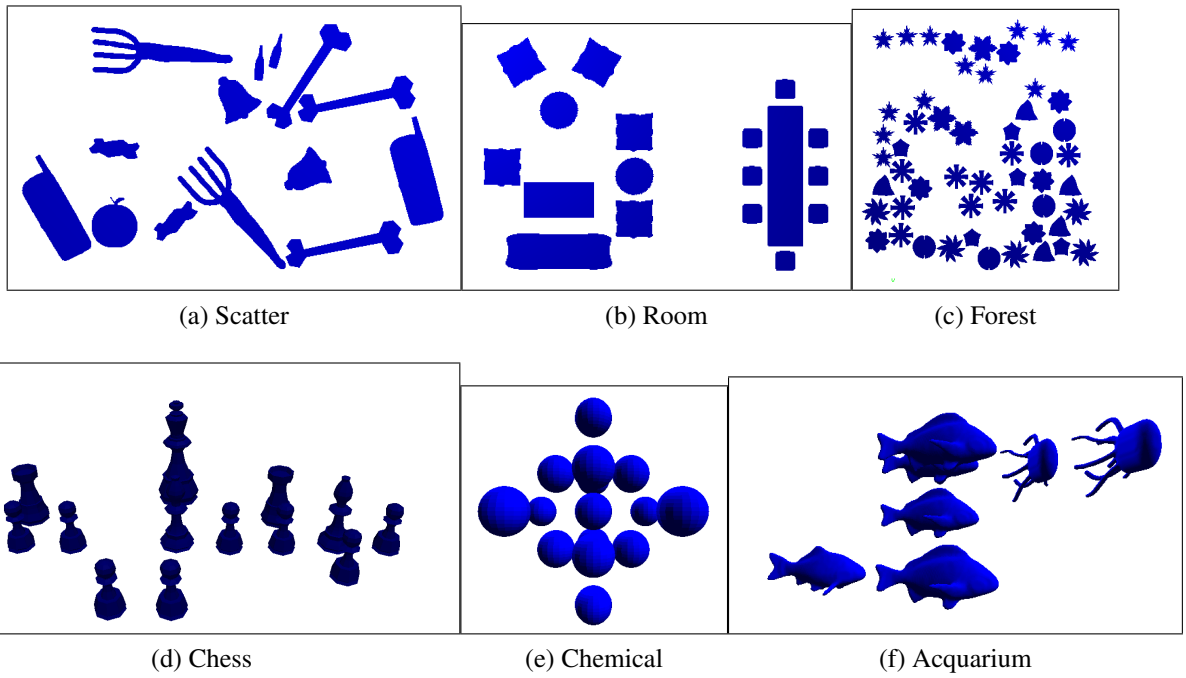
Figure 3.5: Effect of local refinements. An example aggregation (a) may be refined near different focus points (b-c) and focus-edges (d). Focus-elements are drawn in black. Original objects in blue and the aggregated region is shown in green.

Fig. 3.5 compares the resulting aggregation without (a) and with focus-points (b-c) and focus-edges (d). Their placement lightly aggregates the objects near them as compared to objects farther away. Note that the number of levels created in the hierarchy with focus-points and focus-edges

will be less than without.

These focus-elements are customized for a given input environment and may be obtained in a variety of ways. The simplest method is from user input. Focus-points and focus-edges may also be automatically determined as planning proceeds. Few examples of determining the position of focus points and edges in context of motion planning applications is discussed in detail later in Section 3.3.2.4.

## 3.2 Experimental Results



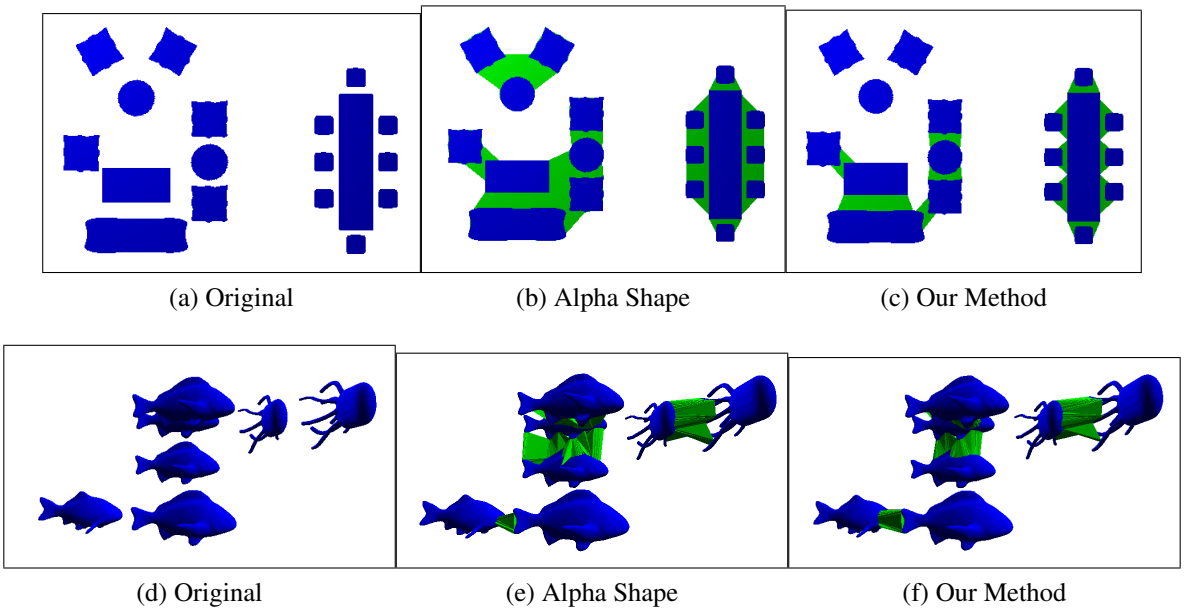
(a) Scatter (b) Room (c) Forest

(d) Chess (e) Chemical (f) Acquarium

Figure 3.6: Experimental environments for evaluation of aggregation hierarchy. (a-c) 2D and (d-f) 3D environments used in experiments.

We implemented our method for environments with 2D and 3D objects. We evaluated our approach in terms of efficiency and shape quality. We compared our shape approximation of the groups of objects with that created by alpha shapes (with $\alpha = \frac{(min+\delta)^2}{16}$ for a fair comparison as $\sqrt{\alpha}$ is the radius of the ball used in the alpha approximation and $min$ is the average of the

minimum distances between the objects). We remove edges in the alpha shapes approximation if they connect vertices from the same object in order to have a close comparison to our method. We use the single-linkage agglomerative clustering of polygons proposed in [17] to compare the construction time of the hierarchy for 2D environments. We used 6 different environments, as shown in Fig. 3.6 (ordered by their total number of vertices). For construction of the CDT, we used the libraries Triangle [69] for 2D and Tetgen [70] for 3D.



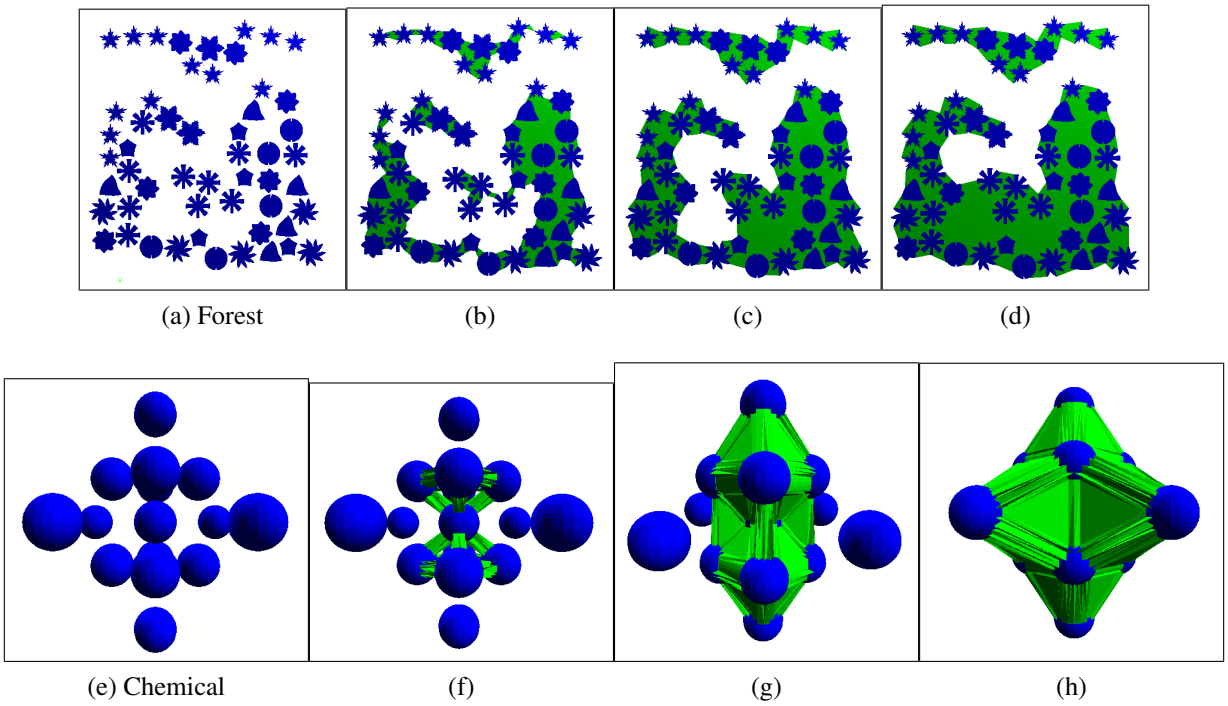
(a) Original (b) Alpha Shape (c) Our Method

(d) Original (e) Alpha Shape (f) Our Method

Figure 3.7: Comparison of shape covers. Original and aggregated environments using alpha shape and our method of approximation with the same $\delta_i$ value for (a)-(c) Room environment ($\delta_i$ = 35), and (d)-(f) Aquarium environment ($\delta_i$ = 7). Original objects in blue and the aggregated region is shown in green.

Fig. 3.7 shows the aggregated objects at a particular level with the same $\delta_i$ using both shape approximation methods (alpha shapes and aggregate function) in 2 different environments. The $\delta_i$ value in these levels for each environment is stated in the caption. For the same $\delta_i$, our method results in an aggregation creates necessary connections (i.e., aggregated model is closer to the original environment) than alpha shapes approximation. This is due to balancing of the value of

the aggregate function with respect to the minimum distance of the objects, variation in distances between the object, and the threshold distance $\delta_i$. Also, the connections formed by our method are narrower than alpha shapes approximation.



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

(e) Chemical (f) (g) (h)

Figure 3.8: Different levels in hierarchy. Different levels in (a)-(d) a heterogeneous 2D Forest environment and (e)-(h) a homogeneous 3D 15-spheres Chemical environment.

Fig. 3.8 shows some of the levels in the hierarchy created by our method for two different environments: a 2D heterogeneous forest environment and a 3D homogeneous chemical environment.

### 3.2.1 Quality Comparison

We use the aggregation hierarchy in decomposition of the free space where the components are the free space difference between consecutive layers of the aggregation hierarchy. We evaluate the quality of the shape cover of the aggregated models in each level of the hierarchy by comparing the ratio of volume of the free space cleared at each level to the volume of the axis-aligned bounding box of the free space. The ratio is computed for each connected component and averaged over

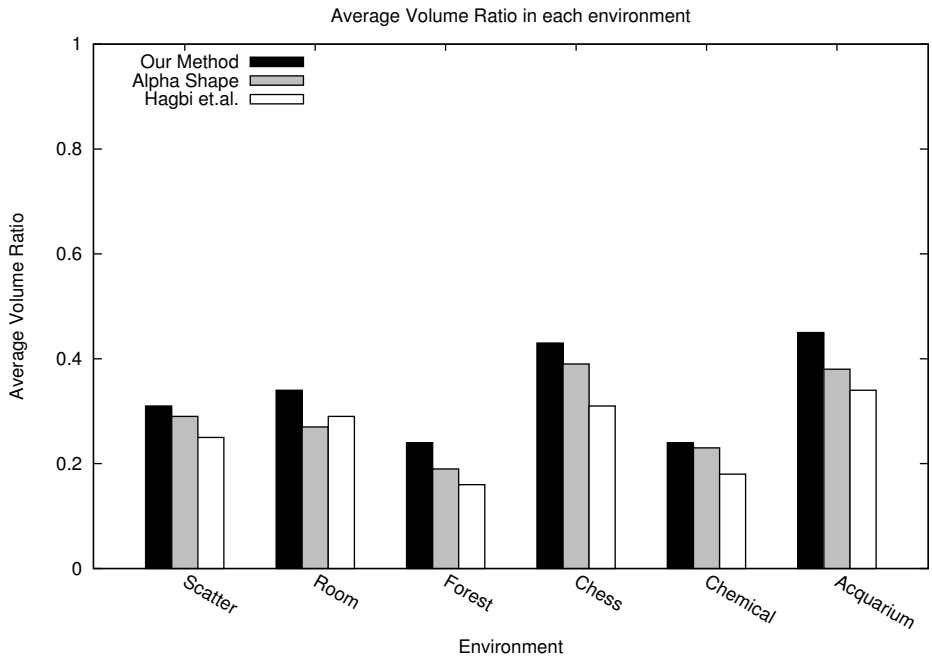



Figure 3.9: Average free space volume ratio of different shape covers for aggregation.

all connected components produced by the hierarchy. Fig. 3.9 compares the average volume ratio measure for the 6 environments.

The figure demonstrates that the measure is less in alpha shapes and the approach used in [34] than our method. This implies that the bounding box of the free space cleared at each level in our method includes less obstacles than alpha shapes and the approximation in [34].

### 3.2.2 Construction Time

Fig. 3.10 shows the construction time of the hierarchy for each environment. The result shows an increase in time with an increase in the number of total vertices in the environment. Also, the time depends on the dimension of the environment. Fig. 3.10 also compares the hierarchy construction time for 2D environments with the single-linkage clustering used in [17] marked as "Chang et.al." in the plot. For 3D environments, the hierarchy (marked as "Chang et.al." in the plot) is created by using the distance metric and clustering approach used in [17] but the aggregated objects are constructed using alpha shapes.

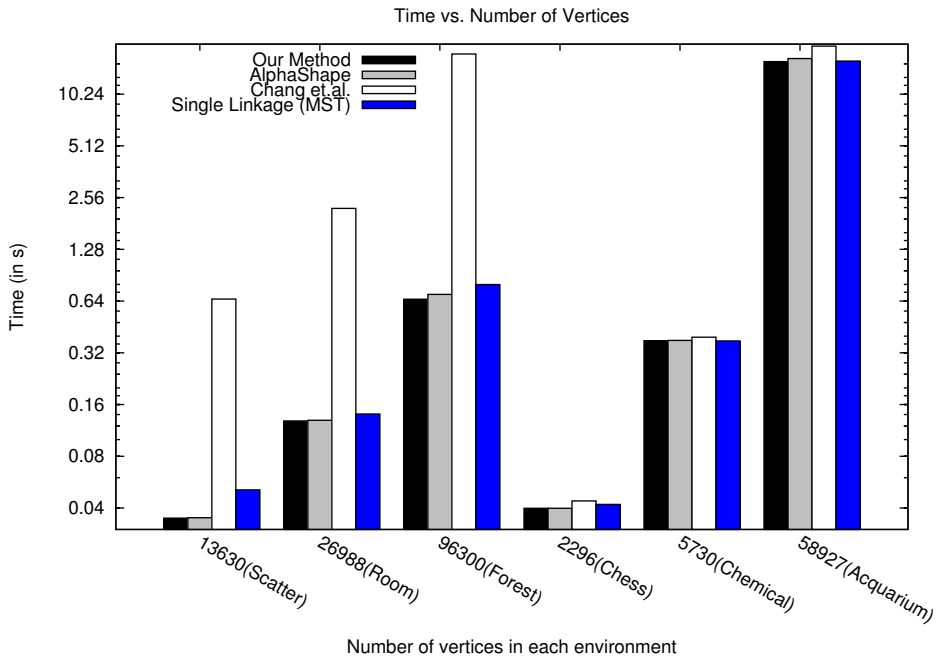



Figure 3.10: Construction time comparison of different aggregation methods.

