Proximity Computations between Noisy Point Clouds using Robust Classification

Jia Pan [1] and Sachin Chitta [2] and Dinesh Manocha [3]
[1]*panj@cs.unc.edu,* [3] *dm@cs.unc.edu, Dept. of Computer Science, University of North Carolina at Chapel Hill*
[2]*sachinc@willowgarage.com, Willow Garage*

***Abstract*— We present a new approach to perform robust proximity queries between noisy point cloud data. Our approach takes into account the uncertainty that arises due to discretization error and noise, and formulates contact computation as a two-class classification problem. We use appropriate techniques from machine learning to compute the collision probability for each point in the input data and accelerate the computation using stochastic traversal of bounding volume hierarchies. We highlight the performance of our algorithm on point clouds captured using laser range finders, active stereo cameras and synthetic datasets.**

I. Introduction

The problems of collision detection and proximity computation are widely studied in different areas, including robotics, physically-based modeling, haptics and virtual environments. In particular, reliable and fast collision detection algorithms are required for robot motion planning, grasping and dynamics simulation to enforce the non-penetration constraints with the environment.

Most of the prior work on collision and proximity queries assumes an exact geometric description of the objects in the scene as a polygon mesh. However, these methods may not work well on robots operating in real-world environments, where only partial observations of the environment are possible based on robot sensors. For example, inaccurate motor control makes a robot deviate from its exact configuration and the sensors tend to add noise to the environment measurements. Current robot sensors including cameras and LIDAR and new devices such as Kinect can easily generate detailed point cloud data (represented as or converted from RGB-D data) of real-world environments. However, it is hard to directly use prior collision detection algorithms which perform a boolean query and compute a yes/no answer on such point cloud datasets. Moreover, exact collision checking may not be suitable in terms of handling uncertainty in perception and control, which also causes uncertainty in collision results. For many robotics applications, such as grasping or motion planning, we need to reduce the risk of physical contacts between the robot and the environment that may result in damages. Hence, we need to develop methods that tend to minimize the probability of collisions.

In terms of collision checking with point clouds, the most natural way is first reconstructing triangle meshes from point clouds and then performing exact mesh collision checking between the reconstructed surfaces. However, this two-step approach suffers from efficiency ($>$ 10s for 10K triangles) and robustness issues that arise in terms of using reconstruction algorithms (e.g. reconstruction quality, sensitive to parameter and noise, etc). Or we can simply expand every point as a sphere or box with suitable radius and approximate the object as a union of spheres [2] or boxes [5] for collision checking. The main difficulty is in terms of automatically choosing different sphere radii or box dimensions for different points. Moreover, these methods do not take into account the two types of uncertainties in the real world point cloud data: *discretization errors* and *position errors* or *noise uncertainty* [3]. Intuitively, the discretization error refers to how these point samples are distributed on the boundary of the surface and is decided by the sampling mode and density of the surface. And the position error measures the imprecision in the coordinates of each point and is decided by the sensor's noise property.

In this paper, we present a novel probabilistic approach to perform proximity queries on such point cloud data sets to handle uncertainty. Our algorithm can handle noisy or inexact point data representations that are gathered using laser range finders or active stereo cameras. In order to handle the point cloud data with noise, we reformulate the collision detection problem as a two-class classification problem, where points of different objects belong to different classes. The collision probability is directly related to the separability of the corresponding two-class problem, which can be elegantly and efficiently solved using support vector machines (SVMs). We accelerate the computation using bounding volume hierarchies and perform a stochastic traversal of the hierarchies that takes into account the noise and uncertainty. We test our algorithm on point clouds from PR2 sensors, Kinect and synthetic data sets. Our method can provide robust results for probabilistic collision detection and its runtime performance is similar to that of hierarchy-based collision detection algorithms for triangle meshes (e.g. 500-1000ms for 10K points).

