CMU Amazon Picking Challenge

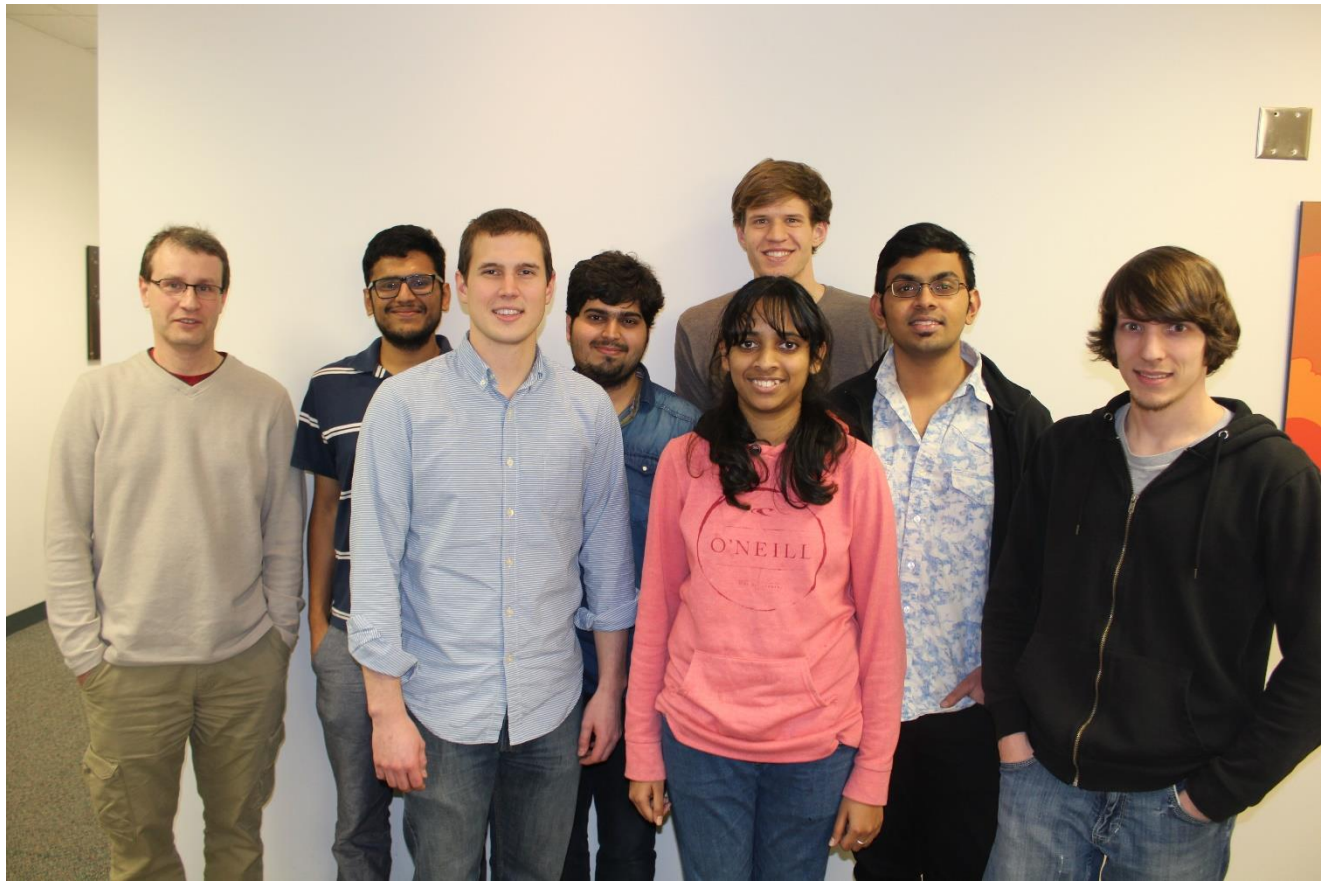
Team Harp (**H**uman **A**ssistive **R**obot **P**icker)

