ECE 590.04 Spring 2015
Planning/Control Team Report

Ben Burchfiel, Brenton Keller, Miles Oldenburg, Xiao Tan
Duke University

May 2, 2015

Contents

1 Summary

The planning and control team is responsible for planning movement of the Baxter robot between given points and sending the control commands to actually move the robot. In addition the team was also responsible for preparing ideal grasps for every known object in the Amazon Picking Challenge.

Team members also went outside the initial requirements to help with calibration, testing, and integration.

2 Initial Plan and Milestones

The following goals were set during class discussion:

- "Ideal world" single object planning with known good grasps
- Test grasps (relative poses and finger closing motions)
- Find an ideal vantage point for each bin with our actual cameras
- High quality motion planning (especially for small movements)
- "Wiggle the vantage point" action
- Recovery actions for failed grasps (safe home configurations, safe retraction motions)
- Determine a collision avoidance margin
- Task planning: when to go for grasp, adjust vantage point, etc
- *Push to the side planning

Progress was made on the majority of initial goals and is described in section 3.2.

3 Implementation

3.1 Planning Code

The team was given a python interface from the systems integration team which contained several well defined functions that needed to be implemented. Our job was to meet the requirements of the interface adding whatever code and classes that were needed.

The team worked in close coordination with the systems integration team to implement several tasks.

All of the limb movement was planned using a a Single-Query Bidirectional Lazy planner (sbl) from the Klampt library.

Move to Vantage Point

Given a desired bin from the integration team, we planned the robot to move from the current configuration to a configuration where the bin contents could be viewed by the 2D camera and the 3D sensor.

Grasp Object in Bin

Given the desired object and bin, we evaluate all known grasps for that object and sort them by the distance the robot would have to move to grasp the object. In sorted order, a plan is computed for the hand to grasp the object until a feasible plan is found. After the plan is found we move the arm to the pre-grasp position 7 cm away from the object. From the pre-grasp position we find an inverse kinematic solution to place the hand around the object. We then set the configuration of the robot to the inverse kinematic solution and close the gripper to complete the grasp.

Move Object to Order Bin

Once the object has been grasped, the object needs to be removed from the bin and dropped into the order bin at the appropriate height. At this point, we extend our planning model of the robot arm to include the object so that the object is also taken into account when performing collision avoidance. First, we find an inverse kinematic solution to lift the object up 5 cm to avoid hitting the lip of the shelf when removing the object. From the lifted position we plan a path to move the object to a designated "safe area" away from the shelf. This safe area was determined empirically by manually positioning Baxter with one of the largest objects in its gripper. The safe area is defined as being 50 cm away from the middle of the shelf in the y-direction, 50 cm away from the back of the shelf in the x-direction, and at the bin vantage point height in the z-direction. The orientation of the hand is rotated 45° about the z-axis. From this safe point it is theoretically easier to plan movements to place objects in the order bin because it is far away from the shelf and in a position where none of Baxter's joints are near their limits. Once a plan has been created to move to the safe area, we plan to place the object in the order bin. Once the object is below the penalty drop limit then the gripper is opened for the object to be released into the order bin.

Move to Initial Pose

This method creates a plan from the robot's current configuration to a default pose. Designed to be used in error recovery, it is useful in restarting a process over from a position that is known to be safe and where new plans can be computed quickly.

3.2 Progress on Initial Goals

"Ideal world" single object planning with known good grasps

"Ideal world" in this case means in a world where calibration and sensing are perfect and without error. In an ideal world the location of the shelf, order bin, and objects would all be known. The sensors would be calibrated to return accurate data and physical limb and gripper movement would also be well calibrated. Good grasps would also be available to ensure the object could be picked up without damaging the object or dropping it prematurely.

At the time of writing we were able to move a limb from the initial pose to the vantage point of a bin, search for possible grasps and pick the best one, grasp the object, pull the

object out of the shelf, and place it in the order bin. A description of experimentation with the box of crayons is available in section 4.1

Test grasps (relative poses and finger closing motions)

In this task we used the provided grasp editor program and went through all of the objects that will be used for the challenge and made grasps for them using the reflex gripper. We attempted to make at least one grasp for every object and in other cases created multiple. Ben modified the editor to allow for creating grasps by interpolating between two points. This modification allows us to create many grasps at one time. For example, creating grasps along one side of the box of crayons is a tedious process using the editor when we want to have the exact grasp just shifted along one axis. If we create the beginning and ending grasps at one end of a side then we can automatically create grasps in between.

