Matthew Ryan
Developmental Robotics Project Proposal

Abstract

As technology advances, industrial robots are becoming more and more collaborative. These new robots, called co-bots, are designed to work alongside humans and perform similar tasks in unstructured environments. Compared to traditional industrial robots, these new breed of robots aim to be easier to train / program. It is expected that a user can roll a robot up to an existing work cell designed for a human and just start making use of it. A method where one of these new cobots could self discover the affordances of a workcell and map them to a set of parameterized templates in a behavior tree is presented.

Introduction / Background

Traditionally manufacturing robots have worked in cages and work cells specifically designed for them. As robots become more collaborative, they have started working alongside humans in work cells that were not designed with the robot in mind. Many of the new collaborative manufacturing robots, such as Rethink Robotics's Sawyer, are meant to be moved from work cell to work cell frequently, potentially performing a different job at each cell.

Sawyer was designed as a robot that you can teach by demonstration. A user can press buttons on the arm and “teach” the robot what actions to perform without any programming experience required. One of the features of sawyer is a special mode called “Zero G” where the user can grab a cuff on the end of the arm, and move the robot around freely. The robot will compensate for the weight of the arm due to gravity, and allow the user to easily move the arm. Sawyer also has a built in camera on its wrist that can be used for part detection and localization.

Intera is the software platform that runs Sawyer. Intera uses behavior trees, a technology that is widely used in the video game industry to create artificial intelligence for non player characters, to provide a rich task description and guided task execution. Behavior trees are a directed graph, commonly used i where nodes in the graph represent some action the robot will take. Flow of execution in a behavior tree is left to right, top to bottom. An example of a behavior tree in Intera is depicted in figure [1], in which the robot will move between two positions in space with a delay between the first and second move. Behaviors can be reused across tasks.

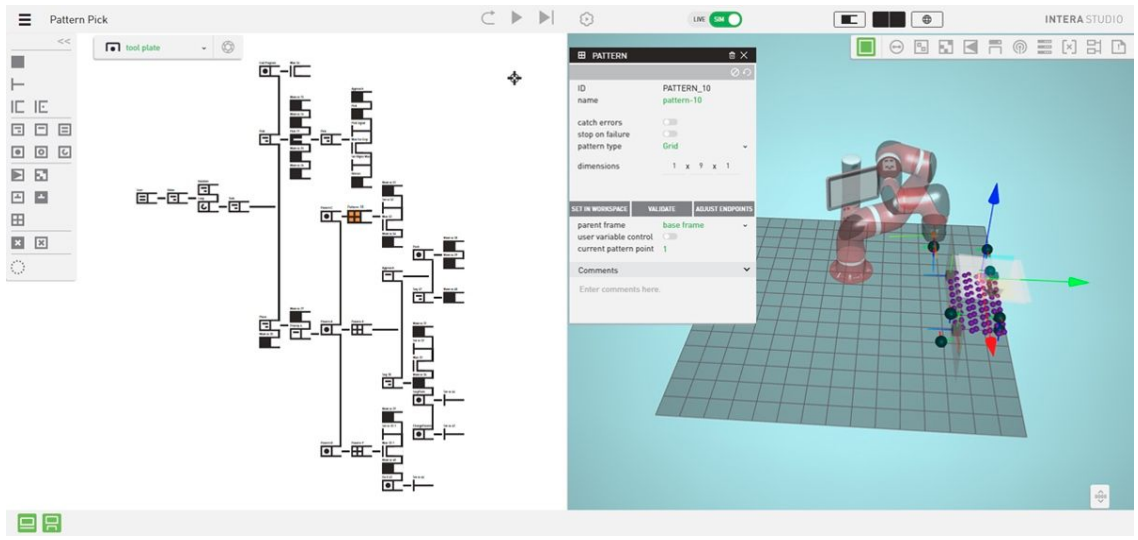


Fig 1 - An example of Intera Studio with a behavior tree

Behaviors in Intera are parameterized - for example a “move node” contains the joint position and cartesian pose of where it wants the robot to go.