Graduate Developers

- Abhishek Bhatia
- Alex Brinkman
- Lekha Walajapet
- Feroza Naina
- Rick Shanor

Search Based Planning Lab

- Maxim Likhachev
- Venkatraman Narayanan
- Andrew Dornbush



System Overview

https://youtu.be/ko8Cr8kBc-Q

Why the Picking Challenge

Great Learning Experience for MRSD

Exposure to huge variety of robotic domains

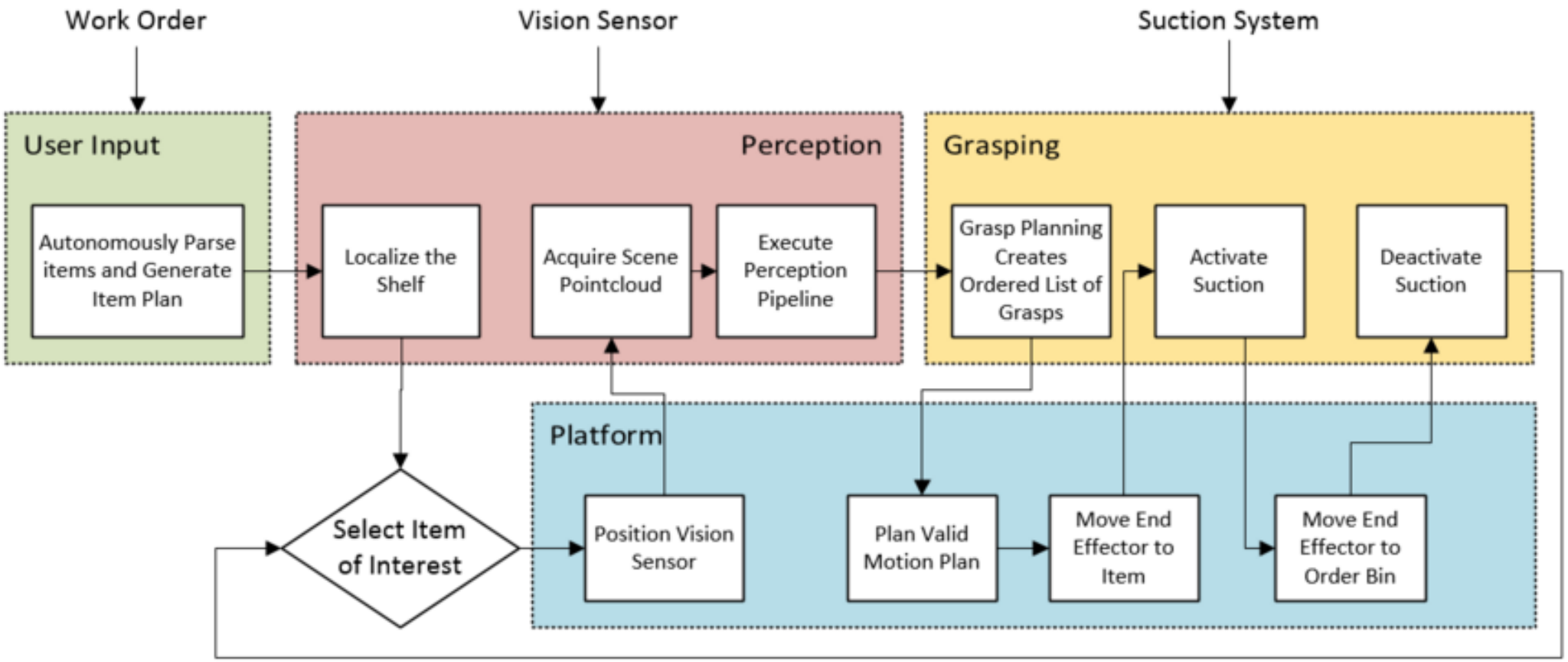
Fits the MRSD schedule

Real world application of SBPL planners (Perch, ARA*)

Visibility for CMU, SBPL, and team

It’s fun!.. Sometimes…

Functional Architecture

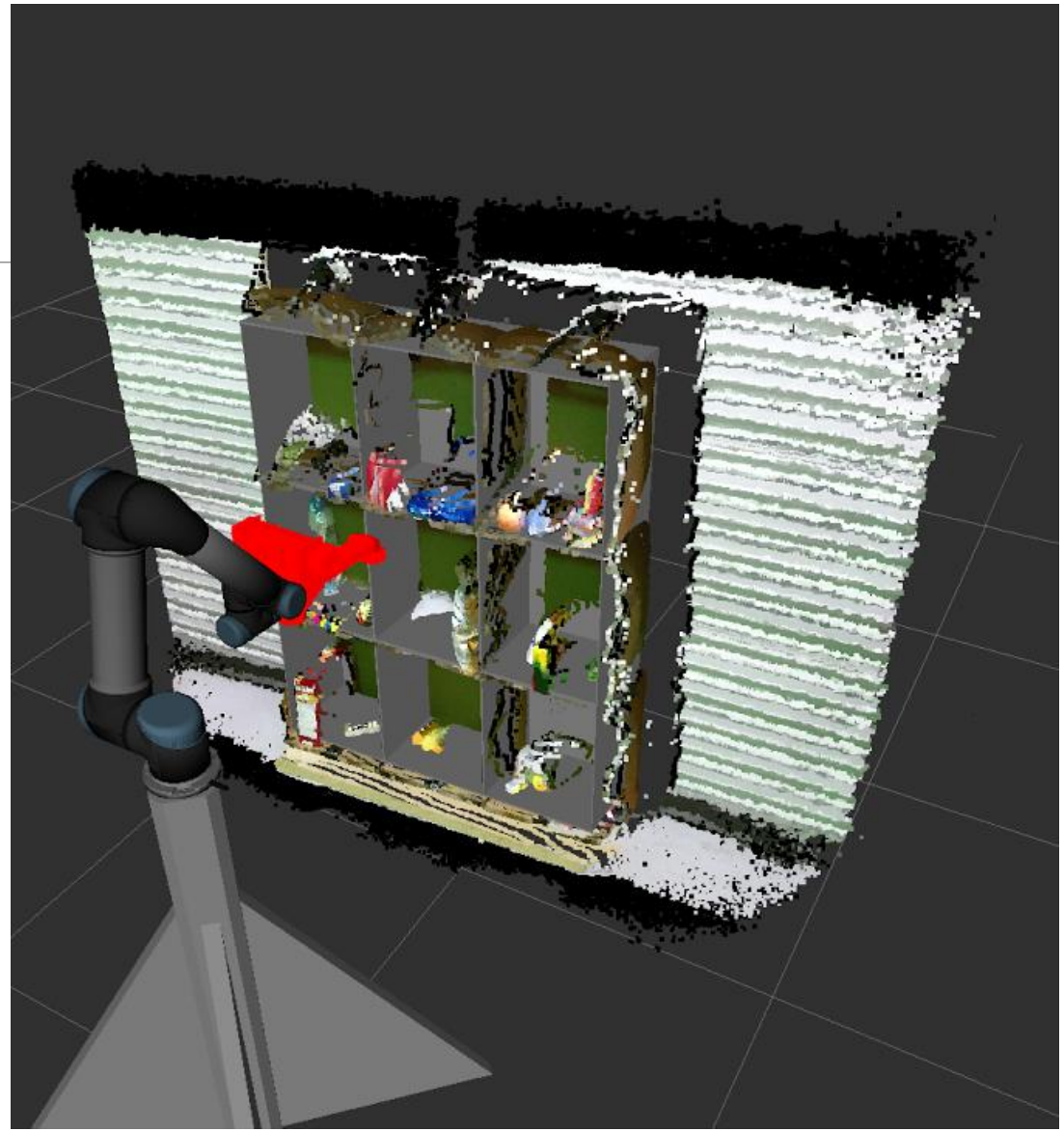


Suction Gripping

- High flow low pressure vacuum system
- Custom suction cup mounted to UR5 wrist
- Capable of acquiring 36 / 38 items in list
- Simplifying the grasping problem. Most teams converged on similar designs.
- Make suction tube co-axial with last DOF to help planning problem
- Find a quiet vacuum