Our method of aggregation is faster (even using alpha shapes to approximate the groups) than the method stated in [17]. As stated in [17], the complexity is heavily dependent on updating the distance metric and inter object distance calculation. We used a brute-force method to find the minimum distance between two objects by computing the distance between every pair of vertices and finding the minimum. We can use a minimum spanning tree algorithm over the neighborhood graph to create a single-linkage agglomerative clustering hierarchy in less time. The number of levels in the untuned hierarchy created by the agglomerative single-linkage clustering is limited by number of objects in the environment and hence, does not represent the evolution of free space as completely as the hierarchy created using our method. As shown in Fig. 3.10, the single-linkage clustering (shown as "Single-Linkage (MST)" in the plot) takes almost same time as our algorithm.

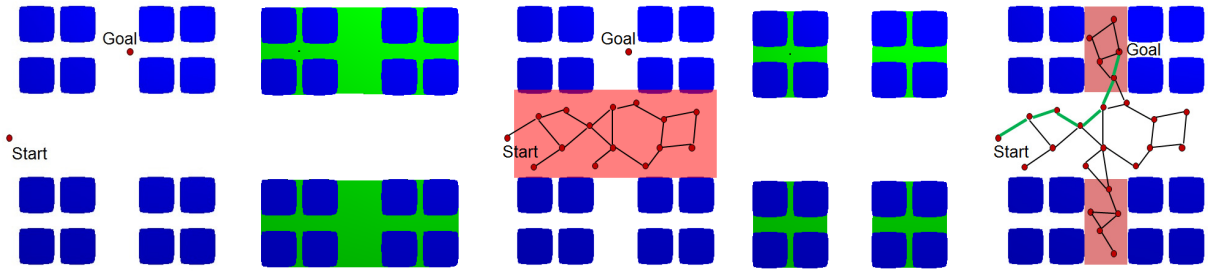
Hierarchy tuning using volumetric tuning does not improve the time efficiency. This is because the time is heavily dependent on the complexity of the CDT construction algorithm and finding the level values and free space volume difference. As both the approximations using alpha shapes or our method are a linear time operation dependent on the number of CDT edges/faces, differences

in this function are not reflected in the time efficiency. However, if an aggregate function required computation that does not depend linearly with number of edges/faces in CDT, the complexity will increase or decrease accordingly with the level of aggregation.

### 3.3 Application to Motion Planning

We apply hierarchical aggregation to workspace obstacles to improve the performance of sampling based motion planners. The aggregation hierarchy not only simplifies the number of obstacles to handle for collision checking, it also represents the evolution of the free space as one moves down the hierarchy. We take advantage of this free space evolution identification by focusing sampling in these regions.

#### 3.3.1 Overview




(a) Original Environment (b) Level N-Aggregation (c) Level N-Planning (d) Level N+1-Aggregation (e) Level N+1-Planning

Figure 3.11: Application of aggregation hierarchy in a motion planning problem. Given a motion planning problem in an environment (a), use the levels in the aggregation hierarchy (b) & (d) to apply the planning strategy by sampling in the regions freed at that level (c) & (e) until the problem is solved (e). Original objects in blue and the aggregated region is shown in green. Sampling regions are marked as translucent red. Reprinted, with permission from [7], ©2016 IEEE.

The core idea behind applying the aggregation hierarchy is to iterate through the various levels of the hierarchy until the motion planning problem is solved. Given an input planner, we start at the highest (coarsest) level of the hierarchy and call the input motion planner operating at that aggregation level. If the planner has not solved the problem yet, we iteratively reduce the aggregation

level (i.e., increase the level of detail) and call the planner again on the new aggregation level. At each reduction, we retain the collective progress of the planner, identify areas to focus planning based on the difference between the approximation models of the different levels, and continue planning. If the problem is not solved on reaching the lowest level of aggregation (i.e., the original input problem), then the number of samples created is insufficient to solve the problem. So to create more samples, we repeat the entire process starting back at the coarsest level while always retaining the progress made thus far. Algorithm 3 provides a sketch of the process.

**Algorithm 3** Hierarchical Sampling for Motion Planning

**Require:** An environment $E$, a motion planning problem $MP$, and a sampling based planner $P$.

1: Create the aggregation hierarchy, $H(E) = \{l_0, l_1, ..., l_n\}$ where $l_i$ is a level in hierarchy.
2: $i = 0$.
3: **while** $MP$ not solved **do**
4: Let $R_i$ be the set of regions freed at level $l_i$.
5: Let $B_i$ be the set of bounding volumes for $R_i$.
6: Apply $P$ to $MP$, restricting new samples to $B_i$ but allowing connections in all of $E$.
7: $i = i + 1$.
8: **if** $i = n$ **then**
9: $i = 0$

As stated earlier, the regions of space freed by the consecutive levels in the hierarchy can be easily computed. They are the difference in free space between the aggregated environments in the current level and its parent level. In Fig. 3.11c and Fig. 3.11e, these regions are shown in translucent red. Restricting sample creation to the freed regions of space not only focuses the sampling space, but it also increases the probability that the new samples are valid. We can approximate the freed regions using a variety of bounding volumes such as bounding boxes, bounding spheres or convex hulls to ease sampling.

Although the regions of space freed at a particular level might be disconnected (e.g., regions $A$ and $B$ in Fig. 3.11e), this is not an issue since we are only restricting the creation of new samples, not their connection. As shown in Fig. 3.11e, regions marked $A$ and $B$ are disconnected.

For our implementation, we generate samples in the set of axis aligned bounding boxes as an approximation of the regions freed at each level. One issue in using axis aligned bounding boxes is that the improvement in performance observed is dependent on the orientation of the obstacles with respect to the world coordinate frame. We could directly sample in the sets of triangles/tetrahedra removed at the current level but it might incur computation overhead as these sets might not be convex. Hence, in this initial implementation we use axis aligned bounding boxes because they are fast to compute, easy to derive sampling bounds on, and completely cover the freed regions.

#### *3.3.1.1 Use of Local Refinements in Aggregation while Planning*

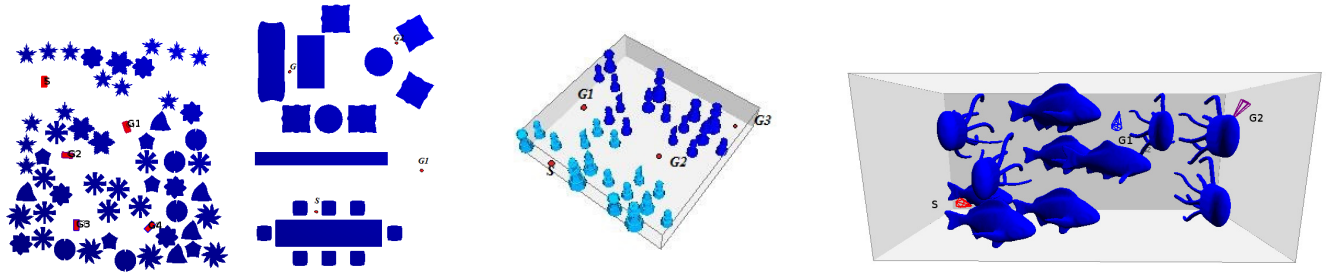
The basic aggregation hierarchy can block crucial narrow regions (important with respect to the solution path) at many of its levels requiring planning at the lowest (finest) levels. This affect possible performance gains as planning is artificially forced to occur at the finest levels of detail to find the path. We use the local refinements through focus points and focus edges to refine the hierarchy near them.

Waypoint placement is often used to guide planning [71, 72, 73, 74, 75] and provides a natural placement of focus-points to restrict aggregation. In our implementation, we use query tours as the user-defined focus-point or focus-edge input.

Focus-points and focus-edges may also be automatically determined as planning proceeds. For example, to find a path between the start and the goal, they must belong to the same connected component in the roadmap. Thus, it is natural to focus planning to connect the component containing the start ($CC_s$) to the component containing the goal ($CC_g$). We then define two focus-points at each iteration: the configuration $s' \in CC_s$ closest to the goal and the configuration $g' \in CC_g$ closest to the start. When working with focus-edges, we instead use the straight lines: between $s'$ and $g'$, between $s'$ and the witness/closest configuration in $CC_g$, and between $g'$ and the witness/closest configuration in $CC_s$. Unlike user-defined placement, these automatic methods will dynamically adapt their location to the planner's progress.

### 3.3.2 Results

In this section, we study the performance improvement provided by our method using several different planners: uniform sampling (BasicPRM) [46], GaussianPRM [50] and Obstacle Based PRM (OBPRM) [76]. These were selected as they produce a variety of sampling distributions: uniform (BasicPRM) and near obstacle surfaces (GaussPRM and OBPRM). Although the sampling distributions are very different, our aggregation hierarchy improves the planning efficiency for all of them.




(a) Forest (b) Room (c) Chessboard (d) Aquarium

Figure 3.12: Environments for evaluation of aggregation hierarchy in motion planning. (a),(b) 2D environments and (c),(d) 3D environments used in the experiments. Reprinted, with permission from [7], ©2016 IEEE.

We study several different environments, see Fig. 3.12. The Room and Chessboard environments are for point robots in 2 and 3 dimensions. The robot for the Forest environment (Fig. 3.12a) is a rectangular rigid body constrained to the plane, and thus it has 3 degrees of freedom. The robot for the Aquarium environment (Fig. 3.12d) is a free-flying pyramid with 6 degrees of freedom. Each environment shows the query or tour it needs to solve.

In our experiments, we tune the hierarchy such that the volume of the difference in free space between the levels is at least 10% of the total volume of the environment. This prevents creation of an unmanageable number of levels in the hierarchy.

Each method is run until either the query is solved or 10,000 samples are generated. All methods use Euclidean distance, straight-line local planning, and a $k = 5$-closest neighbor connection strategy. We compare the total planning time and the sampling success rate (computed as percentage of valid samples) for each method. All results are averaged over 10 runs.

#### *3.3.2.1 Performance*

Fig. 3.13a shows the running time of our method with respect to the different underlying sampling based motion planners. It also shows the part of the time used in the creation of the hierarchy. Times are normalized to the time taken by the planner without the aggregation framework.

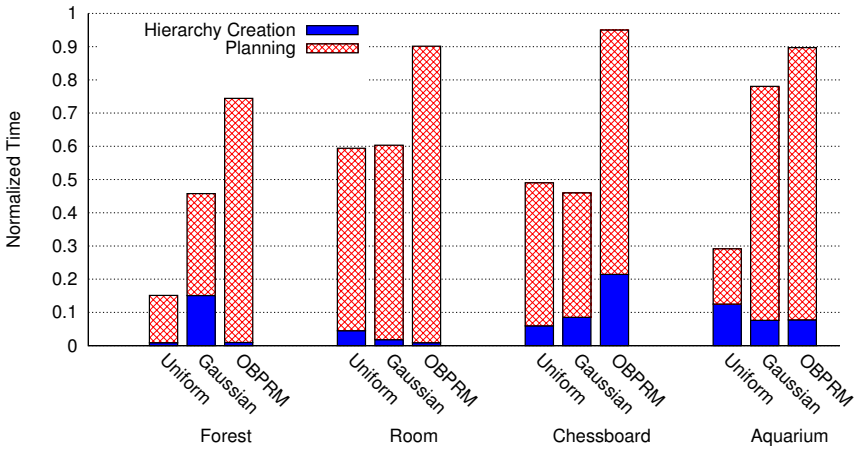
Across all environments and all planners, the aggregation framework is faster than the original planner. In half the cases, the aggregation framework is less than 60% of the original planning time. Note that while aggregation occurs in the workspace, performance improvements extend to rigid body cases as well (i.e., the Forest and Aquarium environments) where $\mathcal{C}_{space}$ and workspace are not the same. The time taken to create the hierarchy is relatively small (less than 25% of the total planning time in most of the cases).

Fig. 3.13b shows just the portion of the running time spent generating samples. Again, times are normalized to the time taken by the underlying planner to sample without the aggregation framework. We see a similar pattern of improvement in the sampling time as in the overall running time. Thus, most of the performance improvement comes from the effect of aggregation on sampling.

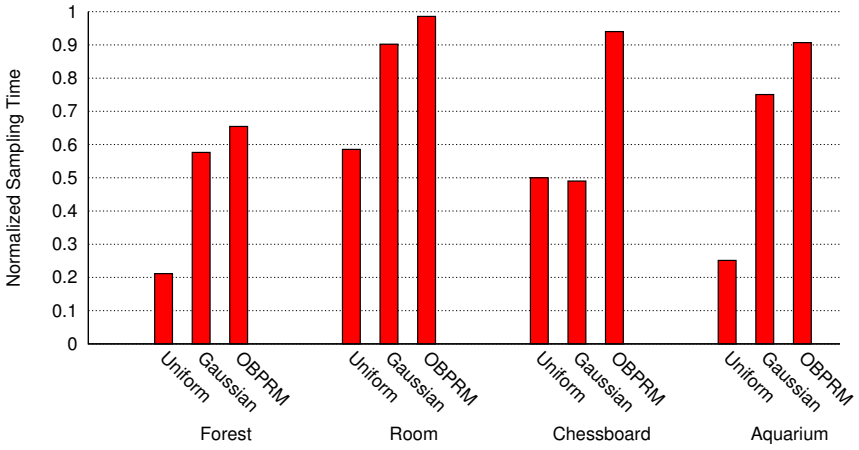
Except for OBPRM sampling, the sampling time is improved significantly through using the aggregation hierarchy framework. This is the result of restricting the sampling in regions with low obstacle density as compared to sampling the entire space.

#### *3.3.2.2 Percentage of Valid Samples*

Fig. 3.14 compares the sampling success rate with and without using aggregation hierarchy. The success rate is computed as the percentage of valid samples out of all sampling attempts. We normalize the success rate to that of BasicPRM without the aggregation framework in each




(a) Total Planning Time




(b) Sampling Time

Figure 3.13: Time efficiency with aggregation hierarchy and various underlying planners. Normalized (a) total running time and (b) sampling time of different aggregation methods with various underlying planners for each environment. Data is normalized to the time spent by each underlying planner without the aggregation framework in each environment. Reprinted, with permission from [7], ©2016 IEEE.

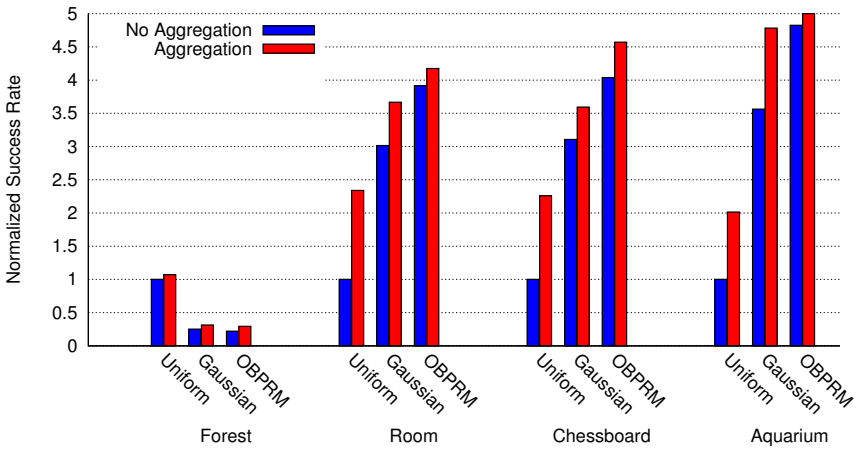



Figure 3.14: Sampling success rate with aggregation hierarchy and various underlying planners. Data is normalized to the sampling success rate of BasicPRM without the aggregation framework in each environment. Reprinted, with permission from [7], ©2016 IEEE.

environment.

We see that the success rate improves using the aggregation hierarchy irrespective of the original success rate of the underlying sampling method. By identifying regions freed between aggregation levels and focusing sampling there, the framework boosts the success rate for all the samplers.

#### *3.3.2.3 Comparison to Quad/Octree Decomposition*

We compare the performance of using decomposition of the free space based on aggregation hierarchy to that based on quad/octree. Similar to aggregation hierarchy, we use the quad/octree decomposition of the free space and initialize planning at the root. At each level the planning is performed on the quadrants/octants that is completely in the free space (i.e, the leaf nodes representing free space at each level). Similar to aggregation hierarchy, iterate through each level in the octree hierarchy till the solution of the problem is achieved.

