

# Efficient Probabilistic Collision Detection for Non-Convex Shapes

Jae Sung Park, Chonhyon Park, Dinesh Manocha

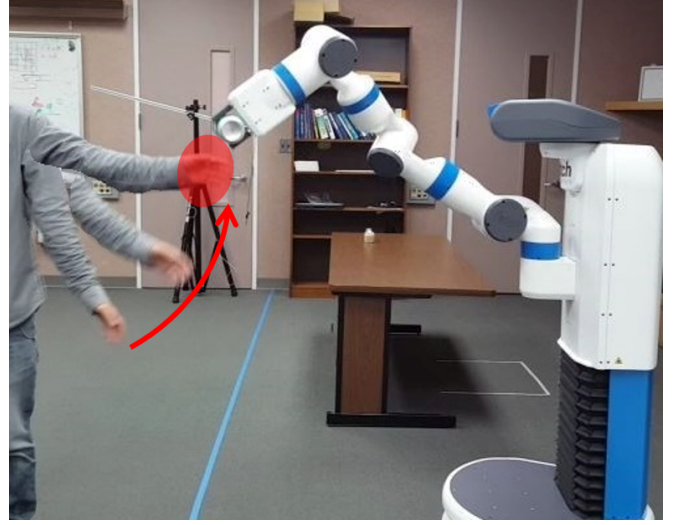
**Abstract**—We present new algorithms to perform fast probabilistic collision queries between convex as well as non-convex objects. Our approach is applicable to general shapes, where one or more objects are represented using Gaussian probability distributions. We present a fast new algorithm for a pair of convex objects, and extend the approach to non-convex models using hierarchical representations. We highlight the performance of our algorithms with various convex and non-convex shapes on complex synthetic benchmarks and trajectory planning benchmarks for a 7-DOF Fetch robot arm.

## I. INTRODUCTION

Collision detection is an important problem in many applications, including physics-based simulation and robotics. In robot motion planning, collision detection is regarded as one of the major bottlenecks. There is extensive work on faster collision checking for convex shapes, hierarchical algorithms, and methods for deformable models [1], [2], [3]. These prior collision detection techniques assume an exact representation of the geometric objects. They perform exact interference tests between the primitives and return the overlapping features.

In many applications, including robotics, virtual environments, and dynamic simulation, exact representations of the primitives are not easily available. Rather, the object representations are described using probability distribution functions. This may occur because the environment data is captured using sensors and only partial observations are therefore available. Furthermore, the primitives captured or extracted using sensors tend to be noisy. In this case, the goal is to compute the *collision probability* of two or more objects when one or more object representations (e.g. positions, orientations, etc.) are represented in terms of probability distributions [4], [5], [6]. In many robotics applications, such probabilistic collision queries are performed on the imperfect representations due to the uncertainties. For example, the planning of robot motions in dynamic real-world environments has to compute safe robot trajectories that avoid collisions with moving obstacles. In this case, the future obstacle positions are not known exactly and typically predicted using probability distributions. In other cases, the environment is typically represented using point clouds with some error distribution.

Typically, these probability distributions are approximated using Gaussian distributions and the collision computations are performed using probability distribution functions (PDF). The resulting probabilistic collision detection for objects with



(a)

**Fig. 1: Trajectory planning for robot avoiding human arm using our probabilistic collision algorithm:** The moving human arm is tracked using a point cloud sensor and the positional error is represented as a Gaussian probability distribution. The 95% boundary of Gaussian distribution is represented by the red circle. We use our novel probabilistic algorithm to guarantee that the collision probability is less than 5% at any state during the given trajectory. In order to handle such non-convex objects and compute tight bounds for probabilistic collision detection, we found that a hierarchy of OBBs (oriented bounding boxes) provides the best results in terms of running time and probabilistic bounds.

Gaussian distributions is defined using a confidence level, and probability can be evaluated using numerical integration or stochastic techniques. However, accurate techniques based on Monte Carlo integration are too slow for realtime applications. Current algorithms for probabilistic computations are based on circular or spherical approximations [7], [8], [9], but they tend to be rather conservative (in terms of bounds) for general non-convex shapes.

**Main Results:** In this paper, we present a fast probabilistic collision detection algorithm for general non-convex models. Our approach is applicable to any models represented in terms of inexact polygons or noisy point cloud data in which we are given two general shapes and the error is represented using Gaussian probability distributions. We compute hierarchical representations of non-convex models using simpler bounding volumes and present a novel hierarchical probabilistic collision detection algorithm. Moreover, we present

fast and reliable algorithms to perform probabilistic collision detection on simple shapes such as AABBs (axis-aligned bounding boxes), OBBs (oriented-bounding boxes), k-DOPs (k discretely oriented polytopes), and convex polytopes. We have evaluated the performance of our probabilistic collision detection on many synthetic and real-world benchmarks (captured using the Kinect). Moreover, we have integrated them with trajectory planning, and highlight the performance for real-time motion prediction and planning for a 7-DOF Fetch robot. We also evaluate them on complex synthetic benchmarks and observe that the hierarchies of OBBs or k-DPS provide the best balance between tight bounds and running times.

The rest of the paper is organized as follows. We survey prior work on exact and probabilistic collision detection algorithms in Section II. Section III gives an overview of the problem of probabilistic collision detection, and we present our algorithm for convex and non-convex polygonal models in Section IV. We analyze these algorithms and highlight their performance on complex benchmarks in Section V.

## II. RELATED WORK

In this section, we give a brief overview of prior work on exact and probabilistic collision detection algorithms.