II. Algorithm

The main *pipeline* of our system consists of three steps: 1) Obtain raw data from sensors and filter the point clouds to remove points on the robot and reduce the shadow effect [5]; 2) Compute the separating surface between two point clouds by estimating the noise from sensor parameters; 3) Estimate the collision probability for each point and then the overall collision probability between two point clouds. Moreover, we

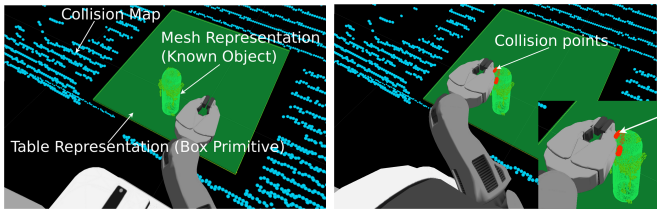

Fig. 1. *A visual representation of the collision information generated by the sensors on the PR2 robot. (Left) The environment for collision checking includes the points in the collision map (in light blue), mesh representations for known objects detected through visual sensing (in green). (Right) A representation of the collision points (shown by red spheres) between the gripper and the object on the table. We use our probabilistic algorithm for robust collision checking on noisy point clouds at interactive rates.*

use bounding volume hierarchies to accelerate the computation and recompute them for dynamic environments.

Our basic idea is to reformulate the point cloud collision detection problem as a two-class classificiation problem, which is originated from the basic definition of *collision free*: Given two point clouds $C_1 = \{\mathbf{x}_i^1\}_{i=1}^{n_1}$ and $C_2 = \{\mathbf{x}_i^2\}_{i=1}^{n_2}$ with n_1 and n_2 elements, respectively, C_1 and C_2 are collision-free if and only if there exists an oriented seperating surface P so that C_1 is completely on the one side of P while C_2 is on the other side. Therefore, we can transform the collision checking into the problem of finding the optimal surface that can separate the two point clouds as much as possible. We also require some additional smooth constraints on the separating surface to filter out the surfaces that are unlikely in the real world applications.

To reformulate the collision problem, we first rewrite $\mathbf{x}_i^l$ with $l \in \{1, 2\}$ as $(\mathbf{x}_i, c_i)$, where $\mathbf{x}_i = \mathbf{x}_i^l$ and $c_i = (-1)^{l+1} \in \{-1, 1\}$ denotes which object the point $\mathbf{x}_i$ belongs to. As a result, we have $n_1 + n_2$ elements in $\{(\mathbf{x}_i, c_i)\}$. Next, in the terminology of machine learning, finding an optimal separating surface will correspond to finding an optimal classifier that can minimize the expected risk (i.e. the classification error) on the classification problem whose data is drawn from $\{\mathbf{x} : \mathbf{x} \in S_1 \bigcup S_2\}$ and its *training set* is $\{(\mathbf{x}_i, c_i)\}$. However, unlike typical machine learning algorithms which only deal with cases where $(\mathbf{x}_i, c_i)$ are specified exactly, we also need to take into account the noise in $\mathbf{x}_i$. Our solution is based on the maximum-likelihood (ML) scheme, i.e. the optimal surface should maximize the probability on the observed inputs $\{(\mathbf{x}_i, c_i)\}$. Usually we formulate the separating surface P as a parameterized surface with parameter θ (e.g. P is $\{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$ if P is a plane and $\theta = \{\mathbf{w}, b\}$. Or P is $\{\mathbf{x} : \mathbf{w}^T\Phi(\mathbf{x}) + b = 0\}$ if P is nonlinear and Φ is a function mapping to $\mathcal{R}^3$), therefore the ML scheme can be solved by following optimiation problem about θ:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_i \ln \sup_{\mathbf{x}_i'} p(\mathbf{x}_i', c_i; \theta) p(\mathbf{x}_i | \mathbf{x}_i'; \Sigma_i), \qquad (1)$$

where $p(\mathbf{x}_i', c_i; \theta)$ is used to model the discretization error and $p(\mathbf{x}_i | \mathbf{x}_i'; \Sigma_i)$ is used to model the position error.