The quad/octree decomposition is constructed from the voxel map representation of the envi-

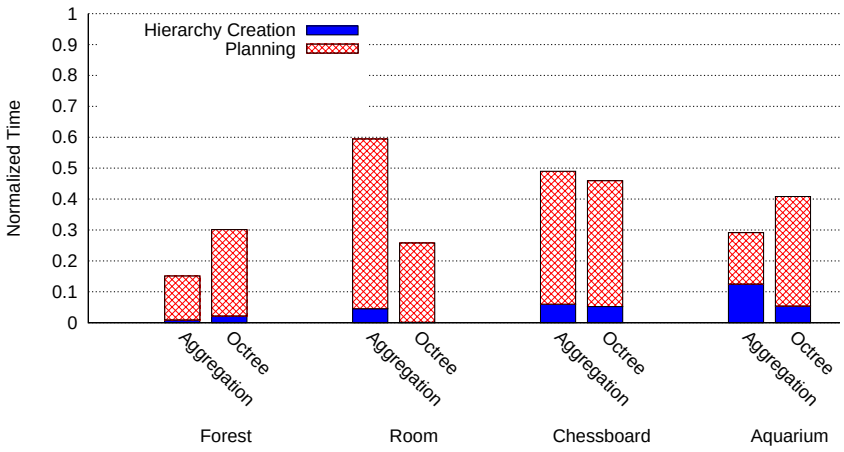



Figure 3.15: Planning time of different decomposition methods for free space. Data is normalized to the planning time of BasicPRM without the decomposition framework in each environment.

ronment. As the construction of the voxel map is implementation dependent, we do not report this time in the hierarchy creation (initialization) time for fair comparison with the aggregation hierarchy. Fig. 3.15 compares the running time to solve the motion planning problems in Fig. 3.12. The underlying planning is Basic PRM (uniform sampling).

Using a decomposition of the free space based on the aggregation hierarchy of the obstacle space, we get performance benefit either better or comparable to that using quad/octree decomposition (with exception of the Room environment). In environments such as Room and Chessboard where the number of obstacles are few and mostly axis-aligned, we observe better performance improvement on using quad/octree decomposition. For environments such as the Forest and Acquarium, the performance improvement using aggregation hierarchy is better than quad/octree decomposition of the free space. This can be attributed to the non-axis aligned nature of the free space and cluttering of the obstacles which causes division of the otherwise contiguous free space at different levels in the octree decomposition based on boundaries of the octants. Whereas in aggregation hierarchy, the free space is divided whenever there is a change in the distance between

the obstacles which captures the wider regions irrespective of the alignment and placement of the obstacles with respect to the environment axes.

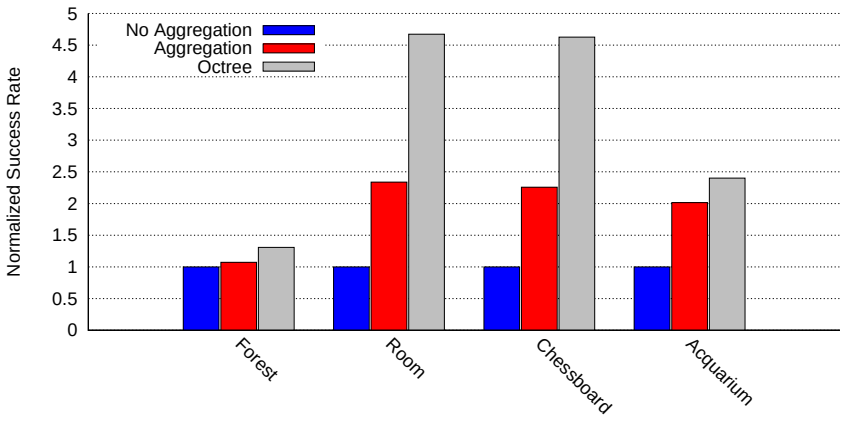



Figure 3.16: Success rate of different decomposition methods for free space. Data is normalized to the success rate of BasicPRM without the decomposition framework in each environment.

Fig. 3.16 compares the sampling success rate for different decomposition of the free space: by aggregation of obstacle space and octree decomposition. The data is normalized to the success rate of Basic PRM planning (uniform sampling) in each of the environments. The figure indicates the success rate is higher on using octree decomposition. This can be attributed to the completely free octants or sampling regions at the higher levels of the hierarchy compared to the sampling regions in the aggregation hierarchy. The sampling success rate with aggregation hierarchy is comparable to the octree decomposition in Forest and Acquarium environment. This is because of the added dimensions of the rotation parameters of the robot's body causing the sampling regions even in the octree decomposition to contain configuration space obstacle and thereby affecting the sampling success rate.

To summarize, use of octree decomposition of the free space is recommended where the groups

of obstacles are nearly axis-aligned with respect to the environment's axes. Example environments will be organized or structured environments like office or city plan. For other unstructured environments, using aggregation hierarchy for planning in free space provides a comparable or better performance improvement.

#### *3.3.2.4 Results of Local Refinements in Aggregation Hierarchy*

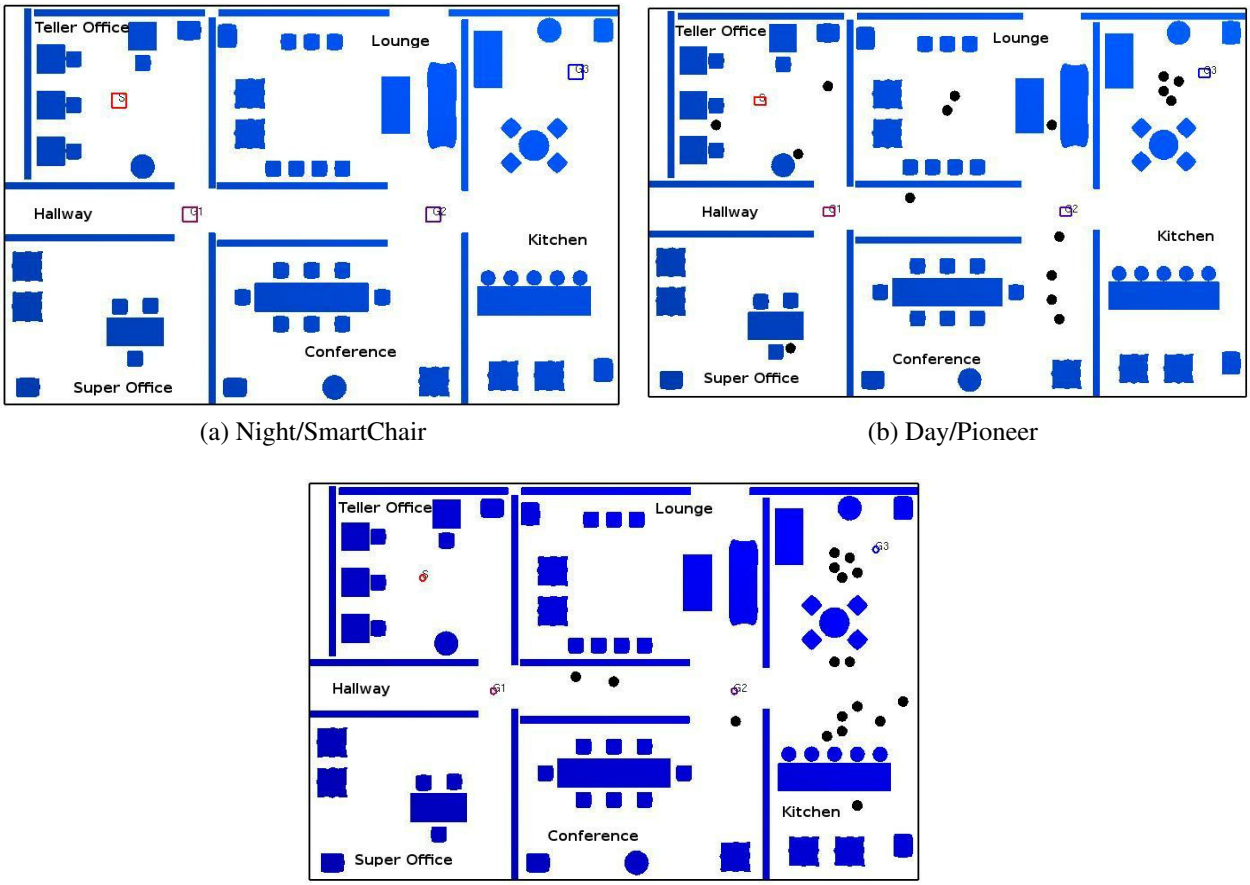
Remember, the objective of using local refinements is to utilize the benefits offered by aggregation hierarchy in environments containing narrow or cluttered areas (such as narrow passage ways, cluttered furnitures, etc).

We showcase our refinement tools on a typical office floor (Fig. 3.17) containing several office rooms, a kitchen, a lounge and a conference room connected by a hallway. We look at three typical occupancy levels: low at night, moderate during a workday and high for a celebration in the kitchen. People or agents are displayed as black disks. We also model three different robots of varying size: a SmartChair [77] for the mobility impared (large), a Pioneer P3-DX robot [78] to collect trash from trash bins (medium) and an iCreate robot [79] to vacuum the floor (small). We also show the effect of refinements in Acquarium environment (Fig. 3.12d).

We compare the impact of refining the hierarchy from user input and from automatic focus-element placement against the basic (unrefined) aggregation hierarchy in the office set of environments.

Fig. 3.18 shows the total planning time of basic aggregation and various focus-refined schemes for different underlying sampling based motion planners in each environment. Times are normalized to the time taken by the underlying planner without the aggregation framework. Note that the time also includes the time taken to create the hierarchy.

Aggregation of any form improves the total planning time across all environments and independent of the underlying planner used (i.e., all normalized times are $< 1$). In all cases, focus-point and focus-edge refinement improves performance over basic aggregation for both the user-defined set and the automated set. We also see that focus-edge refinement consistently performs better than focus-point refinement (regardless of placement scheme). This is due to the fact that edges better




(a) Night/SmartChair

(b) Day/Pioneer

(c) Celebration/iCreate

Figure 3.17: Environments to show effect of local refinements. Planar office environment with three different occupancy levels (low/night, moderate/day, high/celebration) and three different robots (large/SmartChair, medium/Pioneer, small/iCreate). Each environment shows the query tours used. People are displayed with black disks.

represent the regions between roadmap connected components where planning should target.

Interestingly, automatic focus-point/focus-edge detection out-preforms or is comparable to user-defined placements in every situation even though the user-defined sets are placed at/between every point in the query tour input. This is because automatic placements change throughout planning to adapt to the current progress of the planner. In particular, the planner should focus on areas of need which often change throughout planning. Static user-defined placement (as in the query tours) typically identify critical passages, but they fail to identify areas of good coverage and poor connectivity as this is planner-specific and changes over time. Automatic placement, however, fo-

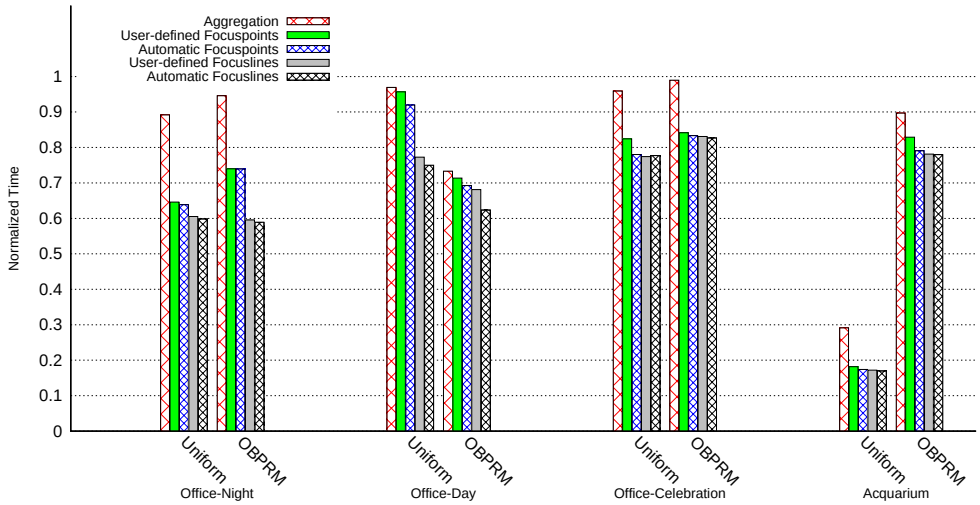



Figure 3.18: Total running time of basic and different focus-refined aggregation methods. Data is normalized to the time spent by each underlying planner in each environment without the aggregation framework (non-aggregation bars not shown for clarity).

cuses on these areas precisely which proves more important to overall planning performance. They also eliminate any requirement for additional user input and are adaptable to changing query tours.

Fig. 3.19 compares the sampling success rate for each aggregation scheme (no aggregation, basic aggregation, aggregation with user-defined focus-refinements, and aggregation with automatically placed focus-refinement). The success rate is computed as the percentage of valid samples out of all sampling attempts. We normalize the success rate to that of BasicPRM without the aggregation framework in each environment.

Except for OBPRM sampling in the office night-time scenario, the sampling success rate is improved through using any version of the aggregation hierarchy framework. This is the result of restricting the sampling in regions with low obstacle density as compared to sampling the entire space. Using the refinements either user-defined or automatic causes a decrease in success rate compared to the aggregation framework without any refinements. This is due to the sampling re-

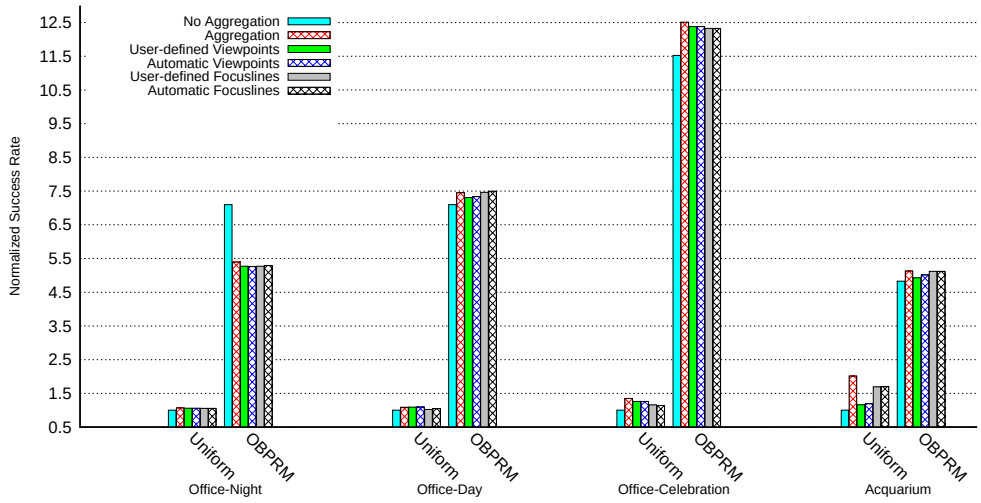



Figure 3.19: Success rate of basic and different focus-refined aggregation methods. Data is normalized to the sampling success rate of BasicPRM without the aggregation framework in each environment.

gions in the aggregation framework with refinements containing more obstacles than that without any refinements. However, the success rate using aggregation with refinements is generally comparable to that without aggregation framework in crowded environments. Even though the sampling success rate for OBPRM in the office at night decreases with aggregation, this is more than made up for by focusing sampling in regions at the proper level of detail as can be seen by improved total planning times (see Fig. 3.18).

# 4. SHAPE PRIMITIVES APPROXIMATION [1]

Simple shapes such as shape primitives (e.g., sphere, box) are often preferred in approximating complex shapes. The convexity and the parametric definition of the shape primitives is useful for quick containment checks.

In this work, we place primitive shapes inside any bounded space such that they are completely inside the space and represent the topology of the space. This encodes the information about the connectivity of the space in the approximation. Instead of using a single type of primitive in the skeleton, we use of variety of them. This makes the approximation applicable to a variety of applications from rectilinear spaces in simulation applications to tubular structures in biological applications.

In this chapter, we describe how the shape primitive skeleton is constructed and how they are used to improve the performance of motion planning algorithms.

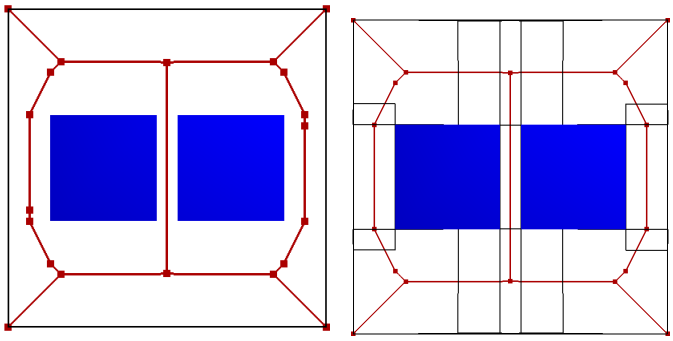
## 4.1 Approach

In this section, we provide the details of the shape primitive skeleton construction method. We use a topological skeleton of a bounded space and annotate it with shape primitives. The annotation is done by finding a best fit shape primitive from a candidate set for each skeleton vertex and edge.