### A. Collision Detection using Bounding Volume Hierarchies

There is extensive work on exact collision detection algorithms for geometric models represented using triangles or higher order primitives. These include efficient algorithms for convex polytopes [1], [10] and general algorithms for non-convex shapes using bounding volume hierarchies [2], [11]. Many techniques have been proposed to improve the efficiency of collision detection using Bounding Volume Hierarchies (BVHs). Some of the commonly-used hierarchies are based on bounding volumes such as spheres [12], [13] or axis-aligned bounding boxes (AABBs) [14]. Other algorithms that use tighter-fitting bounding volumes include oriented-bounding boxes (OBBs) [15], discrete oriented polytopes (k-DOPs) [16], or their hybrid combinations [17].

There is a trade-off between different bounding volume algorithms. The simple bounding volume algorithms have lower overhead in terms of overlap test, but can result in a high number of false positives and exact tests between the primitives. Tighter-fitting bounding volume algorithms may involve more complex overlap tests, but the tighter bounds reduce the number of false positives. The relative performance of different BVH-based algorithms varies based on the complexity and shape of geometric models and their relative configurations.

### B. Probabilistic Collision Detection

A common approach to checking for collisions between noisy geometric datasets is to perform exact collision checking using enlarged volumes that enclose the original object primitives [18], [19]. Many approaches handling point cloud sensor data convert the data into a set of boxes and check for collisions between the boxes and a robot [4]. Other

approaches generally enlarge the object bounding volumes to compute a new bounding volume for a given confidence level. This may correspond to a sphere [20] or a Sigma hull [21]. However, the computed volume overestimates the collision probability.

Other approaches have been proposed to perform probabilistic collision detection on point cloud data. Bae et al. [5] presented a closed-form expression for the positional uncertainty of point clouds. Pan et al. [6] reformulated the probabilistic collision detection problem as a classification problem and computed per point collision probability. However, these approaches assume that the environment is mostly static. For realtime collision detection, probabilistic collision detection is performed using broad phase data structures that handle large point clouds [9].

Some approaches approximate the collision probability using numerical integrations or stochastic techniques [22], [23], which require a large number of sample evaluations to compute an accurate probability. Guibas et al. [24] evaluate collision probability bounds using numerical integrations in multiple resolutions, which can avoid unnecessary numerical integrations in the high-resolution. The collision probability between objects with Gaussian distributions can be approximated using the probability at a single configuration, which corresponds to the mean [7] or the maximum [8] of the probability distribution function (PDF).

## III. OVERVIEW

In this section, we define the symbols and notation used in the rest of the paper and give the problem statement of probabilistic collision detection.

### A. Symbols and Notation

We use upper case symbols, like  $A$  and  $B$ , to denote 3D input primitives. We use boldface letters, such as  $\mathbf{p}$  or  $\mathbf{x}$ , to represent vectors. The probability that an *event* can occur is denoted as  $P(event)$ . We represent Gaussian distribution as  $\mathcal{N}(\mathbf{p}, \Sigma)$ , where  $\mathbf{p}$  is a mean vector and  $\Sigma$  is a covariance matrix. The probability density function of the Gaussian distribution  $\mathcal{N}(\mathbf{p}, \Sigma)$  is represented as  $p(\mathbf{x}, \mathbf{p}, \Sigma)$ .

### B. Background

Probabilistic collision detection algorithms are used to perform collision checking between two or more objects when the objects are not represented exactly and some of the input information such as positions or orientations of polygons or point clouds are given as probability distributions. The output could be a binary answer corresponding to whether or not those objects overlap. In another case, it could be a probability value that corresponds to the probability that an overlap occurs for the given input. This probability value is useful in many applications where a binary value is not enough to evaluate the given input.

In many applications, probability distributions are approximated using Gaussian distributions, which allow efficient computation using probability distribution functions (PDF). However, Gaussian distributions have non-zero probabilities

in the entire problem space, which always result in non-zero collision probabilities. Therefore, the probabilistic collision detection for objects with Gaussian distributions is defined using a *confidence level*, which is a threshold probability that determines the binary output of the collision detection (active collision or not) from the computed collision probability (e.g. 0.99).

In general, the exact collision probability is not computable even for a simple collision scenario between circles with a Gaussian distribution (see Sec. III-D). The probability can be approximated using a numerical integration or a stochastic technique such as the Monte Carlo method [22], [23], which requires a large number of sample evaluations to compute an accurate probability.

Therefore, efficient approximation methods have been proposed to compute the collision probability with tight bounds, without the evaluation of a large number of samples [7], [8], for a limited type of bounding volumes such as circles or spheres. In this paper, we show that our novel algorithms based on OBBs or k-DOPs provide better solution

### C. Problem Statement

We assume that the two input primitives,  $A$  and  $B$ , are each represented as a polygon or triangle mesh. We also assume that one Gaussian distribution (PDF) is available for each object. In this paper, we consider the cases when  $A$  and  $B$  are both convex polytopes, or general non-convex shapes. If the input is given as 3D point cloud data with some error distribution, we pre-compute a mesh representation and the appropriate Gaussian distribution from the point cloud data.

