FRASIER: Fostering Resilient Aging with Self-Efficacy and Independence Enabling Robot

Team Northeastern's Approach to 2019 RoboCup@Home

Tarik Kelestemur, Maozhen Wang, Naoki Yokoyama, Joanne Truong, Anas Abou Allaban, and Taskin Padir

College of Engineering, Northeastern University
360 Huntington Ave., Boston, 02115 MA, USA
http://robot.neu.edu/frasier/

Abstract. This report describes Team Northeastern's progress to meet the requirements of the 2019 RoboCup@Home Domestic Standard Platform League (DSPL). We present our ongoing research efforts on affordance-based navigation method for efficient path planning in cluttered environments. We also discuss our results on manipulation of objects, 2D and 3D perception, speech and face recognition. We demonstrate our results using the Toyota Humman Support Robot platform awarded to our team in 2017.

1 Introduction

Our overarching goal in this research and development effort is to advance the capabilities of Toyota Human Support Robot (HSR) for a successful team performance and technology demonstration at the 2019 RoboCup@Home DSPL. We will achieve this goal by (1) leveraging our team's robotics competition experience from the DARPA Robotics Challenge (DRC), NASA Sample Return Robot (SRR) Centennial Challenge, NASA Exploration Robo-Ops Challenge, and Intelligent Ground Vehicle Competition (IGVC), (2) developing a systematic model-based task validation methodology, (3) implementing novel perception based navigation, mobile manipulation and human-robot interaction techniques. Successful completion of this project will not only progress the technological readiness of autonomous personal service robots for practical applications but also contribute new knowledge and methods to the RoboCup@Home DSPL community.

Our research at the Robotics and Intelligent Vehicles Research Laboratory (RIVeR Lab) at Northeastern University made research contributions in experimental robotics for disaster response, service and space exploration; human-in-the-loop robot control; and whole-body motion planning and control for humanoid robots. RIVeR Lab led the Team WPI-CMU in the DARPA Robotics Challenge (DRC) from 2012-2015. More recently, we have been selected by NASA to receive a Valkyrie humanoid robot [1].

[1] https://www.nasa.gov/press-release/nasa-awards-two-robots-to-university-groups-for-rd-upgrades

2 Research Plan for RoboCup@Home DSPL

2.1 Affordance-based Navigation

The robot navigation as of today behaves differently from human-beings. With the passive obstacle avoidance implemented in classical robotic navigation, the mobile robots may have to take long detours or even fail navigating when unexpected obstacles appear, which is quite normal in our living environment. Unlike robots just avoiding all obstacles detected on the path, we human-beings are capable of handling the unexpected obstacles in a more active way, such as moving the obstacles to clear the path. We proposed a navigation algorithm based on affordance that enables HSR to remove unexpected obstacles on the way to destination.

The concept of affordance was first introduced by psychologist J.J. Gibson [1]. It describes the actions that an object can afford to the perceiving agent. In the case of navigation, affordance can be simplified to the mobility, such as pushability, pullability or liftability of the obstacle. For example, a chair affords being pushed away and a cup affords being picked up. We further developed the research conducted regarding affordance detection in [2] and applied it to navigation. The affordance is extracted by two consecutive steps: detection and validation. For a new obstacle, point cloud processing is first performed to find out basic geometry primitives that the obstacle consists of, such as planer, cylinder or sphere. Those primitives then are bonded with potential interactions and become affordance candidates. For example, HSR assumes large vertical plane as pushable and small cylinder as liftable. HSR then validate the affordance candidate by performing the potential interaction. The feedback from the wrist force sensor is then be used to tell if the obstacle can be moved as assumed. Our navigation algorithm is specialized for in-door environment where a global map of static obstacles is provided. The problem we want to solve is when a robot met a new obstacle on its way to the goal, what the robot should do with the new obstacle. The work flow of our algorithm can be described as following: the home configuration of large and unmovable furniture is provided to HSR as a global map. When HSR is navigating from Point A to Point B, it calculates a free path based on global map. While HSR is executing the path, a new obstacle appears on its way. With affordance detection and validation, HSR can first determine if this new obstacle is movable. If the obstacle is movable, HSR can just move the obstacle and continue executing the original path. Otherwise, HSR can take a detour just like classical navigation.

At current stage, HSR is able to validate liftability and pushability for unmet obstacles and take advantage of extracted affordance to clear the path instead of taking a detour. However, the re-position for movable obstacles is not defined yet, which will be our future work. Another future work will be including physical properties in the affordance model, which can better guide the robot to interact with the obstacles.

3 Accomplishments To-Date Towards Qualification

In this section, we describe our methodology in completing the RoboCup@Home DSPL tasks. We present our results using the Toyota Human Support Robot (HSR) platform which was awarded to our team in 2017.