### 4.1.1 Skeleton

The topological skeleton of a bounded space is a graph embedded in the space that represents the topology of the space (i.e., the same number of connected components and holes as the space). The skeleton vertices are points in the space and the skeleton edges are curves connecting the skeleton vertices (for example the red graph in Fig. 4.1a). Examples of skeletons are the medial axis in 2D [80], Reeb graphs [81] and the Mean Curvature Skeleton in 3D [82].

[1] The description of the method and some experimental results are reprinted from a post-peer-reviewed, pre-copyedited version of the work: "Fast Collision Detection for Motion Planning using Shape Primitive Skeletons " by M. Ghosh, S. Thomas, N. M. Amato, in the Proceedings 2018 *International Workshop on the Algorithmic Foundations of Robotics* (WAFR),[8].



(a) Example Skeleton (b) Shape Primitive Skeleton

Figure 4.1: Example of curve skeleton and corresponding shape primitive skeleton. (a) Topological skeleton of a bounded space (shown in red) and (b) corresponding shape primitive skeleton of the space with primitives (boxes in this case) annotated to the skeleton. The obstacles are shown in blue and the primitives are transparent with black boundaries.

#### *4.1.1.1 Preparation of Skeleton for Annotation*

In order to place shape primitives to provide a good coverage of the space, the topological skeleton should have the following characteristics:

- *Medially Placed:* The skeleton should lie on the medial axis. As medial skeletons are the farthest from the boundary, placing shape primitives on a medial skeleton ensures good coverage.

- *Line Skeleton:* It is hard to place shape primitives on curved edges. The skeleton should be simplified to a line skeleton.

- *Clearance Annotated*: In order to ensure proper and maximal inscription, the skeleton should be annotated with clearance information (i.e., distance to the nearest point on the boundary).

- *Boundary Mapping:* To find the best fit shape primitive type, the skeleton should contain the mapping to the corresponding points on the boundary. These are generally the points in the boundary that generates a point (vertex or intermediate point in edges) in the skeleton.

### 4.1.2 Annotation of Shape Primitives

Given a skeleton, we place the shape primitives on each skeleton component (vertices and edges). We iterate through each skeletal component based on a priority score. A best fit geometric shape primitive from a set of candidate shape primitives is placed if the skeleton component is not already covered by another shape primitive. Algorithm 4 provides an overview of the algorithm.

**Algorithm 4** Shape Primitive Skeleton

**Require:** Object $O$, Vertex Shape Primitive Candidates($VP$), Edge Shape Primitive Candidates($EP$), Scoring Function(SCORE($el$))
**Ensure:** Set of Shape Primitives $P$
1: $G(V, E) \leftarrow$ Skeleton($O$)
2: Let skeleton elements, $EL = V \cup E$
3: Sort $EL$ based on SCORE($el$)
4: **for** every $el$ in $EL$ **do**
5:  **if** $el$ is in $V$ **then**
6:  $p$ = BEST-FIT-SHAPE-PRIMITIVE($VP$,$el$)
7:  **else**
8:  $p$ = BEST-FIT-SHAPE-PRIMITIVE($EP$,$el$)
9:  **if** $p$ does not overlap significantly with a previously placed shape primitive **then**
10:  Insert $p$ in $P$

In our implementation, the skeleton components are scored based on the unoccupied volume surrounding the skeletal component. The volume is approximated using a grid map of the environment. The score of a skeletal component is the number of unoccupied grid cells that are at minimum clearance distance away from the component. The scores are updated for neighboring skeleton components after placement of a shape primitive.

#### *4.1.2.1 Placement of Candidate Shape Primitives*

In our implementation, the candidate set for vertices includes spheres and boxes and for edges includes ellipsoids, boxes and capsules (swept volume of a sphere along an edge). The following describes the shape primitives considered in the candidate set in our implementation and how to place them and determine their parameters:

- *Sphere:* We use a sphere as a vertex candidate primitive. The center of the sphere is at the vertex and the radius is the minimum clearance at the vertex.

- *Box:* We use a box as both a vertex and an edge primitive candidate. For vertices, the x-axis is aligned along the minimum clearance boundary point and the semi-range as the minimum clearance. In 2D, the y-axis range is then computed by adjusting the box such that it does not overhang the boundary. In 3D, projecting the boundary points associated with the vertex on the plane with the computed x-axis as the normal passing through the vertex. The y-axis and z-axis are then computed as x-axis and y-axis are determined for boxes placed on vertices in 2D.

  For edges, the x-axis is aligned along the edge and semi y-axis range is the minimum clearance along the edge. In 3D, y-axis is oriented along the minimum clearance point and flattened across z-axis without overhanging the boundary. After determination of y and z axes, the box is then expanded along x-axis without overhanging the boundary.

- *Capsule:* We use a capsule as an edge candidate primitive. The edge represents the backbone line segment of the capsule and the minimum clearance along the edge represents the radius.

- *Ellipsoid:* We use an ellipsoid as another edge primitive candidate. The x-axis is aligned along the edge with semi-axis range for x-axis as sum of half the length of the edge and the minimum clearance of the end-vertices. The center of the ellipsoid is considered to be the center of the edge. In 2D, the value of the semi-axis range for y-axis is computed as the minimum value that would not go beyond the clearance at any interpolated point on the edge. For 3D, we place a spheroid with y-axis aligned to a vector connecting one of the minimum clearance boundary point perpendicular to the edge. The y-axis semi-range value is assigned as the same value as the spheroid's y-axis semi-range. The z-axis semi-range is found by expanding the spheroid along z-axis without overhanging the boundary.

The boundary mapping of the skeleton is useful in expanding or contracting the primitives

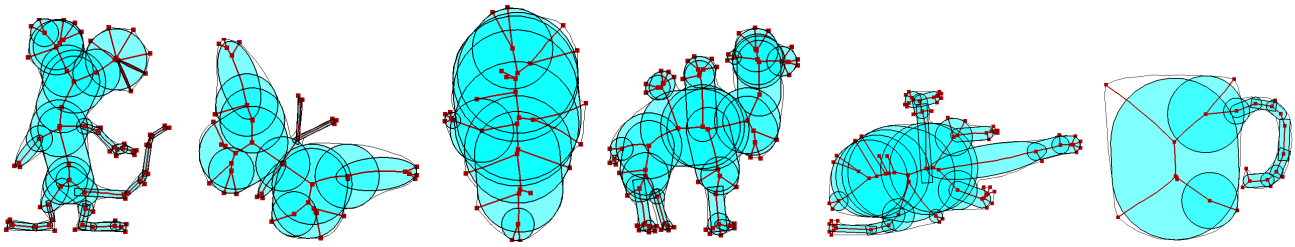
along any of the computed axes.

## 4.2 Experimental Results

In this section, we demonstrate how our method can be used to place primitive shapes inside objects and bounded spaces. We evaluate our approximation scheme in terms of the number of primitives placed and coverage (i.e., percentage of total volume covered). We approximate the volume computation by placing a grid over the environment and adding up the volume of the cells occupied by the shape primitives.

We used the CGAL library [83] for our skeleton implementations: Segment Delaunay Graph (2D) and Triangulated Surface Skeletonization (3D). We do not compare to any bounding volume method because bounding volumes go beyond the boundary of the shape whereas our primitives are fully inscribed. We also do not compare to general packing methods as primitives in packing methods are non-overlapping while we allow overlapping primitives.

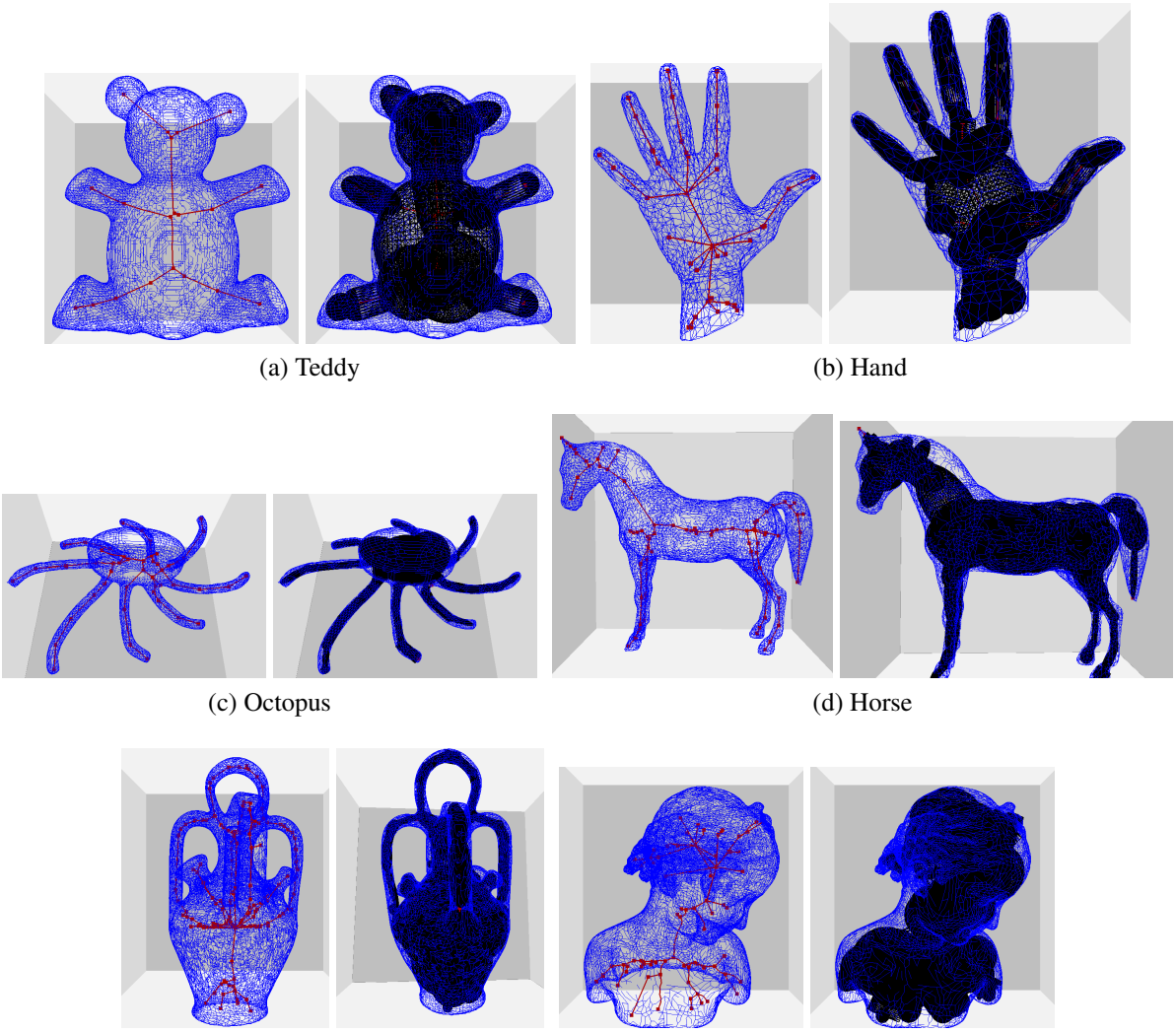
### 4.2.1 Coverage



(a) Rat (b) Butterfly (c) Face (d) Camel (e) Chopper (f) Cup

Figure 4.2: Our placement of primitives in 2D shapes. Area covered by the primitives are shown in cyan.

Figs 4.2 and 4.3 show the placement of primitives in a set of 2D polygons and 3D polyhedra. Table 4.1 provides the number of primitives of each type placed and the coverage obtained. As seen in Table 4.1, our method of primitive placement provides good coverage (over $80\%$ in 2D shapes and over $70\%$ in most 3D shapes) with none of the primitives going beyond the boundary.



(a) Teddy (b) Hand

(c) Octopus (d) Horse

(e) Vase (f) Bust

Figure 4.3: Our placement of primitives in 3D shapes.

The coverage is low in 3D shapes as compared to the 2D shapes. This can be attributed to an increase in dimensionality causing a looser fitting of the primitive volumes without going beyond the boundary.

The data shows that spheres and capsules are the dominant primitive shapes in most of the cases. Spheres are the most dominant primitive shape where the underlying skeleton is curvilinear causing the approximated line skeleton to have edges with small lengths (smaller than the clearance at the end vertices). The small edges cause the skeleton vertices to score higher or comparable to

| | #Spheres | #Ellipsoids | #Capsules | #Boxes | Coverage |
|---|---|---|---|---|---|
| Rat | 11 | 6 | 10 | 10 | 90.94 |
| Butterfly | 7 | 3 | 2 | 2 | 93.83 |
| Face | 8 | 0 | 2 | 0 | 96.74 |
| Camel | 20 | 1 | 6 | 4 | 95.66 |
| Chopper | 12 | 2 | 7 | 3 | 94.47 |
| Cup | 3 | 0 | 5 | 3 | 85.98 |
| Teddy | 6 | 5 | 3 | 0 | 82.18 |
| Hand | 6 | 12 | 5 | 3 | 69.95 |
| Octopus | 10 | 0 | 32 | 6 | 72.95 |
| Horse | 25 | 26 | 7 | 8 | 82.01 |
| Vase | 43 | 7 | 25 | 5 | 91.09 |
| Bust | 49 | 1 | 7 | 12 | 81.93 |

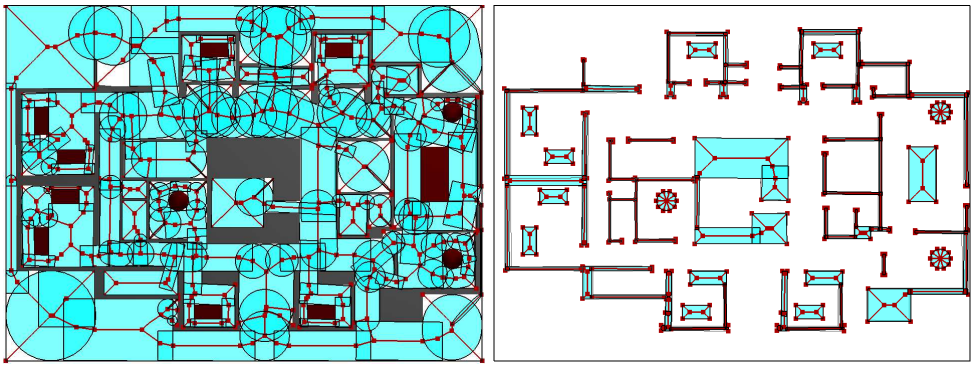
Table 4.1: Number of primitives and coverage for various models.

that of the edges and hence spheres are placed. This can be observed in the Camel and the Face model in 2D and in the Bust and the Vase model in 3D.

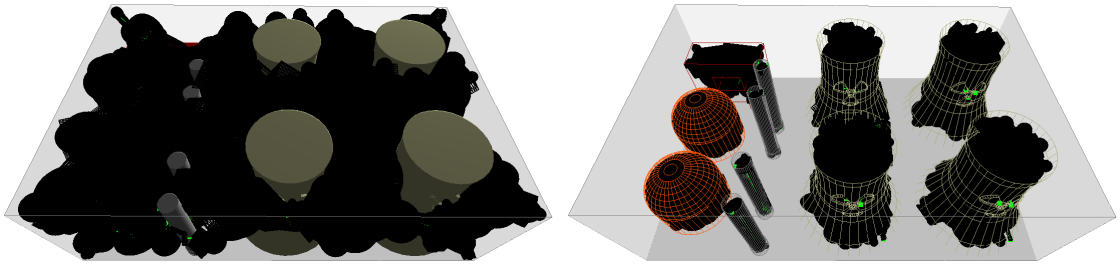
Capsules are the next dominant primitive shape (especially in 2D) due to low clearance variances along an edge or the presence of low clearance points closer to the center of the edge. In 3D, boxes and ellipsoids are placed where the cross-sectional aspect/axial ratio of the boundary along the edge is higher than 1. Boxes are favored in flattened areas with near uniform clearance distribution along the edge such as the handles of the Vase and tail of the Rat model.