A positional displacement vector  $\epsilon$  is applied as translational operator to the volume  $A$ , which is denoted as  $A + \epsilon = \{\mathbf{a} + \epsilon | \mathbf{a} \in A\}$ . We assume a Gaussian distribution assumption on  $\epsilon$  with zero mean and covariance matrix  $\Sigma$ . We define the output collision probability as

$$P((A + \epsilon) \cap B \neq \emptyset) \quad (1)$$

$$\epsilon \sim \mathcal{N}(\mathbf{p}, \Sigma).$$

The collision event is equivalent to

$$\begin{aligned} (A + \epsilon) \cap B &\neq \emptyset \\ \iff \exists \mathbf{a} \in A, \mathbf{b} \in B \quad s.t. \quad \mathbf{a} + \epsilon &= \mathbf{b} \\ \iff \exists \mathbf{a} \in A, \mathbf{b} \in B \quad s.t. \quad \epsilon &= -\mathbf{a} + \mathbf{b} \\ \iff \exists \mathbf{x} \in (-A) \oplus B \quad s.t. \quad \epsilon &= \mathbf{x} \\ \iff \epsilon \in (-A) \oplus B. \end{aligned}$$

Then, the probability is formulated as

$$P(\epsilon \in (-A) \oplus B) \quad (2)$$

$$= \iiint I(\mathbf{x} \in (-A) \oplus B) p(\mathbf{x}, \mathbf{0}, \Sigma) d\mathbf{x} \quad (3)$$

$$= \iiint_V p(\mathbf{x}, \mathbf{0}, \Sigma) d\mathbf{x}, \quad (4)$$

where the function  $I(\mathbf{x})$  and the obstacle function  $p(\mathbf{x}, \mathbf{p}, \Sigma)$  are defined as,

$$I(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is true} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad (5)$$

$$p(\mathbf{x}, \mathbf{p}, \Sigma) = \frac{e^{-0.5(\mathbf{x}-\mathbf{p})^T \Sigma^{-1} (\mathbf{x}-\mathbf{p})}}{\sqrt{(2\pi)^3 \|\Sigma\|}}, \quad (6)$$

respectively.

### D. Collision Probability Approximation

For a sphere  $A$  of radius  $r_2$  with  $\epsilon \sim \mathcal{N}(\mathbf{p}, \Sigma)$  and a sphere  $B$  with radius  $r_1$ , the exact probability of collision between them is given as the integration of  $\epsilon$  in  $V$ , where  $V$  is a sphere of radius  $r_1 + r_2$ . It is known that there is no closed form solution for the integral given in (4), even for spheres.

Du Toit and Burdick [7] approximate (4) as

$$P(\epsilon \in ((-A) \oplus B)) \approx \iiint_V 1 d\mathbf{x} \cdot p(\mathbf{x}, \mathbf{0}, \Sigma) \quad (7)$$

$$= \frac{4\pi}{3} (r_1 + r_2)^3 \cdot p(\mathbf{x}, \mathbf{0}, \Sigma). \quad (8)$$

However, this approximated probability can be either smaller or larger than the exact probability, and it is hard to give any guarantees in terms of lower or upper bound on the computed probability.

Park et al. [8] compute  $\mathbf{x}_{max}$ , the position that has the maximum probability of  $\mathcal{N}(\mathbf{p}, \Sigma)$  in  $V$ , and compute the upper bound of (4) as

$$P(\epsilon \in ((-A) \oplus B)) \approx \frac{4\pi}{3} (r_1 + r_2)^3 \cdot p(\mathbf{x}_{max}, \mathbf{0}, \Sigma), \quad (9)$$

where  $\mathbf{x}_{max}$  is computed as a solution of

$$\mathbf{x}_{max} = \arg \min_{\mathbf{x}} \{(\mathbf{x} - \mathbf{p})^T \Sigma^{-1} (\mathbf{x} - \mathbf{p}) + \lambda \mathbf{x}^2\} \quad (10)$$

with a Lagrange multiplier  $\lambda$ , using a one-dimensional numerical search. This approach guarantees that the approximation never underestimates the collision probability, but is limited to isotropic objects such as circles or spheres.

## IV. PROBABILISTIC COLLISION DETECTION

In this section, we present a fast algorithm for probabilistic collision detection between convex polytopes. Furthermore, we extend to non-convex models using convex decomposition and bounding volume hierarchies.

### A. Probabilistic Collision Detection for Convex Polyhedrons

We extend the probabilistic collision detection for any two 3D convex polyhedrons  $A$  and  $B$ . We transform the volume  $V$  in (4) by  $(\Sigma)^{-\frac{1}{2}}$  to normalize the Gaussian distribution, i.e.,

$$\begin{aligned} &\iiint_{V'} p(\mathbf{x}, \mathbf{0}, \mathbf{I}) d\mathbf{x} \\ &= \iiint_{V'} \frac{1}{\sqrt{8\pi^3}} \exp\left(-\frac{1}{2} \|\mathbf{x}\|^2\right) d\mathbf{x}, \end{aligned} \quad (11)$$

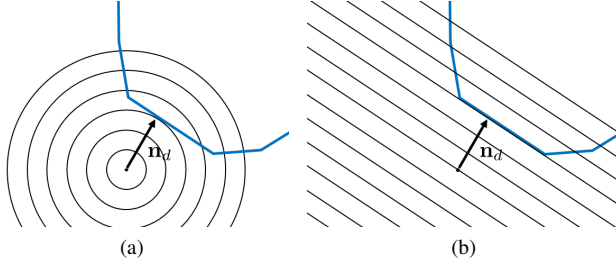


Fig. 2: (a) Contour plots of the bivariate Gaussian distribution, whose covariance matrix is normalized to an identity matrix. The minimum displacement vector  $\mathbf{n}_d$  from the center to the blue polygon is computed using the GJK algorithm. (b) Contour plots of upperly bounded function  $\mathbf{F}$  whose contours are perpendicular to  $\mathbf{n}_d$ . This function behaves like 1D Gaussian distribution.

where  $V' = (\Sigma)^{-\frac{1}{2}} V$ .