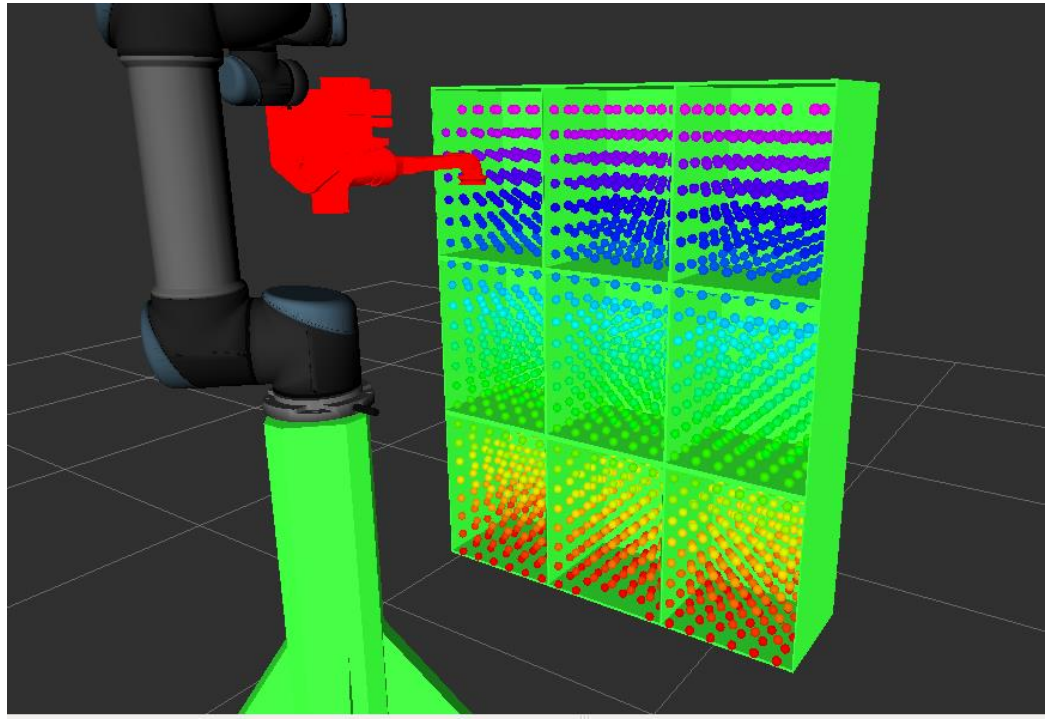
Simulation and State Control

- SMACH state controller manages robot actions
- Actions simulated and visualized in RVIZ to aid in development
- RVIZ visualizations help validate robot actions
- SMACH is not recommended, userdata is not handled efficiently



Configuration Space Analysis

- Ensured UR5 feasibility
- Brute force configuration space search allows visualization of workspace within bins
- Optimized placement of robot origin with respect to shelf
- This did not indicate how difficult the planning problem would be.
- Recommend a UR10 to increase configuration space
- Linearly actuated base or end effector would also help

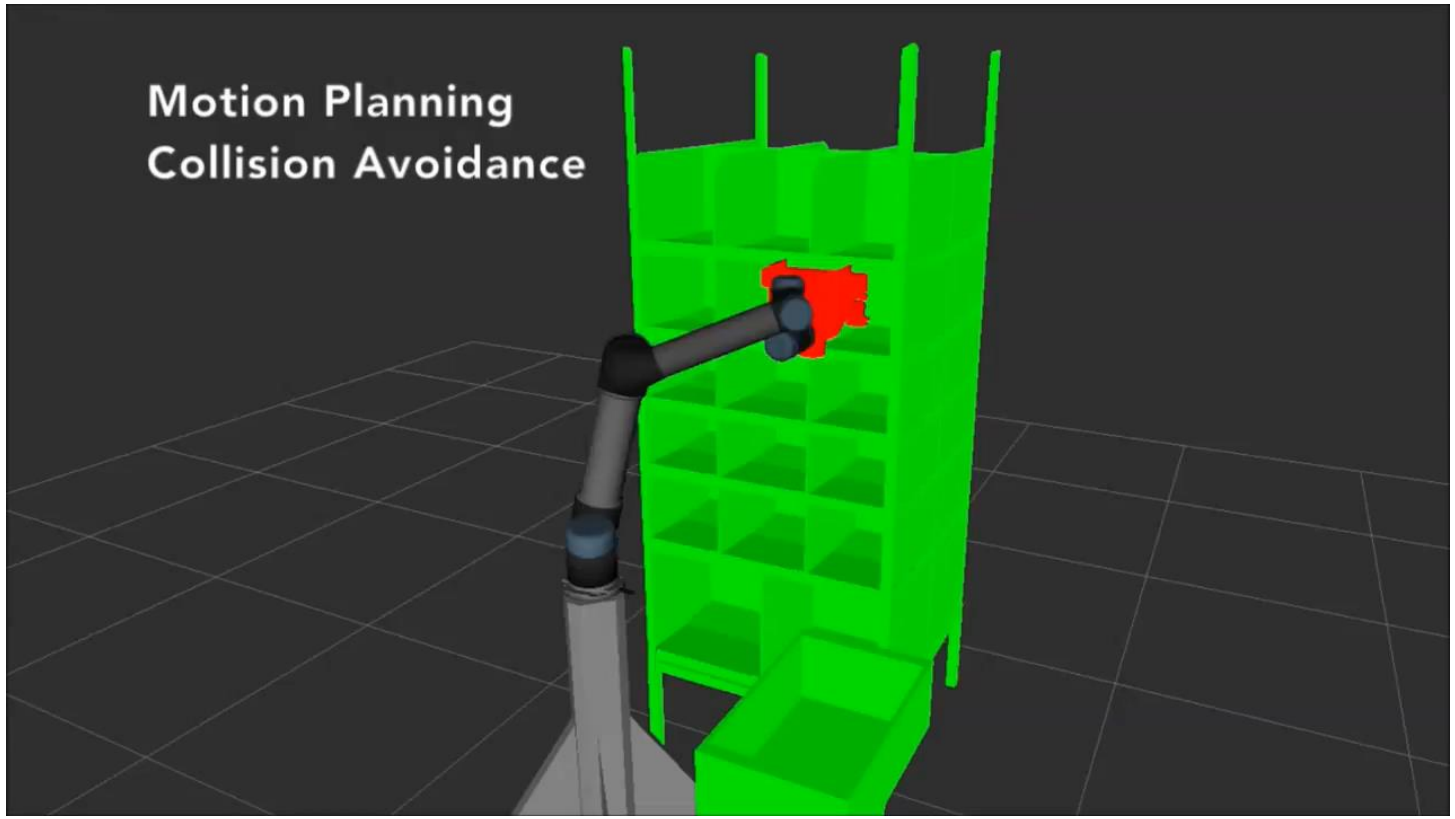


Localization

- Images captured from multiple perspectives merged using robot kinematic chain
- ICP Algorithm minimizes error between point cloud and shelf CAD model
- Algorithm runs twice during competition
- Worked sufficiently well for our needs, but recommended future algorithm constraining shelf height, etc.

