Grasp Planning with Baxter

EECS C106B Project 3

Osher Lerner Will Panitch Enya Xing

I. Abstract

We present a method for planning, analyzing, and executing generalized grasps on a variety of objects, using a variety of pose estimation and evaluation techniques. In doing so, we also provide a method for estimating the 3D position of a colored cube in a scene, as well as for solving optimization problems involving the friction cone by reformatting them from nonconvex problems involving nonlinear terms to a discretized linear combination of terms. We hope that this work will provide a comprehensive introduction to grasp estimation in robotics.

II. Methods

A. Cube Reconstruction

For all grasps except the cube, we used visual identification tags to identify the pose of the object to be grasped. However, for our simplest object, the cube, we tried to use the geometry of our camera system to locate and identify the object.

To do so, we made use of the ROSPy and OpenCV-Python libraries in order to implement a simple pipeline [?]:

Algorithm 1 Cube localization

```
$I \leftarrow$ Image from camera
$P \leftarrow$ Camera projection matrix
$K \leftarrow$ Camera intrinsic matrix
$D \leftarrow$ Known height of table
$H \leftarrow$ Convert $I$ from BGR to HSV
$T_{WC} \leftarrow$ Transform from world frame to camera frame
$t \leftarrow 0$
while The nonzero pixels of $I$ are not contiguous do
    $t \leftarrow t + 1$
    Pixels of I where $saturation < t \leftarrow 0$
end while
$E \leftarrow$ Detect the edges $I$ with a Canny detector
$C \leftarrow$ Find the corners with a Shi-Tomasi corner detector on $(E)$
if $Length(C) > 4$ then
    $C \leftarrow C$ [:4]
else if $Length(C) < 4$ then
    Move the arm and repeat from the beginning
end if
$p_c \leftarrow$ use $P^{-1}$ to project $C$ back into 3D space
$p_w \leftarrow$ use $T_{WC}^{-1}$ to transform $p_c$ to the world frame
Scale $p_w$ so that $z(p_w) = D$
return $p_w$
```

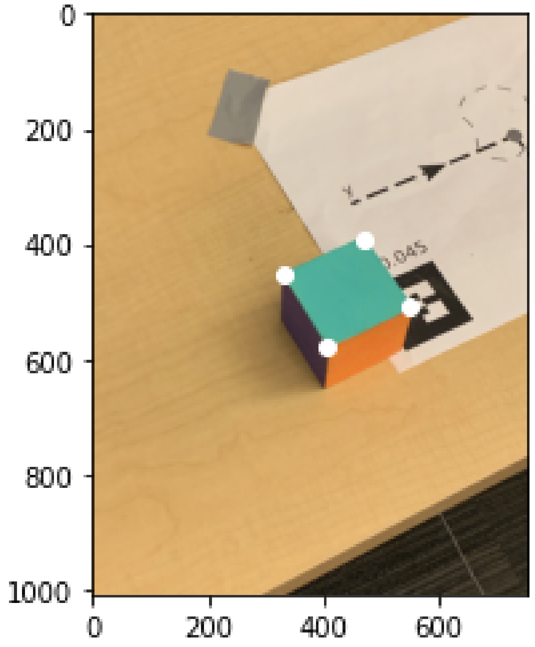


Fig. 1. ube localization

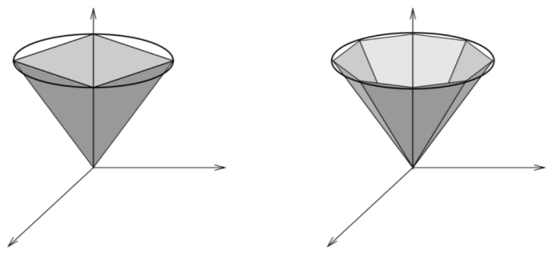
We used OpenCV's cvtColor, Canny, goodFeaturesToTrack, and drawContours functions, in addition to the ROS tf, sensor messages, and geometry messages resources to communicate information between different parts of our pipeline. We found that our cube detection algorithm successfully generated reasonably accurate estimates of the cube's position. However, the grasps that were computed under each further method were not feasible, due to errors in the way that sampled points were evaluated when they were part of the same surface 1.

