TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

A STUDY OF 3D OBJECT DETECTION AND CLASSIFICATION

by

Brandon Maas

A thesis

Presented to the faculty of

Towson University

in partial fulfillment

of the requirements for the degree

Master of Science

Department of Computer and Information Science

Towson University
Towson, Maryland 21252

May, 2016

TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

THESIS APPROVAL PAGE

This is to certify that the thesis prepared by Brandon Maas

entitled A study of 3D object Detection and classification

has been approved by the thesis committee as satisfactorily completing the thesis requirements for the degree Master of Science
(for example, Master of Science)

Chairperson, Thesis Committee Signature — Date: 4/28/2016

Committee Member Signature — Date: April 28, 2016

Committee Member Signature — Date: 4/28/2016

Committee Member Signature — Date: 4/29/2016

Committee Member Signature (Dept Chair) — Date: 4/28/16

Dean of Graduate Studies — Date: 5-2-16

Acknowledgements

I would like to extend my sincere gratitude to my faculty advisor, Dr. Darush Davani, for his guidance and encouragement over the past three years. I also want to thank Frederick Ackers for his assistance in leading our team during the DARPA Robotics Challenge, and his insight into artificial intelligence and computer vision as related to my thesis research. Finally, I want to thank my fiancé, Megan Randolph, for supporting me throughout my master's degree work.

Abstract

A STUDY OF 3D OBJECT DETECTION AND CLASSIFICATION

Brandon Maas

This work intends to provide a vision and artificially intelligent recognition system design that can be used in a robotics platform for use in the virtual robotics competition as well as any platform requiring advanced visual processing. This work explores the DARPA robotics competition and analyzes the requirements of each task in the competition. These requirements set a good basis for developing requirements for visual object recognition system. Next we research a number of technologies that would be targeted for integrating into this system such as the Atlas sensor suite, the robotics operating system, Gazebo simulator, and the point cloud library. The system proposed expands on an existing system proposed by Bedkowski et al. and adds the ability to recognize more intricate, non-flat objects through the use of the point cloud library and robotics operating system.

Table of Contents

List of Figures

Introduction

DARPA Robotics Challenge Overview

The DARPA Robotics Challenge (DRC) is competition that is open to the public with the main goal of fostering robotics innovation and setting a line in the sand to indicate the level of progress reached by the robotics community. The competition theme centered on the development of robotic systems capable of assisting humans in responding to natural and man-made disasters. To that effect, the challenge consists of three events that place equal emphasis on software and hardware. [1]

The first event was the Virtual Robotics Challenge (VRC). This event tested a team's software ability to effectively guide a simulated robot, the ATLAS by Boston Dynamics, through three sample tasks in a virtual environment. The first task entailed programming the robot to manipulate its bipedal form to traverse rugged terrain. The second task entailed dexterous manipulation of a fire hose by grasping the hose nozzle, attaching it to a valve, and opening said valve. The third and final task involved guiding the bipedal into a vehicle, drive the vehicle through a course, and exit the vehicle. [2]

The second event, the DRC trials, took qualifying teams from the VRC event and from Tracks A and D (Teams that had already possessed robotic platforms) and had them guide their robots through eight individual, physical, and more complex tasks that tested mobility, manipulation, dexterity, perception, and operator control mechanisms. The tasks were as follows: [3, pp. 5-6]

1. Drive a utility vehicle at the site.
2. Travel dismounted across rubble.

3. Remove debris blocking an entry way.
4. Open a door and enter a building.
5. Climb a ladder and traverse an industrial walkway.
6. Use a tool to break through a concrete panel.
7. Locate and close a valve near a leaking pipe.
8. Replace a component such as cooling pump.

The third and final event, the DRC finals, took place with significantly increased challenges and task adjustments. The course, pictured in Figure 1, was simplified significantly in order to make the 60 minute time limit a realistic goal for the participating teams; driving courses would be shorter, only one door would be encountered, and instead of a steep flight of stairs, a gentle slope of a few stairs would be used. However, in an effort to simulate a real disaster scenario three new requirements would be levied against the teams and their robotic platforms. They would have to provide a platform that was self-contained in terms of power source i.e. no power supply from an external source. Communications with the robotic platform would undergo random and significant interference. The robot would have to sustain functionality throughout the interference and, in the interest of time, remain autonomous towards completion of the task in progress. The third requirement would prevent the teams from using a belay of any kind throughout the course. This would increase the likelihood of falling and with it, catastrophic system failure. [4]

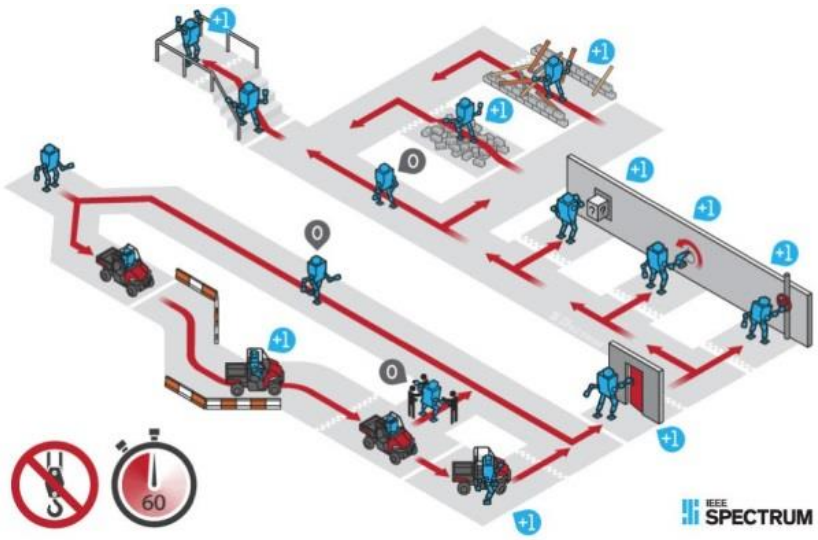

Figure 1: DRC Finals Course [5]

It is important to fully understand how computer aided vision can play a role in assisting both the human operator and the robot itself. The human operator must have situational awareness in order to execute rapid and informed decisions, many of which undoubtedly mean the difference between life and death in a disaster scenario. The robot itself would, ideally, be capable of detecting movement and identification of stationary objects in the environment. This information could be used to inform the operator of threats in the environment, remaining tasks to be completed, and even identifying humans in need of rescue. In the event of communication interference or total loss communications, the robot will need to carry on with its tasks. It is conceivable that a mission plan, from which a robot would follow in the event of communications loss with the operator, would contain visual targets. This places critical value on computer aided vision to provide the robot with the level of autonomy needed to complete the mission. As it relates to the DRC final events, here is how computer aided vision would aid in completing each task which ultimately sets the basis for requirements for the visual system described in this thesis. Not all robotic abilities are necessarily required to complete the task

since a remote operator will be maneuvering the robot, however, in the interest of maximizing situational awareness, desired capabilities will still be listed.

Task 1: Drive a utility vehicle at the site

- The utility vehicle must be identified as a utility vehicle capable of being driven.
- In order to enter the vehicle, the robot must of course be able to fit into the vehicle. Equipped with its own mechanical specifications, analysis of vehicle dimensions should reveal if the vehicle can be driven.
- Vehicle operation will require identification of key vehicle components needed for driving.
- While driving, the robot must identify a route or routes to the destination assuming the terrain was determined to be passable via vehicle.
- If the route is not passable by vehicle then the robot will attempt to follow the route on foot.
- Regardless if traveling on foot or vehicle, potential obstructions and hazards must be identified (e.g. fire, rugged terrain, path blocked, etc).

Task 2: Exit the vehicle

- Once the destination has been reached the robot must identify a safe location to exit the vehicle.
- The robot must identify surfaces on both the ground and the vehicle with which to hold on to or step on for support.

Task 3: Travel dismounted to and through the entry way

- The robot must identify the next objective which, according to either mission plan or remote operator, would be a door. It must then ascertain the existence of a mechanism with which to open the door and then grasp, manipulate, and push the door open via said mechanism.
- The robot must determine that the entryway is traversable after opening the door and then move through it.

Task 4: Locate and close a valve near a leaking pipe

- The robot must be able to identify that a pipe is indeed leaking a liquid or a gas. Then it must be -able to trace the length of the pipe and identify a valve that it is connected to.
- Once the valve has been detected and identified the robot must approach the valve, grasp it, and rotate it closed.