Equation 1 is directly related to the problem of *robust classification* [1]. If P is a planar surface, this problem can be solved efficiently using a SVM-alike algorithm [4]:

$$\begin{aligned} \underset{\mathbf{w}, b, \xi_i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & c_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i + r_i \|\widetilde{\Sigma}_i^{1/2}\mathbf{w}\|, \quad \forall 1 \leq i \leq n; \\ & \xi_i \geq 0, \qquad \forall 1 \leq i \leq n, \end{aligned} \qquad (2)$$

where λ is a regularity term used to control the 'smoothness' of the separating plane.

When P is nonlinear (i.e. not a plane), we can turn Equation 2 into its dual form and apply the kernel trick [6] and compute an optimal nonlinear classifier.

Once the optimal $P = f(\theta, \mathbf{x})$ is calculated, we can compute the collision probability for each point, i.e. the probability that one point $\mathbf{x}_i$ lies on the wrong side of the separating surface:

$$\mathbb{P}_{\mathbf{x}_i' \sim \mathcal{N}(\mathbf{x}_i, \widetilde{\Sigma}_i)}(c_i f(\mathbf{x}_i') \leq 0) = \operatorname{cdf}(-c_i f(\mathbf{x}_i) / \|\widetilde{\Sigma}_i^{1/2} f'(\mathbf{x}_i)\|). \qquad (3)$$

Next, we can compute the collision probability between two point clouds based on these per-point collision probability. If a yes-no collision decision similar to traditional exact collision detection methods is required, our algorithm uses two thresholds $A \geq 0.5 \geq B$:

1) if collision probability $> A$, we report *collision-free*
2) if collision probability $< B$, we report *in-collision*
3) if collision probability is between A and B, we report *in-contact*.

For exact proximity computation on meshes, in-contact is used to describe the collision state that the two objects have common points with zero volume in 3D space. For the probabilistic proximity query, in-contact is used to describe the state that we do not have enough information to justify it as collision-free or in-collision. When collision-avoidance is critical for the underlying applications, we can use large conservative value for A and small conservative value for B to achieve higher guarantees.

We also use bounding volume hierarchies for point clouds to accelerate our probabilistic collision detection algorithm and the overall running time is similar to performing collision and proximity queries between triangulated models.

III. Result

We evaluated the performance of our algorithm on synthetic data set corresponding to a moving piano in a room with tables. We generated a point cloud by sampling the polygons and adding some noise. We used the PQP package to perform exact collision detection and separating distance query between the exact, triangulated model and compare the results with probabilistic collision detection on the resulting point cloud (see Fig 2). We see a high correlation between our results and the actual separating distance, and it varies based on the level of noise. This shows that our approach is quite robust and even works well in degenerate configurations, e.g. when the two objects are barely touching or very close to each other. Such configurations are more susceptible to noise and

the exact collision detection algorithms are very sensitive to these configurations.

We have applied our probabilistic collision detection to the point cloud data generated for manipulation using the PR2 robot. Point cloud data on the PR2 robot is generated from a scanning laser range finder (*Hokuyo Top-URG(UTM-30LX)*) and an active stereo camera (*WGE-100*). There aren't many algorithms or software implementation for collision checking between noisy point clouds. As a result, we compare our algorithm with the implementation in ROS (based on ODE) and exact collision detection on reconstructed meshes. The collision checking procedures used in ROS [5] are currently based on the collision checking implementation available as part of ODE software package [1]. The input to the collision checker is a combination of mesh models of the robot and objects in the environment and the collision map. The points in the collision map are represented as axis-aligned box primitives whose lengths are equal to the resolution at which the collision map is maintained. The current representation of the collision space considers every point in the collision map to be a potential obstacle. Thus, noise in the sensor data can frequently lead to false positives, i.e. the detection of potential collisions in parts of the environment where there are no obstacles. There is no robust criterion to compute the box size, which is a function of noise, so we can't compare all the features of our method with ODE collision checking. We also use a reconstruction algorithm to compute a triangle mesh from the point clouds and perform triangle-based collision detection as well as separating distance computation using PQP. Note that the reconstruction scheme only provides an approximate answer in this case.