Ellipsoids are favored in regions where the minimum clearance boundary lies near the end-vertices of the edge. For example, in regions like the nose of the Rat model, the body and wings of the Butterfly model, the palm region Hand model ellipsoids are favored over capsules or boxes. This is due to drastic decrease in clearance value at the end-vertices. In 3D, the shape of cross-section across the edge determines the placement of ellipsoids and boxes. An elliptical cross-section places an ellipsoid, otherwise a box is placed.

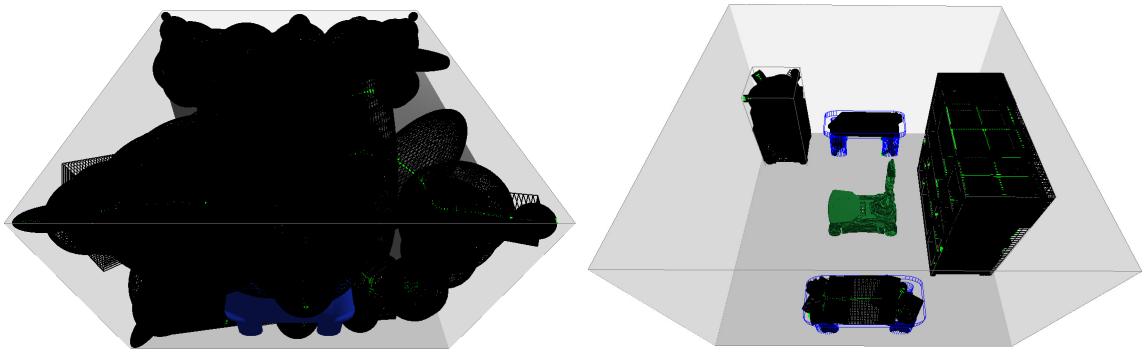
We also demonstrate how our method can place primitives in any bounded space (free and obstacle space in environments used in motion planning problems). Figs. 4.4 and 4.5 shows how we use the approximation in following environments: Office (floor plan of the Computer Science



(a) Office

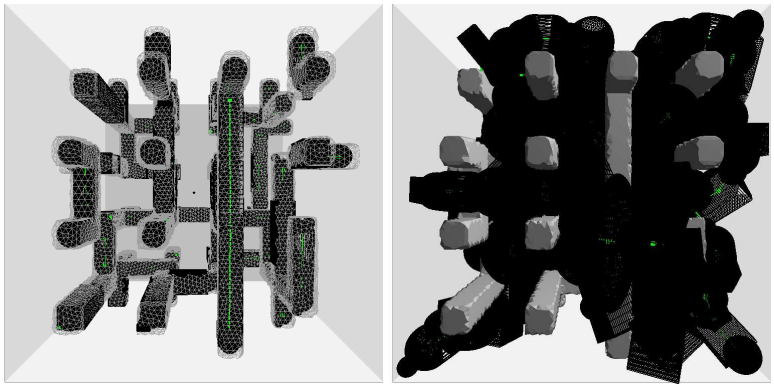


(b) Nuclear

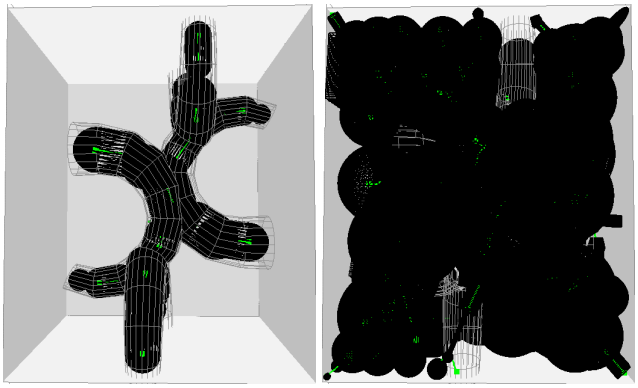


(c) Room

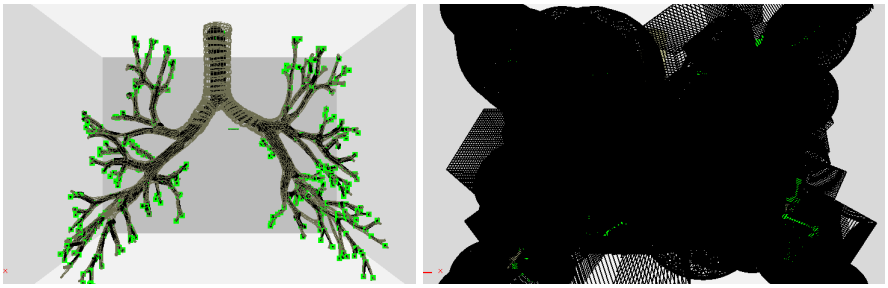
Figure 4.4: Shape primitive skeletons for both spaces in different navigation environments.



(a) Tunnels



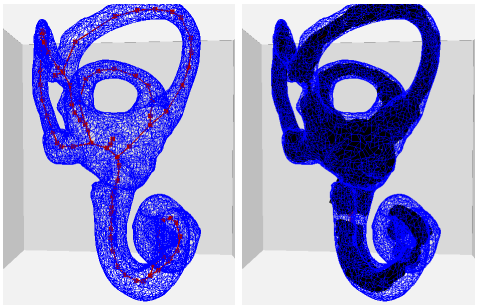
(b) Tubes



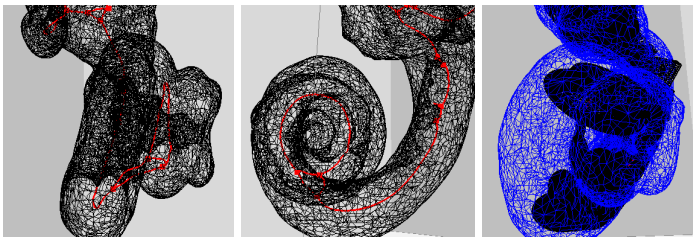
(c) Bronchi

Figure 4.5: Shape primitive skeletons for both spaces in different tubular environments.

building at Texas A&M University), Nuclear power plant, Room, Rectangular Tunnels, Tubes and Bronchi. As observed in the figures, the connectivity of the space is represented in the approximation.



(a) Skeleton and Primitives



(b) Issues

Figure 4.6: Issue with curvilinear skeleton. Curvilinear nature of the skeleton does not span every region of the object and hence affects the coverage in inner ear model.

The environments are generally more rectilinear (with exception of Nuclear plant, Bronchi and Tubes) than the shapes in Figs. 4.2 and 4.3. This causes placement of more boxes on the edges and vertices. If the underlying topological skeleton does not span certain regions (e.g., left corners in Tunnels obstacle space), no shape primitives will be placed in these regions. Also, if the skeleton is not medially placed, the clearance information along the edges are affected which in turn causes placement of skinny primitives in otherwise clear regions (for example, in Tunnels obstacle space). For 3D environments, the curvilinear nature of the underlying skeleton affects the size and fitting of the primitives. For example, in the lower part of the inner ear model (Fig. 4.6a) where the curvilinear skeleton does not span the model (due to cup-like or conch shell-like nature),

the skeleton primitive does not span the entire uncovered region (Fig. 4.6b).

In summary, our method was able to place primitives in both polygonal shapes or any bounded object with a good coverage of the inner volume. Capsules and spheres are preferred in most of the shapes whereas boxes are placed in virtual simulation environments.

### 4.2.2 Comparison of Shape Primitives Types

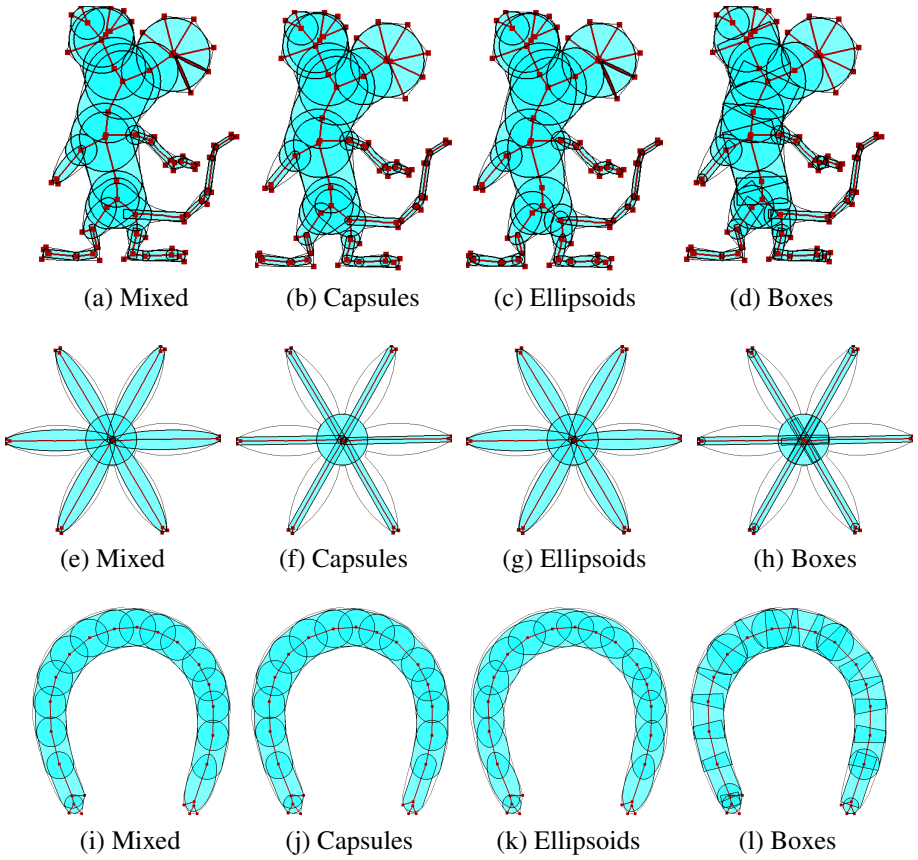



Figure 4.7: Effect of using mixed primitives in 2D shapes. We vary the edge primitives candidate sets for Rat (a-d), Flower (e-h) and Horseshoe (i-l) models. Coverage by the primitives is shown in cyan.

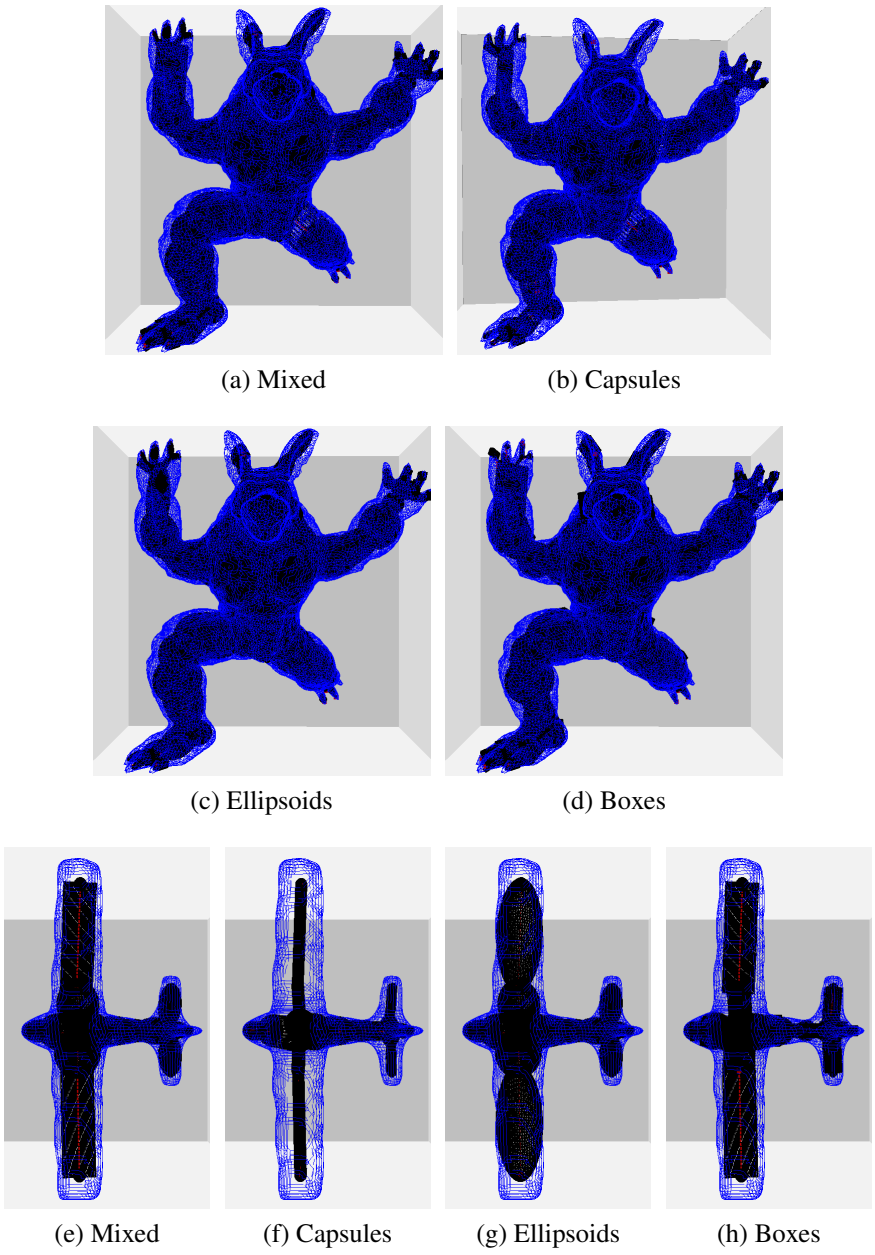
In this section, we compare various shape primitives used in the edge candidate set for seven models: Rat (our method places a combination of all edge primitives), Flower (our method places

ellipsoids) , Horseshoe (our method places capsules mostly), Armadillo (our method places a combinations of all edge primitives), Airplane (our method places ellipsoid and boxes mostly), Fertility (our method places capsules mostly) and Elk (our method places ellipsoids and capsules mostly). We use four different types of candidate set for the edges: Mixed (capsules, ellipsoids and boxes), Capsules (only capsules), Ellipsoids (only ellipsoids) and Boxes (only boxes). The vertex candidate set contains only spheres and boxes for all the four variants of edge candidate set just as before.

Fig. 4.7, Fig. 4.8 and Fig. 4.9 show how primitives are placed in these seven models with different edge candidate sets. Table 4.2 compares the number of spheres and edge primitives placed as well as the coverage provided.

As observed from the figure and the table, using a mix of edge primitives generally provides good coverage with small number of primitives. Using only capsules as edge primitive provides the least number of primitives but at the cost of coverage. The least number can be attributed to the coverage provided by the semi-sphere end caps of the capsule to the end-vertices causing no primitive to be placed at the end-vertices in most cases. The coverage is affected due to its less volume along the edge-length compared to boxes or ellipsoids. For elliptical shapes such as Flower, ellipsoids provides maximum coverage. Note that capsules or boxes are generally preferred when using mixed primitives in edge candidate sets. For a general shape such as Rat and Armadillo, the mixed primitives candidate set was able to place maximum volume primitives for edges which ensured good coverage while keeping a low number of primitives.