Task 5: Use a tool to cut through a drywall panel

- The robot must identify any tools present at the site that are also suitable for a cutting task.
- The robot must grasp tool correctly such that it can be operated.
- The robot must identify a black circle and subsequently a sufficient radius length from the center of the circle in order to identify an initial point for beginning the cut.
- The robot must guide the cutting tool from the starting point and complete a circular cut around the black circle.

Task 6: Surprise task

- Identify all objects in the area.

- Determine if any objects can be recognized as devices previously encountered.
- If the object can be opened in some way then position self closer to the object and attempt to open.
- In the event that the object is recognized as a button then proceed to push the button.

Task 7: Traverse rubble or debris laden terrain

- Identify obstacles in the way that are obstructing the path to the destination.
- Identify a path with the least resistance in the path, possibly by determining the path with the most horizontal surfaces.

Task 8: Travel up a stairwell

- Identify a stairwell and approach, leaving enough space to begin traversing the stairs.
- Repeatedly identify horizontal surfaces of the stairs as elevation increase to continue negotiating the stair climb.
- Identify the final step and look for additional horizontal surface with which to move out onto to gain stable footing.

These actions outlined previously may appear trivial and mundane but that is of course only true to a human being, or at least those who have graduated into the developmental stages where this is all enacted naturally. Implementation of a system capable of performing a fraction of the requirements listed above is far from trivial. Each action that we take for granted must be decomposed into simple and fundamental procedures in order to understand the problem. Now, after having completed a brief collection of requirements, we need to identify the capabilities of the Atlas robot's sensor suite as well as any additional sensors that may be deemed necessary.

We will also explore the various computational solutions for utilizing the output provided by the sensor suite.

Background Research

Atlas Sensor Suite

The Atlas robot was developed for the DRC by Boston Dynamics and funded by DARPA. The robot was developed with the goal of advancing disaster response robotics. While the robot was largely constructed by Boston Dynamics, two sets of hands were built by iRobot and Sandia National Labs, and the sensor suite or head (Figure 2) of the Atlas was built by Carnegie Robotics. [6] [7]

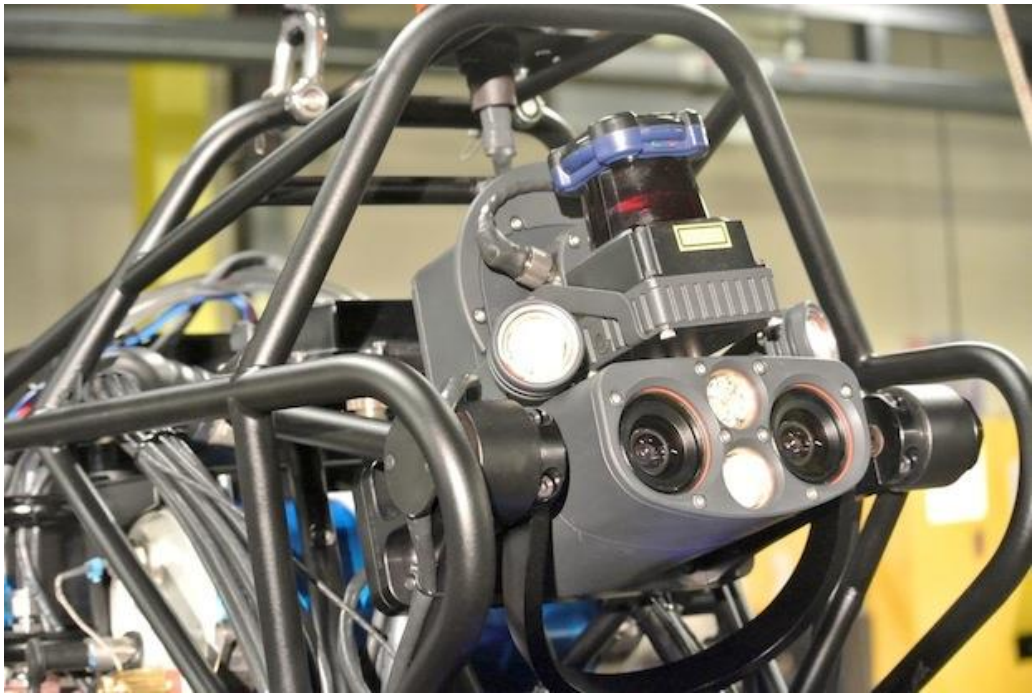

Figure 2: Atlas Head Sensor Suite [6]

The Atlas head, Carnegie Robotics MultiSense SL, is a tri-modal (laser, 3D stereo, video), high-resolution, high data rate, and high accuracy 3D range sensor. It is capable of facilitating numerous robotics applications including autonomous vehicles, 3D mapping, and workspace understanding, just to name a few. The head's processing capabilities are augmented with a low

power FPGA which is coupled with a gigabit Ethernet interface making it capable of outputting massive amounts of preprocessed visual data. The head consists of two primary sensors, the Hokuyo UTM-30LX-EW laser and a MultiSense S7 high resolution stereo camera. [8]

The Hokuyo UTM-30LX-EW, referred to as simply the Hokuyo, uses a laser source to scan a 270^0 semicircular field (Figure 3). The Hokuyo works by basically measuring the distance to any objects in its range (approx. 30m) during each angular step. The measurement data and angular step are then communicated over gigabit Ethernet and formatted into a proprietary data transfer protocol. The Hokuyo is mounted on a spindle for axial rotation at a user specified speed. That advantage, given that the Hokuyo laser itself provides two dimension scans, is in being able to provide three dimensional scans as transformed by odometry simultaneously received by the axial rotational mechanism, the spindle encoder. [9]

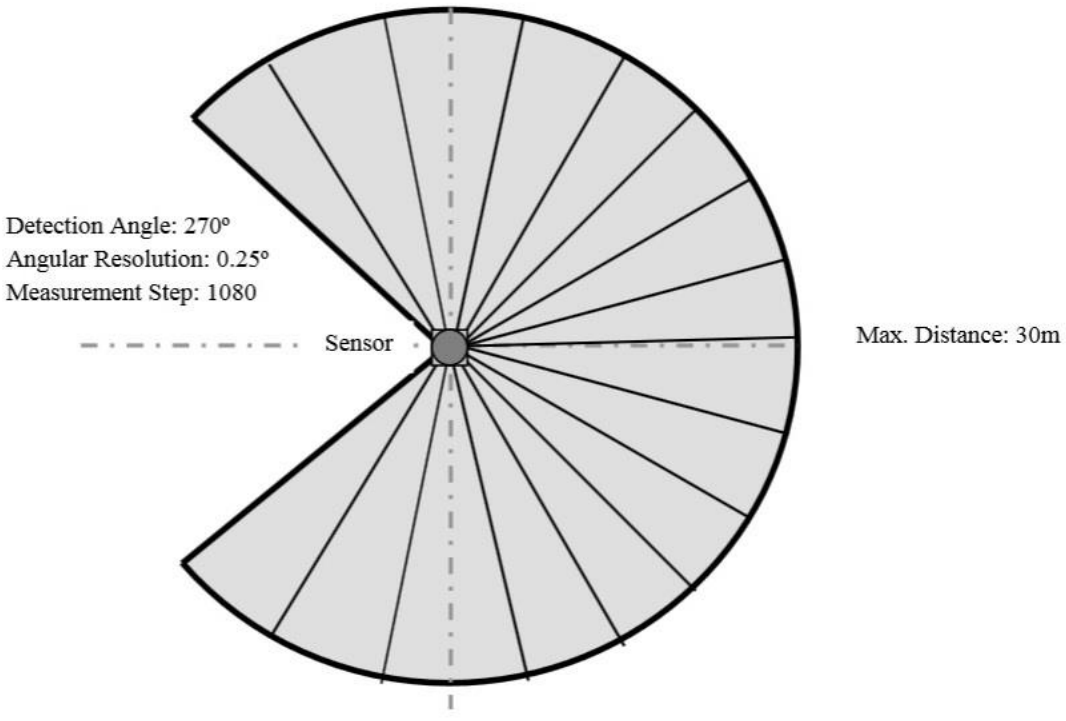

Figure 3: Hokuyo Sensor Scan Range [9]

The MultiSense S7 is a stereo camera which is a camera with two or more lenses and sensing elements which are used to mimic human depth perception. The S7 is equipped with infrared lighting for illuminating low light environments. Human-like perception is implemented within the processing unit of the S7 using the semi-global-matching algorithm. [10] [8]

Stereo matching is a technique for finding corresponding pixels in a pair of images such as those generated by a stereo camera. This allows us to reconstruct a three dimensional depth image of the original scene via triangulation and using known intrinsic and extrinsic orientation of the cameras. Semi-global-matching is a computationally efficient algorithm that allows the MultiSense S7 to generate depth or disparity images (Figure 4) on the sensor's processing hardware which removes the need for external processing of raw unfiltered and unrectified data and freeing external platforms to perform other post processing functions. [11] [12]

It is worth noting that disparity or depth images are expressed in the form of 3D point cloud data. That is to say each pixel is formatted to contain, at minimum, a 3-dimensional vector as well as color values of either RGBA or HSV formats.