B. Wrench Optimization

In order to evaluate various grasp metrics, we implement a generic optimization algorithm to find the forces at the contact points that can generate a given wrench.

To specify the contact forces to optimize over, we discretize the 3D friction cone using basis vectors along the edges of a regular pyramid approximating the cone defined by the friction coefficient μ (and a vector along the surface normal). We append an additional basis vector corresponding to torque resistance.

Fig. 2. Discretized friction cones for num-facets=4 and 8



We describe our contact forces as coefficients of these basis vectors for each contact point, and compute the total wrench by computing the linear combination of these basis forces for each point, then combining contacts by transforming the forces to wrenches in the center-of-mass frame.

To obey the friction constraints, we constrain our coefficients corresponding to 3D force basis vectors to be non-negative (to remain within the cone). Further, we restrict the torque coefficients to be less than γ times the total normal force (computed from the other coefficients).

Finally, we constrain the total wrench to equal the desired wrench. We then optimize our coefficients to meet these constraints while minimizing the total squared L2-norm of the contact forces. This ends up looking like

$$\begin{aligned} f^* &= \arg\min_f \|f\|^2 \\ &\quad \text{s.t. } 1 \le i \le n_{\text{contacts}} \; 0 \le j \le n_{\text{facets}}, \;\; \alpha_{i,j} \ge 0 \\ &\quad \text{s.t. } 1 \le i \le n_{\text{contacts}} \; \alpha_{i,n_{\text{facets}}+1} \le \gamma \sum_{j=0}^{n_{\text{facets}}} \alpha_{i,j} \\ &\quad \text{s.t. } GF\alpha = W \end{aligned}$$

We use CasADi for the optimization backend, and using our choice of variables this is a simple linear optimization problem. We detect that our constraints are infeasible if the optimizer does not converge and throws an exception (as is an intended in the CasADi design).

C. Gravity Resistance

To compute gravity resistance, we simply run our Wrench Optimization problem with the desired wrench being the force in the z-direction opposite to gravity. The converged contact forces f are then used to compute the cost J.

D. Ferrari-Canny

To compute the Ferrari Canny quality metric, we generate N=10 random unit wrenches (by projecting random samples of the 6D cube onto the sphere) and run the Wrench Optimization problem for each, using the outputted contact forces norm as a measure of local quality. We then compute the grasp quality by taking the minimum of the inverse square root of these local qualities.

E. Robust Force Closure

We compute the Robust Force Close metric by sampling N=20 perturbed contact points and recording the success rate of the Force Closure algorithm. We perturb the contact points by adding a random vector in $\mathcal{R}^3$ (sampled from a zero-mean Gaussian) to both ends. We then project these 3D coordinates onto the nearest point on the mesh (using trimesh's built in functions) and recompute the surface normals.

We use a standard deviation of 1.5cm. Larger values resulted in wildly different grasps, especially grasps in which the two contact points were perturbed onto the same point (or other infeasible relative positions). $\sigma = 1.5$cm was stable, and a sensible scale for actual robot errors.

F. Grasp Planning Algorithm

To generate grasps, we first sample N=1000 vertices. We then sample 500 random pairs from these as candidate grasps, filtering for vertices 3cm above the table and pairs whose distance is within the bounds of the gripper. We score each of these grasps using the selected metric, normalize the scores, and select the top 5.

We then compute the feasibility of the gripper to reach these contact points. For each selected grasp, we rotate through 100 evenly spaced orientations around the axis that goes through the 2 contact points. At each candidate gripper orientation, we compute the position of the gripper tips during the grasp, the position of the tips when the gripper is fully open right before closing onto the contact points, and a point at the center of the top of the gripper "U" shape. For each of these points, we check for intersections between the mesh and the line segment stretching towards the approach of the gripper. If none are found, this assures that the gripper will not unwantedly hit the object on its approach, and that it will not hit prematurely when closing the gripper. If there are no intersections, we select the grasp orientation as one to execute. If there are intersections, we drop the candidate grasp and resample and rescore 500 more grasps until we find 5 high-scoring non-intersecting grasps.