In 3D models, our method places ellipsoids or boxes for flat regions such as in Airplane and Elk models and capsules in others such as in Fertility model. Our method places boxes in the wings of the Airplane model and ellipsoids in the tail or rudders. The loss of volume due to eccentricity of the ellipsoids is more for longer edges (along the wings) and causes placement of boxes rather than ellipsoids. For the Elk model, the boxes provide better coverage than the mixed type of primitives. This can be attributed to the greedy placement of primitives on the edges which considers the local optimum than the global optimum. However, the number of primitives are more with just boxes

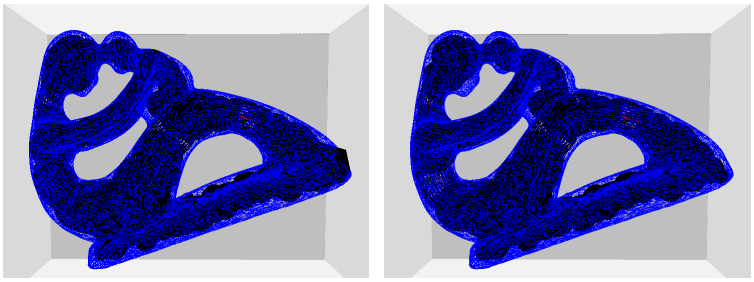


(a) Mixed (b) Capsules

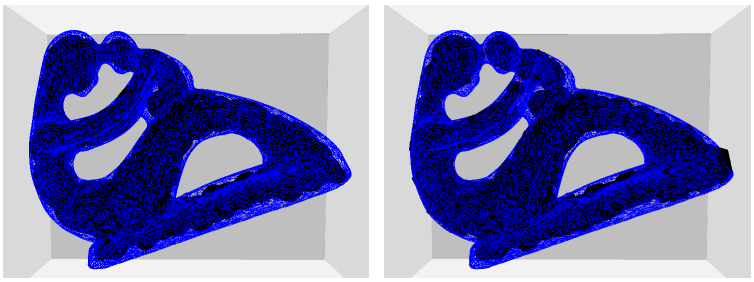
(c) Ellipsoids (d) Boxes

(e) Mixed (f) Capsules (g) Ellipsoids (h) Boxes

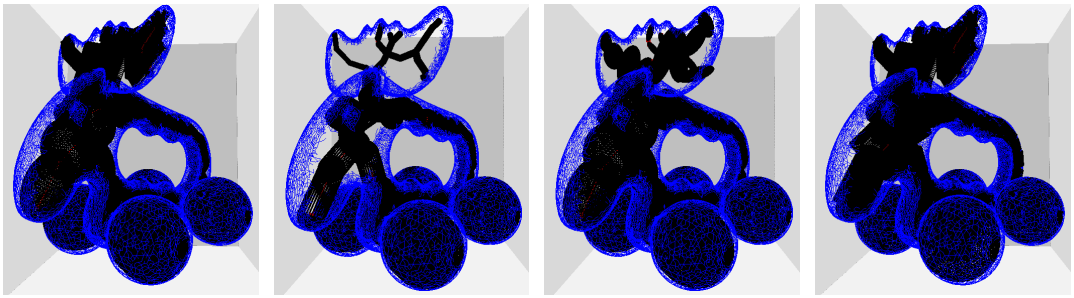
Figure 4.8: Effect of using mixed primitives in 3D zero genus shapes. We vary the edge primitives candidate sets for Armadillo (a-d) and Airplane (e-h) models.



(a) Mixed (b) Capsules



(c) Ellipsoids (d) Boxes



(e) Mixed (f) Capsules (g) Ellipsoids (h) Boxes

Figure 4.9: Effect of using mixed primitives in 3D non-zero genus shapes. We vary the edge primitives candidate sets for Fertility (a-d) and Elk (e-h) models.

| | | #Vertex Primitives | #Edge Primitives | Coverage(%) |
|---|---|---|---|---|
| Rat | Mixed | 11 | 26 | 93.94 |
| | Capsules | 11 | 24 | 93.27 |
| | Ellipsoids | 14 | 25 | 93.89 |
| | Boxes | 18 | 26 | 93.34 |
| Flower | Mixed | 1 | 6 | 81.10 |
| | Capsules | 1 | 6 | 43.20 |
| | Ellipsoids | 1 | 6 | 81.10 |
| | Boxes | 6 | 6 | 43.14 |
| Horseshoe | Mixed | 1 | 16 | 94.28 |
| | Capsules | 1 | 16 | 93.57 |
| | Ellipsoids | 5 | 11 | 91.44 |
| | Boxes | 7 | 17 | 94.07 |
| Armadillo | Mixed | 19 | 27 | 77.75 |
| | Capsules | 19 | 26 | 71.94 |
| | Ellipsoids | 21 | 29 | 73.73 |
| | Boxes | 24 | 33 | 72.40 |
| Airplane | Mixed | 8 | 12 | 72.44 |
| | Capsules | 6 | 12 | 71.94 |
| | Ellipsoids | 8 | 12 | 70.40 |
| | Boxes | 16 | 13 | 72.24 |
| Fertility | Mixed | 64 | 48 | 80.53 |
| | Capsules | 62 | 50 | 78.39 |
| | Ellipsoids | 62 | 50 | 79.88 |
| | Boxes | 101 | 66 | 79.76 |
| Elk | Mixed | 27 | 44 | 78.02 |
| | Capsules | 26 | 46 | 69.31 |
| | Ellipsoids | 31 | 45 | 76.02 |
| | Boxes | 64 | 63 | 79.24 |

Table 4.2: Number of primitives and coverage for various edge primitives candidate set.

as edge primitives than the mixed type. The sharp edges of the boxes constrain them to extend to small extent beyond the edge end vertices compared to other shapes such as capsules or ellipsoids.

In summary, using a variety of primitive types in the candidate set ensures good coverage of the volume with a low number of primitives in most cases.

### 4.2.3 Construction Time

Table 4.3 lists the construction time of the models in Fig 4.2, Fig 4.3, Fig. 4.7, Fig. 4.8 and Fig. 4.9. The construction time stated in the table includes preparation of the curve skeleton for shape primitive annotation (e.g., finding the clearance and line simplification) and creation of the shape primitive skeleton.

| Models | Construction Time (in seconds) |
|---|---:|
| Rat | 0.04 |
| Butterfly | 0.01 |
| Face | 0.02 |
| Camel | 0.02 |
| Chopper | 0.01 |
| Cup | 0.01 |
| Flower | 0.003 |
| Horseshoe | 0.01 |
| Teddy | 0.45 |
| Hand | 0.35 |
| Octopus | 0.40 |
| Horse | 0.12 |
| Vase | 0.63 |
| Bust | 0.38 |
| Armadillo | 0.59 |
| Airplane | 0.09 |
| Fertility | 0.21 |
| Elk | 0.26 |

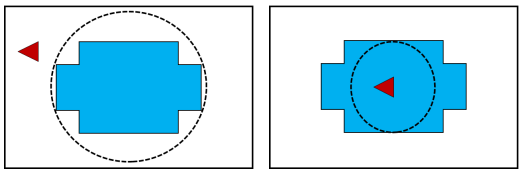
Table 4.3: Construction time for various models.

As the table shows, the construction time varies from $0.01 - 0.04$ seconds for 2D models and varies from $0.1 - 0.6$ seconds for 3D models. The shape of the underlying skeleton affects

the construction time. For models such as Vase, high degree vertices and large clearance values in the underlying skeleton (in the base) causes increased calls to the update of the score for its neighboring edges and vertices which results in an increase in the computation time. The increase in dimension also increases the computation time.

## 4.3 Application to Motion Planning

We use the shape primitive skeleton(s) of the obstacle and/or free workspace to improve the performance of collision detection operations in motion planning applications. The details about the construction of the shape primitive skeletons are stated previously in Sec. 4.1. We construct a collision checker which makes quick decisions (early rejection or acceptance) about the collision status of a moveable object with the surrounding environment (with static obstacles). If the checker cannot decide the collision status, then a full collision detector is called.

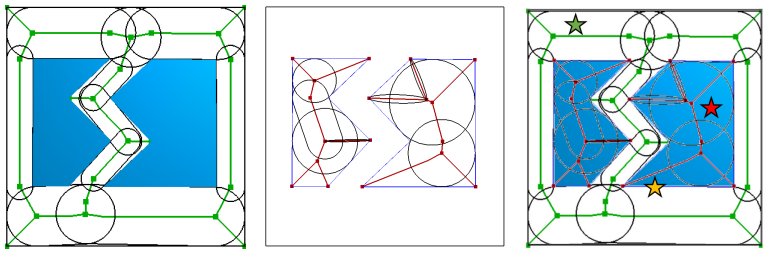


(a) Exclusion Test (b) Inclusion Test

Figure 4.10: Comparison of exclusion and inclusion tests. (a) Exclusion test checks if object is completely outside the bounding volume. (b) Inclusion test checks if object is completely inside the inscribed volume.

Unlike the bounding volume hierarchies where the objects are *bounded* by geometric shape primitives, the shape primitives in a shape primitive skeleton are completely contained *inside* the object. Hence, we perform *inclusion tests* (i.e., if an object is inside a shape primitive, it is inside the object bounding the primitive) as compared to *exclusion tests* (i.e., an object is outside another object if the former is outside all bounding shape primitives of the latter) to determine the collision status. Fig. 4.10 shows a comparative example of exclusion and inclusion tests.

### 4.3.1 Overview



(a) Free Space Primitive Skeleton (b) Obstacle Space Primitive Skeleton (c) Collision Checker

Figure 4.11: Use of shape primitive skeletons of both spaces in collision detection. (a) Shape Primitive Skeleton of the free space (white colored). The underlying skeleton graph is shown in green and primitives are shown with black boundaries. (b) Shape Primitive Skeleton of the obstacle space (object with blue color boundary). The underlying skeleton graph is shown in red and primitives are shown with black boundaries. (c) Collision status for different robot placements(red - in collision, green - collision free, yellow - undecided). The robot is shown as star shaped object.

We first construct the shape primitive skeleton(s) of the obstacle and/or free workspace in the environment. Fig. 4.11a and Fig. 4.11b are examples of the shape primitive skeletons of the free and obstacle space of a given environment.

Fig. 4.12 provides an overview of the algorithm. Given the object's location, we check whether the object is completely contained in any shape primitive. If the object is completely or partially contained in a shape primitive in obstacle space, the object is marked as in collision (placement of red robot in Fig. 4.11c). If the object is completely contained in a shape primitive in free space, the object is marked as collision-free (placement of green robot in Fig. 4.11c). If the object is partially contained in a shape primitive in free space, the object is clipped and the part outside the primitive is further processed. If the collision status cannot be determined (placement of yellow robot in Fig. 4.11c), a full collision detector is called. Algorithm 5 gives a brief sketch of the algorithm.

As robot placements are marked collision free or in collision if they are contained in shape primitives of the shape primitives skeleton, skeletons that provide good coverage of the space will

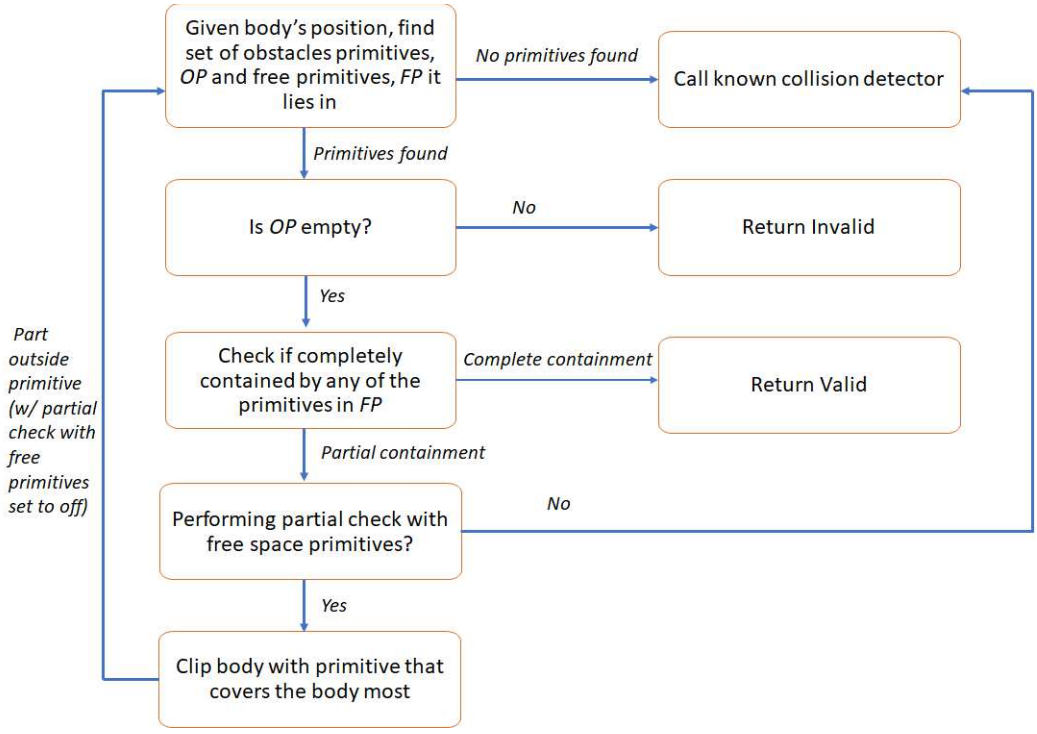



Figure 4.12: Flow chart for the algorithm.

**Algorithm 5** Fast Collision Detection (FastCD)

**Require:** Shape Primitive Skeletons $PS$ (obstacle $OS$ and free $FS$), Moveable object $M$, Collision Detector $CD$, Partial Validation Flag $PV$ (False by default).

**Ensure:** True(not in collision) or False (in collision)

$P$ = FindShapePrimitives($PS$, $M.currentLocation$)
**for** each $p$ in $P$ **do**
  **if** $M$ is completely/partially inside $p$ **then**
    **if** $p$ is in $OS$ **then**
      **return** False
    **else if** $p$ is in $FS$ **then**
      **if** $M$ is completely inside $p$ **then**
        **return** True
      **else if** $PV$ is True **then**
        $M'$ = ClipRobot($M$, $p$)
        **return** FastCD($PS$, $M'$, $CD$, False)
**return** $CD(M)$

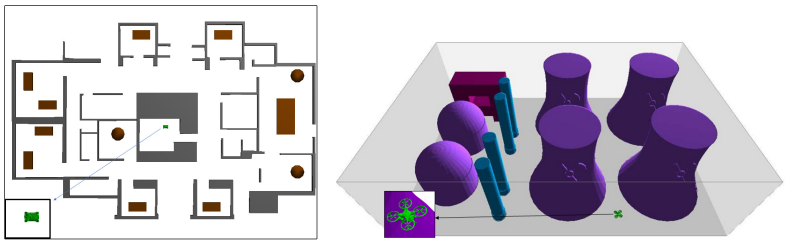
result in fewer indecisive states and fewer calls to the underlying full collision detector. A large number of primitives might provide a good coverage but it will create primitives of size comparable to or smaller than the robot size which will also result in increased number of indecisive states. In such circumstances, partial validation reduces the number of indecisive states.

Below are some of the implementation details used in this work:

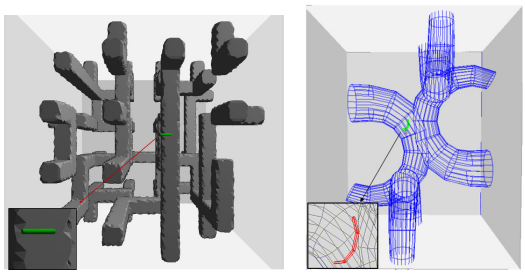
- Containment check of an object with any primitive is done by checking the clearance of the object's center with the primitive.
- To clip the robot, we clip the robot's bounding box with the primitive.
- Axis-aligned bounding boxes of the primitive are used for spatial indexing of the shape primitives, i.e., to find candidate shape primitives for containment check.

### 4.3.2 Results

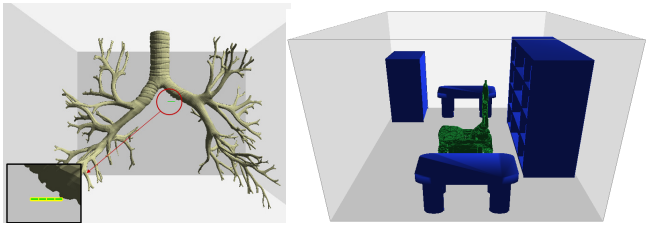
We study the impact of using our shape primitive skeleton on the performance of collision detection methods (see Alg. 5) in the environments shown in Fig 4.13: (a) Office (2D) with a rigid body robot with 3 degrees of freedom (DOFs), (b) Nuclear power plant (3D) with a rigid body robot (6 DOFs), (c) Tunnels (3D) with a rigid body robot (6 DOFs), (d) Tubes (3D) with a free flying linkage robot with 4 links (9 DOFs), (e) Bronchi (3D) with the same robot as the Tubes environment and (f) Room (3D) with a multi-link robot with 8 DOFs (2 positional, 1 orientation and 5 joints). For Tunnels, Tubes and Bronchi environments, inside the tunnel or tubes or bronchi is considered free space and outside is considered obstacle space. We use two different underlying collision detectors: RAPID [36] (which uses OBB hierarchy) and PQP [84] (which uses swept sphere volumes) to show the generality of our approach. For each environment, we use a sampling based motion planning approach: generate a set of random placements of robot and then connect them using a straight line local planner to build a roadmap. Both the placements of robots (samples) and the edges in the roadmap (intermediates in the edge) are checked for collision with and without the use of our shape primitive skeleton approach and with and without the partial intersection check in free space (as described in Sec. 4.3). We use three different strategies for roadmap generation:



(a) Office (b) Nuclear Power Plant



(c) Tunnels (d) Tubes



(e) Bronchi (f) Room

Figure 4.13: Environments for evaluation of collision checker. Robot is shown in green color and in inset.