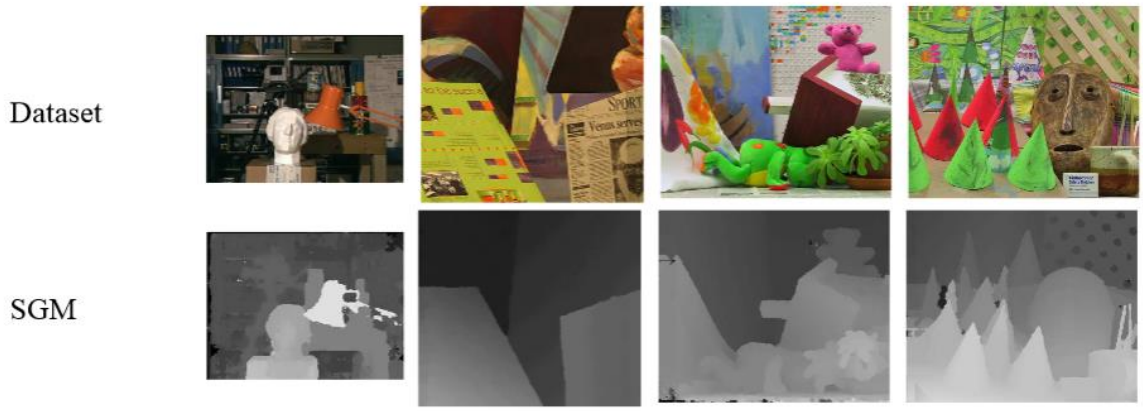

Figure 4: Depiction of input data set and application of semi-global-matching to generate disparity images. [12]

Robotics Operating System

The Robotics Operating System (ROS) is a quasi-operating system for robotic systems which, more appropriately, functions as middleware on top of an existing operating system. To date the only supported operating system for ROS is Linux. However, much like an operating system, ROS provides services including hardware abstraction, low-level device control, interprocess communication mechanisms, and package management. At its core, ROS was designed to communicate via a publisher/subscriber model where information is multicast or published to all recipient processes or subscribers. ROS also contains a software suite consisting of tools and libraries for running your code across any number of systems. [13]

The ROS runtime model is a peer-to-peer graph (Figure 5) or network of processes, potentially across physically separate processing nodes, which communicate via the ROS communication infrastructure. The primary components that we will be interacting with and developing for our visual system are Nodes, Master, Parameter Server, Messages, Topics, Services, and Bags. [14]

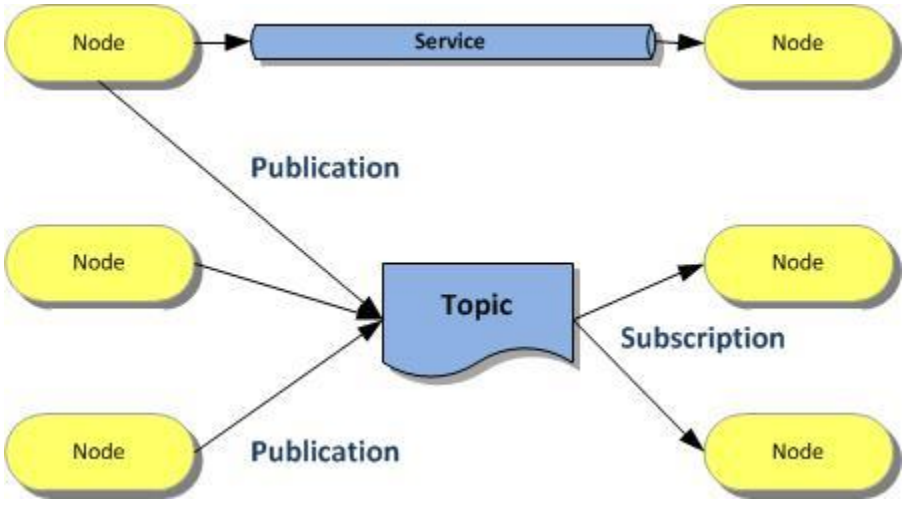

Figure 5: High level diagram of ROS constructs [15]

Nodes are processes that perform the computational work of the robotics application. They are combined together to form a graph and communicate with one another via topics, services, and/or the parameter server. These nodes are meant to operate at a fine-grained scale in order to encapsulate the minimum necessary functionality to complete a task. This has the added benefit of fault tolerance where if any node crashes it does not bring down the entire robotics application. Much of the visual system design proposed later in this paper will center on a collection of new and existing nodes. [16]

The master provides naming and registration services to the nodes in the robotics application. It also tracks publishers and subscribers to topics as well as services. The primary role of the master is to allow nodes to register themselves with ROS and also to allow nodes to find other nodes. Finally, once registration has taken place the master will establish peer-to-peer communications between the nodes. For this reason, this master is often the first running process with the ROS infrastructure. [17]

The parameter server, which is actually served by the master, provides a shared, multi-variate dictionary that is accessible via network APIs. The parameter server is intended to provide nodes with the ability to store and retrieve parameters during runtime. It is, however, not designed for high performance and so it is best used to store configuration parameters that accessible by the robotics application as a whole. That said, the parameter server only supports 32-bit integers, booleans, strings, doubles, iso8601 dates, lists, base64 encoded binary data, and dictionaries. [18]

Topics are named buses through which nodes send and receive messages. Topics have anonymous publish/subscribe semantics which means that nodes do not know who they are

sending a message to or receiving a message from. If a node requires the information in a topic then they subscribe to it. If a node will instead send information in a topic then they will publish it. There is practically no limit to the number of publishers and subscribers of a given topic. The visual system design proposed later will contain several new and existing topics. [19]

Messages are data structures that are composed of typed fields. Nodes communicate with each other by publishing messages to topics. [20]

Should the need arise for request/response messaging, which often does occur, services provide a means for this two way communication between ROS nodes. [21]

Bags are created by a ROS tool which subscribe to one or more topics and store the serialized message data in a file as it is received. These bag files are able to be played back to the topic(s) of the ROS application from where they originated. This has the benefit of collecting data from a simulated or real source and allow to play them back later for a variety of purposes such as when equipment is unavailable or during testing. [22]

ROS is an immensely useful framework for the development of robotic systems. The visual system design proposed later will be designed to use the existing ROS API for the MultiSense SL provided by Carnegie Robotics.

Gazebo and DRC Simulator

Gazebo is the simulation framework chosen for use during the Virtual Robotics Challenge event. Gazebo is a powerful simulation framework that supports a number of physics engines, 3d rendering engine, sensor and noise data generation, includes robot models, and allow for remote hosting for server side simulation processing which in effect decouples the rendering system from simulation processing. [23] [24]

For the Virtual Robotics Challenge, DARPA, along with OSRF, provides a wealth of resources for our usage in the competition. The simulation resources provided were entire disaster environment models, an Atlas model complete with physics characteristics and simulated sensors, as well as ROS nodes and services for interacting with the robot and its subsystems. [23]

3D Object Recognition Algorithms

Segmentation

Point cloud segmentation is the process of classifying point clouds into one or more homogeneous regions. This is a challenging problem due to high levels of redundancies (many similar features exist in a scene which may be part of a completely separate object), uneven sampling density which might be due to sensor noise or interference, and the lack of any notion of structure in the point cloud data. [25] The notion of the detection of primitive shapes is a common problem with many proposed solutions in the computer vision community. One of the most important algorithms used to perform point cloud segmentation is the Random Sample Consensus or RANSAC.

