Project Proposal

**Background:** While developing new robots, it is important that we can make robots which can still complete tasks if interfered with. This would allow for safe human-robot-interaction, as robots would successfully maintain a proper distance from any potential collisions. In collaboration with the Artificial Intelligence Lab at Stanford, Fabrizio Flacco developed a model for collision avoidance on a 7-dof KUKA Light-Weight-Robot IV using a Microsoft Kinect sensor. The control method they use for their endpoint remains consistent: assuming the location of the endpoint, they use inverse kinematics to determine where the rest of the joints need to be to position the arm of the robot in the correct fashion. The robot is designed to move along this endpoint, and when it is driven off course because its sensor detects incoming motion, the control law determines what position the robot should return to following its avoidance of a collision. However, they have different kinematic equations from what is used for the endpoint for the control points in the arm of the robot which drive the motors.

For my project, I will be working under the supervision of Alessandro Roncone to develop a new collision-avoidance system for the Baxter robot. Similar to the Flacco work, Baxter will be able to avoid collisions based on perception using a Microsoft Kinect sensor. This system will also allow Baxter to react to potential contact forces as a backup to its perception recognition.

The novelty of my project lies in two areas. For one, there is not a robot system that
currently exists which integrates both perception and contact into its collision-avoidance behavior. Second, I will look to develop a new method for the control laws computing the endpoint and control point behavior. As aforementioned, the Flacco lab used different kinematic equations to determine the position of the endpoints and control points. My project will look to develop a single equation describing the behavior of both. This control law will be far more flexible, allowing for easy integration into other robot systems looking to develop the same collision-avoidance behavior.

**Previous Work:** I have now been working in the ScazLab on this project for two semesters. Two semesters ago, I worked on the image processing components of the project, which involves using a Kinect Camera to identify potential collisions.

To begin calibrating the Kinect sensor to the Baxter robot, I first needed to bridge Windows and Ubuntu software so that Kinect images could be communicated to Ubuntu client-side code, necessary because the Kinect v2 runs on Windows software, while the ROS software is written on Ubuntu. For this, I relied on a Github repository maintained by Michael Tsang, a member of the USC Robotics Research Lab. I modified the code from this repository to send depth messages, which I could then use for further processing on the Ubuntu client.

On the client side, I wrote code to process the depth messages sent by the server. The depth message data was sent as a single array of integers, where each value of the array is a depth value for a specific pixel. After converting the Kinect depth message data to an image, I publish the image as a sensor_msgs/Image message, the proper format for depth image messages in ROS. With this image now on the client, I use the pixel values to cluster the data using a k-means clustering algorithm, which isolates the Baxter robot from its background and also potential collisions.
This past semester, I worked on implementing a controller for the Baxter robot, one that could precisely move Baxter from one position to another. I built two types of controls: the first was a position controller, one that moves Baxter’s arm by indicating what positions the joints should go to. I then built a velocity control, which, contrary to a position controller, moves Baxter’s arms by indicating the velocities at which Baxter’s joints should move.

To implement these controllers, my code uses the TRAC_IK C++ library, a library that computes the positions of the joints based on the end effector position using inverse kinematics. In short, the controller continuously listens to a ROS topic for messages indicating Baxter’s desired end effector pose. If Baxter is not in the desired pose, this control loop incrementally moves Baxter toward the desired position; the algorithm computes inverse kinematics on the slightly changed end effector position, and then it sends a ROS command to move the Baxter robot’s joints accordingly. This process goes on until the end effector reaches the desired position, at which point the condition within the infinite loop is no longer true.

Following the design of the position controller, I then moved on to developing a velocity controller for the Baxter robot. The reason for doing so was because a velocity controller provides the user far more control over the Baxter robot’s motion. This is because Baxter does fewer internal calculations with velocity commands as opposed to position commands, meaning the velocity messages get processed faster and result in smoother trajectories. There are multiple ways to design a velocity controller, though I settled with computing velocities based on positional inverse kinematics. Thus, given the end effector's position, I computed the desired joint positions. Then, based on the joints and end effectors’ current position I computed the necessary velocity to travel the given distance in the time frame of my program.

**Plan for Semester:** The next part of next semester’s work will be implementing a custom inverse kinematics algorithm with collision avoidance, the most original part of my research.
This custom inverse kinematics controller will allow Baxter to move its arm along a trajectory, adjusting mid-trajectory for potential collisions. The primary way I plan to implement this is using the IpOpt convex optimization library. The IpOpt optimizer will help with my work because it allows me to optimize the solution to an equation while adhering to constraints. Thus, I can use the solver to find an optimal solution for joint velocities while at the same time adhering to constraints of maximum/minimum joint velocities in addition to the joint positions not being able to go to certain locations due to an oncoming collision.

Finally, I will have to do some further work on the image processing. Currently, I can process the depth messages to determine clusters in the image through a k-means clustering algorithm. From this data, I need to determine which object is the Baxter robot. I currently plan on using a basic machine learning algorithm, one which is trained on several models in which I know exactly which cluster the Baxter robot is. Having figured out which cluster is the Baxter robot, detecting collisions will be fairly simple – if an object comes close to Baxter, then it must be an oncoming collision. In order to complete my project, I will have to bridge my image processing work with the custom inverse kinematics controller, allowing Baxter to successfully avoid collisions detected by the Kinect camera sensor. With the complete collision avoidance system in place, I would like to implement a testing framework to ensure the system’s robustness. Building tests for ROS systems is complicated, as it will involve simulating the publishing and subscribing of nodes to various topics without the nodes actually running.

**Timeline:**

*Weeks 1-3:* Porting code from iCUB IpOpt linear solver to Baxter robot

*Week 4-5:* Implement collision avoidance factor for controller using IpOpt solver

*Weeks 5-6:* Finalize work on collision avoidance system and submit paper to IROS conference
**Weeks 8-10:** Complete image processing component to create a real time collision avoidance system

**Weeks 11-13:** Implement a robust testing framework for the software

**Deliverables:** Ultimately, my goal for this project is to have a full collision avoidance system integrated on the Baxter robot. This would take the form of code in a ROS package that can be used to avoid collisions on any robot system. This package will be distributable code, not just to be used in the Scaz Lab. Specifically for this semester, my advisor and I have set a goal to submit a paper to the International Conference on Intelligent Robots and Systems (IROS) conference, which will be held in Vancouver Canada in September, 2017. This will involve completing all of the aforementioned components of the project without linking the image processing component.

**References**