Activity and Object Affordance Detection

Sean D'Andrea

Abstract
The goal in this project is to create an online version of an offline system that classifies human activities as well as affordances of objects in a dynamic scene. The algorithm has five distinct sections. The first, data collection, is implemented in ROS as well as the last three, segmentation, feature generation, and SVM. Object classification, which is part of another project, is not yet integrated into the online system. Experiments demonstrate the system has about 55% accuracy on activity detection and a little less than 80% accuracy when detecting object affordances. A short explanation of these numbers follow. Finally, there is a brief discussion of how the system might be expanded, primarily by integrating object detection and causing a proper reaction in a controlled robot.

Introduction
Something we take for granted in our day to day lives is the ability for humans to understand the context of a situation. The primary means by which we determine our activities is by interpreting the objects and actions surrounding us all the time. Take a simple example. It's morning and you've just woken up. You see your brother pouring a dry bowl of cereal, and he mumbles something at you that you were too tired to process. Instinctively, however, you know what he wants is the milk jug next to you to put in his cereal, so you pass it to him. We think robots should be able to do this too.

Specifically, my project extends the extensive work by Hema Koppula in what she calls activity and object-affordance detection. Hema has created an offline algorithm that handles the perception side of reactivity described above. Specifically, it can take in 3D kinect data and eventually output the affordances of objects in the scene as well as what actions people are taking from a list of about 10.

My job, then, was to extend this work to be online. The algorithm is not real time, in the sense that the robot would seem to be reacting quickly. However, my work has been to create a pipeline in which a robot can stare passively at a scene and through communication between multiple constantly-running ROS nodes, figure out what is going on without outside intervention. Due to time limitations as well as other reasons, the system currently has the first portion and the latter portion working, with a component missing in the middle, which I will expand upon below.

Approach

Framework
The framework takes advantage of the ROS topic interface. Each portion of the perception code subscribes to a ROS topic that the previous portion writes to. When new data shows up, it runs the next batch of the algorithm. A tag determines which nodes sent the message, and a unique identifying number as well as some metadata is passed between algorithm nodes so it can know where to look for data. Details of the framework are explained below.

Perception

The offline (as well as the online) algorithm has five main components as follows:

1) Raw Data Collection
2) Object Classification
3) Segmentation of Skeleton Tracking
4) Deriving Features from the Object and Segmentation Data
5) Classification of the Features using SVM

Data Collection

Before we can begin processing what is being perceived, the algorithm first needs to detect the raw data from its environment. Thankfully, the Kinect provides a great means by which to acquire the data we need. The open source Open NI interface, already built into ROS, was a wonderful launching point from which to build the data collection code. Our code collects 2 types of data: skeleton data and rgbd data. The latter is simply the color at each pixel with attached depth data at each pixel. This is basically a 3-dimensional raw image of the scene. Furthermore, I combined Hema's implementation with the ROS implementation to gather skeletal information. This skeletal information contains the location of major joints in the human body at any given frame. I was also able to successfully integrate Hema's visualization code too, so that while the Kinect is reading in data, the user sees what the Kinect sees, including depth data and the skeleton that is being tracked by the device.

Object Classification

Simply put, object classification is the piece missing from my project at the moment. Under Hema's original implementation, frames were tagged with bounding boxes that would allow the algorithm to provide temporal features for later classification; basically, it let the algorithm know where the object was and how it moved. This is because previous work in tracking 2D images has met limited success. Recently, Rudhir Gupta has been doing work to explore object tracking using 3D data. The hope is that by including depth information, we can accurately follow objects without needing human intervention. Due to time constraints on both sides, however, his project and my project were never integrated.

Segmentation

Given the immense amount of data, it was clear that we wouldn't need every single frame in order to compute the various features. Thus, the segmentation code splits the stream of data into manageable chunks, which are then characterized by several key frames.

Feature Generation

Once we have gathered the three pieces above, we need to massage the data into features which SVM can work on. Features are generated for each segment, as well as between segments to track movement and other temporal information. These features are basically the positions of the objects and people in the scene, as well as the relationships between the objects and people. For a detailed explanation, I strongly encourage looking at Hema Koppula's work, though unfortunately, much of it is currently in review.