Evaluating the integration of Gaussian distribution over volume  $V'$  is still difficult, but a tight upper bound of the integral can be efficiently computed by replacing the probability distribution function with another one, which makes the integration computation easier. To define another function for computing the upper bound, first we compute the minimum displacement vector  $\mathbf{d}$  between convex shapes  $A'$  and  $B'$  which are transformed from  $A$  and  $B$ , respectively, by  $(\Sigma)^{-\frac{1}{2}}$ .  $\mathbf{d}$  is computed efficiently by the GJK algorithm [1]. Let  $\mathbf{n}_d$  be the unit directional vector. Then, by the Cauchy-Schwarz inequality,

$$(\mathbf{x} \cdot \mathbf{n}_d)^2 \leq \|\mathbf{x}\|^2.$$

Thus,

$$\begin{aligned} & \iiint_{V'} \frac{1}{\sqrt{8\pi^3}} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) d\mathbf{x} \\ & \leq \iiint_{V'} \frac{1}{\sqrt{8\pi^3}} \exp\left(-\frac{1}{2}(\mathbf{x} \cdot \mathbf{n}_d)^2\right) d\mathbf{x}. \end{aligned} \quad (12)$$

The benefit of replacing  $\|\mathbf{x}\|^2$  with  $(\mathbf{x} \cdot \mathbf{n}_d)^2$  is that the latter term, the projection of  $\mathbf{x}$  to the direction of  $\mathbf{n}_d$ , behaves like a 1D parameter instead of 3D, as shown in Fig. 2.

The divergence theorem is used to compute the upper bound on collision probability (12).

$$\iiint_{V'} \text{div}(\mathbf{F}) dV = \oint_{S'} (\mathbf{F} \cdot \mathbf{n}_S) dS,$$

where  $\mathbf{F}$  is a vector field,  $S'$  is the surface of  $V'$ ,  $dS$  is an infinitesimal area for integration, and  $\mathbf{n}_S$  is the normal vector of  $dS$ . We want the divergence of  $\mathbf{F}$  to be equal to the function inside the integral in (12). Let's define  $\mathbf{F}$  as

$$\mathbf{F}(\mathbf{x}) = \frac{1}{2\pi} \left( 1 + \text{erf}\left(\frac{\mathbf{x} \cdot \mathbf{n}_d}{\sqrt{2}}\right) \right) \mathbf{n}_d,$$

where  $\text{erf}()$  is the Gaussian error function. The directional derivative of  $\mathbf{F}(\mathbf{x})$  along any directional vector orthogonal to  $\mathbf{n}_d$  is zero because  $\mathbf{F}$  varies only along  $\mathbf{n}_d$ . The divergence of

$\mathbf{F}$  thus becomes  $(\partial \mathbf{F} / \partial \mathbf{n}_d)$ , and this is equal to the function in (12).

The right-hand side of the divergence theorem makes the integration efficient for a convex polyhedron  $S'$ . It is formulated as

$$\sum_i \oint_{\triangle S_{i1} S_{i2} S_{i3}} (\mathbf{F} \cdot \mathbf{n}_i) dS,$$

where  $\mathbf{n}_i$  is the normal vector of the  $i$ -th triangle  $\triangle S_{i1} S_{i2} S_{i3}$ . Because the error function integral over a triangle domain is another hard problem, the upper bound on the integral is evaluated instead, that is

$$\begin{aligned} & \oint_{\triangle S_{i1} S_{i2} S_{i3}} (\mathbf{F} \cdot \mathbf{n}_i) dS \\ & \leq \sum_i \left( \max_{j=1,2,3} \mathbf{F}(S_{ij}) \cdot \mathbf{n}_i \right) \text{Area}(\triangle S_{i1} S_{i2} S_{i3}). \end{aligned} \quad (13)$$

Summing up the values for all triangles gives a tight upper bound of collision probability, as defined in (1).

This gives higher value than expected collision probability, as compared to that computed using Monte Carlo methods. Monte Carlo methods take  $n_{MC}$  samples,  $\epsilon_i$ , from position error  $\mathcal{N}(\mathbf{0}, \Sigma)$ .

*Theorem 4.1:* As the number of Monte Carlo increases to the infinity, the approximated probability by Monte Carlo method is upperly bounded by Equation (13).

*Proof:* The collision probability approximated by Monte Carlo methods follows a binomial distribution  $B(n_{MC}, p^*)$  divided by  $n_{MC}$ :

$$\frac{1}{n_{MC}} I(\epsilon_i \in ((-A) \oplus B)) \sim \frac{1}{n_{MC}} B(n_{MC}, p^*),$$

where  $B(n, p)$  is a binomial distribution and  $p^*$  is the exact collision probability given as (11). The expectation of Monte Carlo approximation is

$$\begin{aligned} & \mathbb{E} \left[ \frac{1}{n_{MC}} B(n_{MC}, p^*) \right] \\ & = \frac{1}{n_{MC}} \mathbb{E}[B(n_{MC}, p^*)] \\ & = \frac{1}{n_{MC}} n_{MC} p^* = p^* \\ & = \iiint_{V'} \frac{1}{\sqrt{8\pi^3}} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) d\mathbf{x} \\ & \leq \iiint_{V'} \frac{1}{\sqrt{8\pi^3}} \exp\left(-\frac{1}{2}(\mathbf{x} \cdot \mathbf{n}_d)^2\right) d\mathbf{x} \\ & = \sum_i \oint_{\triangle S_{i1} S_{i2} S_{i3}} (\mathbf{F} \cdot \mathbf{n}_i) dS \\ & \leq \sum_i \left( \max_{j=1,2,3} \mathbf{F}(S_{ij}) \cdot \mathbf{n}_i \right) \text{Area}(\triangle S_{i1} S_{i2} S_{i3}). \end{aligned}$$

The variance of Monte Carlo approximation converges to

**Algorithm 1**  $p_{col} = \text{ConvexPCD}(A, B, \Sigma)$ 

: Compute the upper bound on collision probability between convex polytopes  $A$  and  $B$ , given a covariance matrix  $\Sigma$ .