RANSAC was proposed by Fischler et al., in 1981. It is an iterative method that is used to predict as, accurately as possible, parameters of a mathematical model from dataset containing statistical outliers. The algorithm assumes that all of the data we are looking at is comprised of both inliers and outliers. The inliers can be explained by the model with a specific set of parameter values, while the outliers do not fit the model in any circumstance. An additional and essential assumption is that a procedure that can optimally estimate said parameters of the chosen model from the data is available for use in RANSAC. [26] The RANSAC algorithm functions as follows.

First we define the inputs to the RANSAC algorithm which are the following.

- A set of observed data values from a sensor or simulated data set. [26]
- A parameterized mathematical model that can be fitted or applied to the input data set. [26]
- Confidence parameters are supplied that describe the likelihood of outliers and being present among others. [26]

With these inputs the RANSAC algorithm iterates over a randomly selected subset of the original dataset. For each iteration, RANSAC, at first, considers all data from the subset as hypothetical inliers. Each point of the random subset is tested as follows:

1. The specified model is fitted to the hypothetical inliers.
2. All other data in the subset is then tested against the fitted model. If a point fits in the estimated model, then that point is also considered a hypothetical inlier or rather becomes part of the consensus set.
3. The estimated model is considered good if a suitable number of points have been determined to be part of the consensus set.
4. The good model is now re-estimated from all hypothetical inliers. This is done because the model was only initially estimated with a random subset of the full data set.
5. The model is finally evaluated by calculating the error of the inliers relative to the model.

This algorithm is repeated a number of times and with the number of repetitions being determined by the user. Each repetition will either produce a model that is either rejected because too few points are part of the consensus set or a model was accepted because it was refined and was part of a larger consensus set than the previous. Figure 6 shows a 2D dataset (a). The result

(b) of the RANSAC algorithm, in this case, would produce a model that describes the line which is shown superimposed on the inliers. The outliers are shown in red and inliers are shown in blue. [26]

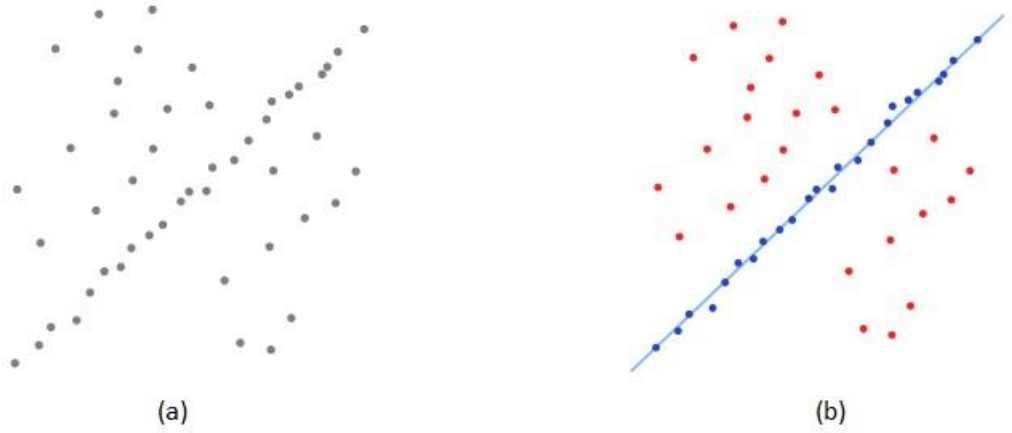

Figure 6: 2D RANSAC - Line Model

It is worth noting that the output from the MultiSense SL will be an organized point cloud as opposed to an unorganized point cloud. RANSAC algorithms have been largely targeted at segmentation of unorganized datasets, i.e. datasets without structure relative to each point. An organized point cloud has the form of an organized image-like structure where the data is split into rows and columns. The advantage of organized point cloud datasets is that relationships between points and their neighbors are more clearly identified and nearest neighbor operations are performed far more efficiently. [27] Since we will be processing organized point clouds, I have researched the work of Trevor, et al., [28] that takes advantage of organized point cloud data.

The work proposed by Trevor, et al., [28] is a component based approach to point cloud segmentation that processes an organized point cloud only. As stated earlier, an organized point

cloud possesses more efficient structure and is output from the MultiSense SL sensor in this form as opposed to unorganized. The method is described as follows. The organized point cloud is represented by $\boldsymbol{P}$ and points referenced by $\boldsymbol{P}(x, y)$ and neighboring points expressed as $\boldsymbol{P}(x - 1, y)$ and $\boldsymbol{P}(x, y - 1)$. Since the point cloud is organized, all nearest neighbor accesses are performed in constant time. The algorithm functions by segmenting an organized point cloud $\boldsymbol{P}$ into a group of segments, $\boldsymbol{S}$. This grouping occurs by creating an integer label $\boldsymbol{L}$ for each point in $\boldsymbol{P}$. The label for a point $\boldsymbol{P}(x, y)$ is given by $\boldsymbol{L}(x, y)$. Two points will then have the same label $\boldsymbol{L}(x, y)$ if $\boldsymbol{P}(x_1, y_1) \in \boldsymbol{S_i}$ and $\boldsymbol{P}(x_2, y_2) \in \boldsymbol{S_i}$. Now in order to apply a label we must compare the two points in some way. This is done with a comparator function shown below.

$$\boldsymbol{C}\big(\boldsymbol{P}(x_1, y_1), \boldsymbol{P}(x_2, y_2)\big) = \begin{cases} true\ if\ similar \\ false\ if\ not \end{cases}$$

Equation 1: Segmentation Comparator

If true, then both points will receive the same label. However, if false, then the largest assigned integer label will be increase by one and the unlabeled point will receive that new label. The comparator function $\boldsymbol{C}$ is chosen based on what type of segmentation function we are trying to perform (e.g. planer, cubic, spheroid, etc).

Point Pair Feature & Complex Shape Histogram

The system implemented by (Bedkowski, Majek, Maslowski, Kaczmarek) [29] will be used as a basis for designing a vision system in the Atlas capable of more advanced object recognition capabilities. Their approach was as follows.

Critical to 3D object recognition is the ability to extrapolate points of interest or features of an object or scene. This function is usually accomplished via feature point algorithms such as

Normal Aligned Radial Feature (NARF), Point Feature Histogram (PFH), Fast Point Feature Histogram (FPFH), and others. The authors in [29] had decided to use an improved but simplified Point Pair Feature (PPF) descriptor known as Complex Shape Histogram (CSH). CSH is simplified by categorizing points into two semantic classes {flat, not flat}. The point is classified as a flat shape when its nearest neighbors can be approximated by a local plane. This has the benefit of identifying planar areas in a point cloud very quickly, but at the same time, this is a significant limitation if you need to recognize nonplanar objects. [29]

CSH considers all pairs of points from a point cloud dataset. The features considered are the Euclidean distance between points (Equation 2), the combination of shape features {flat, not flat}, {flat, flat}, {not flat, not flat}, and the angle between normal vectors computed for pairs of points (Equation 4). [29]

$$d(u,v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + (u_3 - v_3)^2}$$

Equation 2: Euclidean distance between u and v

$$\boldsymbol{u} \times \boldsymbol{v} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$$

Equation 3: Cross product for calculating surface normal

$$\angle(u,v) = \tan^{-1}\left(\frac{\|u \times v\|}{u \cdot v}\right)$$

Equation 4: Angle between normal vectors u and v

The point pair feature **F**, which describes the relative position of two oriented points, is defined as the following 4D tuple: [30]

$$F(\boldsymbol{m_1}, \boldsymbol{m_2}) = \left(\|\boldsymbol{d}\|_2, \angle(\boldsymbol{n_1}, \boldsymbol{d}), \angle(\boldsymbol{n_2}, \boldsymbol{d}), \angle(\boldsymbol{n_1}, \boldsymbol{n_2})\right)$$

Equation 5: 4D Point Pair Feature

where $(\boldsymbol{m_1}, \boldsymbol{m_2})$ is the point pair, $\boldsymbol{d} = \boldsymbol{m_2} - \boldsymbol{m_1}$, and normal vectors of $\boldsymbol{m_1}, \boldsymbol{m_2}$ are $\boldsymbol{n_1}, \boldsymbol{n_2}$. The feature **F** is used create a global model description which consists of all model point pair features and represents a mapping from the point pair feature space to the model. [30]