Motion Planning

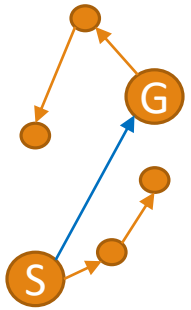
- MoveIt! software package manages arm kinematics and path planning
- Base, end effector, and shelf modeled as collision objects
- Hierarchy of planners was implemented
  1. Lookup Table
  2. Cartesian Plan
  3. Simple E-graph
  4. SBPL (ARA*)
  5. OMPL (RRTConnect)



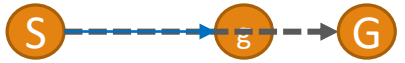
Planner Hierarchy

- A planning request prioritizes planners and returns the first successful plan augmented by velocity scale factor

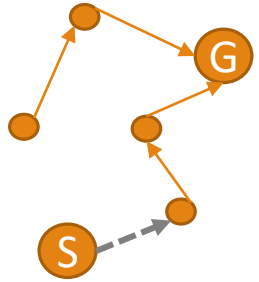
1. Lookup Table
   - A directed graph from joint state -> goal pose learned from single queries
   - Pros: Fast and consistent. Mitigated Hose entanglement
   - Cons: Manual process to learn paths, difficult to delete specific learned plans
   - Improvements: Bidirectional Graph, DOT visualization, debugging tools



2. Cartesian Plan
   - End Effector moves in straight line interpolated from current pose to goal pose
   - Partials solutions can be returned if desired
   - Pros: Fast, good solution quality
   - Cons: only applies to a limited set of joint configurations
   - Improvements: Better understand Moveit! implementation or Build our own



3. Cartesian Snap
   - Andrew's coupled version that performs a Cartesian plan to the nearest node on lookup graph
   - Pros: Improved in-bin planning times, good solution quality
   - Cons: Very painful to learn the in-bin plans using single-query planning
   - Improvements: Automate trajectory saving. Quantify Solution quality for our problem



Planner Hierarchy

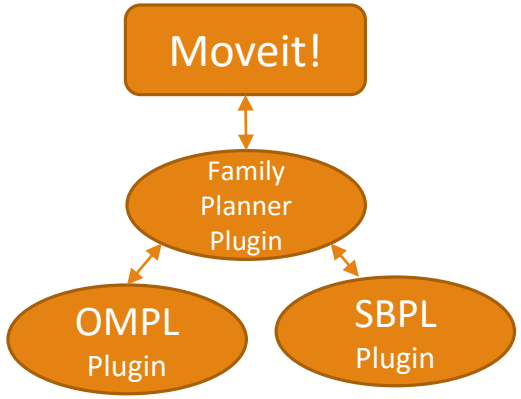
Initial development used OMPL RRT planners. SBPL Integration required family planner workaround (courtesy of Andrew)

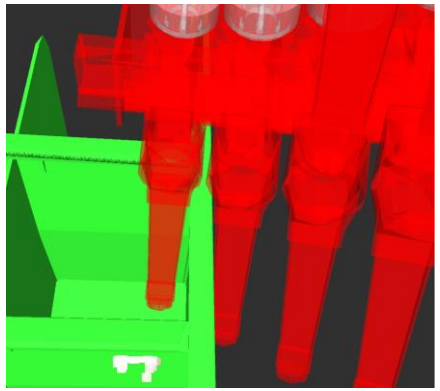
4. SBPL (ARA*)
   - Used SBPL ARA* for single queries
   - Pros: planning in-bin
   - Cons: Trajectory execution was jerky, Planning times longer. Sometimes optimal solutions were undesired (bin 6 solutions inverted hose 360 Deg).
   - Improvements: Modify SBPL plugin to allow arbitrary planning groups. Quantify Solution quality
5. OMPL (RRT Connect)
   - OMPL Randomized planners for single queries
   - Pros: Fast, randomization was often helpful. Planning outside of bins
   - Cons: non-deterministic, solution quality varied greatly

Planning through walls was an intermittent issue but fixed by upsampling collision checking

UR5 high joint velocity disables were a problem despite slowing down trajectory execution

