3D Point Cloud Object Classified for ROS

Kory Kraft[1], Austin Nicolai[2]

Abstract— **Many challenging robotics tasks require the ability to manipulate the local environment to complete a task. Often, due to the difficulty, these tasks are accomplished using shared autonomy or teleoperation. The need exists for robots to be able to automatically identify and manipulate the local environment on their own. We implemented a prototype object classifier module for ROS. To test with, we classifed four common manipulatives found in many homes.**

I. Introduction

Robotics is experiencing an upsurge in research, industry, and popular culture due to recent advances in basic robotic abilities. Driverless vehicles are on the road, and seem to be, in more ways than one, right around the corner. However, even with advances in software architecture, computing hardware, and mechanical hardware, there is still a large gap between what popular culture envisions robots doing and what robots can actually do today.

Similarly to computers, there exists tasks that robots can do better than humans, and tasks that humans can do better than robots. The ideal is to have fully autonomous robots operating in the world without the need for human intervention. There are numerous obstacles preventing that from being the current reality. In order to deploy robots in the real world as quickly as possible, to the benefit of humans, shared autonomy must be exploited to the fullest extent. Finding ways to leverage the advantages of both parties will lead to accelerations in both robot deployment and ability.

Simplifying and streamlining the robot-human interface while using what is readily available in current hardware and software will lead to a better, more fluid shared autonomy experience in less time. Robots are built to interact with the physical world at a level beyond most computer systems. To this end, they must sense the environment in order to interact with it. The Microsoft Kinect RGB-D camera is a low-cost solution for many computer vision tasks required by a robot. On the software side, the Robot Operating System (ROS), is a software system and set of libraries that currently handles basic information processing for robots in a hardware-independent manner.

What is missing between the two is a simple framework with a clean interface that unites the current hardware and software capabilities for the purposes of object classification, that will then be used for object manipulation during a shared autonomy session. To this end, we built a prototype program that utilized ROS-formatted data streams from a Microsoft Kinect along with the PyBrain machine learning package to classify four common manipulative objects: a standard light switch, a door-handle, a door-knob, and a drawer handle. These objects were classified using 3D point cloud information obtained via a Kinect sensor. Having a streamlined data to classification program will improve the current state of the art shared autonomy approach for object manipulation by removing a previously time-consuming step on the part of the human user.

*All authors are with the School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA.

[1]Corresponding author: `kraftko@onid.oregonstate.edu`

[2]Corresponding author: `nicolaia@onid.oregonstate.edu`

II. Related Work

A. Shared Autonomy

In the world of robots, there is a spectrum of autonomous function. On one end, there is a fully autonomous robot without the need for human intervention in the direction of basic tasks. Think, a driverless vehicles that only need to be told a destination, and then the rest is taken care of by the vehicle. On the other end is remote teleoperation that requires a human expert to constantly inform and direct the robot. Think, a scientist sitting a desktop using a joystick to control all the movements of a research submersible. Shared autonomy lies in the middle between these two extremes.

Shared autonomy requires that a robot be able to do some things on its own (e.g. raise its gripper hand or drive forward), but leaves room for a human in the loop to direct or give meaning to the robot's actions.

(wait for paper recommendations from Matt R.)

B. Object Classification

Austin could you write this section?

C. ROS

The Robot Operating System (ROS), which has some of its deepest roots in the Stanford AI group and former Willow Garage, has become the leader in open-source robotics software (ros website and paper). ROS is written largely in C++ and Python and runs ontop of Ubuntu.

It is used by hobbyists, researchers, and major industrial manufacturers, and has a large pool of active developers (ros website). The software library is vast, and due to the hardware agnostic nature of the system, ROS can run on many different robot platforms. ROS includes tools for basic robot sensing, movement, and simulators that allow programmers to test code before deploying it on a physical robot. Although ROS is guided by Open Source Robotics Foundation (OSRF), ROS relies on the larger community to maintain and extend its capabilities (http://www.osrfoundation.org/projects).

ROS includes packages for teleoperation, using the keyboard and mouse, or even operating the robot utilizing a Razer Hydra controller and Oculus Rift headset (surrogate website). ROS also includes MoveIt!, a path-planning and object manipulation software (http://moveit.ros.org/).

Given all of the benefits of ROS, we were somewhat surprised to find that there was not a streamlined, one-step program and interface to classify objects based on 3D point cloud information that could then be used in conjunction with a shared autonomy approach and MoveIt!.

