Team Harp: Amazon Picking Challenge Stowage Task

16-662 Robot Autonomy, Carnegie Mellon University
Abhishek Bhatia, Alex Brinkman, Feroze Naina, Ihsane Debbache, and Rick Shanor
Project Video Link: https://youtu.be/SN3n3u0Wb2s

Introduction

The stowage task for the Amazon Picking Challenge requires autonomous decision making to control arm motions, end effector actuation, perception, and classification. The goal is to pick 12 items out of an unstructured bin and place them onto a 12 bin shelf and return the contents of every shelf bin. To accomplish this task, a UR5 6 DOF robotic arm and suction end effector system is used to manipulate items. A head mounted Kinect 2 for Windows RGB-D sensor is used to identify grasp points out of the unstructured bin and a base mounted Kinect used for item identification after grasping.

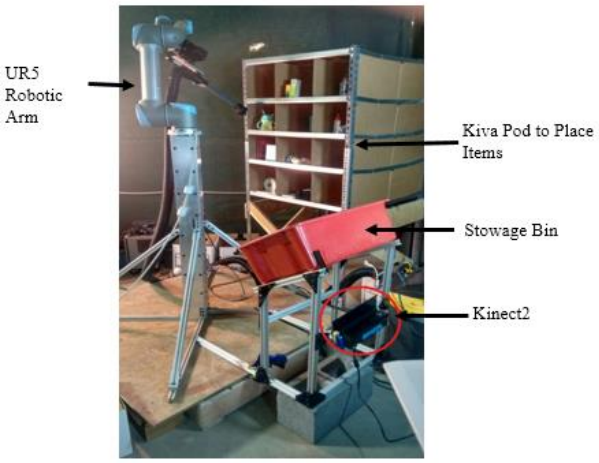

Figure 1: System Depiction

System Setup – Hardware

The Arm is a UR5 manipulator from Universal Robots and comes with its only controller and power supply. The Base and end effector are custom designed and fabricated. The grasping system consists of a suction cup and vacuum equipped with a pressure sensor that is controlled by a desktop. The Kiva pot shelf and red stowage bin are provided by Amazon and will be the same used in the competition. There is a computer-controlled actuator underneath the stowage bin to tilt the stowage bin as needed.

System Setup – Software

The UR5 comes equipped with a touch screen interface for typical manufacturing assembly. We connect to the controller over LAN socket to router and control the robot from a networked computer. ROS is used to manage the sensors, world model, controllers, and task-level executives. The TF package is in charge of managing the robot state, collision objects, and target item grasp points, and drop off set points. Caffe is used to train and get predictions from the Convolution Neural Network used for identification. PCL and OpenCV are used for the various pointcloud segmentation operations and image filtering operations as needed. Once the competition is completed, our code will be available on Github for general use.

Arm Planning

Arm planning is implemented through ROS and Moveit! using Open Motion Planning Library planners. A URDF of the robotic arm is provided by the manufacturer and modified to include our custom end effector. The default planning group starts at the base mounting point and extends through all 6 joints to the tool attachment mount. Our planning group is extended to include the end effector to enable target poses defined in the world coordinates for suction cup locations. Collision models for the mounting frame, order bin, and kiva pod shelf are loaded in using the MoveIt! planning scene interface. Plan details are specified using Moveit! like maximum planning time and which OMPL planner to use.

Each single query plan took approximately 5 seconds to find acceptable paths. Given the competition time constraint of 15 minutes, planning between set points would take too long and inhibit the effectiveness of the robot. A feature to precompute the path plans was developed and implemented in our system. The trajectory replay feature learns paths when learning mode is enabled. Whenever the trajectory-playback execution order is specified, the feature looks through the learned database for similar start configurations and goal poses and execute the stored trajectory. Otherwise, a Cartesian or single-query plan must be computed.