Since one of the main goals of Sawyer is to be easily trained in new tasks it would be compelling to have the robot be able to look around its surroundings with its built in camera, find common devices it knows about, and use the object affordances it knows about in order to help the user train a new task. For example, if the task is for the robot to tend a CNC lathe machine, it may have to press a button the machine to start the lathe. Instead of having the user kinesthetically train each motions of the button press with the robot, the robot could look around with its built in camera, identify (and localize) the button using machine learning techniques and map what it knows about the affordances of a button and then present those to the user as predefined behaviors. So the robot sees the button, knows it has a “push” affordance, localizes it in space and then creates a custom behavior to push that button. The behavior is then available for a user to just insert into the task.

Related Work

This work lies at the crossroads of many different research areas. Behavior trees provide an easy to understand, but extensible way to program and teach a robot. At Rethink Robotics, work on a behavior tree paradigm for controlling robots started in 2015, and was inspired in large part by work done by Disla [4] in the video game industry. In the video game industry, graphic and level designers who do not know how to program are able to use behavior trees to control the way their characters act in complicated environments. This is similar to the goal Rethink has for behavior trees - to allow people who don’t know how to program to teach a robot how to complete a complicated task. Rethink is not alone in use of behavior trees for robot control, however, as Paxton et al [5] have developed a behavior tree framework for collaborative robot control called COSTAR that is in the process of being commercialized.

Affordances, first introduced by Gibson [2], describe all the possible actions an object may provide to some agent. Gibson argues this concept is fundamental to how humans learn to interact with the world around them, and is currently being applied in robotics research across various interests [8,9,10]. In our case we plan on making use of affordances that already exist and are defined, and not learning them. However since the concept of affordances is so essential to how humans operate in the world, it makes sense for a robot to try and locate and inform its human companion what affordances it knows about and can perform.

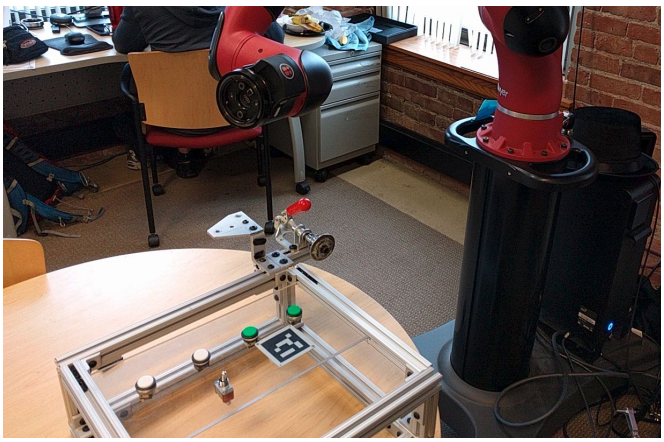
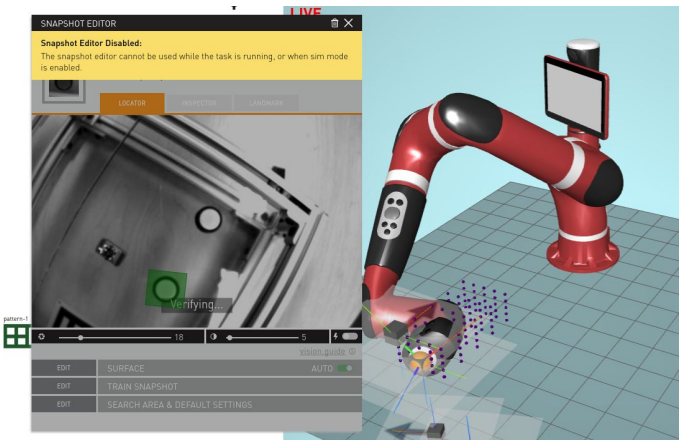
In the last two decades, due to advances in hardware and algorithms use of deep learning and neural networks has taken off. These new systems can be used to detect faces in images[1], process natural language, or find early stage cancer. Google recently open sourced its own internal deep learning / numeric computation software framework, TensorFlow [6]. This framework supports various learning methods and has an Object Detection Framework API [7] that allows users to use their own data sets to train models.