**Input:** two convex shapes  $A$  and  $B$ ,

**Output:** Upper bound on collision probability  $p_{col}$

- 1:  $A' = (\Sigma)^{-\frac{1}{2}} A$
- 2:  $B' = (\Sigma)^{-\frac{1}{2}} B$
- 3:  $\mathbf{n}_d = \text{GJK}(A', B')$  to define  $\mathbf{F}$
- 4:  $V' = \text{MinkowskiSum}(A', B')$
- 5:  $p_{col} = 0$
- 6: **for all**  $i$  of  $V'$  **do**
- 7:   Add Equation (13) to  $p_{col}$
- 8: **end for**
- 9: **return**  $p_{col}$

zero:

$$\begin{aligned}
 & \text{Var} \left[ \frac{1}{n_{MC}} B(n_{MC}, p^*) \right] \\
 &= \frac{1}{n_{MC}^2} \text{Var} [B(n_{MC}, p^*)] \\
 &= \frac{1}{n_{MC}^2} n_{MC} p^* (1 - p^*) \\
 &= \frac{1}{n_{MC}} p^* (1 - p^*) \xrightarrow{n_{MC} \rightarrow \infty} 0.
 \end{aligned}$$

Thus, the Monte Carlo approximation converges to the exact collision probability as the number of samples increases, and the approximation value is bounded by Equation (13). ■

Algorithm 1 describes how the upper bound on collision probability for two convex shapes is computed. The two input shapes are transformed so that the variance of Gaussian distribution becomes isotropic. Then, GJK algorithm is used for finding the minimum displacement vector  $\Sigma$  between  $A'$  and  $B'$ . The Minkowski sum  $V'$  is computed and then the upper bound on collision probability is directly computed from its all triangles. Computing the Minkowski sum (line 4) has a time complexity linearly proportional the number of output vertices. The loop (line 6-8) also has a time complexity linearly proportional to the number of output vertices. In the worst case, the time complexity can be  $O(n^2)$ , where the number of vertex in  $A'$  and  $B'$  is  $O(n)$ . However, special types of convex shapes can be used for efficient computation. k-DOPs have  $k$  planes surrounding the objects, and the Minkowski sum computation takes  $O(k)$  instead of  $O(k^2)$ , because the triangles are parallel to corresponding ones. AABB is special case of k-DOPs with  $k = 3$ . OBB has 3 pairs of parallel rectangular faces, so their Minkowski sum can be efficiently computed. As a result, we have much simpler and faster algorithms to compute the collision probability for these widely used bounding volumes.

### B. Probabilistic Collision Detection for General Shapes

In order to handle non-convex objects efficiently, we decompose them into many convex shapes and build bounding volume hierarchies (BVHs) to reduce the number of operations for every convex-convex shape pair.

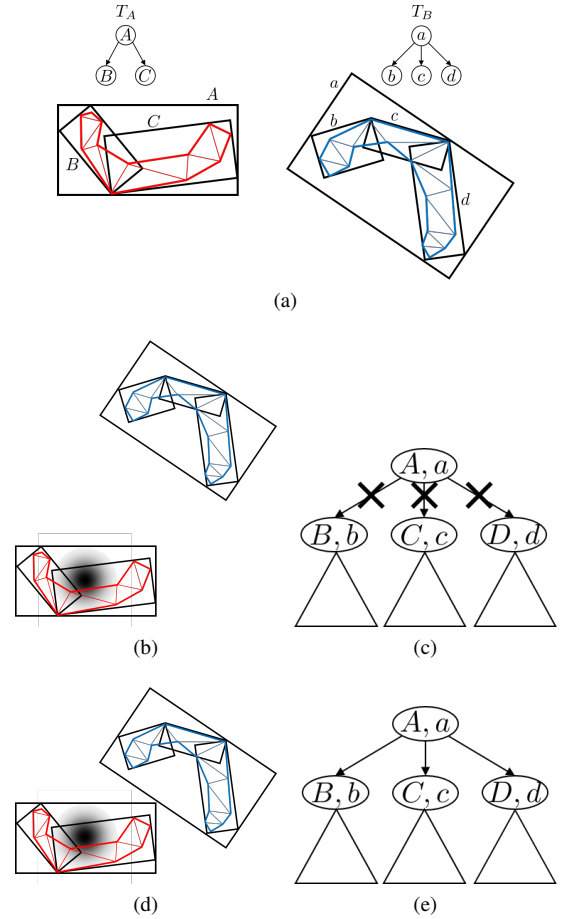


Fig. 3: A 2D example of BVHs and their traversal during probabilistic collision detection. The non-convex shapes are shown as red and blue polygons, and they are bounded by OBB trees in (a). The probability distribution of displacement of the red object is drawn at the center of the bounding box in (b) and (d). BVH traversal (b-c) stops or (d-e) continues, which is determined by comparing the upper bound on collision probability and the confidence level  $\delta_{CD}$ .