As shown in Figure 3, our result matches well with the exact collision detection algorithm, especially, there is a high correlation with the computed separating distance. Furthermore, we notice that the collision probabilities computed by our approach change slowly when the noise increases. It is more robust as compared to the yes-no result computed by ODE on the point clouds, which is likely to frequently switch between collision-free and in-contact configurations, when the noise level changes. Moreover, from Figure 2 and Figure 3, we observe that configurations with the same distances to the obstacles can have large spread in the computed collision probabilities. The reason is that distance is only a partial measurement of collision status while our collision probability is a more complete description about collision status and provides more detailed information about the relative configurations, which is useful for applications such as grasping in constrained environments.

We also test our algorithm on the Kinect data provided by the Robotics and State Estimation Lab of the University of Washington. The data is a point cloud for a building environment as shown in Figure 4. We compare the result of our method with exact collision detection on reconstructed meshes. The reconstructed mesh for the environment and the robot (a piano) are shown in Figure 5. Figure 6 shows the comparison result: our result matches well with the approximate groundtruth provided by exact collision on reconstructed meshes. Similar to the previous two experiments, we observe configurations with the same distances to the obstacles can have large spread in the computed collision probabilities.

[1]http://opende.sourceforge.net/wiki/index.php/Manual_(Collision_Detection)

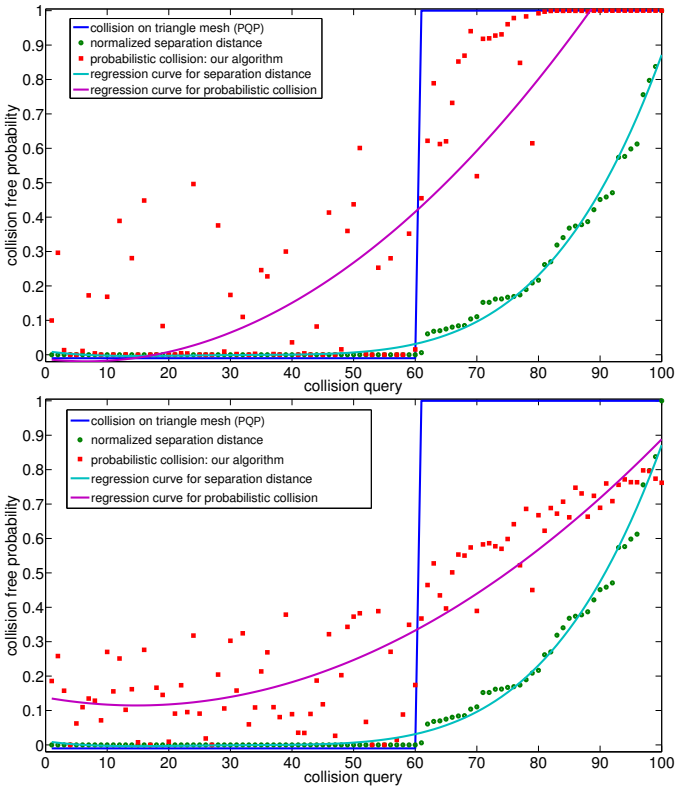

Fig. 2. *Comparison between the results for* 100 *random queries between prior collision detection algorithms for exact triangle meshes and our algorithm on the noisy point clouds generated using synthetic datasets.*

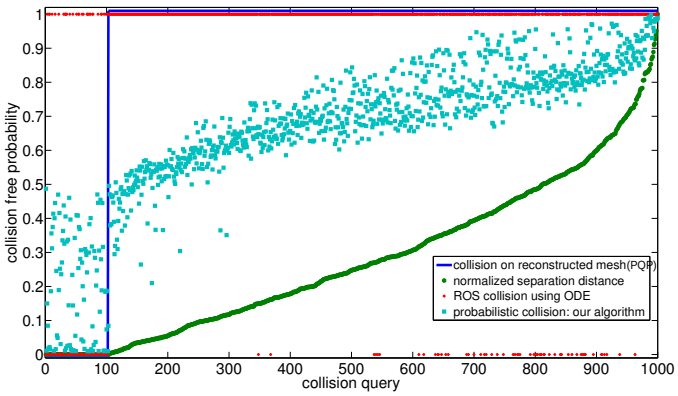

Fig. 3. *Robust Classification of contact status between the point clouds generated using PR2 sensors. We use* 5000 *random proximity queries.*