3.1 Perception

Object Detection and Segmentation We use the Mask R-CNN framework [3] for object detection and segmentation. In order to gather a sufficiently large amount of training data in an efficient and rapid manner, our approach involves generating an exhaustive artificial training set afflicted with various types of noise. By adding noise that may appear in the images taken by the camera, we train our object detection model to be more robust against them during inference [4].

To generate the artificial training images, videos of each object rotating atop an automatic turntable are recorded. This is done at various camera elevations to capture all angles of the object from different perspectives. Background subtraction is then performed on each frame of the video in order to extract all contours of the object. These extracted contours are then randomly chosen, scaled, and rotated, before being placed on an image of scenery similar to the types of environments the HSR may operate in. For each image, randomly chosen contours from a few randomly chosen classes are used in order to teach our model how to best distinguish the objects from each other when they appear together, which is especially helpful for objects that are similar in appearance. Occlusions, up to a certain percentage threshold, are allowed and are properly recorded when annotations are created. Each of these composite images are then afflicted with random adjustments in lighting and artificial image noise. Once a composite is generated, a corresponding annotation that records the object instance's class and the pixel-wise contour is created in COCO format [5].

Person Detection OpenPose [6] was used to find the keypoints (i.e. right elbow, left knee) of each person in the camera frame. Each person could then be segmented by their head, torso, and legs. Using three convolutional classifiers, their gender, age, and emotion could be inferred from their face. Using the DeepFashion dataset [7], we trained an InceptionV3 model to recognize the clothes in a picture. By feeding this model images of each person's torso and legs, their upper and lower clothes could be inferred. Finally, by using a color histogram conditioned on pixels near particular keypoints, the color of each clothing could also be inferred. An example of the output can be seen in Figure 1.

Point Cloud Processing In order to plan collision-free trajectories, a complete model of the environment needs to be generated. The processing starts with down-sampling the raw point cloud. Later, we use the RANSAC algorithm implemented in [8] to segment planes in the point cloud, e.g., shelf racks, table.

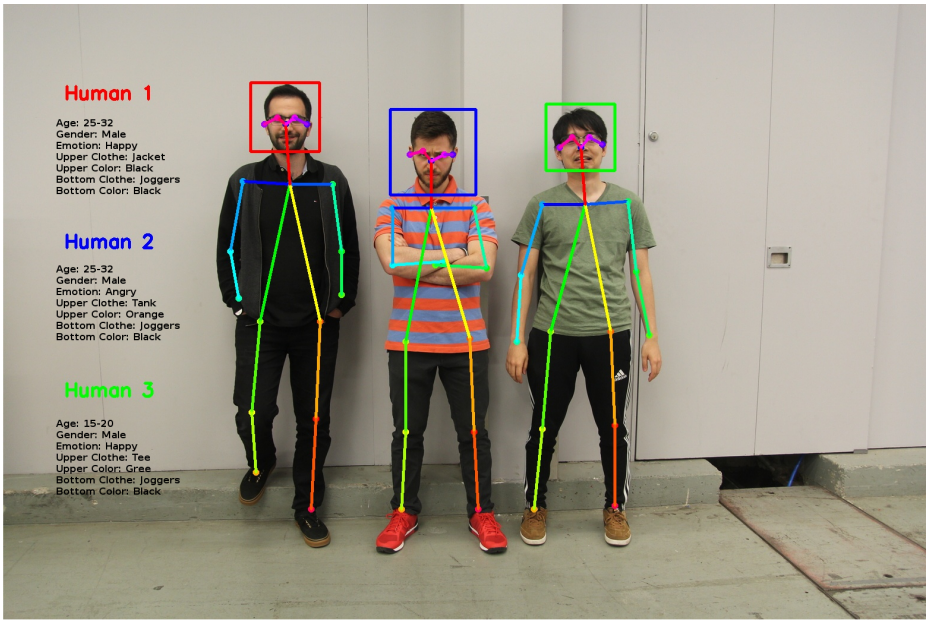

Fig. 1. Person Detection

After segmenting primitive shapes, we use Hierarchical Approximate Convex Decomposition method [9] to generate mesh of the remaining point cloud in the scene. For estimating the shape and size of the objects, we implement various algorithms. Once the object of interest is segmented in the RGB domain using Mask R-CNN, the correspondence of the RGB and the depth sensor is used to get point cloud cluster of the object. The object cluster is then fed into the RANSAC algorithm to estimate its primitive shape. The position and orientation of the object cluster is found using principal component analysis as explained in [10] which is then used for grasp synthesis. Figure 2 shows the result of the complete point cloud processing and the HSR model.