Our implementation of arm control implements a ROS blocking server that can accept planning details and execution orders. The motivation was to abstract planning as another step in our state controller so that infinite planning loops are not possible. Multiple plans and planning details can be tuned for the needs of each motion plan request. The available execution orders are trajectory-playback, Cartesian plan, strict Cartesian plan, fast single-query, and slow single-query. Trajectory-playback is a feature that replays precomputed trajectories. The Cartesian plan is a fast planner that uses FastIK to quickly solve the inverse kinematics of the arm at short intervals along the path. Straight-line paths are computed from start and goal poses. Additional waypoints can be specified and the Cartesian planner will attempt to find collision-free, straight-line plans between each waypoint. The Strict version of the Cartesian execution order forces the arm to be able to travel to the goal state whereas the normal Cartesian execution order will result and execute a partial path. Finally, the single query execution order uses the OMPL planner to perform a traditional motion planning query. The only difference between fast and slow planners is the allowable planning time. The OMPL planner could be specified for each request. RRT* was our default planner for its ability to find valid plans and improve them as time allows. RRT plans were fast but caused large swinging motions that are not desired for our picking application.

Grasp Point Selection

To determine grasp surfaces and valid grasp points, we first determine clusters corresponding to a set of points that are part of the same smooth surface. This is done using the region growing segmentation and clustering algorithm. We used the base code provided as part of the 'Region Growing Segmentation' pointcloud tutorial [1] to generate valid clusters. Once the clusters are identified, the clusters are scored and sorted to find the best grasp point. The score is computed for each clusters based on the maximum number of points, height of each cluster, area of the horizontal surface of each cluster (x-y axis), and direction of normal. The cluster with the maximum score is selected as the best cluster. The grasp point is then determined as the centroid of the best cluster, with the height of this point updated as the maximum height of the cluster.

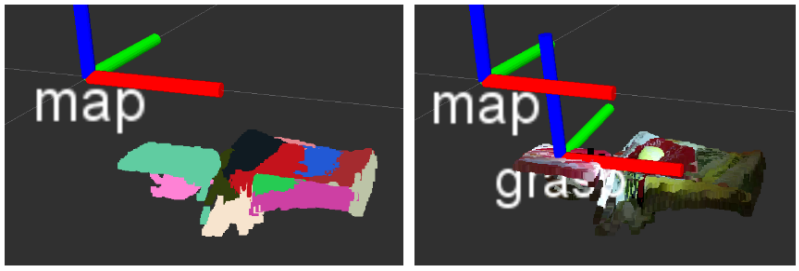

Figure 2: RGB Segmentation and Grasp Selection

Grasping Execution

Once grasp points are identified, the arm is moved to a setpoint to prepare for the pickup. The suction system is turned on and the variation in pressure is recorded. The grasp detection pressure is set as the measured average pressure minus 2.5 times the standard deviation. Using this dynamic setting ensures we are robust to small changes in pressure, hose configuration, height above sea level, and other factors that may influence unrestricted vacuum pressure. Using the 2.5 sigma boundary ensures we are as sensitive as possible to grasp detection without risking false positives. With the suction system on, the arm is commanded over the item and iterated down onto the item until pressure feedback is detected or the goal has been reached. If the grasp was successful, the arm moves the item over the identification camera to perform item identification. Otherwise, the arm is commanded over the bin where the stowage bin is reimaged and another attempt is made. First pass picking success using this approach is approximately 90%.

Item Identification - Database Generation

Any approach for item identification requires a large amount of training data to create a classifier. To generate this data for 40 items, a turntable was created to aid in database generation. The turntable consists of an actuated turntable, capable of 360 degrees of rotation and an actuator to vary the view angle of the Kinect v2 RGB-D sensor. The turntable is shown in Figure 3.

Figure 3: Turntable for Dataset Collection

Manually masking each input image would have required a lot of time so HSV thresholding and convex hull filters were applied to the images to

automatically mask the input images. Each HSV filter had to be tuned by hand for each item. Items that contained green elements on the edges required a different background color. Specular items against the green background would reflect green hues so a cardboard background was chosen for specular and green items.

