Signed-Distance Collision Detection using a Predictive Net: Final Report

Cade Parkison, Braxton Smith
April 24, 2019

***Abstract*—ABSTRACT**

***Index Terms*—CNN, Signed-Distance Collision Function**

I. Introduction

FOR robotic planning systems to work properly they must accurately reason about their environment. One key aspect of this is checking for collisions given a joint configuration; not just of the robot with itself but also with the obstacles within its workspace.

Traditional collision detection in robotics typically involves computing the forward dynamics/kinematics to determine where a robot is located. These queries can be computationally expensive and limit the speed at which robotics can plan in real time.

Other approaches to real-time collision detection use hierarchical data structures to speed up the detection time. These typically use bounding volume hierarchies to represent the environmental objects in ways that make the collision computations more efficient but error on the side of caution and eventually converge to the traditional method above (when the accuracy is needed). While this does speed things up, the calculations involved are still expensive.

More recent work has used machine learning techniques to formulate the question posed to various "prediction nets" in terms of collision detection and avoidance. Many try to inform the motion planning algorithm of areas where collision is less probable. Examples include [1], [2], [3].

Hybrid's of the two exist in which a neural net is used to speed up collision checking. "Fastron" [4], developed by Nikhil Das, Naman Gupta and Michael Yip, uses kernel perceptrons (a non-linear variant of the perceptron) to represent the systems configuration space and an efficient search for collision status changes in a time-variant environment. While shown to be much better than traditional methods, significant problem formulation is required for this approach.

To avoid the need for costly problem reformulation, we propose directly training a predictive net on sampled point-cloud, joint configuration and signed-distance data from a highly-accurate simulated robot. The job of the net will be to take in a given set of point-cloud and joint-configuration data and output the smallest signed-distance to collision.

II. Data Collection

In order to collect the data required to train our Nerual Network, we developed a automated data collection system using ROS. We collected simulated collision data using the KUKA LBR4 robot arm inside the Gazebo simulator with the DART physics engine. In the environment we placed a table in the robots workspace where we placed various objects. This can be seen in Figure 1. These everyday objects were sourced from the Bigbird [5] dataset. We generated over 100 random environments, each with varying number of objects placed uniformly randomly in the robot workspace. We placed a simulated depth camera above the robot workspace to generate RGB and depth images similar to what would be collected with a Kinect camera in the real-world.

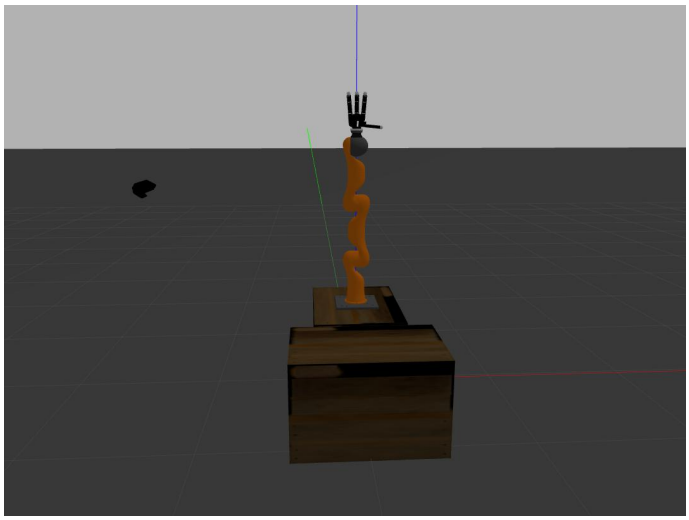

Fig. 1. Robot Environment

For each of the generated environments a number of robot configurations were chosen uniformly at random. We then moved the robot into these configurations at which point the data collection could begin. Once the robot arm is stationary, the RGBD image data was captured from the depth camera above and saved to a file. At this location we also capture the signed distance to collision using ROS Moveit(https://moveit.ros.org/). This package uses the geometry of the robot and the 3D mesh of the environment to determine if the robot is in collision using a kinematic-based collision detector. The signed distance to collision is later used as our ground truth data for training the NN.

While setting up the automated system above, we ran into many problems that made collecting data considerably time consuming. Some of these problems included interfacing the various software components and automating the environment spawning of both the robot and the interactable objects.

III. Model Architecture

We define a deep neural network which takes as input a joint configuration and point cloud image and predicts as output the signed distance to collision. We first pass the image through two convolution layers and one max pooling layer. We process the joint configuration through one fully connected layer. Next we concatenate the image features with the joint configuration features. This data is then passed through a convolution layer, then a max pooling layer, followed by two fully-connected layers before a final logistic regression output layer.

Cross Entropy was used as the loss function and the Adam optimizer was used for the optimization during training of our Network.

IV. Experimental Evaluation

Unfortunately, due to the issues we faced with data collection, we had little time for rigorous experimental evaluation.

Our model was trained with a small subset of all possible robot and environment configurations. This meant that the model did not generalize well to unseen configurations. Our belief is that with much more data our model would be better at predicting in these new environment configurations.

The model we trained was specific to the KUKA LBR4 robot arm. If our approach was to be used on a robot with different geometry, a new model would need to be trained with data specific to that environment. In addition, one specific camera location was used in training our model. Any changes in the location of the depth camera would lead to inaccurate results.

V. Future Work

In the future, we wish to improve our data collection procedures to collect larger amounts of data. This will allow us to better train our model in the hopes of better generalization.

In addition to collecting more data, we wish to try preprocessing the point cloud images. Using point cloud segmentation to subtract the unwanted backgrounds in our images should give us more usable data. Other forms of image processing could also help our model distinguish between objects and find the signed distance to those objects.

Another possible improvement in our Neural Network could be the use of an autoencoder. By adding this to the beginning of our network, we could reduce the number of dimensions and speed up the time needed for training. We would also like to explore other possible improvements in our network, such as expanding the depth, width, and type of layers. Time permitting, further tuning of our hyperparameters should also improve our results.

VI. Conclusion

We presented an approach for collision detection formulated as regression in a learned deep neural network. Our approach allows for more data-efficient collision detection compared to the current costly kinematic based methods. While our predictive results leave room for much improvement, the automated data collection method we established will be useful in future work for various learning methods in robotics.

References

[1] J.H. Graham, et. al, *A neural network approach for safety and collision avoidance in robotic systems*. University of Lousiville, 1996.

[2] P. Long, et. al, *Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation*. IEEE Robotics and Automation Letters, 2016

[3] A. Sharkawy, et. al, *Human-Robot Collision Detection Based on Neural Networks*. 2018

[4] N. Das, et. al, *Fastron: An Online Learning-Based Model and Active Learning Strategy for Proxy Collision Detection*, University of California, 2017.

[5] Ashutosh Singh, Jin Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. *Bigbird: A Large-Scale 3D Database of Object Instances*. In IEEE Intl. Conf. on Robotics and Automation (ICRA), pages 509516, 2014.

[6] Q. Lu et. al., *Planning Multi-Fingered Grasps as Probabilistic Inference in a Learning Deep Network*. University of Utah, 2017.