An integrated perception pipeline for robot mission execution in unstructured environments

Narayanan, Priya, Yeh, Bryanna, Holmes, Emma, Martucci, Scott, Schmeckpeper, Karl, et al.

Priya Narayanan, Bryanna Yeh, Emma Holmes, Scott Martucci, Karl Schmeckpeper, Christoph Mertz, Philip Osteen, Maggie Wigness, "An integrated perception pipeline for robot mission execution in unstructured environments," Proc. SPIE 11413, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, 1141318 (21 April 2020); doi: 10.1117/12.2560699

Event: SPIE Defense + Commercial Sensing, 2020, Online Only, California, United States

An integrated perception pipeline for robot mission execution in unstructured environments

Priya Narayanan[a], Bryanna Yeh[b], Emma Holmes[b], Scott Martucci[b], Karl Schmeckpeper[c], Christoph Mertz[d], Philip Osteen[a], and Maggie Wigness[a]

[a]CCDC Army Research Laboratory, Adelphi, MD, USA
[b]Johns Hopkins: Applied Physics Lab, Laurel, MD, USA
[c]University of Pennsylvania, Philadelphia, PA, USA
[d]Carnegie Mellon University: The Robotics Institute, Pittsburgh, PA, USA

ABSTRACT

Visual perception has become core technology in autonomous robotics to identify and localize objects of interest to ensure successful and safe task execution. As part of the recently concluded Robotics Collaborative Technology Alliance (RCTA) program, a collaborative research effort among government, academic, and industry partners, a vision acquisition and processing pipeline was developed and demonstrated to support manned-unmanned teaming for Army relevant applications. The perception pipeline provided accurate and cohesive situational awareness to support autonomous robot capabilities for maneuver in dynamic and unstructured environments, collaborative human-robot mission planning and execution, and mobile manipulation. Development of the pipeline involved a) collecting domain specific data, b) curating ground truth annotations, e.g., bounding boxes, keypoints, c) re-training deep networks to obtain updated object detection and pose estimation models, and d) deploying and testing the trained models on ground robots. We discuss the process of delivering this perception pipeline under limited time and resource constraints due to lack of a priori knowledge of the operational environment. We focus on experiments conducted to optimize the models despite using data that was noisy and exhibited sparse examples for some object classes. Additionally, we discuss our augmentation techniques used to enhance the data set given skewed class distributions. These efforts highlight some initial work that directly relates to learning and updating visual perception systems quickly in the field under sudden environment or mission changes.

Keywords: robot visual perception, object detection, keypoint detection, learning from small data, operation in unstructured environments

1. INTRODUCTION

The recently concluded Robotics CTA, a collaborative technology alliance program, led by the Combat Capabilities Development Command (CCDC) Army Research Laboratory (ARL) with several academic and industrial partners, addressed research and development requirements to enable the transition of robotic unmanned systems from research laboratories to natural unstructured environments for Army relevant applications. The focus of this program was to develop technologies required to permit inanimate systems to perform in a seemingly human fashion and make these systems an integral part of the team. In future Army operational scenarios, these ground systems are expected to navigate through highly challenging environments such as urban clutter and dense vegetation to carry out complex missions. The robots will also be fitted with manipulators/limbs that will be used for clearing obstacles (e.g., rubble, fallen trees) enabling them to navigate through unstructured and challenging terrain. The human teammates will provide guidance and command to the robotic forces to help the robots plan their course of action and trajectories.

To address these future requirements, some of the key capabilities developed during the course of the RCTA program are OPTEMPO mobility in dynamic and unstructured environments, dynamic mission execution within human-robot teams, and robust manipulation of generic objects. Many software components are integrated to

Corresponding Author
P. Narayanan: priya.narayanan.civ@mail.mil

Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II, edited by Tien Pham, Latasha Solomon, Katie Rainey, Proc. of SPIE Vol. 11413, 1141318
© 2020 SPIE · CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2560699