Once the HSV threshold was tuned per item, a convex hull routine was applied to the image to create cleaner filters. The convex hull operation finds the smallest polygon that contains all of the pixels that pass the HSV color filter. This ensures that holes in the masks do not appear in the final mask which could cause the identification classifier to learn fictitious features. Figure 3 shows the output of each operation

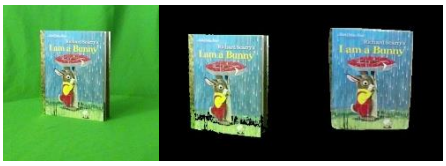

Figure 4: Original image (left), the HSV color filter image (center), and the final mask after Convex Hull (right)

In total, approximately 100 masked images were captured for each of the 40 items in the item dictionary. These images were rotated, distorted, mirrored, lightened, and darkened to create approximately 400,000 images for future classifier training.

Item Identification – Online Segmentation

Online segmentation is required to produce the mask similar to the images created for the database. Since the robotic system geometry is known, a cubic region of interest is created and pointcloud points that fall outside this region of interest are removed from the image. Care was taken to ensure the region of interest was large enough to capture full images of the item without including bad pixel values Figure 4 shows the raw and masked images.

Figure 5: Raw (left) and Masked (right)

Item Identification - SLIC Superpixels

Superpixel generation is a color based segmentation technique. A superpixel is defined as a group of pixels with similar characteristics. To solve our problem, we are using the SLIC (Simple Linear Iterative Clustering) superpixel algorithm to generate superpixels from an input image, primarily because of its low computational overhead [2]. Considering the scope of this project, we did not modify much within the superpixel generation pipeline and directly used the off the shelf package provided by scikit-image 'skimage.segmentation.slic' that takes in a RGB image (3D array) and outputs an integer mask indicating segment masks [4]. Once we generate the valid superpixels for each input (masked) image, we input these superpixels to the CNN described in sections below for training and testing/identification purposes. Figure 6 shows the output of the superpixel operation.

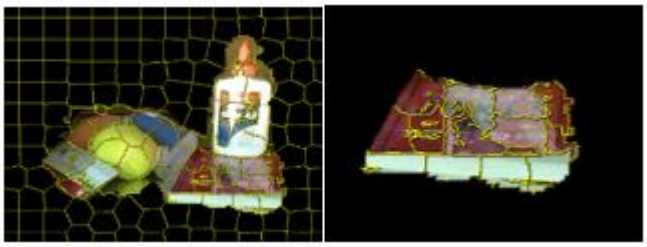

Figure 6: SLIC Superpixels

Item Identification - CNN for Superpixel Identification

To capture the large amount of appearance variation from item to item, a convolution neural network (CNN) was chosen to be the classifier. The popular deep net AlexNet was selected since its architecture and training made it well suited to the Amazon picking application. The CNN takes in a 255x255 RGB pixel array and passes the image through a series of five convolutional layers that filter and preprocess the image for distinctive features. Next, three fully-connected layers are trained to detect the identifying features for each item.

Our implementation of AlexNet requires an additional output layer to map the 1000 class predictions to the 40 class predictions needed for our application. This final layer consists of an SVM classifier with no initial training. The CNN takes in a 255x255 RGB image of each superpixel and outputs the probability that the superpixel belongs to each class. This is repeated for each superpixel in the segmented image to create an Nx40 prediction matrix for N superpixels.

The CNN was trained on images collected by our turntable apparatus and segmented into superpixels. The initial parameters were used for the net and our 400,000 training images were used to adjust the parameter and output SVM later using a learning rate of .001 and batch size of 250 training images. The model was loaded into an Nvidia Titan GPU for training and requires approximately 6 hours of computation to retrain the model.

Item Identification - Local Item Prediction

The goal of local item prediction is to identify the grasped item out of the 12 possible items in the request list to guide the robot executive decision-making. The probability of correct identification is low for a 12 choose 1 decision, but still useful to the operation of the system.

The output of the CNN produces an Nx40 matrix of predictions for N superpixels and the 40 items in the item dictionary. From this output, the items not contained in the 12 item request list can be ignored and the probabilities for the remaining 12 items renormalized. Finally, the average prediction is taken over all the superpixels for each item. This results in a 1x12 matrix of confidence for each of the 12 possible items. The local prediction is made as the item with the greatest confidence.