We define a confidence level  $\delta_{CD} < 1$  (usually higher than 0.90), an indication of when the traversal in BVHs stops. The confidence level means that we can confidently say that there will likely be no collision if the collision probability is less than  $1 - \delta_{CD}$ . To be more specific, the upper bound on the collision probability of two bounding volumes of current traversing nodes is computed as described in Section IV-A, and is compared to  $1 - \delta_{CD}$ . If the collision probability is less than  $1 - \delta_{CD}$ , it means that the probability of collision at this level of bounding volume is sufficiently low. Otherwise, traversing further to the child nodes is needed to compute a tighter bound on collision probability.

Fig. 3 shows two cases in BVH traversal: stop and continue. In Fig. 3(b), the two bounding volumes are so far away that the upper bound on collision probability is already sufficiently below the threshold  $1 - \delta_{CD}$ . Therefore, traversal stops in the BVHs in Fig. 3(c), resulting in the reduction in traversing. On the other hand, in Fig. 3(d), the



---

**Algorithm 2**  $p_{col} = \text{GeneralPCD}(T_A, T_B, \Sigma, \delta_{CD})$

---

: Compute the upper bound on collision probability between non-convex polytopes  $A$  and  $B$ , given BVHs  $T_A$  and  $T_B$ , a covariance matrix  $\Sigma$  and a confidence level  $\delta_{CD}$ .

---

**Input:** two BVHs  $T_A$  and  $T_B$ , confidence level  $\delta_{CD}$ ,

**Output:** Upper bound on collision probability  $p_{col}$

```

1:  $v_A = T_A.\text{root}$ 
2:  $v_B = T_B.\text{root}$ 
3:  $p_{root} = \text{ConvexPCD}(v_A, v_B, \Sigma)$ 
4: if  $p_{root} < 1 - \delta_{CD}$  or both  $v_A$  and  $v_B$  are leaf nodes
   then
5:   return  $p_{col} = p_{root}$ 
6: end if
7: if  $v_A$  has children and  $\text{Vol}(v_A) \geq \text{Vol}(v_B)$  then
8:   return  $p_{col} = \sum_{c_A:\text{child}} \text{GeneralPCD}(c_A, T_B, \Sigma, \delta_{CD})$ 
9: else
10:  return  $p_{col} = \sum_{c_B:\text{child}} \text{GeneralPCD}(T_A, c_B, \Sigma, \delta_{CD})$ 
11: end if

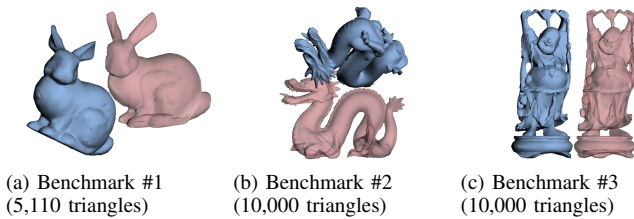
```

---

two bounding volumes are so close that they are likely to collide with each other, though the actual shapes are not so close. In this case, the traversal continues to the children for tighter bound computation.

Algorithm 2 describes how the BVHs are traversed and when the traversal stops to reduce the number of computations. Two BVHs  $T_A$  and  $T_B$ , corresponding to any type of bounding volumes (e.g., spheres, AABBs, OBBs, k-DOPs, or convex hulls), are traversed simultaneously from roots to leaf nodes. The upper bound on collision probability is computed for the convex volume of current nodes at line 3 as described in Section IV-A. The traversal stops when the collision probability is sufficiently low compared to  $1 - \delta_{CD}$  or it has reached the leaf nodes (line 4). Otherwise, traversal continues to the child nodes of  $T_A$  or  $T_B$ , depending on the size of two bounding volumes, in lines 7-11.

## V. RESULTS



**Fig. 4: Geometry Models and Obstacle Position Distributions.** We evaluate the performance of different algorithms with different non-convex shapes: (a) Benchmark #1: bunny, (b) Benchmark #2: dragon, and (c) Benchmark #3: buddha. The red objects have Gaussian distribution errors in position.

In this section, we describe our implementation and highlight the performance of our probabilistic collision detection algorithms on synthetic and real-world benchmarks. Furthermore, we compare the performance of different bounding

volumes (i.e. spheres, AABBs, OBBs, k-DOPs, convex hull) in terms of the probability bounds and the query time. We use the following performance measurements to evaluate the performance:

- **Bounding volume approximation error (BVE):** It is measured as the ratio of the volume of bounding shapes to that of the original or underlying shapes, which is always greater than or equal to 1. The closer this ratio is to 1, the better the bounding volume approximates the original shape. This ratio is low for tight fitting bounding volumes such as convex hull and high for spheres.
- **Collision probability upper bound (CP):** We compute the upper bound on collision probability between two nearby shapes based on Gaussian probability distribution. The actual collision probability cannot be computed analytically, so it is approximated by the Monte Carlo method. In practice, Monte Carlo methods can take a long time, but we assume that they provide the most accurate solutions. The upper bounds computed by our algorithms are expected to be higher than the Monte Carlo approximation. Our goal is to efficiently compute the tightest upper bound on collision probability with different bounding volumes for non-convex shapes.
- **Computation time (Time):** Each algorithm has a different time complexity, so it is important to evaluate the trade-off between the tightness of a collision probability upper bound and time complexity. All the timings reported in this paper are measured in milliseconds (ms) and generated on a single CPU core of Intel I7 processor.