The feature generation step is following by a number of scripts for massaging data, the most notable of which is a binning process which allows us to ensure homogeneity between the sampled frames within and between segments.

SVM
The final step is a Support Vector Machine. The support vector machine determines, for each segment, the affordance of every object as well as the action by all humans in the scene. The list of affordances is 'movable', 'stationary', 'reachable', 'pourable', 'pourto', 'containing', 'drinkable', 'openable', 'placeable', 'containable', and 'closeable' while the actions are 'reaching', 'moving', 'pouring', 'taking', 'eating', 'drinking', 'opening', 'placing', 'closing', and 'null'. The learning step is done offline, so the model is provided to the online algorithm. The svm classification is done using svm light using some scripts to ensure classifications occur per segment.

Results
As stated above, both the Kinect portion as well as the Segmentation, Feature Generation, and SVM parts of the code are integrated. In order to test the system, I created a simple program which I called 'ping' which sent the notifications the Segmentation code would expect. This created a cascade which would cause the rest of the system to run from segmentation all the way to classification. I have run the code base on five sample sets and present the results below.

	126141638	126141850	126143115	129112015	129112342	Averages
Action	50.00%	47.50%	40.91%	80.00%	58.82%	55.45%
Object 1	80.56%	72.50%	68.18%	90.00%	91.18%	
Object 2	80.56%	72.50%	77.27%	80.00%	82.35%	
Object 3	63.89%	77.50%		100.00%	94.12%	
Object 4				85.00%		
Object (average)	75.00%	74.17%	72.73%	88.75%	89.22%	79.97%
Segments	36	40	22	20	34	

The data requires a bit of context however. The object classification looks pretty good, but it's worth noting that the classifications are based on an offline system where bounding boxes are provided to help accurately position the object in time and space. It is also significant that the most common classification in many trials is stationary, which intuitively seems pretty easy to detect and doesn't give us terribly much information. That being said, roughly 80% is quite a good number.

Unfortunately, the action detection is a little less stellar. It is definitely worth noting that the algorithm produces a certainty value which was not factored into the above chart which represents how sure the algorithm is of its answer. To give an idea, on trial 126141850, one of the lowest scoring trials, the average certainty assigned to correct classifications was .729, whereas the average certainty for incorrect classifications was .617, which is not much lower, but is certainly indicative of a trend.

Finally, since this project was primarily about building infrastructure and less about improving the underlying algorithms, the fact that the infrastructure works is itself incredibly significant.

Example
Here we see a visualization of a correctly classified segment. Below is frame 346, the middle frame of segment 20 in activity 126141638. (In this example, there are 21 frames in this segment, which spans from frame 296 to 396.) The algorithm correctly classified the subject as pouring, the jug as pourable, the bowl as pourto, and the cereal as stationary.

Future Work
While a great foundation has been laid, there are definitely steps that could be made to improve the system. Initially, I had wanted to integrate both the object detection code as well as using the code's output to trigger a simple response in a ROS run robot. Thus, implementing both of these would be a great place to look. Furthermore, those parts which are implemented could be made more robust; for example, the binning script could be moved from MATLAB into C code, and the code could be made to better handle file structure exceptions and work around old runs.

Conclusion
Though not as far reaching as originally envisioned, the project does a great job bridging the gap between all of the different components. I personally felt it to be much more convenient to test everything once the pipeline had been set up and I could simply wait for all of the pieces to run one by one. Moreover, I have understated it a little in this paper, but it is important that the data collection code has also been moved into ROS; the only reason I was unable to integrate it in a more meaningful way is that it would be fairly meaningless to run the ROS code, do the object detection offline, and then restart the system to finish the segmentation, feature generation and classification. I believe this will provide a strong foundation for future work to integrate all the components necessary for robots to react in meaningful ways in our day to day lives.

References
Koppula, Hema, “Human Activity Learning using Object Affordances from RGB-D Videos.” *In Review.*