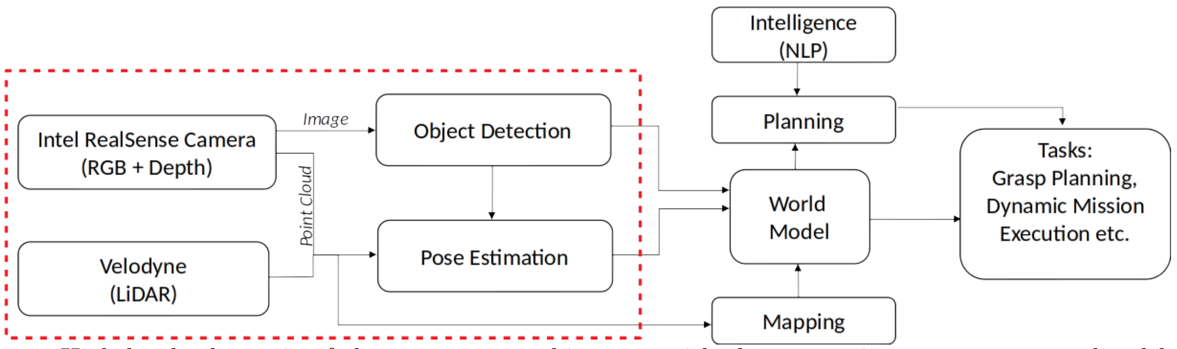

Figure 1: High level schematic of the autonomy architecture with the perception components outlined by the dashed red line. We discuss the details of the perception stack in this paper and briefly describe how perception provides input to the overall system to accomplish high level tasks.

achieve these capabilities, as seen in Figure 1, but an advanced perception system is a crucial component to provide context for the unmanned system as it reasons, makes decisions, re-plans, and maneuvers throughout these unstructured environments. We discuss the perception system, indicated by the dotted red line in Figure 1, used within the RCTA program to provide robust onboard processing for object detection and pose estimation of objects to aid in higher level human-robot teaming capabilities.

Learning perception models is a well-studied area of research with benchmark datasets and challenge problems addressed by the larger research community. However, the commercial applications of these learned models and techniques is largely orthogonal to real-world Army applications. More specifically, there is limited object of interest overlap between the different applications, and there is a large domain gap between readily available training data within the commercial sector and that of the Army relevant unstructured operating domain.

This application gap necessitates the development of a larger perception pipeline that includes a strategy for rapid data collection, annotation, training, and deployment of perception modules that can be used to modify the onboard perception system when it degrades over time or when the robot begins operation in a novel environment. In this paper we present the developed perception pipeline used for the RCTA program to address this need including the perception challenges associated with unstructured environments such as clutter and small or rare objects, and challenges met when trying to rapidly collect labeled data such as label noise, verification, and class imbalances.

The rest of this paper is outlined as follows. Section 2 outlines the motivating human-robot teaming application and describes the military relevant characteristics of the environment and objects the perception system must learn. The general perception tasks are outlined in Section 3 followed by a description and evaluation of the perception pipeline in Section 4. In Section 5 we describe how the perception system has been used for successful dynamic mission execution. An overview of existing perception research and datasets commonly used in commercial and academic settings is provided in Section 6, with a discussion on how this contrasts with the unstructured environments used to test capabilities developed within the RCTA. Finally, we conclude with some thoughts on future work in Section 7.

2. MOTIVATING MISSION EXECUTION SCENARIO

The perception pipeline outlined in this paper is largely motivated by intelligence, surveillance and reconnaissance (ISR) tasks that can be executed by a human-robot team. The combination of multiple ISR tasks can be thought of as a dynamic mission in the sense that the information retrieved by completing one task will likely be relevant in determining the next task that needs to be accomplished. In this section, we describe the environment characteristics, the interaction between the human-robot team, and the robot teammate for the motivating scenario.

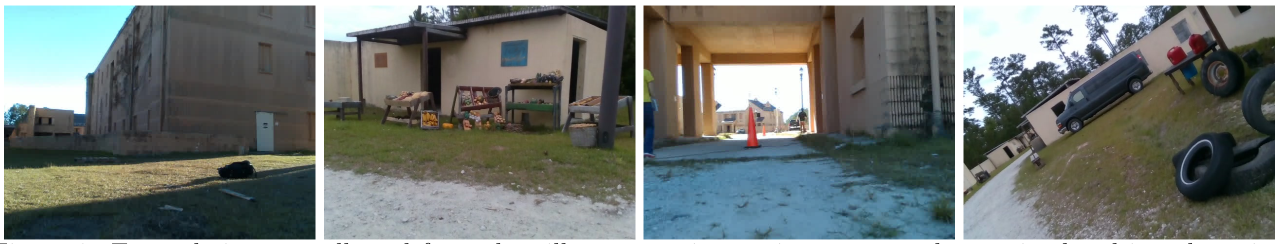
Figure 2: Example images collected from the village operating environment used to train the object detection models.

2.1 Mission Execution Environment

In the context of the RCTA program, the operating environment for human-robot mission execution was meant to resemble military relevant characteristics, e.g., highly complex or unstructured features. More specifically, mission execution was performed in environments that 1) do not resemble urban cities found in many autonomous driving research,[1,2] and 2) contain unique object classes that appear infrequently, if at all, in existing object detection benchmarks.[3–5] Most experimentation was performed at a military training facility that is meant to resemble a small village. The environment contains several building structures, gravel roads, and various objects. There is a gravel road that runs through a market area of the environment, which is particularly object-rich. As the robot travels down the path through the market, the items each shop displays easily fall into the field of view and provide many landmarks that could be used to test mission execution.