➢ Need trajectory smoothing or optimization for UR5 on all returned solutions



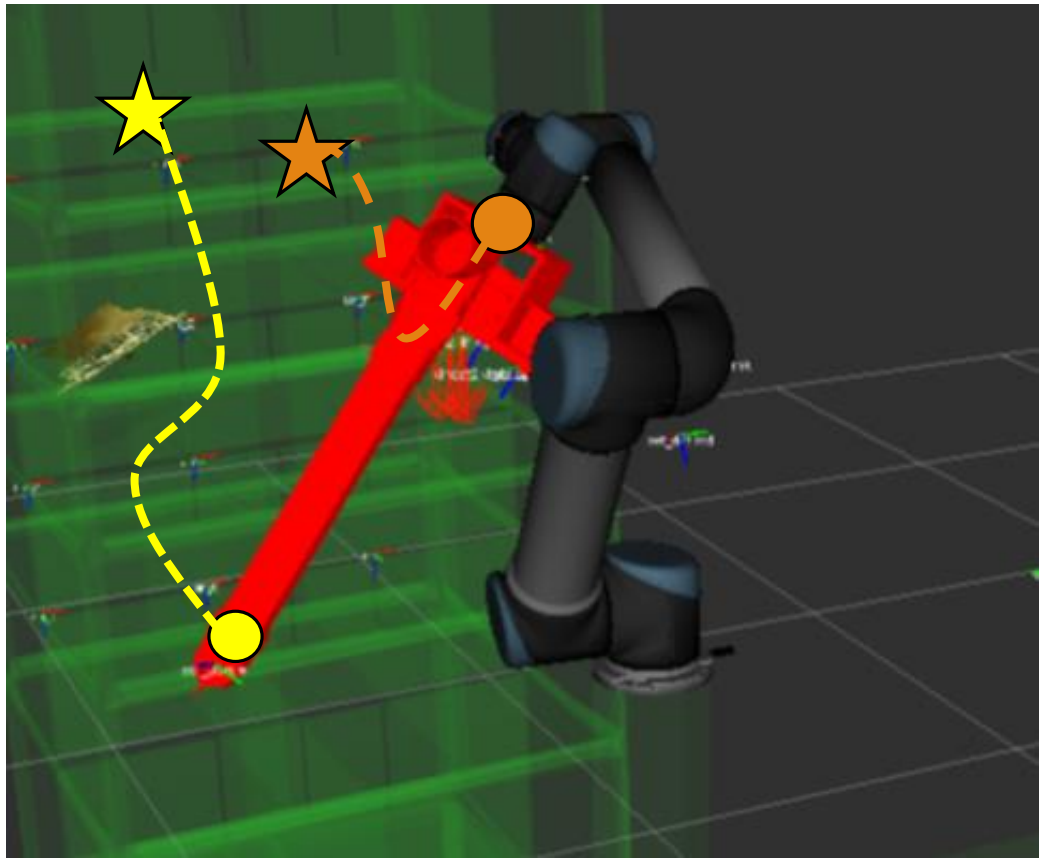


Before Upsampling

SBPL Planning

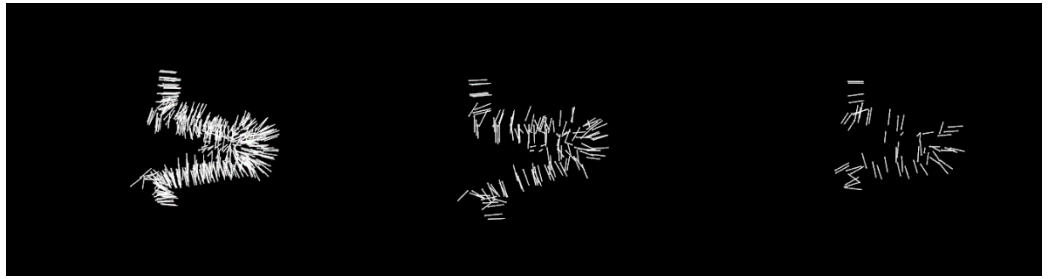
Required addition of 2-finger heuristic to solve for goal orientations

A well designed system should require minimal planning

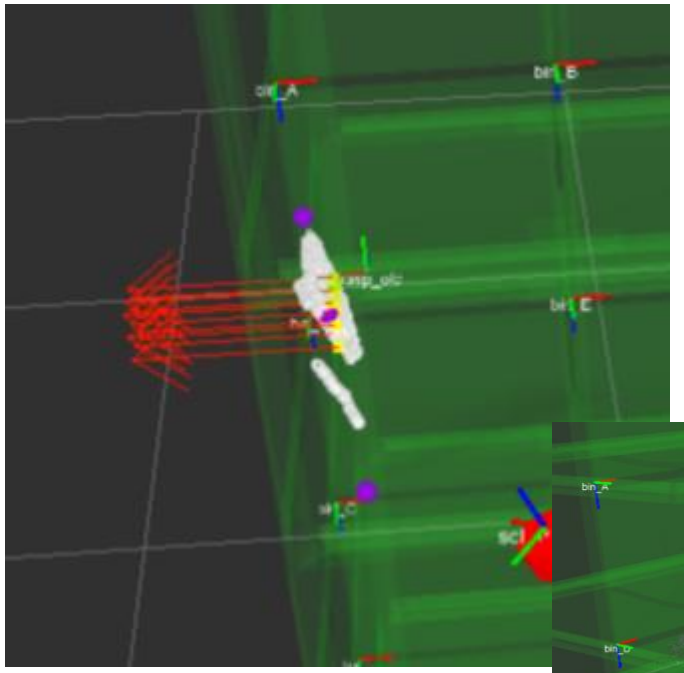


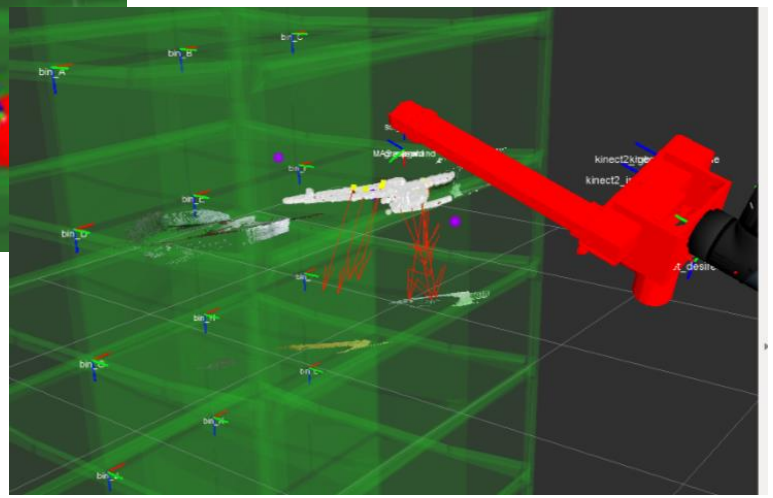
https://youtu.be/qqdvioAipnU

Grasp Planning



1. Perform normal estimation
2. The centroid and bounding points of the pc are computed
3. Select Grasp Primitive (top-down or sideways)
4. sort best normals
   - 100 candidates, ordered
5. Compute EE poses (RPY randomization added)
   - ~100 pose-sets
6. Check pose sets against fastIK and only keep feasible goals
   - 5 – 25 candidates feasible, ordered
7. If too few candidates, generate default poses (At least 12 guaranteed candidates)





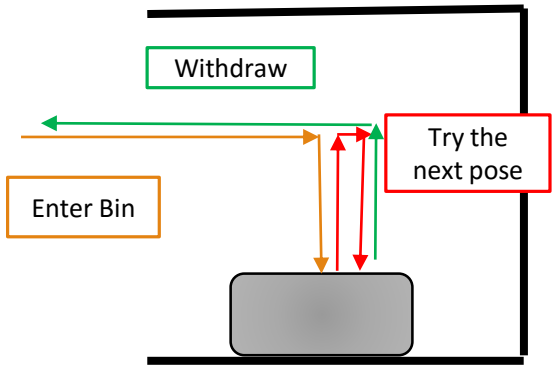
Grasp Execution

Every pose is attempted by following process

snap - snaps the target pose to the limits of the bin
set – update visual pose marker
call - move_arm_server to move the arm
sleep - sleep briefly to reduce concurrent
move_arm_server request

The execution ends when :
- Suction sensor determines success
- The end of the pose list is reached
- The amount of time executing grasp > 1 minute

Most of the time was spent in this operation. Single query plans inside the bin are slow.