Item Identification - Global Item Prediction

The goal of global item prediction is to identify the entire set of items out of the 12 possible items in the request list. During autonomous operation, the robot will grasp and pickup each of the items and perform local prediction on each. After all images are captured, segmented, and passed through the CNN, global prediction compares the full set of prediction probabilities to make the best overall prediction.

The global item prediction begins with the Nx40 matrix of predictions for each of the 12 items. As with local prediction, impossible items are removed from the superpixel matrix. As before, each matrix is averaged across superpixels and renormalized to get the average item confidence.

Next, the global prediction finds the most confident item in all of the image confidence tables and assigns that image to the item label. The predictions for the selected item label is removed from the remaining images and each image confidence table is renormalized. Now the process repeats. With the set of remaining items, the highest confidence prediction is taken and assigned to the correct image. The predictions for the best item keep getting removed until all items have been assigned to images.

Item Identification – Results

The item identification process was tested by acquiring several images for each item, serving as ground truth, and choosing 12 images at random to do item identification. The segmentation and CNN operations were applied to the images and then global prediction was performed on the output prediction tables. The study was performed on 10,000 random permutations of images and compared to the ground truth. Overall, 75% accuracy was achieved by this method. A

confusion matrix was created comparing the global prediction results against the ground truth. Figure 6 shows several easily confused items from the confusion matrix.

Figure 7: Examples of Several Confusion Sets

Place Operation

The place operation occurs once the item has been grasped and imaged for identification. The arm is commanded to one of the 12 shelf bins and then translates into the center of the bin. The suction system is disengaged and the item deposited in the bin. The arm withdraws and moves to look down into the stowage bin, ready to pick up the next item.

Results

The current system performs first-pass picking with 90% accuracy. Local prediction is approximately 58% accurate and global item prediction is 75% accurate. Overall, we can pick and place approximately 84% of items and according to the competition rules score full points on approximately 63% of attempts. These results are promising but the system will require refinements before the competition in the end of June.

Future Work

Plans to improve the performance of the robot are focused on the identification and stowage operations. Identification will be improved by revisiting the segmentation to reduce data loss in the pointcloud that could be reducing the confidence of the predictions. The CNN will be trained on more data as we acquire more testing data to better reflect the images it will be actually see at runtime. Finally, multiple viewpoints may be acquired to ensure there is at least one good image of the item.

The other main area of improvement is the stowage operation. Currently, there is no pose estimation of the grasped item and the placement on the shelf goes to the center of the shelf. This works well for most of the items but larger items will need to be handled better. We want to implement a simple pose estimate by approximating a bounding box from the identification pointcloud. From that, a simple offset may be added to the shelf placement for improved placing.

References

[1] PointCloud Library http://pointclouds.org/documentation/tutorials/region_growing_segmentation.php
[2] Achanta, Radhakrishna, et al. Slic superpixels. No. EPFL-REPORT-149300. 2010.
[3] Wikipedia. 2016 https://en.wikipedia.org/wiki/Lab_color_space
[4] Scikit-Image. 2016 http://scikit-image.org/docs/dev/api/skimage.segmentation.html#skimage.segmentation.slic
[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
[6] The Quadratic-Chi Histogram Distance Family . Werman Ofirpele. http://www.ariel.ac.il/sites/ofirpele/publications/ECCV2010.pdf