To execute a pick-and-place, we use MoveIt to navigate through the following series of waypoints: default overhead pose, 20cm behind desired grasp, desired grasp location, close grippers, 20cm up, 15cm over in the y direction, 20cm down, release grippers, 32cm up.

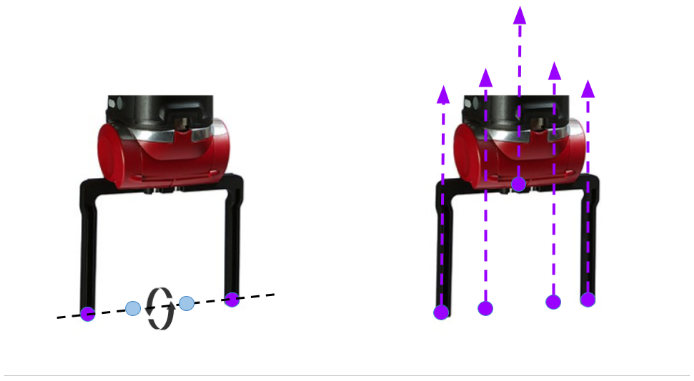
III. Experimental Results

A. Cube

IV. Discussion

The pawn was the easiest object to grasp for us as it did not have any holes or hollow parts on the inside, meaning that the gripper most likely was able to find successful grasping points on either side of the pawn. The nozzle however, had a few hollow portions and overall was more irregularly shaped. Our gripper had a hard time closing enough on the nozzle, and quite a few of the grasp points were on the underside of the nozzle, making them infeasible. We were able to obtain good

Fig. 3. Left: Example grasp orientation. Contact points in blue, gripper tips when open in purple. Right: Grasp approach lines tested for intersection with object mesh.



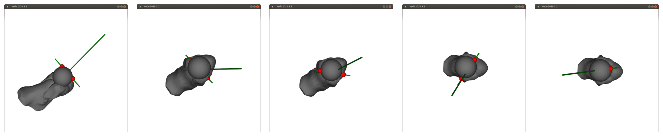


Fig. 4. Selected pawn grasps using the Robust Force Closure metric

cube recognition, but were unable to successfully integrate it into grasping.

An edge case that was not anticipated was the grasps on the bottom of the nozzle. We did not run into any issues with the bottom of the pawn being selected as a grasp point. Another difficulty we faced was with the nozzle with grasp points being selected inside the nozzle, but in actuality were hard for the gripper to actually be able to access the point. These grasps tended to fail a lot and we were unable to filter them out successfully, but would be a great avenue for future work.

The grasp metrics predicted scores and actual success did not seem to exactly match. As humans, we could often gauge the likelihood of success of a planned grasp before executing it. When the metric was confident in a grasp we knew would