Vision: Segmentation

Geometric Filtering - Shelf contents isolated from shelf based on localization results and Kinect Pose

Segmentation - Shelf content clustered into shelf/not-shelf regions.

SegNet CNN was trained on acquired data but eventually not used.

Raw Geometric Filtering images were passed to identification CNN

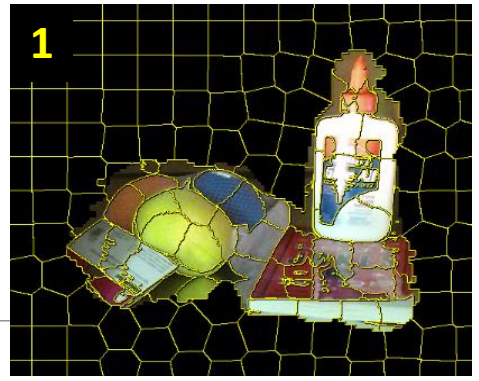
Vision: Identification CNN

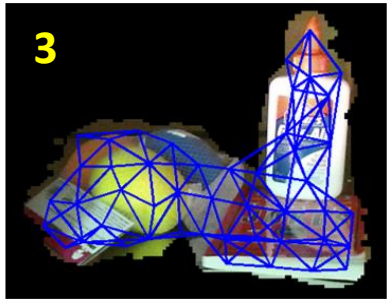
SLIC (1) Divide image into small “superpixel” segments based on colors and edges

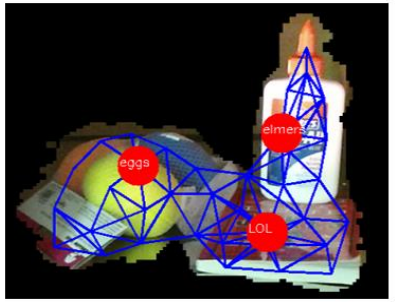
Segment Identification (2) Individual segments are classified using AlexNet CNN

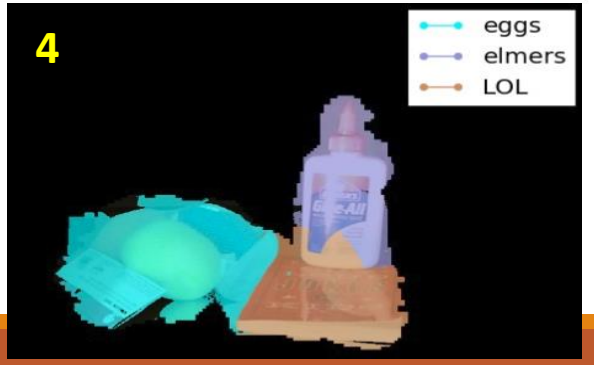
Graph Generation (3) Neighboring “superpixels” are connected, forming a graph

Item Identification (4) Individual “superpixel” identification outputs are merged to solve for an optimal scene









Vision: Database Generation

A turntable was developed to automatically rotate items and scenes and record images.

100 - 200 images were collected for each item

Images were rotated, mirrored, colored skewed to upsample to add variety to the dataset

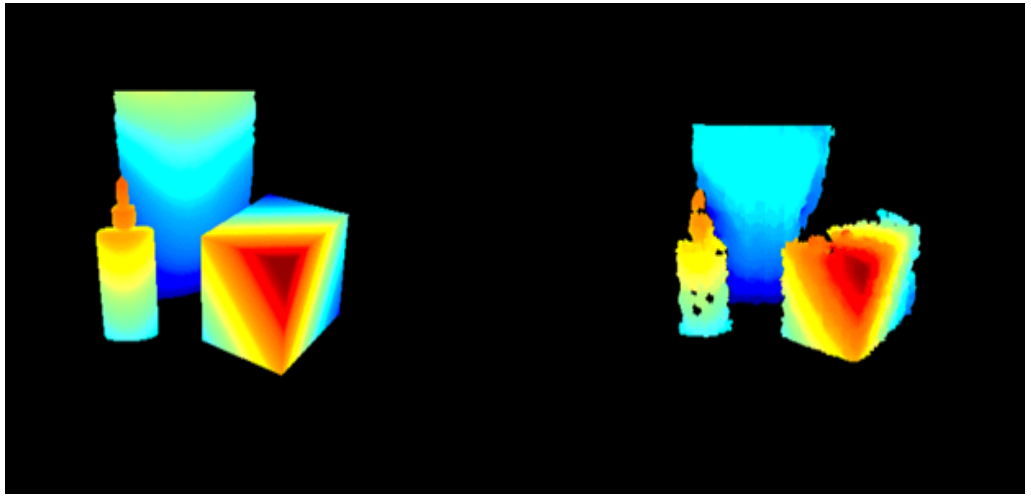
6,000 Raw images -> 400,000 training superpixels

Vision: Perch

Perch: Geometry based search recognition and location solver developed by Venkat Narayanan and the Search Based Planning Lab

Perch was enabled on 4 of 40 items at the final competition. The pose estimate of these items was great. Perch processing took approximately 60 seconds