The global model description, or rather the mapping of point pair feature space to the model, can be described formally by the following equation:

$$L: \mathbb{Z}^4 \rightarrow A \subset M^2$$

Equation 6 Mapping of PPF space to model [30]

Where the four dimensional point pair features, defined by equation 4, are mapped to set **A** of all pairs $(\boldsymbol{m_i}, \boldsymbol{m_j}) \in \boldsymbol{M^2}$ that define an equal feature vector. The global model description is implemented as a hash table which is indexed by 4D sample feature **F**. All model features $\boldsymbol{F_m}(\boldsymbol{m_i}, \boldsymbol{m_j})$ that are similar to a given scene feature $\boldsymbol{F_s}(\boldsymbol{s_i}, \boldsymbol{s_j})$ can then be searched in $O(\mathbf{c})$ time by using $\boldsymbol{F_s}$ as a key into the hash table. [30]

The next part of this method takes an arbitrary reference point $\boldsymbol{s_r} \in \boldsymbol{S}$ from the scene and assumes that it lies on the object that we are trying to detect. If this assumption is correct then there is a point $\boldsymbol{m_r} \in \boldsymbol{M}$ that corresponds to $\boldsymbol{s_r}$. After aligning points $\boldsymbol{m_r}$and $\boldsymbol{s_r}$ and their normal vectors, the object can be rotated around the normal vector of $\boldsymbol{s_r}$ to align the model to the scene. This motion from the model space into the scene space is described by a point on the model and a rotation angle $(\boldsymbol{m_r}, \boldsymbol{\alpha})$. This pair is known as the local coordinates with respect to $\boldsymbol{s_r}$. [30]

Now a model point pair $(\boldsymbol{m_r}, \boldsymbol{m_i}) \in \boldsymbol{M^2}$ is aligned to a scene point pair $(\boldsymbol{s_r}, \boldsymbol{s_i}) \in \boldsymbol{S^2}$ according to the closeness in similarity of their feature vector **F**. The transformation from local model coordinates to scene coordinates is expressed by

$$\boldsymbol{s_i} = \boldsymbol{T_{s\to g}^{-1} R_x(\alpha) T_{m\to g} m_i}$$

Equation 7: Transform from local model to scene coordinates. [30]

Finally, a voting scheme is utilized to determine the optimal local coordinates such that the number of points in the scene that lie on the model is maximized. This is done via a method similar to the Generalized Hough Transform. As shown in figure 7, a reference point $\boldsymbol{s_r}$ is paired with every other point $\boldsymbol{s_i}$ in the scene. The point pair feature **F** is calculated for $(\boldsymbol{s_r}, \boldsymbol{s_i})$. The point pair feature **F** is used to index into the global model description hash table which returns a set of point pairs that have similar distance and orientation. Next, for each point pair on the model matched to the point pair in the scene, the local coordinate $\boldsymbol{\alpha}$ is calculated by solving equation 6. A vote is then cast for the local coordinate $(\boldsymbol{m_i}, \boldsymbol{\alpha})$ which is simply the largest accumulator row. From there, all rows within a configured range of the optimal coordinate's row are chosen as shown in figure 7. [30]

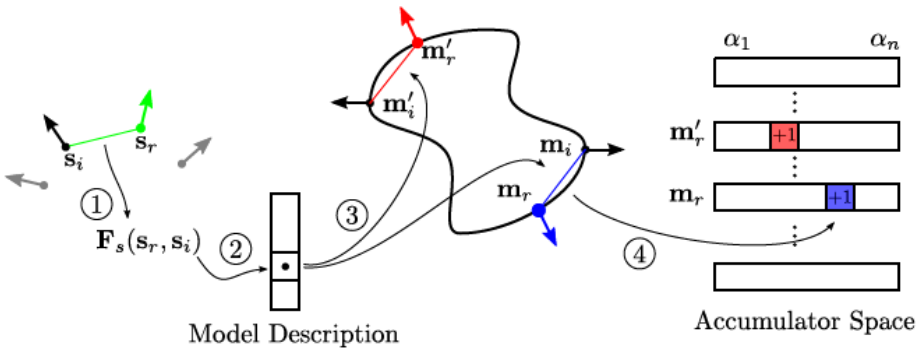

Figure 7: Point Pair Features - Voting Scheme [30]

Coming back to complex shape histogram (CSH) proposed by [29], CSH improves upon the more general Point Pair Feature (PPF) method by approximating a local plane within a predefined range of nearest neighbors. [29]

Classification

Support Vector Machines

Support Vector Machines (SVM) were first described by Vapnik et al., in 1992 and has quickly cemented itself as a powerful algorithm applied to the problem of classification in the larger context of machine learning. First, I will explain how the SVM is applied to two dimensional and linearly separable image data and expand upon this explanation with an approach to classification of three dimensional image data.

Let's say that we have $\boldsymbol{L}$ training points. Each input of $\boldsymbol{L}$, denoted as $\boldsymbol{x_i}$, has $\boldsymbol{D}$ attributes i.e. has $\boldsymbol{D}$ dimensionality. Each input is considered to be in one of two classes $\boldsymbol{y_i = -1}$ *or* $\boldsymbol{1}$. That said, our training data is of the form

$$\{x_i, y_i\} \quad where \quad i = 1 \dots L, y_i \in \{-1,1\}, x \in \Re^D$$

Equation 8: Formal expression of 2D training data [31]

We assume that the data is linearly separable. This means that if we were to graph $\boldsymbol{L}$ training points, i.e. a graph depicting $\boldsymbol{x_i}$ vs $\boldsymbol{x_j}$, we would be able to draw a line that separates the two classes when $\boldsymbol{D} = 2$ and a hyperplane on graphs of $\boldsymbol{x_1, x_2 \dots x_D}$ for when $\boldsymbol{D} > 2$. [31]

A hyperplane is described by the equation:

$$\boldsymbol{w \cdot x + b = 0}$$

Equation 9: Equation of a hyperplane

where $\boldsymbol{w}$ is normal to the hyperplane and $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin. In other words, a hyperplane is the subspace of one dimension less than the actual space. The

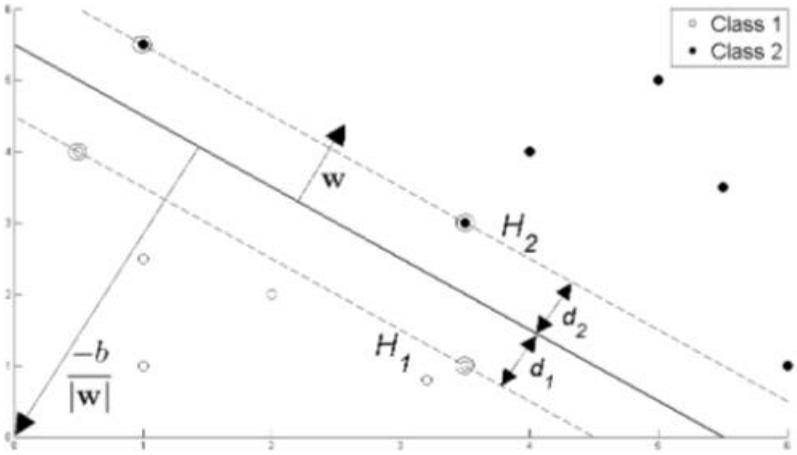

Figure 8: Graph of training data with linearly separating hyperplane [31]

graph shown in figure 8 depicts a hyperplane, a line in this case, that linearly separates the training data into two classes. The support vector(s) are those data points, namely those points on **H1** and **H2**, that are closest to the separating hyperplane and the support vector machine works to orient the hyperplane such that it is equidistant from **H1** and **H2** i.e. the hyperplane is as far as possible from the closest points of both classes. The support vector machine accomplishes this by selecting variables $\boldsymbol{w}$ and $\boldsymbol{b}$ so that training data can described by the following equations:

$$x_i \cdot w + b \geq 1 \; for \; y_i = 1$$

$$x_i \cdot w + b \leq -1 \; for \; y_i = -1$$

Equation 10: Training data description [31]

From these equations we define H1 and H2 where the support vectors reside, that is, the points that lie closest to the hyperplane. The equations are defined below:

$$x_i \cdot w + b = 1 \; for \; H_1$$

$$x_i \cdot w + b = -1 \; for \; H_2$$

Equation 11: Definition of planes H1 and H2 [31]

Referring back to figure 8, the distance from **H1** to the hyperplane is indicated by $\boldsymbol{d_1}$ and the distance from **H2** to the hyperplane is indicated by $\boldsymbol{d_2}$. Since the hyperplane is intended to be positioned equidistantly between **H1** and **H2** it follows that $\boldsymbol{d_1 = d_2}$. This is known as the SVM's margin. The SVM algorithm is therefore an exercise in optimization in determining the maximum margin. [31]

Vision System Design

The vision system design proposed in this document expands upon the capabilities proposed in the system designed by [29] depicted in figure 8.

