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 cope with any potential collisions. In collaboration with the Artificial Intelligence Lab at Stanford, Alessandro de Luca and his lab developed a model for collision avoidance on a 7-dof KUKA Light-Weight-Robot IV using a Microsoft Kinect sensor¹. 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 pose. The robot is designed to move along this endpoint, and when it is driven off course because its sensor detects a collision, the control law determines what position the robot should return to following its avoidance of that 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.

There is also currently software for the iCub robot which deals with reacting to direct contact. Sensors on the skin of the iCub can detect where it is being touched and with how much force. The robot then moves accordingly so as to avoid a potentially damaging interaction.

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 De Luca group, 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
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 De Luca 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 worked in the ScazLab last semester on this same project. I spent my time installing software, learning the ROS middleware for programming with robots, researching current collision avoidance systems, and implementing the first component of my project: the image processing component using a Kinect sensor.

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 the 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. In this repository, there already existed a bridge between Windows and Ubuntu that passed RGB images and speech messages, so I had to modify the code to also send depth messages. The depth data was already there, since the Kinect naturally captures it, so I had to ensure on the server side that it was getting sent as a message as opposed to sitting as superfluous data.

On the client side, I wrote code to handle depth messages sent by the server. The goal was to publish these depth messages as depth images to a ROS topic for use in other ROS
packages. The depth message was passed with a header and a payload, where the header contained information about the image, and the payload contained the data of the depth image as a single array. Each value of the array is a depth value, and the array can be read such that the first \( i \) entries are in the first row of the image, where \( i \) is width of the image in pixels. The next \( i \) entries are the second row of the image and so on. After converting the message to an image, I publish the image as a sensor_msgs/Image message, the proper format for image messages in ROS.

**Plan for Semester:** This semester I will be focusing on the control side of the project, meaning I will be writing the algorithms to handle Baxter’s movement when it needs to avoid a collision. Once this component is complete, I can bridge it together with my image processing work from last semester so that the actual collisions avoided by Baxter are those detected by the Kinect Sensor.

The first component of my semester’s work will involve moving away from the aforementioned Kinect bridge repository maintained by Michael Tsang and the USC Robotics Lab. Over the summer, the ScazLab found a Kinect bridge repository maintained by a CMU robotics lab that is much simpler and more easy to use. I will have to port over the image processing code that I wrote last semester so that it uses this bridge instead. Since the depth message code already exists for in this new bridge, I will have to write a small package that subscribes to the depth topic, then process it in the same way as I had been previously.

I will then continue by learning how to move Baxter with ROS code. My advisor, Alessandro, spent this past summer experimenting with Baxter movement and ROS, so this will involve me capitalizing on his code and customizing it to my needs. I will then begin to build a simple position control to guide Baxter’s motion. Essentially, I will be writing a control that will take in two parameters, Baxter’s current position and the position Baxter will be in. This position
control then works by telling each control point exactly what position to be in at a given moment, eventually moving Baxter into the right overall position.

Following this, I will proceed with the implementation of a velocity control. Instead of coding an algorithm to determine the position of each control point at a given point, this algorithm will determine the velocity of each control point at a given time, still based on where Baxter is and its desired position. Since this control works with velocity as opposed to position, it has far greater control over Baxter’s motion. That is, Baxter’s motion will be smoother, more precise, and more natural because the velocity of the control points is closer to the actual control loop of the robot’s motors.

If I achieve all of the above this semester, I will start integrating the iCub software onto the Baxter robot. A similar framework is under development on the iCub robot that handles collision avoidance, yet this is a major task because the code has to take into account the different hardware and software architecture. The velocity controller will be similar, but I need to encode a different velocity trajectory so as to minimize jerk. Finally, I will implement more functionality so that the controller is doing custom inverse kinematics – that is, the controller will be able to directly compute the desired velocities of the joints, all at the same time as avoiding potential collisions.

**Timeline:**

*Weeks 1-3:* Porting code over to new Kinect bridge written by lab at Carnegie Mellon

*Week 4:* Review code that moves Baxter robot, learn how to use ROS to control robot motions

*Weeks 5-7:* Implement a position control algorithm – control how Baxter moves based on a desired position
*Weeks 8-11:* Implement a velocity control algorithm – control Baxter’s motion to a given position based on velocity information

*Weeks 12-13:* Experiment further with algorithms – write up report on semester’s work

**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 two separate ROS packages: one for avoiding collisions through the Kinect camera sensor, and another for avoiding collisions via a contact sensor. These packages will be open source and freely available to anyone, not just to be used in the ScazLab. Specifically for this semester, I would like to be at the point where I have a velocity control implemented on the Baxter Robot. Though the project will not yet be finished, at semester’s end, I will be able to submit the code that I’ve done, in addition to a write up about my progress this semester. Finally, 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. The deadline to submit a paper for this conference is in April 2016, so I am hoping to have the bulk of my research done in early-mid March.

**References**