3.2 Grasping and Motion Planning

Grasping A heuristic-based grasp synthesis method that uses the geometries of the object and the HSR's gripper is developed. The method takes the object's primitive shape, size and the transformation as the input. A number of grasp poses that are aligned with the principal axes of the object are sampled uniformly over the surface of the object. Pre-grasp positions are found by shifting the sampled points outside of the object along the principal axes by 5 cm. These pre-grasp poses are then filtered out by checking collisions with the environment. Finally, the closest grasp pose to the HSR's gripper is selected.

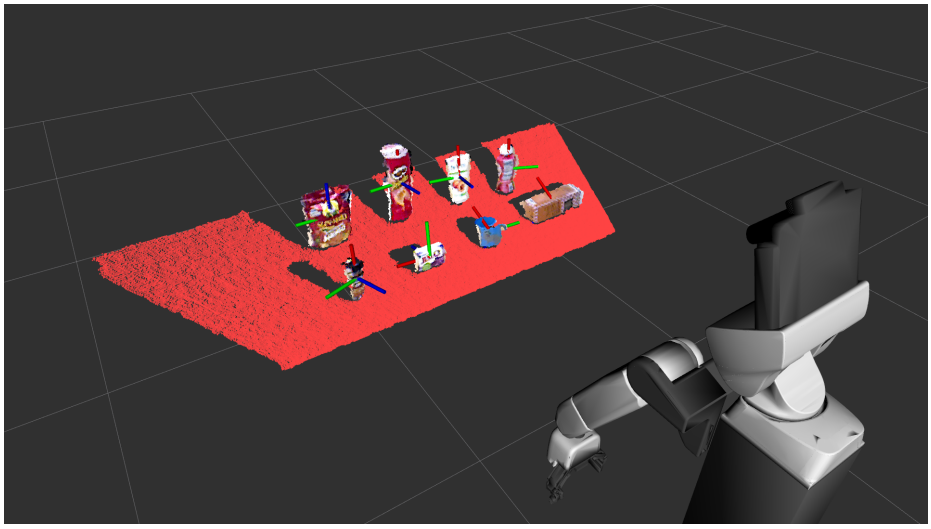

Fig. 2. Tabletop Object Clustering. Red: segmented table plane RGB: object clusters.

Motion Planning The motion planning problem can be formulated as a non-convex optimization problem subject to inequality and equality constraints and can be solved by a nonlinear optimization algorithm such as sequential quadratic programming (SQP). In this study, the decision variables are the joint positions for T time steps, and the cost function is the distance between consecutive configurations:

$$f(q_{1:T}) = \sum_{1}^{T} ||q_{t+1} - q_t||^2, \tag{1}$$

where $q_t \in R^8$ represents the joint configuration at the t-th time step for the 8 DOF kinematic chain. The desired end-effector pose is enforced by an equality constraint such that: $\Delta FK(q_t) = 0$ where FK is the forward kinematics function that calculates the Cartesian end-effector pose given joint positions and $\Delta FK(q_T) = [\Delta x_t, \Delta y_t, \Delta z_t, \Delta roll_t, \Delta pitch_t, \Delta yaw_t]$ is the deviation of the end-effector position and Euler angles from the desired pose. The joint limits are defined as inequality constraints: $(q_t - q^-) > 0$ and $(q^+ - q_t) > 0$ where q^+ and q^- are the maximum and minimum limits of the joint positions, respectively. The Gilbert-Johnson-Keerthi algorithm [11] is used to compute the collision distance, and a hinge-loss function is used to set up the constraints as described in [12]. Once a trajectory is planned, it is smoothed using tension spline interpolation based on the velocity and acceleration limits. Finally, the resulting trajectory is sent to the HSR's position controller as a reference.

To evaluate the performance of our motion planning method, we conducted a real world experiment of a pick-and-place task. The overall experiment is picking an object from a table and placing it onto a shelf within the robot's workspace. The experimental setup is shown in Figure 3. The pick position is selected 3cm

Fig. 3. Experiment setup: (Left) Environment model (Right) Real world

away from the object's center towards the robot and the orientation is aligned with the object's principal axes. The place pose is selected as the middle of the top shelf rack. For a fair comparison, the environment is modeled in the beginning and used for all tests. Two cases are tested:

Two-step Planning A classical two-step approach in which two separate trajectories are planned for each of picking and placing poses. The robot executes the first plan, grasp the object, executes the second plan and places the object. For each planning 5 waypoints are used and the end-effector goals are given as pose constraint at the last waypoint to the optimization.

One-step Planning The whole task is planned in one step where 10 waypoints are used and the pick and place pose are given as pose constraints at the 5th and 10th waypoints, respectively. Since the robot will be moving at the picking waypoint, an external controller is used to check whether the end-effector is reached to picking pose by looking at the difference of current end-effect pose and goal pose. If the error is less than 10^{-4}, the fingers are closed immediately. Each case is executed 15 times.