- Glucose tablets
- Folgers coffee
- Paper towels
- Kleenex tissues

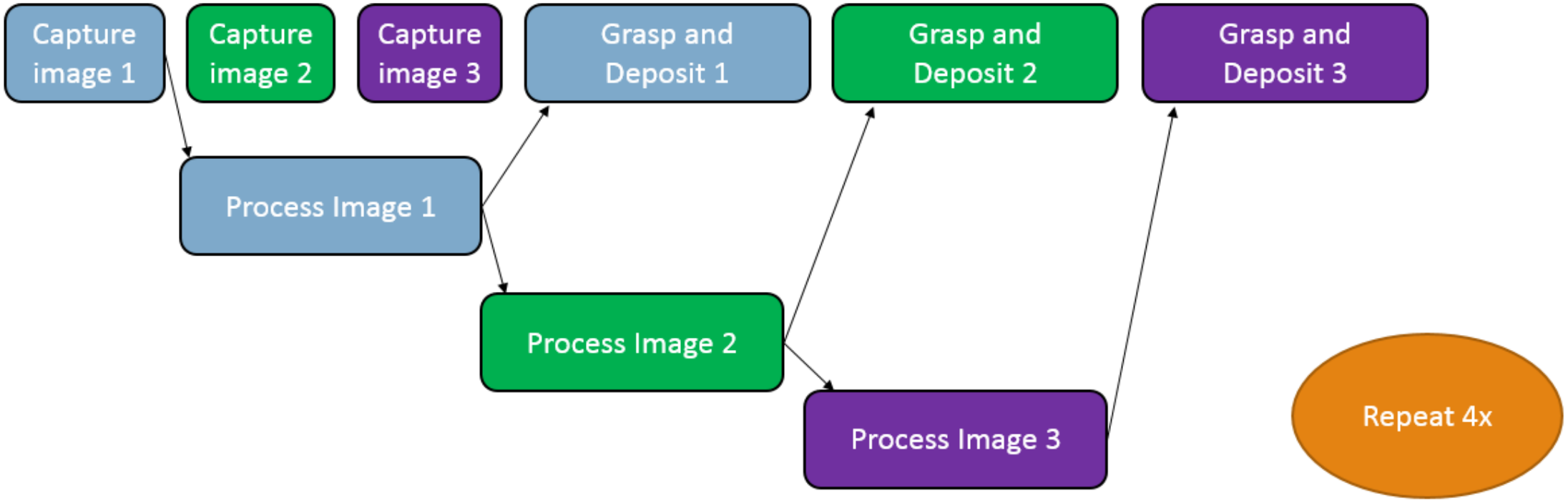


Vision: Parallel Perception

Executive improvement to parallelize vision processing

- Runtime 14:30 min -> 12 minutes

Make subsystems as fast as possible. Best teams could attempt 30+ bins in 15 minute time slot



Vision: Lessons Learned

- Need functionality to determine confidence in perception results during runtime
- Lighting WILL be an issue, so plan accordingly
- Point cloud fusion techniques should be used to generate more dense point clouds
- Reconsider sensor selection

Competition Results:

Stowage

Due to a judging error (we were given the wrong input JSON file), we ended up running our robot twice.

During the first run, we successfully put 7 items back onto the shelf. Our JSON was scored as perfect due to the organizers error, giving us a score of 88.

During the second run, we successfully picked 8 items but were deducted for a misidentification, scoring 77.

Picking

During our picking run, we successfully picked 4 correct items but picked one incorrect item, giving us an overall score of 33 points.

Due to some of our go/no-go criteria on which items to pick to maximize score, we only attempted picking up items in 7 of 12 bins, which was a bit of a disappointment to watch but definitely the right call due to all our previous testing.

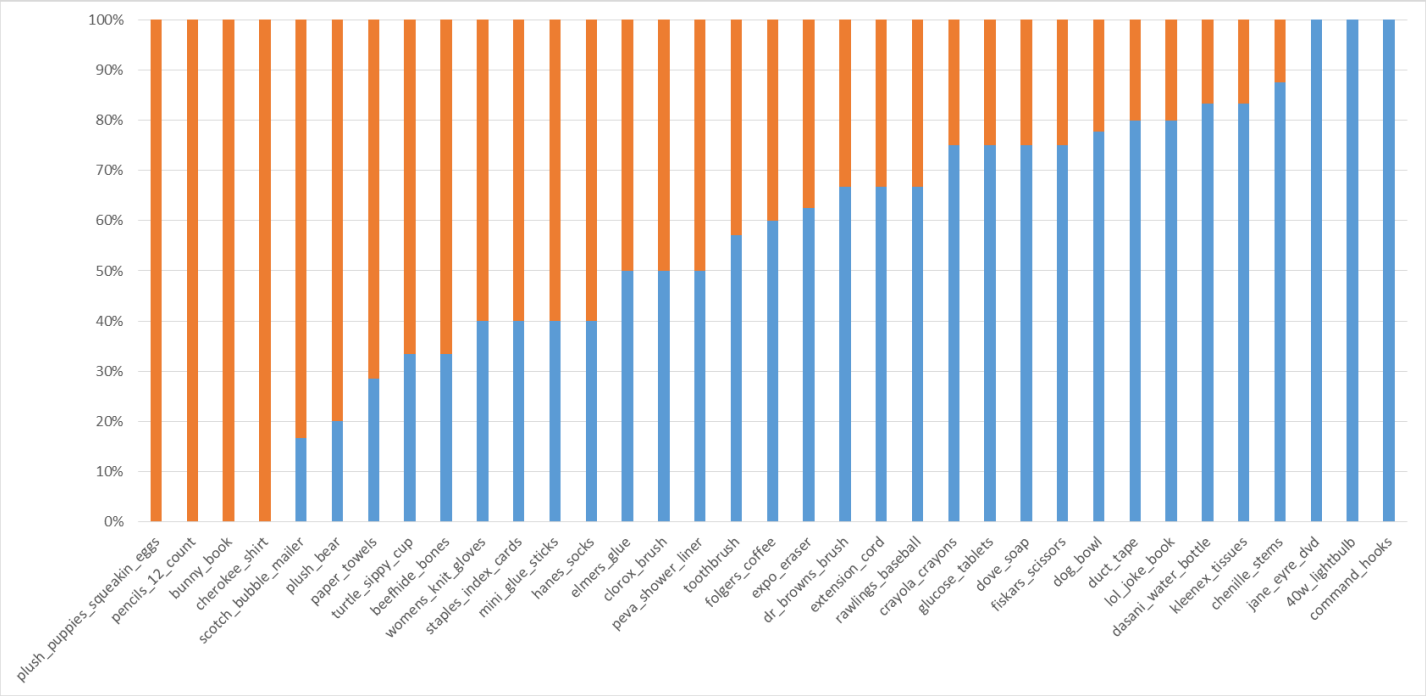
Additional Lessons Learned

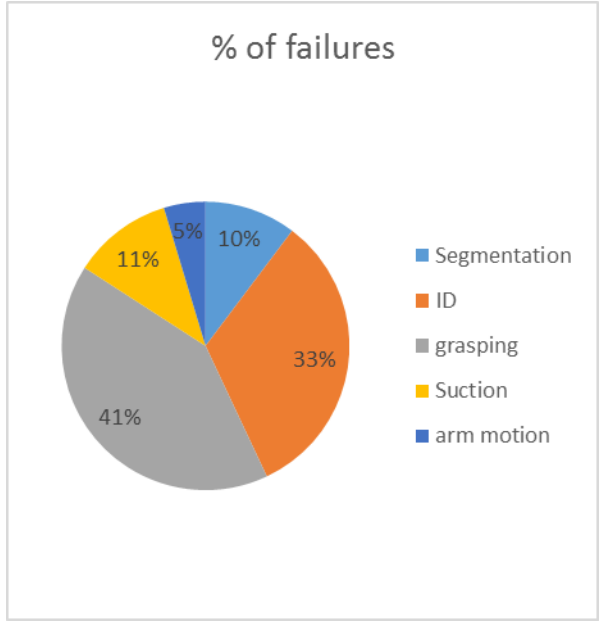
- Improve traceability between off line testing and running the robot
- Dedicate time to optimizing score
- Make everything run as fast as possible
- Keep up good software development throughout the development process
- Plan shipping, competition setup, and system verification well in advance