Find an ideal vantage point for each bin with our actual cameras

The process of finding an ideal vantage point was actually handled by the systems integration team. It was decided that we would have a default vantage point which would be around 20 cm directly in front of the desired bin. In the interface provided by the systems integration team we accept the desired bin and the viewpoint location, then create a plan to move a limb to the provided parameters.

High quality motion planning (especially for small movements)

Initially, the plan was to pre-compute motion plans between a stable home configuration, a predetermined point by each bin on the shelf, and the order bin. These plans would stay far enough away from obstacles that they could be precomputed ahead of time safely. The smaller motions to actually grasp and object would be computed in real time using perception to increase accuracy. During testing, we discovered that motion planning was generally fast, significantly faster than expected. In light of this, we opted to use real-time motion planning for all movements which removes the need to treat gasping as a special high quality plan.

"Wiggle the vantage point" action

Similar to 'Push to the side planning', the motivation for this goal was to get a better vantage point if the object was occluded. We decided not to spend time creating a custom movement because the systems integration team already has multiple vantage points defined. In the case that the perception team needs a better vantage point, the systems integration team calls the 'Move to Vantage Point' method and provides a new vantage point. The perception team can then acquire new point clouds at the new location.

Recovery actions for failed grasps (safe home configurations, safe retraction motions)

If a grasp were to fail and we were able to detect it we can plan to move back to the bin vantage point and retry the grasp. If the grasp continues to fail, or we are unable to move to the vantage point, we can use an error recovery method which plans to move to one of the limbs to any arbitrary position and orientation. We could move to the bin's safe area and then plan to go to another bin, or return to the initial configuration. The actual decision on when to use the error handling and which method to use is handled by the systems

integration team.

Determine a collision avoidance margin

Determining an effective avoidance margin turned out to be more complex than initially expected. Due to the confined shelf space, many potential object orientations and positions require coming into close proximity with other objects (such as the shelf bottom or sides). Conversely, the relative fragility of the reflex grippers means that even small impacts with fixed objects could break our gripper. Balancing the need for safety with the need to reach into confined spaces unfortunately consists of balancing two intrinsically opposed goals. We have not yet determined the ideal trade off, but for now we use a collision detection margin of 5 cm when planning to move to a vantage point and when planning to move from a safe area to the order bin. We chose these two instances to change the collision margins because fine movement is not necessary in these situations, unlike grasping where collisions with the shelf and objects are more likely.

Task planning: when to go for grasp, adjust vantage point, etc

High level task planning is handled by the systems integration team. Given the complexity, desire for multi-threading, and need for error handling the systems integration team handles the overall task planning and calls the planning interface when a certain action is needed. This approach allowed us to concentrate on writing well defined methods without worrying about what happens if our method fails or how to make our code fit in with the rest of the multi-threaded application.

***Push to the side planning**

This task was set as a stretch goal. The motivation for adding it as a goal was in the case that one object was occluded by another. In this case it may be difficult to grasp the object so the idea was to move the object closest to the lip of the bin to the side in order to have a clearer window to grasp the back object. We decided not to spend time implementing this feature because the challenge rules say that,"Items will not occlude one another when looking from a vantage point directly in front of the bin" [1].

4 Experimental Results

4.1 Single Object Planning

To work on this goal we placed the box of crayons in Bin A and in our planning code we manually entered the object and location into our knowledge base. By entering the known real world location into the knowledge base we could work on testing planning to grasp the object as if the perception team reported an accurate position of the item after sensing.

We also concentrated on making sure there were several grasps available for the crayons so that the gripper could easily pick up the item at the orientation we had placed it in the shelf.

[1] http://amazonpickingchallenge.org/details.shtml

We learned several things from testing with the crayons. First, the actual limb location and the simulated limb location used in the planning code were not aligned. The actual limb position tended to be located below the simulated position. We discovered that this was caused by the 3D sensor mounting. We had mounted a 3D camera on the limb with a zip tie that was putting pressure on the sensor that enables easy manipulation of the Baxter arm. We then had to mount the sensor in a different location.

Secondly, we learned about the importance of good grasp configurations. Initially our thoughts were to keep the open configuration slightly larger than the object being grasped so the planner would have a bigger space to work with. Due to errors with the calibration or planning we ended up pushing the crayons back into the shelf instead of neatly enclosing the box. We went back through the grasps and set the open configuration wider so that even with some calibration error the gripper could still position itself around the object. The crayons are also one of of the heaviest objects in the challenge and in our close configuration our grasps did not have enough force to lift the object up after the gripper closed. Again we went back through the grasps and set the close configuration to be tighter so that the gripper would have enough force to pick up the box.