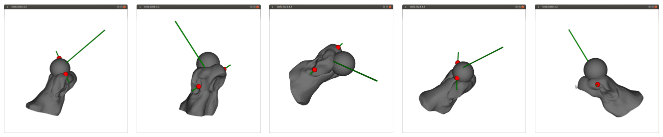


Fig. 5. Selected pawn grasps using the Gravity Resistance metric

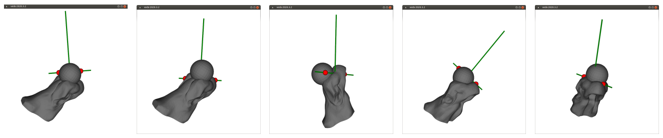


Fig. 6. Selected pawn grasps using the Ferrari Canny metric

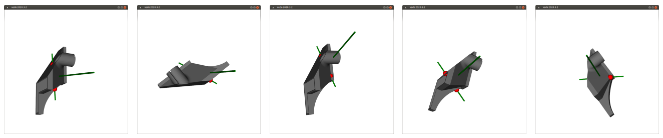


Fig. 7. Selected nozzle grasps using the Robust Force Closure metric

Object	Grasp Metric	Trial #	Score	Success rate
Pawn	Robust Force Closure	1	1.0	1/2
		2	0.7	1/1
		3	0.36	1/1
		4	0.36	1/3
		5	0.24	0/2
Pawn	Gravity Resistance	1	0.0	1/2
		2	0.0	1/2
		3	0.0	1/2
		4	0.0	1/2
		5	0.0	1/2
Pawn	Ferrari Canny	1	0.0	1/2
		2	0.0	1/2
		3	0.0	1/2
		4	0.0	1/2
		5	0.0	1/2
Nozzle	Robust Force Closure	1	1.0	1/2
		2	0.7	1/2
		3	0.36	1/2
		4	0.36	1/2
		5	0.24	1/2

Fig. 8. Table of results for 5 sample grasps from our recorded trials for different objects and metrics. Since at times we only kept results from successful grasps, or those achievable by MoveIt, this data may be skewed.

fail, we could usually explain the reason. Gravity resistance only cared about the efficiency of lifting forces, and would often choose feeble grasps where the object easily slipped out. Ferrari Canny improved on this qualitatively, since it measured against many different wrenches. Yet this metric was very susceptible to grasps based on precise surface normals. We know the noise in grasping would mean the contact would not exactly match the desired one. Robust force closure combated this best, and thus had the highest success rate. It seems resistance to noise was the most important quality of a grasp. Still, the robust force closure did not account for the full shape of the gripper, and assumed two point contacts as opposed to long surface areas, which led to further failure modes.

Our custom grasp planning algorithm is not theoretically complete, as we only check for intersections along 5 important line segments along the inner and outer bounds of the gripper's trajectory. We found its collision detection satisfactory for our grasps, and computationally efficient to the point that the run-time was negligible compared to other parts of our pipeline. However, to account for all possible protrusions we may miss, we should devise a mathematical computation to detect intersections of the object mesh and the full 3D volume swept out by the gripper throughout the approach and grasp. Further advancements could plan a more complex approach trajectory to reach possible grasps using rotations around the object protrusions.

For cube reconstruction, we found that simplifying the problem tended to work very well. For example, instead of estimating the 3D position of the cube, we were able to color-segment out a single face and use its known depth to localize those points. We also found success using blurring and edge detection as part of our pipeline, which allowed us to more definitively estimate the location of the faces. However, we found many challenges involved with using the transforms to reproject the points, and found that our grasps tended to be consistently slightly off in the same direction. Additionally, our grasp metrics proved to be ineffective at differentiating good grasps when presented with the simple cube primitive, and we therefore found little success when using our system for actual grasping.

References

[1] Murray, R. and Sastry S., ”Nonholonomic Motion Planning: Steering Using Sinusoids,” IEEE Transactions on Automatic Control, 1999.
[2] C. Ferrari and J. F. Canny. “Planning optimal grasps.”

V. Appendix

A. Supplementary Materials

Link to GitHub repository with implementations: https://github.com/ucb-ee106-classrooms/project-2-pinky

Link to Google Drive with Videos: https://drive.google.com/file/d/1kYiw5-KUZVam9YsDqBV5nuegIYcm5QwB/view?usp=sharing

CasADI’s Exceptional Design: https://github.com/casadi/casadi/issues/2138

B. Learning Goals

We believe that the learning goals of this assignment are to compare different grasp metrics. We also had the chance to design a grasp planning algorithm, as well as utilize what we learned from our computer vision unit in an application to localize and grasp the 3D cube. Overall, this project was a fun combination of a lot of topics we have touched on this semester: from planning, grasping, then to computer vision.

We ran into a lot of hardware issues (Alan gone, Archytas still there but no longer functional, Ada not being able to access her webcam, and Asimov not being able to open and close his gripper) and thus suffered a lot in the videos we were able to provide for this write-up.

It was mind-blowingly awesome when the robot was able to do grasp successfully though!