Probabilistic Roadmap with Uniform Sampling (PRM) [46], Probabilistic Roadmap with Obstacle Based Sampling (OBPRM) [76] and Rapidly exploring Random Trees with Uniform Sampling (RRT) [47] to study how the distribution of samples affects the performance impact of the shape primitive skeleton. We place 500 valid placements of the robot and attempt to connect each node with its 3 nearest neighbors.

For the Tubes, Bronchi and Room environments, the collision check is done for each moveable part (or link). Self collision detection is performed using the underlying collision detection library and is done at the end for samples which are not colliding with obstacles.

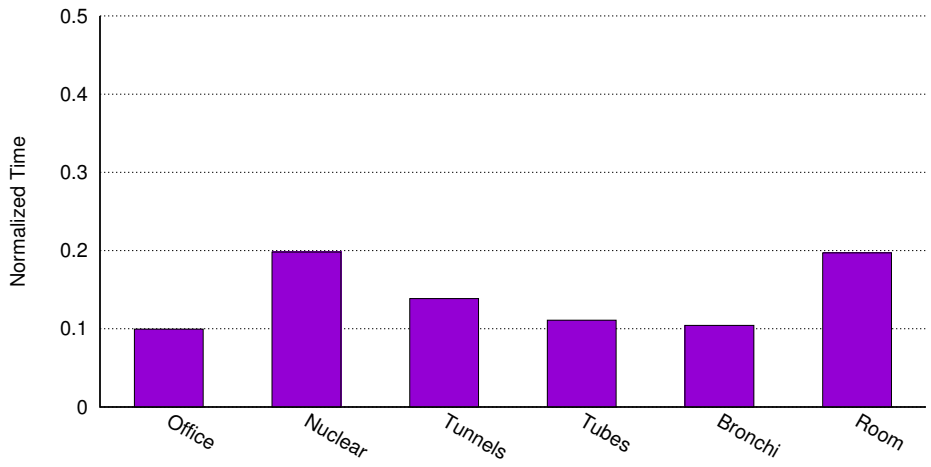



Figure 4.14: Construction time of the shape primitive skeletons of both spaces. The time is normalized to total planning time without the use of the skeletons.

Fig. 4.14 provides the initialization time due to the primitive skeleton construction in both spaces. The time is normalized to the total planning time for generating a roadmap with 500 samples and connections without the use of shape primitive skeleton. The figure shows that the skeleton construction time provides an overhead of $0.1 - 0.2$ times of the planning time without the use of

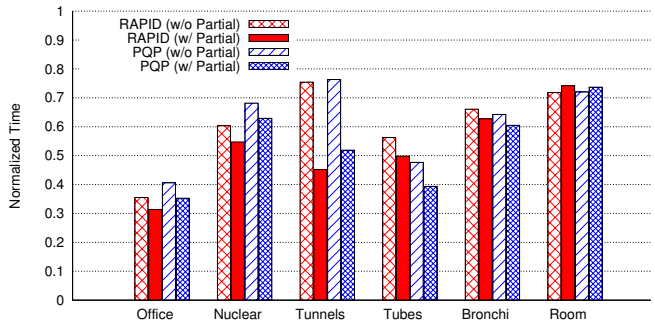
the skeletons. Please note that the construction time also includes the time for clearance computations and other line skeleton preparation pre-processing. For large environments like the Nuclear power plant and environments with large primitives (such as free space in Room environment), the construction time is near to $0.2$ of the planning time without the use of skeletons.

#### *4.3.2.1 Performance*

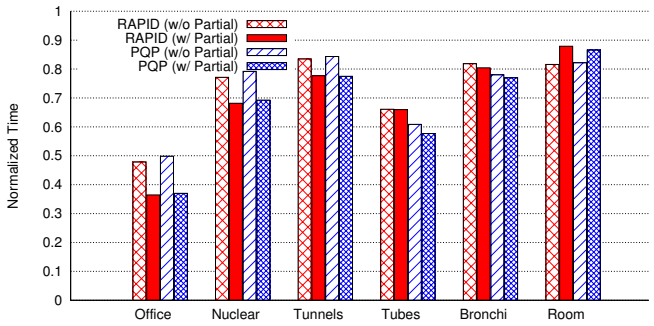
We analyze the performance with respect to the total time taken in collision detection. The collision detection time reported includes the collision detection time for both sampling and local planning (i.e., validation of edges). Fig. 4.15 shows the collision detection time using the shape primitive skeleton normalized to the collision detection time of the underlying collision detection library without using the skeleton for each environment and planning strategy. Fig. 4.16 shows the respective call ratio measured as the ratio of the number of collision detection calls whose collision status is determined by containment checking with the shape primitives to the total number of collision detection calls (complete denotes those without the partial check with free space primitives and partial denotes the same with the partial check with free space primitives).

We observe an improvement the collision detection performance irrespective of the underlying collision detector and underlying planning strategy. The improvement in performance varies from marginal to $70\%$ in some cases, both with and without the partial checking feature.

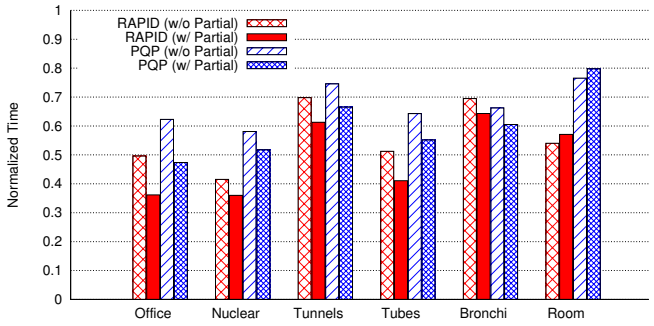
As observed in Fig. 4.15, the use of the shape primitive skeleton (even without the partial check in free space) improves the collision detection time as compared to the underlying CD library. In the Office environment, the status of most of the collision detection calls ($> 90\%$) is determined through inclusion tests with the shapes in the shape primitive skeletons and thereby, we observe the maximum improvement in collision detection performance over the collision detection method without the use of the shape primitive skeleton. Skinny shape primitives (compared to the size of each body in robot) and low coverage of any space by the shape primitive skeleton cause more indecisive states, which causes increases in the number of full collision detection calls and minimal improvement in performance (as observed in the Tunnels and the Room environments), see Fig. 4.16.




(a) PRM

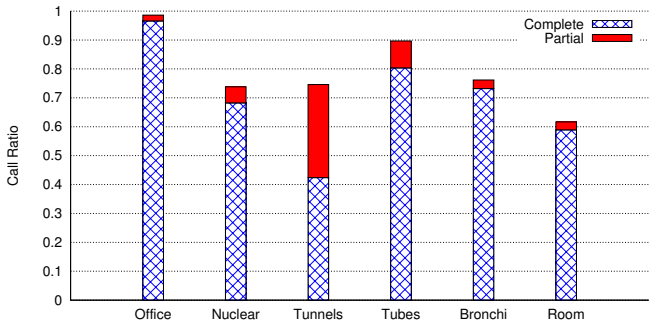



(b) OBPRM

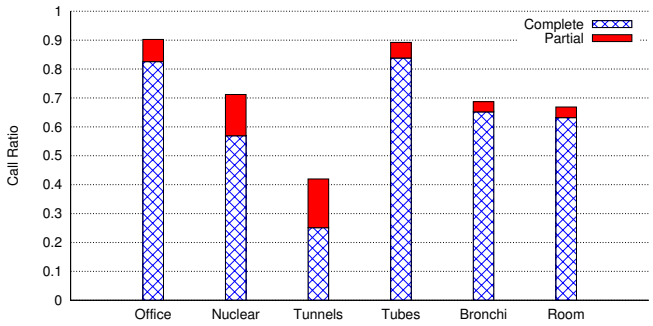



(c) RRT

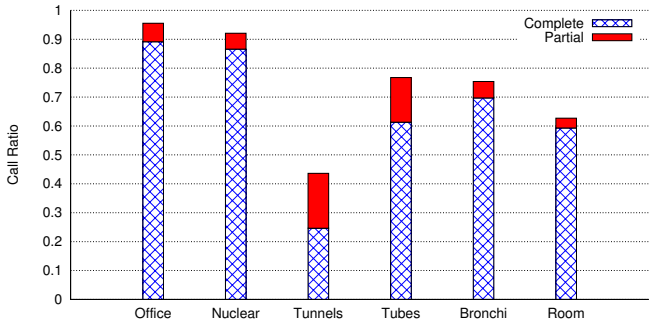
Figure 4.15: Normalized time for different environments and planners. The time is normalized to time taken by the underlying detector alone to build roadmaps with different number of samples for each environment and planning strategy.




(a) PRM




(b) OBPRM




(c) RRT

Figure 4.16: Call ratio to build roadmaps for each environment and planning strategy. The call ratio is number of calls whose status is determined by complete/partial inclusion test to total number of calls.

The motion planning strategy also affects the improvement provided by the shape primitive skeleton. If the robots are placed near the boundary (such as in OBPRM where the samples are generated closer to the obstacle surface), they are near the boundary of the shape primitives causing an increased number of indecisive states (see Fig. 4.16b) and hence an increased number of calls to the full collision detector.

Performance is further improved when the shape primitive skeleton is used with partial checking in free space primitives with the exception of the Room. It is most noticeable in the Tunnels environment where primitives in the free space are skinny compared to the robot size. In the Tubes, the Bronchi and the Room environments, the robot consists of multiple bodies. So even if the primitives in free space are large enough compared to each body size, they are not large enough to validate the entire robot. For the Office environment where most of the samples are already checked through complete inclusion test in PRM, the improvement with partial checking is more noticeable in OBPRM than in PRM.

In the Room environment, the size of the parts and the primitives causes most of the partial checks (which is just for 2 iterations) near the obstacles to fail. This causes an overhead computation and the improvement in performance to be less than that without the use of partial checking. However, even with the overhead computation, there is improvement in performance over that without the use of shape primitive skeletons.

#### *4.3.2.2 Comparison with Skeletons in One Space*

In this section, we show how the shape primitive skeletons at one space impact the collision detection time when compared to the use of skeletons of both spaces. We study the performance of the collision detector with the use of shape primitive skeletons in free space, obstacle space and both free and obstacle skeletons. The collision detection time normalized to the collision detection time without the use of the skeleton.

Fig. 4.17 shows the normalized time of roadmap with 500 samples generated using PRM strategy (uniform sampling) without connecting the samples. As the figure shows, the volume of the space (compared to the total volume) dictates the performance improvement observed in the var-

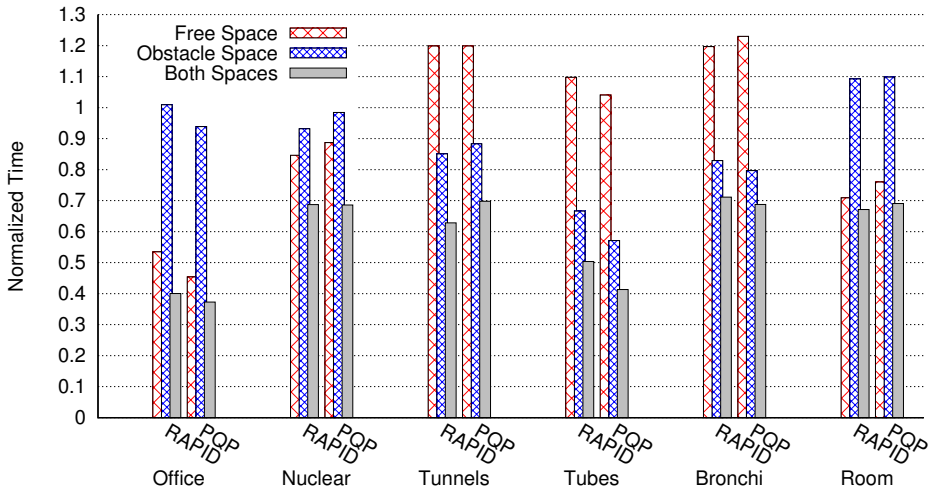



Figure 4.17: Collision detection time for use of skeleton in different spaces with only sampling. The collision detection time in both and one (free or obstacle) is normalized to the collision detection time without use of skeleton.

ious environments. For Tunnels, Tubes and Bronchi environments, there is no performance improvement with use of skeleton in free space only but some performance improvement is observed with the use of skeleton in obstacle space. Similarly, for Office, Nuclear and Room environments the performance improvement is observed with use of skeleton in free space only but no improvement with the use of skeleton in obstacle space. In all the environments, the maximum performance improvement is observed with the use of skeletons from both spaces.

Fig. 4.18 shows the normalized time of roadmap with 500 samples and their connection generated using PRM strategy (uniform sampling). As the figure shows, use of obstacle space skeleton only does not show any performance improvement. However, we observe some performance improvement with the use of free space skeleton only. This can be attributed to the use of the collision detector to validate the edges of the roadmap which is done by validating each intermediate sample along the edges. As the roadmap edges lies in the free space, most of collision detection calls are due to validation edges. However, use of skeleton in both spaces provides the best performance

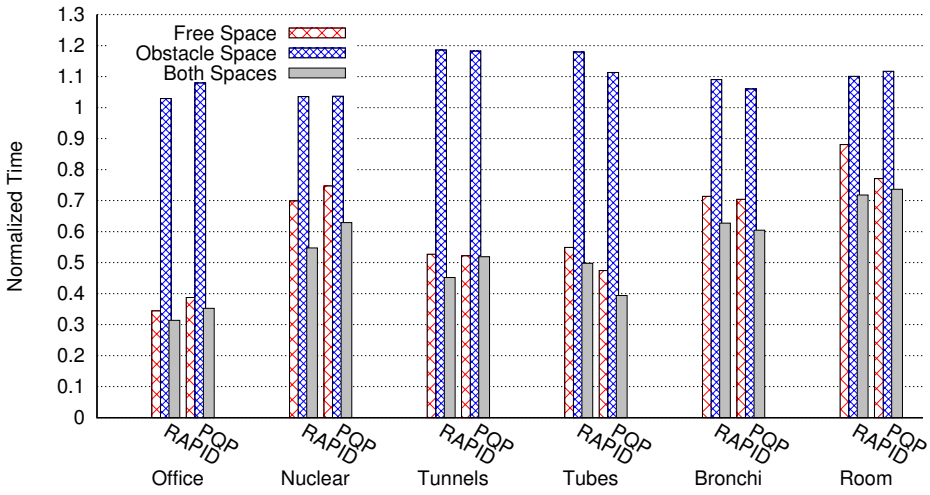


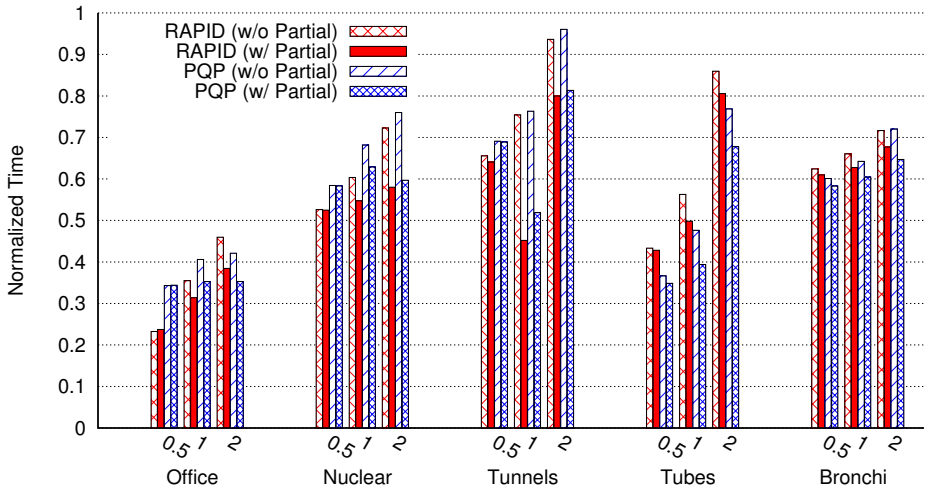

Figure 4.18: Collision detection time for use of skeleton in different spaces with total planning. The collision detection time in both and one (free or obstacle) is normalized to the collision detection time without use of skeleton.

improvement among the three usage of the skeletons.