Existing System

The existing system works by processing output point clouds from a sensor source. The sensor source may either be a real hardware device, such as the MultiSense SL, or a simulation source which conforms to any of the ROS topic structures provided by Carnegie Robotics and included

in the appendix. The output point cloud is preprocessed with a segmentation algorithm to extract points from a scene. Segmentation functions by removing the ground plane and surrounding obstacles. Segmentation is also implemented within the programmable logic of the MultiSense SL FPGA which allows for rapid preprocessing of sensor data. [29]

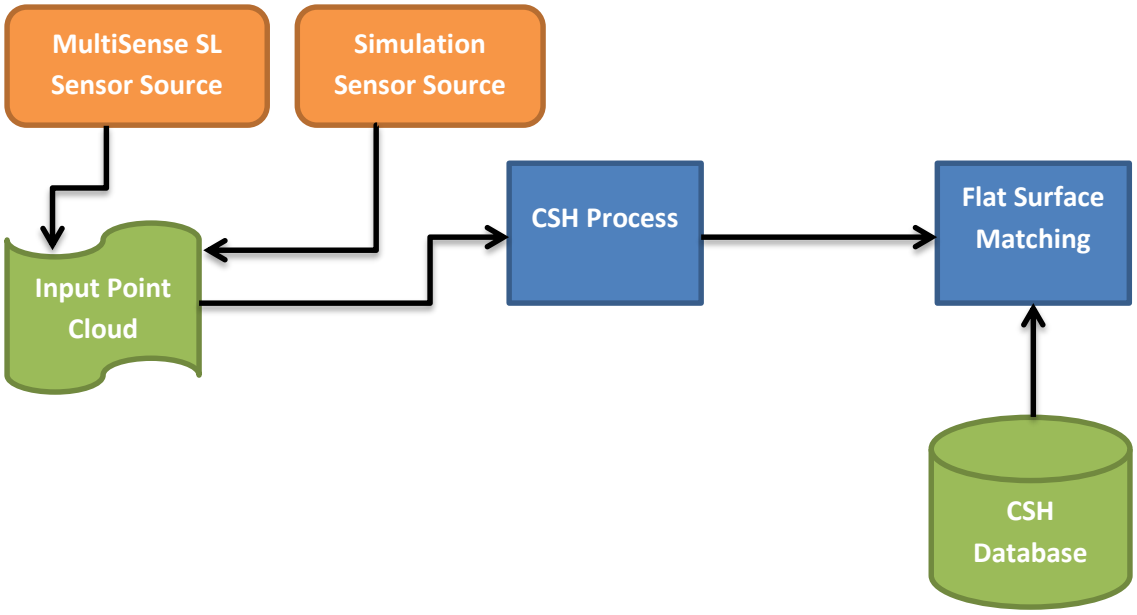

Figure 9: Existing Vision System for Flat Surfaces

The CSH process is executed by a general purpose GPU (GPGPU) and computes CSH in the following three steps. First, the normal vector computation is run using regular grid decomposition. All points are then classified into semantic classes, flat and non-flat. Finally, all CSH features are calculated and used in the next phase, matching. [29]

Matching or classification is performed with a support vector machine (SVM). A support vector machine is a supervised learning model which needs to be trained with the CSH flat surface database. This system utilizes a SVM with a linear kernel function. [29] [32]

Improved System

The improved system that I propose expands on the capabilities of the system proposed by [29] with the addition of capabilities to recognize specific objects and not only flat surfaces. In order for the robotic platform to be capable of acting autonomously in the event of signal compromise or total loss of communication with the teleoperator, the platform must be able to recognize objects themselves. The system improvement is shown in figure 9.

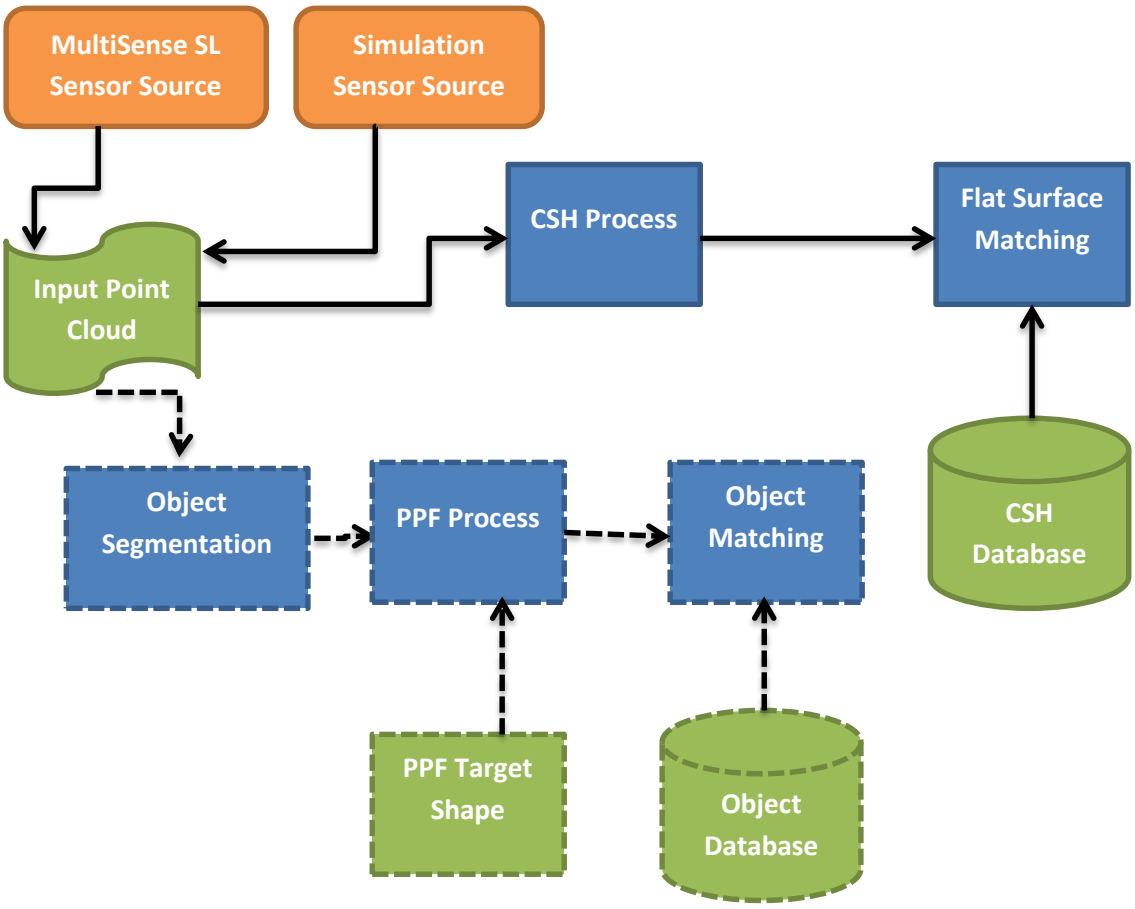

Figure 10: Expanded Vision System

The added components to the system are shown as objects with dotted lines. In addition to the CSH process I am proposing the addition of a system flow in an effort to identify specific objects such as a drill power tool. This flow consists of the additions of an object segmentation module

as described by Trevor, et al., [28] and Point Pair Feature method which was described earlier by Birdal, et al, [33]. This new system flow will be used to identify specific objects in the environment such as a cordless drill, a hose nozzle, or any other complex shape while simultaneous detecting planer surfaces. Finally, a support vector machine will be applied to the output of the PPF process with training data for the desired object, namely a drill power tool.

Improved System Preliminary Demonstration

The improved system has been partially realized as an exercise in learning how to use the point cloud library. The point cloud library is a robust set of algorithms that satisfy a wide range of use cases in computer vision using 3d vision sensors, depth, or range sensors as they are often called. This library is useful for this system as it is open source and has undergone a great deal of peer review. Of particular interest to this system was the object recognition functionality. This functionality was first exercised via a basic implementation that recognized objects using the correspondence grouping algorithm. [34] Figure 11 depicts the application written for this thesis which has been dubbed ORC, object recognition and classification.