The Monte Carlo method provides an approximated value of the actual collision probability by sampling 10,000 points from the obstacle's position distribution in our benchmarks. We expect that is the most accurate available answer for collision probability computations. As a result, all the collision probability computations computed using our approaches compute a higher value than Monte Carlo approximation, but takes significantly less time than Monte Carlo integration. For AABBs, different global coordinate systems can result in different AABB shapes. As a result, we randomly generate 10,000 global coordinate systems from the  $SO(3)$  group to generate the axes and compute the averages over these coordinates.

### A. Complex Non-convex Models

We measure these values for different non-convex shapes shown in Fig. 4. These are very complex synthetic benchmarks in close proximity that are used to evaluate different collision detection algorithms. These models are scales so that the length of the longest bounding ABB is 1m. The positions are determined so that the minimum distance between two shapes is 1 cm or 5 cm, and we compute the collision probabilities for such configurations. The covariance matrix of position error,  $\Sigma$ , is set with a random orientation as its main axes and use 1cm, 3cm, and 5cm as the standard deviation along the axes.

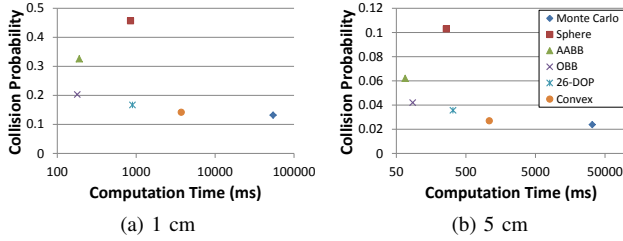


Fig. 5: Performance graph showing the computation time (x-axis) vs. the upper bound on collision probability (y-axis) between the two bunny models away from (a) 1 cm and (b) 5 cm.

Table I and Fig. 5 show the upper bound on collision probability and the query time using different bounding volumes with different approximation errors. We notice that the OBBs seem to provide the best balance between probability bound and the running time. In case of spheres [7], [8]), the running time is higher than AABBs or OBBs, because the culling efficiency of sphere is low because the collision probability bounds are not tight. As a result, the algorithm traverses more nodes in the hierarchy. As compared to OBBs, 26-DOPs provide tighter bounds on the collision probability. However, the query time on 26-DOPs is higher than AABBs and OBBs, because the overhead of computing the Minkowski sum. The performance of OBB is comparable to AABB, but OBBs provide a tighter bound on the probability computation.

### B. Robot Trajectory Planning with Sensor Errors

We have integrated our probabilistic collision algorithms with trajectory planning algorithm and evaluated their performance on a 7-DOF Fetch arm. In our experiments, we compute different bounding volumes for the robot and the obstacles in the scene. Furthermore, the scene consists of a dynamic human obstacles (see Fig. 1) and we assume that the robot is operating in a close proximity to the human. In this case, the robot predicts the trajectory of the human, represents that with a Gaussian distribution and uses probabilistic collision checking to compute a collision-free trajectory. The robot uses a Kinect as the depth sensor, which can represent the human using with  $512 \times 424$  points. We also compute the state of human obstacle model, which is represented using 60 DOFs. It is assumed that the obstacle shape is known in our implementation. The goal of robot trajectory planning is to generate a path where the collision probability is less than a user-specified safety level. In our benchmarks, the safety level is set to 5%, meaning that the collision probability between an obstacle and a robot state at any time along the trajectory should be less than 5%. A tighter bound computed using probabilistic collision detection algorithm increases the search space of the trajectory planning algorithm.

## VI. CONCLUSIONS, LIMITATIONS AND FUTURE WORK

In this paper, we present fast and reliable probabilistic collision detection algorithms for general convex and non-convex shape models. This includes an efficient algorithm for

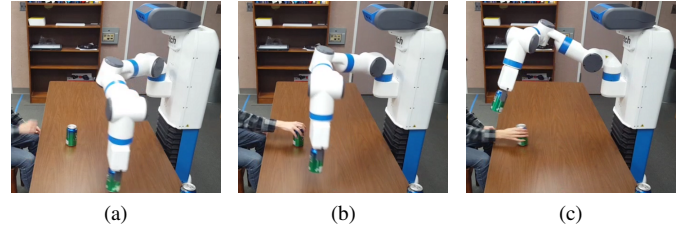


Fig. 6: **Trajectory Planning with a 7-DOF robot:** The robot is used to move the coke cans on a table while avoiding a human user, i.e. the human arm is close to the robot. The robot trajectory is computed using a planner that uses our probabilistic collision detection. At any time along the robot trajectory, the collision probability is less than 5%.

convex polytopes that is based on computing the Minkowski sums of two polytopes. We show that the probability bound computed by our approach is always an upper bound. We also simplify the computations and present optimized algorithms for simpler convex shapes such as AABBs, OBBs and convex hulls. Based on these bounding volumes, we present a hierarchical algorithm for non-convex shapes. We have evaluated their performance on complex synthetic benchmarks and also integrated them with a real-time trajectory planning algorithm. In practice, we observe that OBBs seem to present the best balance between the tightness of bounds and the query times.

Our approach has some limitation. Current formulation is designed for rigid model and the error is represented using a Gaussian probability distribution. The performance of different bounding volumes can vary based on the shape of the objects and their relative configurations. Furthermore, we only take into account only the position error in our benchmarks, and not the orientation error. There are many avenues for future work. Besides overcoming these limitations, we will like to design efficient algorithms with tight bounds for articulated models. It would also be useful to derive similar algorithms for other distributions and evaluate their performance in real-world scenarios.