Problem Formulation and Technical Approach

Given the goal of trying to make robots as easy to train as possible, having a robot be able to look around with its camera, recognize and localize objects, know their affordances and then translate that into an actionable behavior of the robot seems like a good step. This seems like a daunting task. However due to some of Sawyer's innate capabilities and advances in machine learning, this is now a feasible problem to solve. While there are countless objects a robot may encounter in a workcell, my first attempts will make use of various kinds of buttons. They are ubiquitous in a workcell, have a well defined affordance (push) and should be fairly straight forward for ML algorithms to find. However, my approach below should generalize to any object that you can detect and which the robot can act upon. I plan on using Google's TensorFlow + Object Detection API in order to train my models and classifiers.

Data Set Collection

The first step would be picking my buttons, and setting up the robot to collect training sets of images. Sawyer has built in object detection and localization, so it is possible to use Sawyer to collect a properly labeled training set. To accommodate this, I built an "affordance box" which I can install various buttons (and other objects like switches and levers) on, and I have trained a task on Sawyer which will move the built in wrist camera around the box and take samples at varying locations over the box. Sawyer can also be taught to "recognize" an object through a pattern matching algorithm and provide its location in pixel space (and 3d space). Using this feature, I will be able to collect a properly labeled sample set for each button. The robot moves in a predefined grid location and saves an image from the camera and an associated text file with location data for each location. The "affordance box" and this process is shown in figure [2] and [3]

	
[2] Affordance Box	[3] Data set collection task executing

Model Training

Once I have a properly labeled data set of all the buttons I wish to learn, I will need to convert it into a TFRecord format which is required for Google's Tensorflow Object Detection API [7], which I will use to train my button model. Depending on the complexity of the objects I wish to train, it may make sense to have a two stage detection process. One which would generalize objects into things like "buttons," "switches," "doors," etc, and then another stage would could classify a specific button. (Ie, this is button model #243243 from acme company). If I can know some specific things about each button, ie, how much force is required to activate it, I can encode that information in my Object Library, which will be used later on for generating behaviors.

Affordance To Behavior Mapping

For each type of workcell object I wish to train, I will need to encode a set of corresponding affordances. These objects and affordances will live in an Affordance Library that the robot can access. If we stick with the button example, the only actionable button affordance is "push." In the Intera world, this "push affordance" would become a simple behavior tree.

The nodes in the behavior tree would be something close to:

1. Move end effector to location near top of button
2. Approach button until contact is made
3. Start applying a specific amount of force on button
4. Wait for that force to be achieved
5. Move end effector back to start location

In addition we would want to parameterize this behavior tree for location and force required to push the button. If we are keeping track of many different "button" classes, the information

required for each button would be located in the Affordance Library - but we could also have a default behavior for affordances that we do not know the specifics of.

Data Execution Pipeline

Once my model is trained and affordances defined, I will create a program that receives images from the camera and runs the classifiers for any trained objects on them. If it detects any objects, it will look it up in the Affordance library and create a behavior tree snippet on behalf of the user.