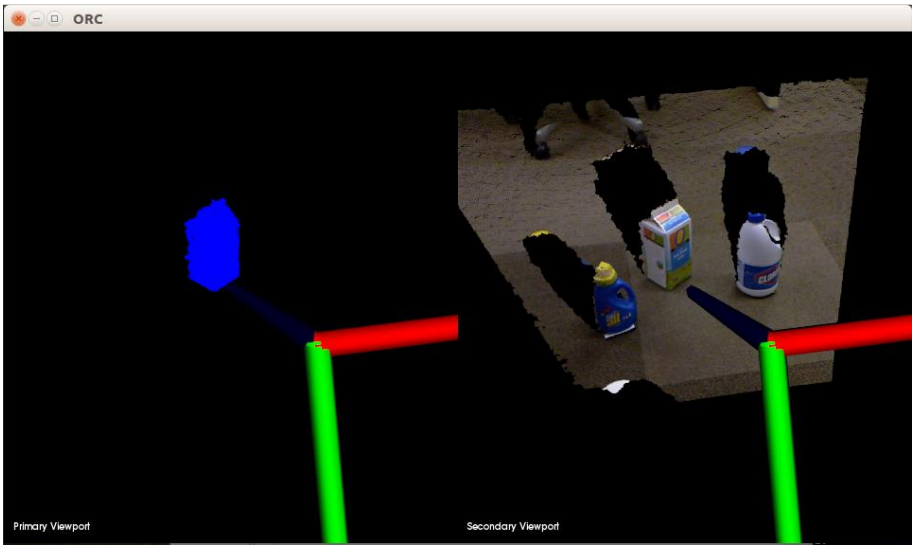

Figure 11: ORC system depicting model and scene

The image shown on the left is a rendering of the milk carton extracted from the scene data on the right through a Euclidean clustering process and saved for later processing and usage. This depiction is used to show the scene and model of interest to be later recognized. Figure 12 is a depiction of correspondence grouping applied to the scene data provided by pointclouds.org [34].

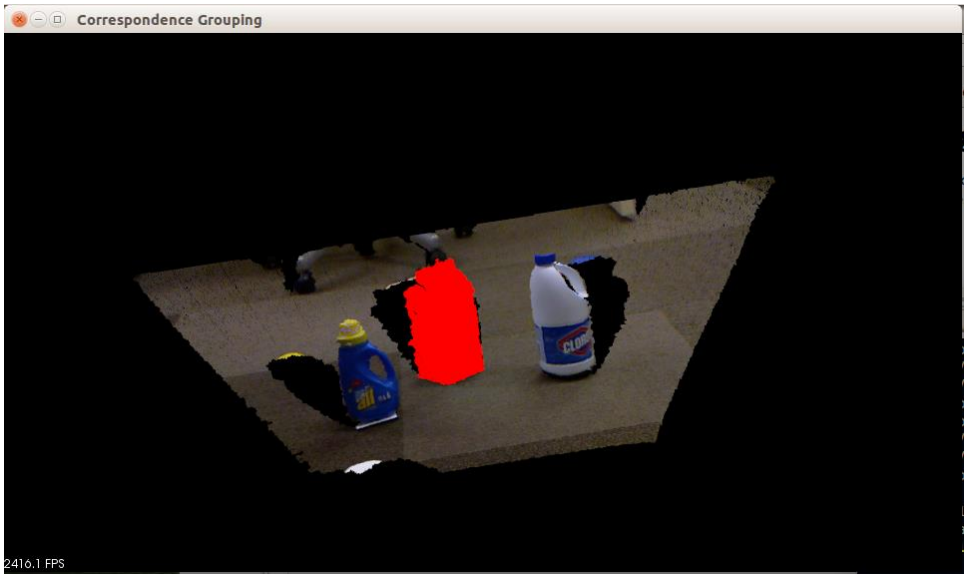

Figure 12: ORC system performing object recognition in scene data

The scene is shown again as in figure 11, however, the ORC system has been modified to include the addition of correspondence grouping functionality to have a baseline comparison of the intended effect when, eventually, the Point Pair Feature method would also be integrated into ORC. The intended effect is that when a specific object is being searched for in a scene, a milk carton in this case, the system will analyze a given scene, as shown above, and provide a positive indication (orange/red highlighting) that the object has been found. Additional information will also be provided such as the orientation of the target model relative to the scene and orientation relative to the sensor itself. This data would aid the robotic system that would utilize the ORC system by using the orientation and spatial data to grasp the target object with a manipulator arm in order to perform a task such targeting and grasping an object and possibly moving said object

as part of some overarching goal. This is only one of a great many potential use cases for such a system.

Challenges

Numerous challenges were faced during empirical study of this proposed improved method. Initial attempts appeared promising with the usage of Gazebo/DRCsim and the Point Cloud Library. I had decided to configure my Linux host with these tools, however, installation became the first overwhelming challenge. Neither tool could be installed directly using host installation tools as directed by the Gazebo/DRCsim install instructions. This was finally resolved but later broken by an Ubuntu update that could not be rolled back. New releases would follow that completely fixed the issue but when the last required update was released, my Gazebo/DRCsim simulation framework would malfunction, yet again. Help on this topic was lacking in the DRC community, perhaps due to a competitive mentality, as can be seen in my online help request. [35]

The Point Cloud Library provided the best opportunity to learn about and test existing. Many papers written in the 3D computer vision community often reference the PCL and provide their work as additions to the library. My research was to be done with the Microsoft Kinect RGB-D sensor which relied heavily on the OpenNI driver for Linux. This posed a challenge for numerous reasons but primarily because this driver was reverse engineered and not working properly or supported by Microsoft. The OpenNI project, started by the company Primesense, was shut down in 2013 [36] when it was purchased by Apple. Progress slowed considerably in the development of the library because OpenNI I was used heavily and the drivers for the Kinect would no longer be permitted for use.

Conclusion

We began with an overview of the DARPA Robotics Competition. The DRC was composed of three main events. The first event, the Virtual Robotics Challenge, sought to demonstrate each team's software development abilities relating to the design and implementation of telepresence and robot control software. The next event would be an initial showcase of the teams' software controlling an actual physical robotics platform. The final event would involve all measure of interference and obstacle to test every teams software and hardware deliveries.

The research involved a closer look at the Atlas sensor suite and the sensors that comprised it. Next we analyzed the functionality of the Robotics Operating System to determine how we might use it for our vision system. Next we looked at Gazebo simulator and the DRC simulator which would be used heavily during the VRC. Finally we looked at the Point Pair Feature method and Complex Shape Histogram method for recognizing flat surfaces with the Atlas's sensor suite, namely the MultiSense SL.

Finally, we presented an existing approach to recognizing flat surfaces. I proposed an enhanced version of the system that included a process utilizing Point Pair Feature method to detect an object in scene and then we applied a SVM to gain confirmation that the drill or other target object was positively identified.

Future Direction and Improvements

There are numerous ways to advance and improve the vision system described in this document. Currently, the expansion of this system's capabilities has been described only in theory and an implementation has not been fully realized except only in part by the authors of [29]. The next

step will be implement the expansion and integrate it with the Gazebo simulator as an initial test of the systems validity.

The machine learning method utilized, SVM, is limited to a binary classification indicating that something is or is not. Other machine methods should be considered to investigate how we might improve accuracy of classification, the amount of entities that can be classified, and how to improve the performance of the classification process.

Finally the hardware interfaces that connect the Atlas sensor suit should be reconsidered. As of now the Atlas sensor suite, the MultiSense SL, contains an FPGA which allows for real-time signal processing of the input scene data. However, the Ethernet interface becomes or will become a bottle neck when improved or additional FPGAs are utilized to perform more of the 3d object recognition work. For that reason I suggest that the sensor suite be integrated into the computing platform via a memory mapped interface such as PCIe which is capable of far greater bandwidth than gigabit Ethernet is capable of. This would be both an expensive and complex endeavor but I am certain that higher speeds will prove to be a significant improvement to robotic platforms that embrace this interface change.

References

[1] DARPA, "Overview," [Online]. Available: http://www.theroboticschallenge.org/overview.