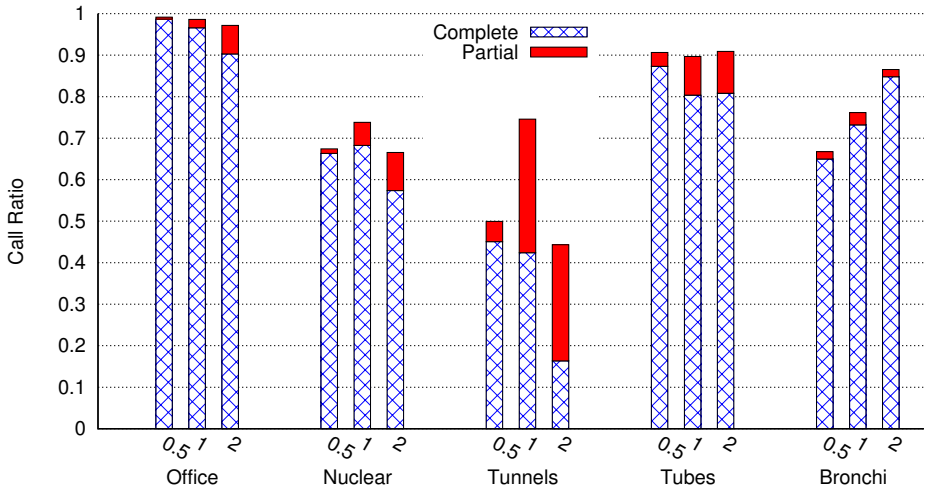
#### *4.3.2.3 Reusability across Different Robots*

In this section, we show how the shape primitive skeleton of the same environment can be reused by robots of different sizes. We study the performance of the collision detector with 3 different robot sizes. We uniformly scaled the robot from the above experiment using 3 scaling factors: 0.5 (reduced), 1 (original) and 2 (enlarged). For the Tubes and Bronchi environments, we scale each moveable part by the scaling factor. We used the PRM strategy to analyze the performance impact.

Fig. 4.19 shows the normalized collision detection time for roadmap with 500 samples. As the figure shows with increased robot size, the performance improvement is reduced (see Fig. 4.19a). This is due to an increase in robot placements near the shape primitive boundary and an increase in calls to the underlying collision detector (see Fig. 4.19b).




(a) Time




(b) Call Ratio

Figure 4.19: Normalized collision detection time and call ratio for different robot scale factor.

# 5. CONCLUSIONS AND FUTURE WORK

In this dissertation, two geometric approximation methods, namely aggregation hierarchy and shape primitives skeleton, are proposed. These approximation techniques are then used to improve the performance of motion planning algorithms.

Aggregation hierarchy groups nearby objects to individual objects. The hierarchy is created by varying the distance threshold that determines how distant the objects are. The aggregation hierarchy on the obstacle space is used to obtain a set of regions in the free space. These regions are then used for sampling in sampling based motion planning algorithms. As these regions have low obstacle density, the sampling efficiency is improved and hence the computation time and success rate of the samplers is improved.

The shape primitive skeleton approximates any bounded space into a set of shape primitives. These shape primitives are of mixed types, completely contained in the space and represent the connectivity of the space. The shape primitive skeletons of both the obstacle and free spaces are then used to improve the collision detection method in sampling based motion planning algorithm. The skeletons are used to make quick early in and out decisions about the collision status. Our results show that this improves the collision detection time significantly for motion planning algorithms in the test environments.

The future work for the aggregation hierarchy includes evaluation of the different hierarchies of the same environment by including other properties besides shortest distance (such as curvature of the tunnel). One future work for the shape primitive skeleton will be to obtain a hierarchical view of the shape primitive skeleton in order to further improve the performance of collision detection. Another future work will be to utilize the connectivity of the skeleton to validate trajectories.

# REFERENCES


[1] A. P. Sampaio, R. Chaine, C. A. Vidal, and J. B. Cavalcante-Neto, "Temporally coherent sculpture of composite objects," *Computers & Graphics*, vol. 58, pp. 118 – 127, 2016. Shape Modeling International 2016.

[2] L.-P. Albou, B. Schwarz, O. Poch, J. M. Wurtz, and D. Moras, "Defining and characterizing protein surface using alpha shapes," *Proteins: Structure, Function, and Bioinformatics*, vol. 76, no. 1, pp. 1–12, 2008.

[3] J. Denny, R. Sandstrom, A. Bregger, and N. M. Amato, "Dynamic region-biased exploring random trees," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, (San Francisco, CA), December 2016.

[4] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Trans. Graph.*, vol. 15, pp. 179–210, July 1996.

[5] E. Plaku, L. Kavraki, and M. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robot.*, vol. 26, pp. 469–482, June 2010.

[6] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Trans. Graph.*, vol. 22, pp. 954–961, July 2003.

[7] M. Ghosh, S. L. Thomas, M. A. Morales, S. Rodriguez, and N. M. Amato, "Motion planning using hierarchical aggregation of workspace obstacles," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, IEEE, 2016.

[8] M. Ghosh, S. Thomas, and N. M. Amato, "Fast collision detection for motion planning using shape primitive skeletons," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2018.

[9] R. Xu, D. Wunsch, *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[10] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, pp. 165–193, Jun 2015.

[11] P. Berkhin, "Survey of clustering data mining techniques," tech. rep., 2002.

[12] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, pp. 68–75, Aug. 1999.

[13] D. Joshi, A. Samal, and L.-K. Soh, "Density-based clustering of polygons," in *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*, pp. 171–178, 2009.

[14] D. Joshi, A. Samal, and L.-K. Soh, "A dissimilarity function for clustering geospatial polygons," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, (New York, NY, USA), pp. 384–387, ACM, 2009.

[15] D. Joshi, L. Soh, and A. Samal, "Redistricting using constrained polygonal clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, pp. 2065–2079, Nov 2012.

[16] P. Pilehforooshha and M. Karimi, "A local adaptive density based algorithm for clustering polygonal buildings in urban block polygons," *Geocarto International*, vol. 0, no. ja, pp. 1–57, 2018.

[17] R. Chang, T. Butkiewicz, C. Ziemkiewicz, Z. Wartell, W. Ribarsky, and N. Pollard, "Legible simplification of textured urban models," *Computer Graphics and Applications, IEEE*, vol. 28, pp. 27–36, May 2008.

[18] J. Xie, L. Zhang, J. Li, H. Wang, and L. Yang, "Automatic simplification and visualization of 3d urban building models," *International Journal of Applied Earth Observation and Geoinformation*, vol. 18, pp. 222–231, 2012.

[19] W. Wang, S. Du, Z. Guo, and L. Luo, "Polygonal clustering analysis using multilevel graph-partition," *Transactions in GIS*, vol. 19, no. 5, pp. 716–736, 2015.

[20] L. P. Gewali and S. Manandhar, "Approaches for clustering polygonal obstacles," in *Information Technology - New Generations* (S. Latifi, ed.), (Cham), pp. 887–892, Springer International Publishing, 2018.

[21] C. L. Wang, "Approximate boolean operations on large polyhedral solids with partial mesh reconstruction," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 6, pp. 836–849, 2011.

[22] F. Bernardini and C. L. Bajaj, "Sampling and reconstructing manifolds using alpha–shapes," in *Proc. 9th Canad. Conf. Comput. Geom.*, pp. 193–198, 1997.

[23] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Trans. Graph.*, vol. 13, pp. 43–72, Jan. 1994.

[24] H.-L. Cheng and X. Shi, "Quality mesh generation for molecular skin surfaces using restricted union of balls," *Computational Geometry*, vol. 42, no. 3, pp. 196 – 206, 2009.

[25] E. G. Teich, G. van Anders, D. Klotsa, J. Dshemuchadse, and S. C. Glotzer, "Clusters of polyhedra in spherical confinement," *Proceedings of the National Academy of Sciences*, vol. 113, no. 6, pp. E669–E678, 2016.

[26] H. Edelsbrunner, P. Fu, and E. P. Mücke, "An introduction to alpha shapes," 1993. In preparation.

[27] H. Edelsbrunner, "Weighted alpha shapes," Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[28] F. Cazals, J. Giesen, M. Pauly, and A. Zomorodian, "Conformal alpha shapes," in *Proceedings of the Second Eurographics/IEEE VGTC conference on Point-Based Graphics*, pp. 55–61, Eurographics Association, 2005.

[29] P. Theologou, I. Pratikakis, and T. Theoharis, "Unsupervised spectral mesh segmentation driven by heterogeneous graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 397–410, Feb 2017.

[30] I. Chiosa and A. Kolb, "Variational multilevel mesh clustering," in *2008 IEEE International Conference on Shape Modeling and Applications*, pp. 197–204, June 2008.

[31] S. Valette and J.-M. Chassery, "Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening," *Computer Graphics Forum*, vol. 23, no. 3, pp. 381–389.

[32] J. Guo, D.-M. Yan, X. Jia, and X. Zhang, "Efficient maximal poisson-disk sampling and remeshing on surfaces," *Computers & Graphics*, vol. 46, pp. 72 – 79, 2015. Shape Modeling International 2014.

[33] S. Borah and B. Borah, "A survey on feature remeshing of 3d triangular boundary meshes," in *2016 International Conference on Accessibility to Digital World (ICADW)*, pp. 57–62, Dec 2016.

[34] N. Hagbi and J. El-Sana, "Carving for topology simplification of polygonal meshes," *Computer-Aided Design*, vol. 42, no. 1, pp. 67 – 75, 2010. Advances in Geometric Modelling and Processing.

[35] C. Tu and L. Yu, "Research on collision detection algorithm based on aabb-obb bounding volume," in *2009 First International Workshop on Education Technology and Computer Science*, vol. 1, pp. 331–333, March 2009.

[36] S. Gottschalk, M. C. Lin, and D. Manocha, "OBB-tree: A hierarchical structure for rapid interference detection," *Comput. Graph.*, vol. 30, pp. 171–180, 1996. Proc. SIGGRAPH '96.

[37] J. Klosowski, M. Held, J. S. B. Mitchell, K. Zikan, and H. Sowizral, "Efficient collision detection using bounding volume hierarchies of $k$-DOPs," *IEEE Trans. Visualizat. Comput. Graph.*, vol. 4, no. 1, pp. 21–36, 1998.

[38] D. Schneider, E. Schömer, and N. Wolpert, "Collision detection for 3d rigid body motion planning with narrow passages," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4365–4370, May 2017.

[39] J. H. Conway and N. J. H. Sloane, *Sphere Packings, Lattices and Groups*. Springer Science and Business Media, 3 ed., 1998.

[40] A. Donev, I. Cisse, D. Sachs, E. A. Variano, F. H. Stillinger, R. Connelly, S. Torquato, and P. M. Chaikin, "Improving the density of jammed disordered packings using ellipsoids," *Science*, vol. 303, no. 5660, pp. 990–993, 2004.

[41] J. H. Conway and S. Torquato, "Packing, tiling, and covering with tetrahedra," *Proceedings of the National Academy of Sciences*, vol. 103, no. 28, pp. 10612–10617, 2006.

[42] S. Li, J. Zhao, P. Lu, and Y. Xie, "Maximum packing densities of basic 3d objects," *Chinese Science Bulletin*, vol. 55, pp. 114–119, Jan 2010.

[43] R. Weller and G. Zachmann, "Inner sphere trees for proximity and penetration queries.," in *Robotics: science and systems*, vol. 2, 2009.

[44] S. Stolpner, P. Kry, and K. Siddiqi, "Medial spheres for shape approximation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 6, pp. 1234–1240, 2012.

[45] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, pp. 560–570, October 1979.

[46] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, August 1996.

[47] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, pp. 378–400, May 2001.

[48] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, (San Juan, Puerto Rico), pp. 421–427, October 1979.

[49] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3d workspaces," in *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*, (Natick, MA, USA), pp. 155–168, A. K. Peters, Ltd., 1998. (WAFR '98).

[50] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, pp. 1018–1023, May 1999.

[51] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "Bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4420–4426, IEEE, 2003.

[52] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proceedings of Robotics: Science and Systems*, (Zaragoza, Spain), June 2010.

[53] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, pp. 1024–1031, 1999.

[54] J.-M. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 3, pp. 4439–4444, sept. 2003.

[55] J. P. van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *Int. J. Robot. Res.*, vol. 24, no. 12, pp. 1055–1071, 2005.

[56] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, vol. 2, pp. 1618–1623, sept.-2 oct. 2004.

[57] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle - an adaptive sampling strategy for prm planning," in *Algorithmic Foundation of Robotics VII*, pp. 35–51, Berlin/Heidelberg: Springer, 2008. Book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), New York City, 2006.

[58] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning via safety certificates," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 767–796, 2016.

[59] F. Schwarzer, M. Saha, and J.-C. Latombe, *Exact Collision Checking of Robot Paths*, pp. 25–41. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[60] B. Lacevic, D. Osmankovic, and A. Ademovic, "Burs of free c-space: A novel structure for path planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 70–76, May 2016.

[61] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semi-definite programming," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[62] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration places," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1469–1475, 2001.

[63] Y. Yang and O. Brock, "Adapting the sampling distribution in prm planners based on an approximated medial axis," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 5, pp. 4405–4410, 2004.

[64] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *Algorithmic Foundations of Robotics X*, vol. 86 of *Springer Tracts in Advanced Robotics*, pp. 297–312, Berlin/Heidelberg: Springer, 2013. (WAFR '12).

[65] J. Bialkowski, M. Otte, and E. Frazzoli, "Free-configuration biased sampling for motion planning," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pp. 1272–1279, Nov 2013.

[66] R. Kimmel, N. Kiryati, and A. M. Bruckstein, "Sub-pixel distance maps and weighted distance transforms," *Journal of Mathematical Imaging and Vision*, vol. 6, pp. 223–233, Jun 1996.

[67] E. W. Chambers, Éric Colin de Verdière, J. Erickson, S. Lazard, F. Lazarus, and S. Thite, "Homotopic fréchet distance between curves or, walking your dog in the woods in polynomial time," *Computational Geometry*, vol. 43, no. 3, pp. 295 – 311, 2010. Special Issue on 24th Annual Symposium on Computational Geometry (SoCG'08).

[68] M. Teichmann and M. Capps, "Surface reconstruction with anisotropic density-scaled alpha shapes," in *Proceedings Visualization '98 (Cat. No.98CB36276)*, pp. 67–72, Oct 1998.

[69] J. R. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," in *Applied Computational Geometry: Towards Geometric Engineering* (M. C. Lin and D. Manocha, eds.), vol. 1148 of *Lecture Notes in Computer Science*, pp. 203–222, Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[70] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, pp. 11:1–11:36, Feb. 2015.

[71] Y. Hwang, K. Cho, S. Lee, S. Park, and S. Kang, "Human computer cooperation in interactive motion planning," in *Proc. IEEE Int. Conf. Adv. Robot. (ICAR)*, pp. 571–576, 1997.

[72] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 529–536, 2000.

[73] A. Vargas Estrada, J.-M. Lien, and N. M. Amato, "Vizmo++: a visualization, authoring, and educational tool for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 727–732, May 2006.

[74] N. Ladeveze, J.-Y. Fourquet, B. Puel, and M. Taix, "Haptic assembly and disassembly task assistance using interactive path planning," in *Virtual Reality Conference, 2009. VR 2009. IEEE*, pp. 19–25, March 2009.

[75] Y. Yan, E. Poirson, and F. Bennis, "Integrating user to minimize assembly path planning time in PLM," in *Product Lifecycle Management for Society*, vol. 409 of *IFIP Advances in Information and Communication Technology*, pp. 471–480, Springer Berlin Heidelberg, 2013.

[76] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 113–120, 1996.

[77] KD Healthcare Company, "KD Smart Chair." `http://kdsmartchair.com/`. Accessed: Sept. 15, 2016.

[78] Omron Adept MobileRobots, LLC, "Pioneer P3-DX." `http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx`. Accessed: Sept. 15, 2016.

[79] iRobot, "iRobot Create 2 programmable robot." `http://www.irobot.com/About-iRobot/STEM/Create-2.aspx`. Accessed: Sept. 15, 2016.

[80] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form* (W. Wathen-Dunn, ed.), pp. 362–380, MIT Press, 1967.

[81] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien, "Surface coding based on morse theory," *IEEE Comput. Graph. Appl.*, vol. 11, pp. 66–78, Sept. 1991.

[82] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," *Computer Graphics Forum*, vol. 31, no. 5, pp. 1735–1744, 2012.

[83] "CGAL, Computational Geometry Algorithms Library," 1997. http://www.cgal.org.

[84] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Distance queries with rectangular swept sphere volumes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 4, pp. 3719–3726 vol.4, 2000.