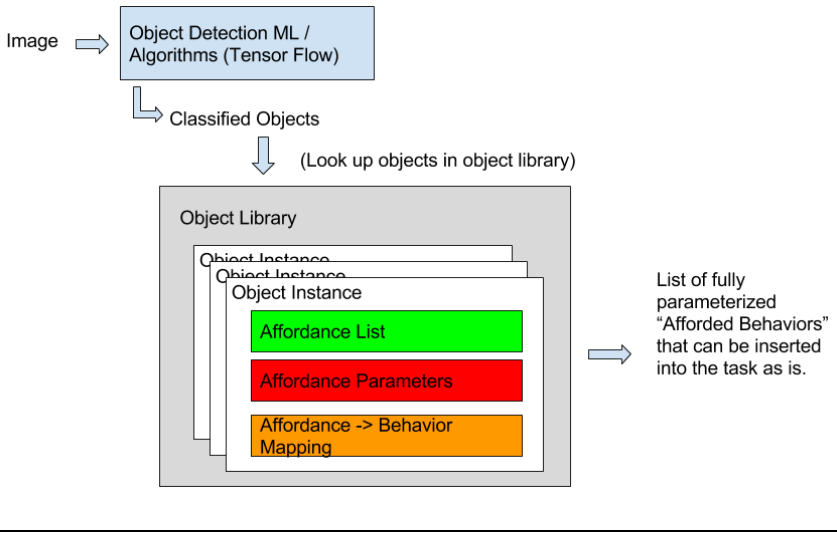


Fig 4 - An example of Intera Studio with a behavior tree

Behavior Parameterization

As previously mentioned, certain things like required force, or dimensions about classified objects can be stored in the Affordance Library. However we still need to learn the 3d location of the object in order to properly interact with it and to provide a fully parameterized behavior tree. Since Sawyer does not have a 3D camera, it would be unable to localize the object without the use of some other mechanism. Luckily, Sawyer comes with its "Robot Positioning System" (RPS). This system consists of a fiducial marker that has fixed dimensions known to the robot. When the robot sees this marker in its wrist camera, it uses the location of the camera and the dimensions of the fiducial to calculate the 3d coordinates of the fiducial. For this work, we will assume that any learned object will have a co-planar fiducial marker in the camera's FOV. This will allow me to calculate the 3d location of the object.

User Interaction / Verification

When training a new task for the robot, a user will have the option to “search for affordances.” For each classified object, the system will generate a fully parameterized behavior tree. The list of behavior trees will be presented to the user in an “Affordance Gallery.” For example, if the robot detects three buttons and a switch, the user would see four available affordances behaviors in their gallery. The user could then insert each behavior into their task or to test / verify that the behavior works. If the robot fails to successfully use the affordance, the user could provide some feedback to automatically adjust the parameters of the behavior and try again until it succeeds.

Experimental Validation

At the end of this project, I expect for the robot to be able to use its camera to find a few pre-learned objects utilizing the deep learning and training capabilities of Tensor Flow. It should then be able to localize and map these objects to predefined affordances / behaviors and automatically perform the afforded action.

Timeline / Schedule

1. Data Collection & Training (2 - 3 weeks)
    a. Pick a few objects (most likely various types of buttons) to be able to search a work cell for
    b. Create a task for the robot to collect labeled training data for all objects
    c. Train, test model
    d. Repeat a - c until I am happy with the results

2. Affordance -> Behavior Mapping (2 weeks)
    a. Create Affordance library and associated data structures
    b. Define Affordances & Behavior Tree templates for each object
    c. Object localization - create code that uses the RPS tool to calculate the location of a classified object. (Assuming it is co-planar to a fiducial marker in the same image)
    d. Code that maps the classified object to a fully parameterized behavior instance

3. User Interaction + Verification (1 - 2 weeks)
    a. Changes to the UI to show “affordance gallery”
    b. Testing + reinforcement learning of each behavior based on feedback from user

References

1. H. A. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, Jan 1998.
2. Gibson, J.J. (1977). ``The Theory of Affordances". Chapter in: Perceiving, acting, and knowing, R. E. Shaw and J. Bransford (eds), Lawrence Erlbaum, Hillsdale.
3. Marzinotto, A., Colledanchise, M., Smith, C., Ögren, P. (2014) Towards a Unified Behavior Trees Framework for Robot Control. In: (pp. 5420-5427). IEEE Robotics and Automation Society
4. D. Isla, “Halo 3-building a better battle,” in Game Developers Conference, 2008
5. Paxton, Chris, Andrew Hundt, Felix Jonathan, Kelleher Guerin, and Gregory D. Hager. "CoSTAR: Instructing collaborative robots with behavior trees and vision." In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 564-571. IEEE, 2017.
6. Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467* (2016).
7. "Speed/accuracy trade-offs for modern convolutional object detectors." Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017
8. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control
9. Stoytchev, A. (2005). "Behavior-Grounded Representation of Tool Affordances", In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, April 18-22.
10. Sinapov, J. and Stoytchev, A. (2007). "Learning and Generalization of Behavior-Grounded Tool Affordances", In proceedings of the IEEE International Conference on Development and Learning (ICDL 2007)