Appendix 1: Confusion Matrix

Prediction vs ground Truth Confusion Matrix	bunny_book	lol_joke_book	scotch_bubble_mailer	glucose_tablets	dasani_water_bottle	rawlings_baseball	folgers_coffee	elmers_glue	hanes_socks	womens_knit_gloves	cherokee_shirt	peva_shower_liner	plush_bear	beefhide_bones	plush_puppies_squeakin_eggs	mini_glue_sticks	chenille_stems	40w_lightbulb	outlet_plugs	toothbrush	dr_browns_brush	command_hooks	turtle_sippy_cup	fiskars_scissors	duct_tape	extension_cord	dog_bowl	3lb_dumbbell	wire_mesh_pencil_holder	clorox_brush	paper_towels	expo_eraser	kleenex_tissues	pencils_12_count	crayola_crayons	jane_eyre_dvd	dove_soap	staples_index_cards
bunny_book	1262	0	24	41	30	0	21	3	8	0	0	16	6	1	0	35	3	45	32	95	8	4	38	141	12	3	68	8	6	0	0	3	3	0	46	0	76	17
lol_joke_book	0	2085	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	2	23	0	1	0	0	0	0	0	0	0	0	0	1	2
scotch_bubble_mailer	33	0	1669	3	0	0	8	0	5	0	0	0	127	4	2	1	0	0	22	1	1	97	18	21	0	0	17	11	32	0	0	0	0	0	5	10	1	0
glucose_tablets	1	0	10	1272	61	28	13	16	5	0	8	1	14	57	1	42	1	92	7	13	17	88	6	81	16	156	7	0	50	0	1	6	0	0	10	1	89	8
dasani_water_bottle	3	0	7	90	1066	2	77	295	18	0	43	8	1	1	0	164	4	71	0	62	18	0	2	6	6	6	11	4	3	82	0	0	54	2	0	4	16	10	45
rawlings_baseball	0	0	0	17	23	2024	1	8	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	8	0	0	1	0	9	0	0	0	0	0	0	1	1	7
folgers_coffee	64	0	4	0	0	1	1553	0	69	45	2	0	0	1	0	1	0	77	1	25	2	14	2	77	63	0	10	0	31	0	1	22	0	0	9	21	0	5
elmers_glue	0	0	17	10	59	17	23	1319	82	0	0	9	0	56	4	81	3	17	2	61	4	62	15	74	12	6	10	1	54	4	1	7	0	0	15	2	45	58
hanes_socks	1	0	18	4	25	3	94	31	1303	0	0	5	0	6	5	45	0	32	2	113	5	54	12	196	79	28	29	2	22	2	0	3	0	0	19	0	48	23
womens_knit_gloves	0	0	0	0	0	0	36	2	0	2025	0	0	0	0	0	0	0	5	0	1	0	0	0	2	1	0	0	0	5	0	0	3	0	0	0	0	1	1
cherokee_shirt	0	0	1	3	18	0	2	0	0	0	2077	1	0	0	0	11	17	0	0	21	4	0	0	5	1	1	4	0	0	0	0	0	0	0	2	0	0	1
peva_shower_liner	0	0	7	3	5	0	12	3	0	0	0	1669	0	2	0	19	2	45	20	8	52	2	8	80	23	2	7	0	2	1	0	0	0	0	17	1	49	111
plush_bear	29	0	19	2	30	0	10	2	7	0	1	1	1402	37	24	24	1	0	28	20	45	19	258	15	3	0	50	4	34	0	0	2	0	0	19	24	5	13
beefhide_bones	0	0	7	31	182	1	2	18	10	0	0	3	0	1595	8	15	1	21	7	21	14	11	12	43	3	31	13	1	2	0	0	2	0	0	11	1	28	20
plush_puppies_squeakin_eggs	1	0	4	0	0	0	1	0	2	0	0	0	29	0	2040	1	0	0	0	12	0	12	6	17	0	0	2	0	9	1	0	0	0	0	1	0	0	0
mini_glue_sticks	0	0	6	47	57	0	4	71	52	0	5	7	2	25	7	940	4	78	3	59	183	42	9	29	35	245	13	3	29	13	11	4	0	0	6	23	60	27