By running this experiment we were able to determine a collision margin that provided a good trade-off between planning time, plan feasibility, and avoiding collisions. As stated earlier, the shelf's collision margins were increased when moving to vantage points and when placing the object in the order bin. We were also able to determine a value for the connection threshold variable in the SBL planner that provided a good trade-off between finding a plan and finding a plan that did not have closely spaced way points.

After making these changes we were able to successfully plan the full retrieval of the crayons from the initial pose to the placing of the crayons in the order bin.

5 Reflection and Remaining Work

5.1 Reflection

The implementation of the planning code posed several challenges. Getting correct grasps to work has been, and continues to be, one of the most difficult tasks. The grasp editor software was useful for creating grasps, but when these grasps are applied to objects on the shelf, they often result in infeasible configurations because of collisions with the robot and the shelf, causing the planner to fail. For larger objects, like the crayons, this isn't much of an issue, but for the smaller objects this continues to present a problem. Unfortunately, there are quite a few small objects in the Amazon Picking Challenge.

Creating collision free plans was another challenge we faced in implementing the code. If the planner was able to find a plan it was collision free in simulation. However, when we ran this plan on the actual robot we found that sometimes the robot still collided with the shelf due to differences in simulation and the real world. This can be attributed to the fact the the collision margins default to zero, so if the real world differs from simulation at all there could be a collision. We tried testing different collision margins to work around this issue, but we found that increasing the collision margin caused the planner to fail more often. We found that with a 5 cm collision margin the planner was still able to create plans and those

plans prevented collisions most of the time. However, even with the 5 cm collision margin the robot still hit the shelf on a few occasions.

Because the planning code ran in it's own process it was highly dependent on the integration team's code. The planning code does not maintain state from action to action and starts fresh each time a method is called. This design decision initially lead to some problems understanding which variables would be available when a method was called and how the return values should be formatted. The integration team was helpful in resolving this issue.

The last major issue we encountered while implementing the planning code was creating smooth paths. Some of the plans that were returned by SBL had only a few way points, but covered large distances. This behavior sometimes resulted in collisions. To better control how the robot moved between far away points we tried to get the planner to use more points that were closely connected by making the connection threshold smaller. However, this caused the planner to fail more often. Recent updates to the hardware layer seemed to have changed the way that Baxter interpolates between configurations and has potentially made this issue less of a problem.

Not only was the task of writing code to plan to pick objects out of the shelf and into the order bin difficult, there were other organizational difficulties as well. For all of us, this semester was the first time we had worked with Klampt and for some of us this was our first time working with Python. Although we had completed a few assignments before this project there was still a learning curve that we needed to overcome. Another organizational difficulty we encountered was managing the source code. Everyone had access to all the code and could change it at any time, which sometimes lead to confusion as to who made changes and why they were made.

Finally, although there was a learning curve associated with Klampt, it was extremely useful in completing this assignment. The ability to choose planning algorithms, set parameters of these algorithms, detect collisions, and configure the robot was necessary to complete this assignment and these features were already implemented in Klampt. If we would have had to write our own planning algorithms from scratch this assignment would have been much more difficult.

5.2 Remaining Work

Scooping motion

We found that grasping flat objects, like books, was extremely difficult, if not impossible with both power grasps and pinch grasps. We believe that flipping these objects out of the shelf is a viable option to retrieve these items. We would start by pulling these objects to the edge of the shelf from behind and then pulling up and towards the robot. This ideally would cause the object to flip over the edge of the shelf were the opposite hand would be waiting with a tray, bucket, etc. to catch the object. We would then "dump" the tray into the order bin.

Additional Grasps

One important area of work is creating more grasps. Ideally, every object would have at least one grasp for each plausible orientation in the bin. For example, the box of crayons has

a cuboid shape so at minimum one grasp on each of its sides should be created to give the planner a variety of choices. For cuboids or other objects with identifiable axes the modified grasp editor should be used to interpolate many grasps in between two points. This would be helpful in cases where the planner might have better success choosing a grasp at the top of an object so the gripper does not hit the shelf.

Another potential method of generating grasps is to manually position Baxter and the gripper around an object inside the shelf. We could use the sensed position of Baxter along with a simulated object to compute a grasp transform that we know will work in the real world.

Unknown Grasps

In the event that a grasp cannot be found for a known object, it is still desirable to make an attempt to grasp the object as a last resort. For this case we worked on integrating the code provided by Dr. Hauser for grasping arbitrary point clouds. This code needs to be integrated so that it takes into account potential collisions with the shelf or other objects.