[2] E. Krotkov, "Virtual Robotics Challenge Rules," DARPA.

[3] DARPA Tactical Technology Office, "Broad Agency Announcement, DARPA Robotics Challenge".

[4] Evan Ackerman, "DARPA Announces Tasks for DRC Finals," IEEE, [Online]. Available: http://spectrum.ieee.org/automaton/robotics/humanoids/darpa-announces-tasks-for-drc-finals.

[5] E. Ackerman and E. Guizzo, "DARPA Robotics Challenge Finals: Rules and Course," IEEE, [Online]. Available: http://spectrum.ieee.org/automaton/robotics/humanoids/drc-finals-course.

[6] S. Cass, "DARPA Unveils Atlas DRC Robot," IEEE, [Online]. Available: http://spectrum.ieee.org/automaton/robotics/humanoids/darpa-unveils-atlas-drc-robot.

[7] B. Dynamics, "Atlas - The Agile Anthropomorphic Robot," [Online]. Available: http://www.bostondynamics.com/robot_Atlas.html.

[8] Carnegie Robotics, "MultiSense SL," [Online]. Available: http://carnegierobotics.com/multisense-sl.

[9] Hokuyo Automatic Co.,LTD, "Scanning Laser Range Finder UTM-30LX-EW Specification," 2011.

[10] Carnegie Robotics, "MultiSense S7," [Online]. Available: http://carnegierobotics.com/multisense-s7.

[11] H. Hirchmuller, "Semi-Global Matching – Motivation, Developments, and Application".

[12] H. Hirschmuller, M. Buder and E. Ines, "Memory Efficient Semi-Global Matching," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science,* vol. 1, no. 3, 2012.

[13] Open Source Robotics Foundation(OSRF), "ROS/Introduction," [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[14] Open Source Robotics Foundation(OSRF), "ROS/Concepts," [Online]. Available: http://wiki.ros.org/ROS/Concepts.

[15] Generation Robots, "ROS - Robot Operating System - Generation Robots," [Online]. Available: http://www.generationrobots.com/en/content/55-ros-robot-operating-system.

[16] Open Source Robotics Foundation(OSRF), "Nodes," [Online]. Available: http://wiki.ros.org/Nodes.

[17] Open Source Robotics Foundation(OSRF), "Master," [Online]. Available: http://wiki.ros.org/Master.

[18] Open Source Robotics Foundation(OSRF), "Parameter Server," [Online]. Available: http://wiki.ros.org/Parameter Server.

[19] Open Source Robotics Foundation(OSRF), "Topics," [Online]. Available: http://wiki.ros.org/Topics.

[20] Open Source Robotics Foundation(OSRF), "Messages," [Online]. Available: http://wiki.ros.org/Messages.

[21] Open Source Robotics Foundation(OSRF), "Services," [Online]. Available: http://wiki.ros.org/Services.

[22] Open Source Robotics Foundation(OSRF), "Bags," [Online]. Available: http://wiki.ros.org/Bags.

[23] N. K. I. C. H. B. S. P. J. H. B. G. S. P. J. L. R. J. M. E. K. G. P. Carlos E. Agüero, "Inside the Virtual Robotics Challenge: Simulating Realtime Robotic Disaster Response," *IEEE Transactions on Automations Science and Engineering,* vol. 12, no. 2, 2015.

[24] Open Source Robotics Foundation(OSRF), "Gazebo," [Online]. Available: http://gazebosim.org/.

[25] A. Nguyen, "3D Point Cloud Segmentation: A Survey," in *Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on* .

[26] pointclouds.org, "How to use Random Sample Consensus model," [Online]. Available: http://pointclouds.org/documentation/tutorials/random_sample_consensus.php.

[27] pointclouds.org, "Basic Structures," [Online]. Available: http://pointclouds.org/documentation/tutorials/basic_structures.php.

[28] S. G. R. B. R. H. I. C. Alexander J. B. Trevor, "Efficient Organized Point Cloud Segmentation with Connected Components".

[29] J. Bedkowski, K. Majek, A. Maslowski and P. Kaczmarek, "Recognition of 3D Objects For Walking Robot Equipped With MultiSense-SL Sensor Head," in *Proceedings of the Sixteenth International Conference on Climbing and Walking Robots*, 2013.

[30] B. Drost, M. Ulrich, N. Navab and S. Ilic, "Model Globally, Match Locally: Efficient and Robust 3D Object Recognition," *IEEE*.

[31] T. Fletcher, "Support Vector Machines Explained".

[32] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, "16.5 Support Vector Machines," in *Numerical Recipes: The Art of Scientific Computing 3rd Edition*, pp. 884-898.

[33] T. Birdal and S. Ilic, "Point Pair Features Based Object Detection and Pose Estimation Revisited," in *IEEE 3D Vision*, 2015.

[34] pointclouds.org, "Documentation - Point Cloud Library (PCL) - 3D Object Recognition based on Correspondence Grouping," [Online]. Available: http://www.pointclouds.org/documentation/tutorials/correspondence_grouping.php.

[35] B. Maas, "Unable to install drcsim and gazebo," [Online]. Available: http://answers.gazebosim.org/question/358/unable-to-install-drcsim-and-gazebo-on-ubuntu-1204-or-1210/.

[36] "OpenNI," [Online]. Available: https://en.wikipedia.org/wiki/OpenNI.

[37] A. Hamacher, "DARPA's Robotics Challenge: A pocket-sized guide to the finals," Robohub, [Online]. Available: http://robohub.org/darpas-robotics-challenge-a-pocket-sized-guide-to-the-finals/.

[38] DARPA, "DRC Trials," [Online]. Available: http://archive.darpa.mil/roboticschallengetrialsarchive/.

[39] W. Community, "DARPA Robotics Challenge," [Online]. Available: https://en.wikipedia.org/wiki/DARPA_Robotics_Challenge.

[40] E. Sofge, "The DARPA Robotics Challenge Was A Bust," Popular Science, [Online]. Available: http://www.popsci.com/darpa-robotics-challenge-was-bust-why-darpa-needs-try-again.

[41] scikit-learn, "Support Vector Machines," [Online]. Available: http://scikit-learn.org/stable/modules/svm.html.

[42] T. Fletcher, "SVM Explained," [Online]. Available: http://www.tristanfletcher.co.uk.

[43] R. W. R. K. Ruwen Schnabel, "Efficient RANSAC for Point-Cloud Shape Detection".

[44] L. A. Alexandre, "3D Descriptors for Object and Category Recognition: a Comparative Evaluation".

Curriculum Vitae

NAME: Brandon Maas

PERMANENT ADDRESS: 8014 Stone Haven Drive, Glen Burnie, MD 21060

DEGREE AND DATE TO BE CONFERRED: Master of Science., 2016

EDUCATION

Collegiate Institutions Attended	Dates	Degree	Date of Degree
Towson University	2011-2016	Master of Science: Computer Science	05/2016
University of Delaware	2003-2009	Bachelor of Science: Computer Science Concentration: High Performance Computing	02/2009

EXPERIENCE

Senior Software Engineer (Technical Lead), March 2013 - Present
Northrop Grumman Systems Corporation, Baltimore, MD

- Designed, implemented, integrated, and tested components of avionics software executing on high-altitude long endurance (HALE) unmanned aircraft platforms
- Designed, implemented, integrated, and tested components of avionics software executing on Black Hawk helicopter platforms
- Successfully conducted several internal research and development (IRAD) efforts, each with goals to demonstrate a technological readiness level (TRL). These IRADs were instrumental in the award of contracts exceeding 400M and one exceeding 1B
- Maintained Top Secret level security clearance

Software Engineer, August 2009 – March 2013
Safenet, Inc., Belcamp, MD

- Implemented new features and improvements on Safenet's HAIPE network encryptor product and facilitated successful NSA certifications of said product
- Lead team in pursuance of FIPS140-2 Certification for Safenet's SC650 smart card product

- Performed extensive research and development on Safenet's security solution targeting Android mobile devices

Lab Technician, June 2006 – June 2008
Bartol Research Institute – University of Delaware, Newark, DE

- Developed bare metal applications that executed on the Motorola HC08 microcontroller used on data acquisition equipment
- Integrated Secure Digital Card storage on data acquisition equipment to be used in the extreme conditions of the Ice Cube Neutrino Observatory at the South Pole, Antarctica
- Developed a primitive file system that was used on the SD card to record and store neutrino sensor data