chenille_stems	0	0	19	0	1	0	3	0	2	0	6	0	0	0	1	1	1984	0	0	9	0	24	1	11	0	0	8	0	2	0	2	3	0	0	1	0	8	0
40w_lightbulb	2	0	22	156	95	2	10	51	80	0	1	82	0	11	2	63	0	996	30	7	58	24	5	52	54	27	25	4	16	1	2	2	0	0	12	0	150	74
outlet_plugs	1	0	56	4	4	1	4	9	4	0	0	8	7	0	2	10	0	9	1818	8	2	4	12	15	5	33	3	6	17	1	0	0	0	0	8	1	22	25
toothbrush	79	0	2	0	144	2	8	2	104	2	2	20	1	225	3	120	30	0	1	1182	60	0	1	17	11	2	2	11	25	21	4	38	0	0	1	0	4	6
dr_browns_brush	0	0	8	4	11	1	4	2	2	0	0	150	3	1	14	272	1	8	2	69	1386	7	18	37	1	13	16	6	15	3	0	0	1	0	3	0	35	9
command_hooks	3	0	15	106	97	2	21	23	33	0	1	0	0	0	0	13	0	92	2	3	45	1358	2	52	30	129	1	0	7	0	35	0	0	0	0	4	14	13
turtle_sippy_cup	46	0	4	30	14	0	0	0	0	0	0	1	281	0	0	21	1	0	0	7	9	0	1623	1	0	0	44	0	0	0	7	5	0	2	6	0	1	0
fiskars_scissors	148	0	4	178	55	11	8	7	114	0	16	1	5	8	8	27	6	25	10	72	2	168	7	893	34	156	5	3	0	1	33	1	0	0	27	0	20	10
duct_tape	85	31	2	0	0	0	113	0	186	0	1	0	0	2	0	0	0	0	1	0	0	18	0	1	1488	0	0	0	49	0	78	1	0	9	0	39	1	9
extension_cord	30	0	20	46	41	2	34	39	14	0	0	24	8	60	9	128	13	73	10	118	91	14	8	84	76	897	18	7	39	4	0	9	0	0	32	2	63	66
dog_bowl	141	0	5	0	0	0	3	0	21	0	0	0	188	0	0	0	0	0	0	4	0	19	6	10	0	0	1489	0	2	7	0	12	0	122	68	0	0	0
3lb_dumbbell	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	73	2008	1	0	0	1	0	0	0	0	0	0
wire_mesh_pencil_holder	32	0	0	3	0	0	10	77	18	2	3	0	0	0	0	7	0	139	55	0	13	0	1	3	24	124	0	0	1438	0	2	17	0	0	5	118	1	37
clorox_brush	0	0	1	1	0	0	1	0	0	0	0	27	0	0	0	4	0	0	1	8	0	0	0	0	0	7	0	3	0	2074	0	0	4	0	2	0	0	0
paper_towels	0	0	2	16	3	0	3	2	2	0	0	3	1	0	0	8	2	2	0	16	0	3	5	47	21	4	16	1	2	0	1918	1	0	0	4	4	18	7
expo_eraser	0	0	2	0	17	0	7	0	1	2	0	0	0	1	0	2	0	3	1	93	0	0	8	6	0	0	55	0	6	0	0	1858	0	0	2	0	2	2
kleenex_tissues	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	2156	0	0	0	0	0
pencils_12_count	1	0	11	0	1	0	0	0	10	2	2	0	1	0	0	0	9	0	0	0	0	0	5	1	0	0	78	0	0	0	0	2	0	1942	10	0	0	0
crayola_crayons	60	0	103	0	0	0	5	2	1	0	0	2	51	0	1	0	0	85	4	11	0	4	3	2	6	0	21	1	0	0	4	0	0	0	1687	1	0	0
jane_eyre_dvd	7	0	1	1	71	0	2	0	3	4	1	1	0	0	0	5	4	1	0	3	10	1	2	10	30	0	2	0	79	0	7	0	0	0	3	1748	1	20
dove_soap	26	0	6	91	64	0	5	73	40	0	0	13	0	8	6	9	0	49	6	5	45	40	7	12	57	95	4	1	62	0	3	7	0	0	14	0	1351	56
staples_index_cards	0	0	8	19	1	6	2	75	13	0	0	98	0	11	0	26	0	151	34	2	28	12	1	11	1	109	1	0	2	0	1	5	0	0	5	0	50	1464

Appendix 2: Amazon Item Dictionary 2016