D. Microsoft Kinect

The Microsoft Kinect was introduced to the marketplace in 2010 (CNET). Since then it has found its way into research labs as well living rooms. It provides RGB-D sensing for less than 100 dollars. It has become a standard RGB-D sensing device due to its low price point, extensive documentation, and large community of adopters. The original Kinect is limited to 0.8 - 4.0m distance range and 57° horizontal and 43° vertical field-of-view.

The device is not flawless; it has a random error in the depth axis ranging from a few mm to 4 cm in a non-uniform distribution [1], [2]. Kinect v2 has been released that provides 3x better depth fidelity and improved small object visualization compared to the original Kinect (http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx).. We used the original Kinect for our project that will provide a baseline for manipulative classification, which will only be improved upon for the future.

Should we say anything about using it only indoors here due to the tech restrictions*?

E. PyBrain

PyBrain describes itselft as "a modular Machine Learning Library for Python." The open-source module focuses on neural networks and offers ways to quickly build and train neural networks, even for beginning programmers. PyBrain includes Python bindings to use other popular products like LibSvm. (https://github.com/pybrain/pybrain,)

PyBrain works well alongside ROS, and depends heavily upon SciPy which is included with rospy.

III. Methods

A. Objects

We decided to classify four common household objects: a standard light switch, a door-handle, a door-knob, and a drawer handle. These objects were selected for their ubiquity in many residences and living environments. (Include picture of the 4 obj here)

B. Data Capture

We chose to capture the data in such a way that can be easily adapted to a near-future operation environment in which a robot will be instructed via a human trainer to "learn" an image that the human is pointing to. Only capturing 5 seconds of data ensures that the capture time will be more seamless, and will not result in the human trainer waiting for enormous amounts of time for the robot to "take in" the scene.

We captured data utilizing the Microsoft Kinect and subscribing to the related ROS topic. The data was recorded into a rosbag format. Even though we did this via a stationary laptop, we were running a virtual instance of ROS to issue the commands. This means that any robot running ROS has access to the same commands and data format. (Include figures from slide here)

For ease of operability, and to have standardized image distances during the first iteration, we decided to detach the Kinect from a robot and place it onto a tripod, with the specific object in the middle of the Kinect's field-of-view. We kept the Kinect at the same height and distance (0.8 m) while capturing the data. Approximately 5 seconds of data were captured for each manipulative (and blank backdrop) at each viewing angle. The three viewing angles, in terms of the horizontal viewing axis were: 90 degrees, and $\pm$ 20 degrees.

The Kinect sensor provides both point cloud and RGB data at resolution of 640 x 480. For every one second of recording, approximately 10 depth registered point clouds were stored. ***Should we say how much data they took up??***

C. Data Parsing

We decomposed the captured image into a 640 x 480 2d color matrix and a 640 x 480 normalized 2d greyscale matrix, using values 0-255. This was done in Python utilizing ROS-pcl, a ROS interface to the open-source Point Cloud Library (PCL), and self-written code (do you want to mention more here? I think we can just show the picture from the slides) (citation to pcl and ros-pcl).

The depth matrix was then cropped to be a matrix of size 192 x 144 (......Austin....?)

D. Learning

The data was randomly split into %80 training data and %20 testing data. The neural network, which relied upon softmax activation function, consisted of 27,648 inputs, one for each value in the normalized depth matrix. There were 25 hidden neurons, and 5 outputs. The outputs were the class labels for the 4 knobs plus the blank backdrop. The neural network was trained using backpropagation for 1000 epochs. Training time lasted approximately 8 hours.

IV. Results

((Include figures from slide..or updated ones if you want to make them)) Overall, classification for the four objects was very high, much greater than random. As can be seen in figure?, if a knob was misclassified, it was classified as a lightswitch. This is probably due to the

Even though the training time on the neural network was longer than we hoped, it was of such duration that the robot could "train" on a new set of objects could take place while a person was away at work, or while a person slept.

V. Future Work

The analysis of both our domain and results presents several avenues for further research.

-package into ros module -tie into shared automony interface.. -train on more objects -pca or some sort of dimension reduction -try against different learning algorithms

A. *Buildit!*

bam!

References

[1] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[2] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. IEEE, 2012, pp. 524–530.