The two cases are compared in terms of planning time, total path length and the execution time. No notable improvements have been observed for the planning time (around 1 second for both cases) and the path length, however, a significant improvement has been recorded for the execution time. For one-step planning, the mean of the execution time over 15 pick-and-place tasks is found to be 20.15 seconds whereas for the two-step planning, it is 26.96 seconds. The standard deviation and mean of execution times are depicted in Figure 4. Since we are using the sum of joint displacements for each consecutive waypoints as the cost function (1), the following joint configurations after picking configuration are close to picking configuration, when it is planning for one-step. Thus, it is shorter for the robot to reach from picking configuration to placing configuration.

Also, one-step planning produces a complete trajectory which results in a better velocity profiling during trajectory smoothing.

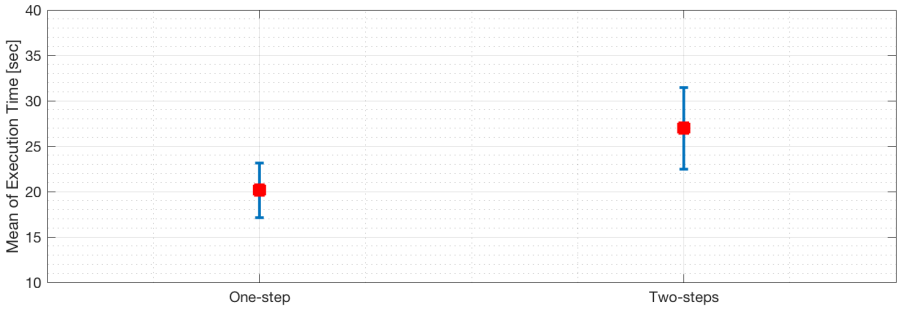

Fig. 4. Mean and standard deviation of execution times

The storing groceries task is a great experiment case for our system since it requires all modules of autonomous manipulation we have developed to work flawlessly in an integrated manner. It is important to note that the configurations of environment is not known beforehand and the participants have only 2 hours access to the objects. Last year at RoboCup@Home 2018, we were the only team in the DSPL to complete the task in time and won the Best Manipulation Skills award.

3.3 Natural Language Understanding

To translate speech into discretized tasks for the robot, we utilize a combination of various neural networks (NN) specialized for NLP. First, we use Google's Speech NN API [13] in order to convert audio into text and to detect hot words to notify the robot that a command is being given. We then use Google's NLP NN API [13] to assign each word in the sentence a label (i.e., verb, location, pronoun, etc.) based on the context provided by the sentence (noun vs. verb, ex. milk vs. to milk). We use this to construct a sequence of tasks for the robot to perform, using verbs as a cue to determine the robot task type and the nouns as cues for locations or corresponding object classes. To classify verbs and nouns that are not explicitly in our dictionary, we use Google's Word2Vec NN [14] to match it to a similar verb in our dictionary. To do this, we compute the cosine between the input word's vector representation to each of the vectors of the words in our dictionary, and check whether or not the largest cosine computed passes a certain threshold. To determine whether or not a perceived sentence is a special pre-defined command, we use the Levenshtein distance to compare it to our list of known pre-defined special commands. Tasks extract related information from the HSR's sensors and known state of the environment during execution to determine feasibility and success.

References

1. James J. Gibson. The theory of affordances chapt. *The Ecological Approach to Visual Perception*, 8., 01 1977.
2. P. Kaiser, M. Grotz, E. E. Aksoy, M. Do, N. Vahrenkamp, and T. Asfour. Validation of whole-body loco-manipulation affordances for pushability and liftability. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 920–927, Nov 2015.
3. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.
4. Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.
5. Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
6. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.
7. Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
8. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.
9. Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 3501–3504. IEEE, 2009.
10. Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.
11. Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.
12. John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.
13. Google. Cloud natural language api, January 2019.
14. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
15. Nicolaus A. Radford, Philip Strawser, Kimberly Hambuchen, Joshua S. Mehling, William K. Verdeyen, and et. al. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 32(3):397–419, 2015.

HSR Software and External Devices

We use a standard Human Support Robot (HSR) from *Toyota*. No modifications have been applied.

Fig. 5. Toyota HSR

Robot's Software Description

For our robot we are using the following software:

- OS: Ubuntu 16.04
- Meta OS: ROS Kinetic
- Person detection: OpenPose
- Motion Planning: OpenRAVE
- Object recognition: OpenCV, Mask-RCNN
- Point Cloud Processing: PCL
- Navigation: Hector SLAM

External Devices

HSR robot relies on the following external hardware:

- system79 Oryx Pro Laptop
- Dell Alienware 15 R3

Cloud Services

HSR connects the following cloud services:

- Speech-to-text: Google Speech API
- Natural Language Processing: Google Dialogflow