Questions

SVE:
Testing Results Breakdown

Overall Results					Failure Breakdown				
Total Runs	Successes	Failures	Overall %		Segmentation	ID	grasping	Suction	arm motion
203	96	107	47%		11	35	44	12	5

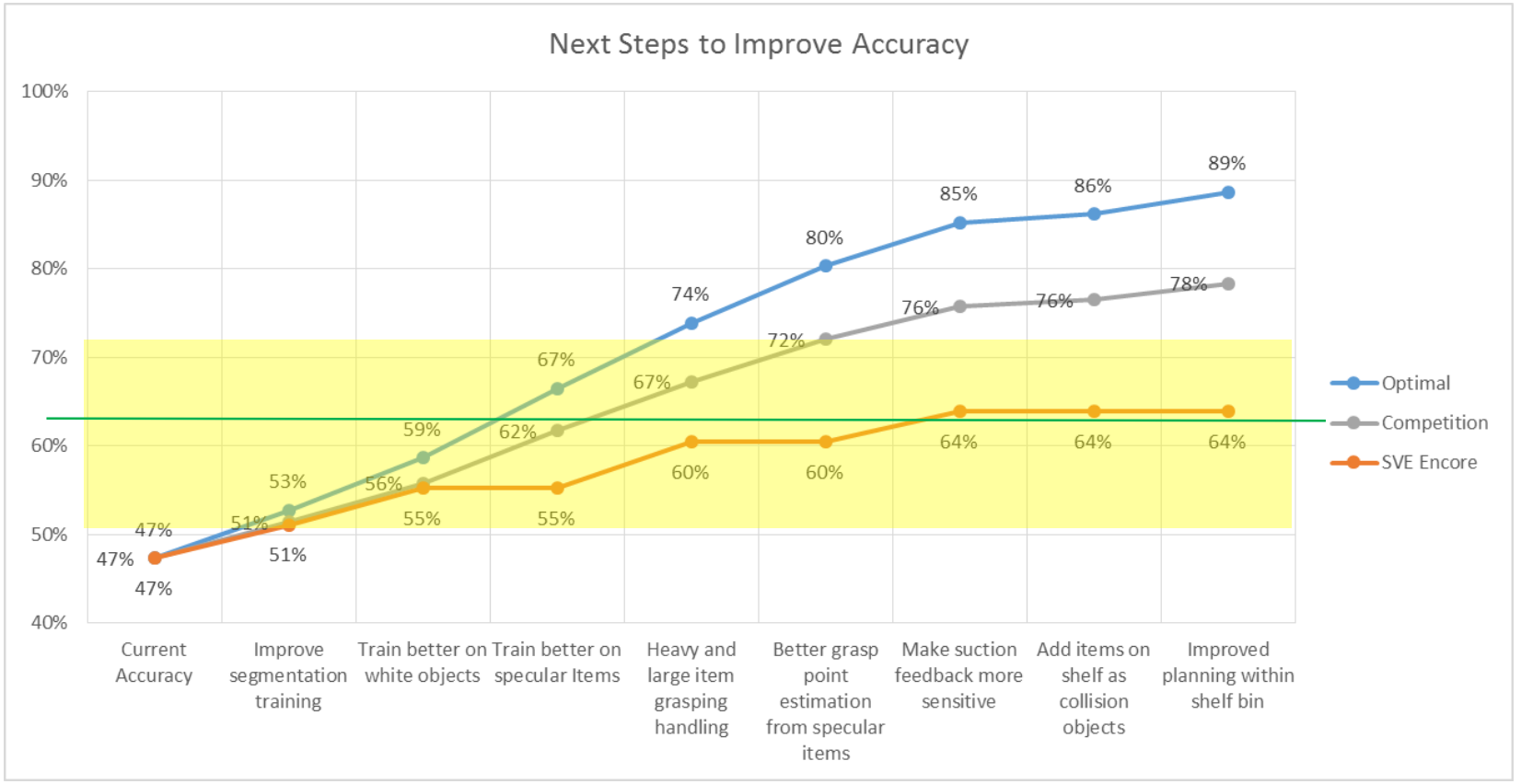




SVE Encore: Status

7	Random Shelf Configurations
244	Grasp attempts
147	Successful Picks
97	Failed Picks

Upper	73.0%
mean	62.4%
Lower	51.9%
std	10.6%



Vision: Stowage Identification

Stowage Identification Architecture 5/7/2016 ----------
2nd slave computer will run kinect and tell the master which bin to go to and provide JSON changes
Filessrc/preprocess_id_image - subscribes to kinect and segments pc to cnnscripts/perception/* - Utilities for running CNN. Server for getting predictions
'stowage_perception_srv'scripts/prediction_utilities.py - utilities for performing prediction. **Can be used for offline simulations**

scripts/stowage_perception_server.py - Main supernode; manages userdata and provides master a server interface
Network-setup.bash - exports ROS_IP and other network params

MessagesTo conform to new proposed architecture, should be contained in harp_apc/apc_msgsMaster-Slave Interface

Stowage_bins2targetapc_msgs/bin[] bin_contents

- 5int8[] tote_contents
- ---int8 target_bin

Stowage Perception Functions, upon service call...Call preprocess_image server to perform pointcloud segmentation

Call CNN to perform predictionUpdate Item belief state

- Perform prediction on all available predictions
- Does not renormalize after taking best guess (it provides betteroverall accuracy)Assign item to shelf bin
- If item is confusable -> send it to the bin with the confusable item
- If item is small -> send it to more dense bins
- If item is large -> send it to less dense bins