## REFERENCES

- [1] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.
- [2] M. C. Lin and D. Manocha, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry: Collision detection*. CRC Press, 2003, pp. 787–808.
- [3] M. Teschner, S. Kimmle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, et al., "Collision detection for deformable objects," in *Computer graphics forum*, vol. 24, no. 1. Wiley Online Library, 2005, pp. 61–81.
- [4] R. B. Rusu, I. Alexandru, B. Gerkey, S. Chitta, M. Beetz, L. E. Kavraki, et al., "Real-time perception-guided motion planning for a personal robot," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4245–4252.
- [5] K.-H. Bae, D. Belton, and D. D. Lichti, "A closed-form expression of the positional uncertainty for 3d point clouds," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 4, pp. 577–590, 2009.

- [6] J. Pan, S. Chitta, and D. Manocha, "Probabilistic collision detection between noisy point clouds using robust classification," in *International Symposium on Robotics Research (ISRR)*, 2011.
- [7] N. E. Du Toit and J. W. Burdick, "Probabilistic collision checking with chance constraints," *Robotics, IEEE Transactions on*, vol. 27, no. 4, pp. 809–815, 2011.
- [8] C. Park, J. S. Park, and D. Manocha, "Fast and bounded probabilistic collision detection in dynamic environments for high-dof trajectory planning," *arXiv preprint arXiv:1607.04788*, to appear in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.
- [9] J. Pan, I. A. Şucan, S. Chitta, and D. Manocha, "Real-time collision detection and distance computation on point cloud sensor data," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3593–3599.
- [10] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1008–1014.
- [11] C. Ericson, *Real-time collision detection*. CRC Press, 2004.
- [12] P. M. Hubbard, "Interactive collision detection," in *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*. IEEE, 1993, pp. 24–31.
- [13] L. He and J. van den Berg, "Efficient exact collision-checking of 3-d rigid body motions using linear transformations and distance computations in workspace," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2059–2064.
- [14] G. v. d. Bergen, "Efficient collision detection of complex deformable models using aabb trees," *Journal of Graphics Tools*, vol. 2, no. 4, pp. 1–13, 1997.
- [15] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtrees: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 171–180.
- [16] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-dops," *IEEE transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.
- [17] A. Sanna and M. Milani, "Cdfast: an algorithm combining different bounding volume strategies for real time collision detection," *SCI Proceedings*, vol. 2, pp. 144–149, 2004.
- [18] J. Van den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "LQG-Obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 346–353.
- [19] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2012.
- [20] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 723–730.
- [21] A. Lee, Y. Duan, S. Patil, J. Schulman, Z. McCarthy, J. van den Berg, K. Goldberg, and P. Abbeel, "Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5660–5667.
- [22] L. Blackmore, "A probabilistic particle control approach to optimal, robust predictive control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, no. 10, 2006.
- [23] A. Lambert, D. Gruyer, and G. S. Pierre, "A fast monte carlo algorithm for collision probability estimation," in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*. IEEE, 2008, pp. 406–411.
- [24] L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman, "Bounded uncertainty roadmaps for path planning," in *Algorithmic Foundations of Robotics VIII*. Springer, 2010, pp. 199–215.



BVH type	BVE			distance between shapes : 1 cm						distance between shapes : 5 cm					
				CP (%)			Computation Time (ms)			CP (%)			Computation Time (ms)		
	#1	#2	#3	#1	#2	#3	#1	#2	#3	#1	#2	#3	#1	#2	#3
Monte Carlo integration	1.00	1.00	1.00	13.2	15.7	6.82	54,300	213,000	108,000	2.38	3.72	0.89	32,800	176,000	92,000
Spheres	4.80	6.28	6.50	45.7	52.8	34.9	847	2,480	2,070	10.3	18.1	10.3	261	1,580	1,030
AABBs	3.28	5.20	4.96	32.6	38.1	31.7	190	578	420	6.21	11.9	7.06	67	213	126
OBBs	1.63	3.16	2.94	20.3	31.7	28.8	179	580	441	4.20	6.23	2.00	86	278	158
26-DOPs	1.26	1.68	1.88	16.7	22.3	12.7	897	3,320	2,637	3.57	2.90	1.27	326	1,010	868
Convex	1.00	1.00	1.00	14.2	18.9	8.09	3,721	10,800	8,780	2.70	4.18	1.12	1,082	2,520	2,330

TABLE I: Performance of different algorithms for complex Benchmarks #1-3 shown in Fig. 4. We measure the bounding volume approximation error (BVE), collision probability upper bound (CP), and query computation time for different bounding volumes. The Monte Carlo integration schemes provides the most accurate result, but is very expensive. On the other hand, spheres (used in prior methods [7], [8]) do not provide tight bounds, as observed with CP values. In practice, OBBs seem to provide the best balance between collision probability bounds and query times.

BVH type	Moving coke cans		Waving an arm	
	CP (%)	Time (ms)	CP (%)	Time (ms)
Monte Carlo	1.00	20,800	1.00	15,200
Sphere	3.81	3,260	4.72	1,730
AABB	3.72	637	3.26	425
OBB	2.50	819	2.80	433
26-DOPs	2.11	1,280	2.71	823
Convex	1.59	8,440	1.39	4,630

TABLE II: Robot motion planning scenarios using collision probability computation as a constraint in terms of trajectory planning. The upper bound on collision probability between an obstacle and the robot trajectory, computed using different bounding volume types, should be less than 5%. The Sphere BVH computes the most conservative bound. We evaluated the exact collision probability and computation time for each trajectory for different bounding volume hierarchies in this real-world scenario. In these scenarios, OBBs provide the best balance between collision bounds and the query time.