IV. Conclusion

We have presented a novel and robust method for contact computation between noisy point cloud data using machine learning methods. We reformulate collision detection as a two-

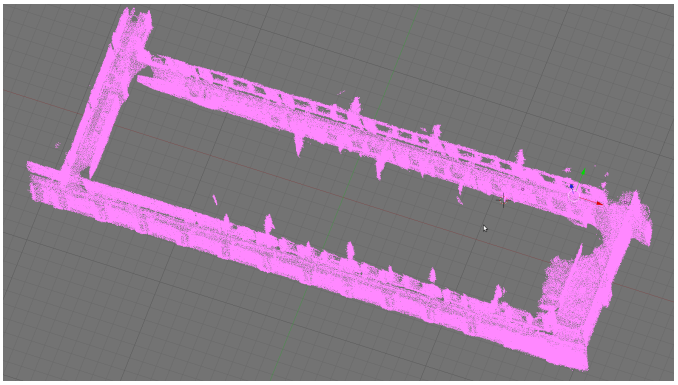

Fig. 4. *The Kinect point cloud used in our experiment.*

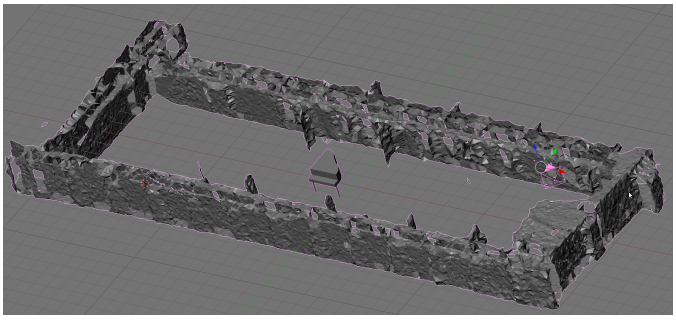

Fig. 5. *The mesh reconstructed from Kinect point cloud and the piano in the environment.*

classification problem and compute the collision probability at each point using support vector machines. We use a bounding volume and perform stochastic traversal to accelerate the computations. We have tested the results on synthetic and real-world data sets and the preliminary results are promising.

There are many avenues for future work. We need to test the performance on different robotic systems and evaluate its performance on tasks such as planning and grasping.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

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

[3] M. Pauly, N. J. Mitra, and L. Guibas, "Uncertainty and variability in point cloud surface data," in *Symposium on Point-Based Graphics*, 2004, pp. 77–84.

[4] P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola, "Second order cone programming approaches for handling missing and uncertain data," *Journal of Machine Learning Research*, vol. 7, pp. 1283–1314, Dec. 2006.

[5] I. A. Sucan, M. Kalakrishnan, and S. Chitta, "Combining planning techniques for manipulation using realtime perception," in *International Conference on Robotics and Automation*, 2010, pp. 2895–2901.

[6] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

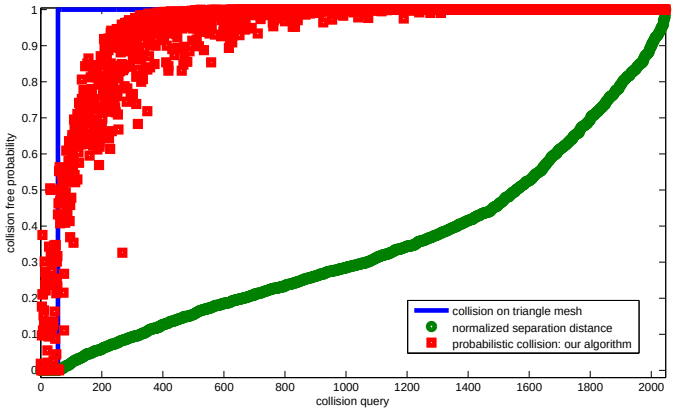

Fig. 6. *Robust Classification of contact status between the point clouds generated using Kinect. We use* 5000 *random proximity queries.*