Figure 2 shows some example images collected from the robot's camera sensor when driving through the environment. All training data for the object detection models discussed in Section 4.4 are collected from the pictured environment.

2.2 Human-Robot Interaction

When operating in the previously mentioned environment, the human will provide natural language commands to the robot teammate describing what the robot should do next in the dynamic mission. It is assumed that the robot and human (either remote or in situ) share some common view of the environment, and mission commands can be issued with respect to landmarks (e.g., objects or buildings) in the scene.

The following are example commands that a human could issue to a robot teammate to direct the robot's next task during a mission.

- `Meet me behind the building on your left.`
- `Report anything found behind the nearest barrel.`
- `Investigate the red car.`
- `Tell me which door the person on your right enters.`
- `Remove the Czech Hedgehog that is blocking the road.`

We note that these complex commands require a robot to be equipped with an autonomy software stack that includes additional components beyond simply perception, such as mapping, planning, intelligence, manipulation etc. A detailed discussion of these additional components is beyond the scope of this paper, (we visually show these connections in Figure 1), but it is evident from these few examples that the successful execution of these commands requires a perception system capable of identifying terrain and detecting objects such as buildings, barrels, cars, people, doors, and Czech Hedgehogs. Further, object pose estimation is required to execute any tasks that require physically moving objects in the scene.

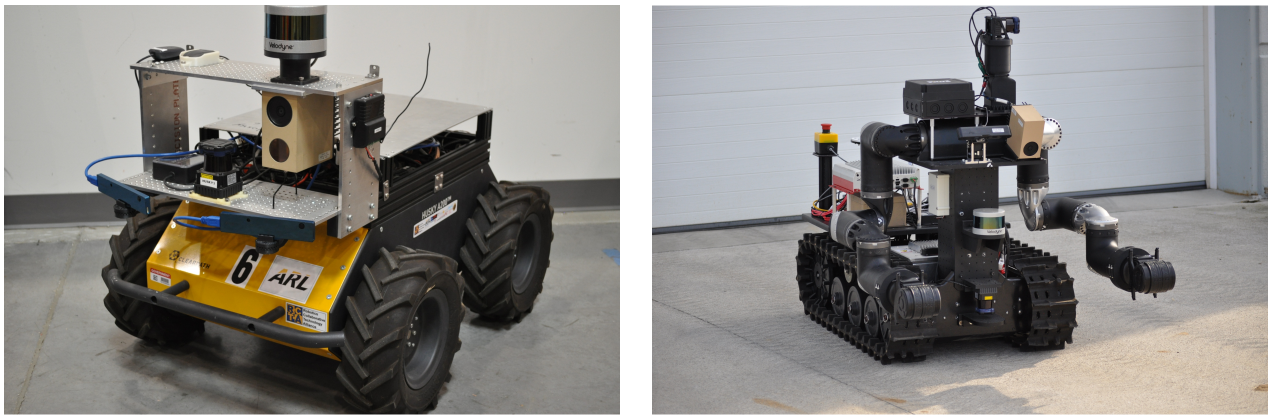
(a) Husky (b) RoMan

Figure 3: The two platforms used for human-robot teaming experimentation: (a) Clearpath Husky and (b) Robotic Manipulator (RoMan).

2.3 Robot Teammates

Two robot teammates were used throughout the RCTA program. Figure 3 shows the Husky platform on the left and the Robotic Manipulator (RoMan) platform on the right. The Husky is manufactured by Clearpath Robotics and is a rugged, outdoor-ready unmanned ground vehicle (UGV) suitable for research and rapid prototyping. The robot has external dimensions of 39 x 26.4 x 14.6 in, weighs 110 lbs, can carry a maximum payload of 165 lbs, and is fully supported by Robot Operating System (ROS). The robot is equipped with a Velodyne VLP16 and Intel RealSense 435 to aid in the perception tasks for dynamic mission execution reserach. Perception inference was run onboard the Husky using an Nvidia GTX 1060, which allowed object detections to run at a rate of $\sim$5 Hz for images with resolution 640x480.

The RoMan is a heavy-duty robotic platform capable of autonomous whole-body mobile manipulation built in-house by CCDC ARL and collaborators as part of the RCTA program. This platform combines 1) a Talon base - a lightweight, unmanned, tracked military robot designed and built by Foster-Miller (owned by QinetiQ North America) for mobility on rough terrain and 2) dexterous RoboSimian arms - built by NASA JPL for manipulation in cluttered environments. The Talon base has a dimension of 34.5 x 22.75 x 33 in, weighs 178 lbs, and can carry a maximum payload weight of 150 lbs. RoMan has an articulated torso as well as a pan-tilt unit (PTU) with a VLP16 and an Intel RealSense color/depth (RGB-D) camera for directed perception. Its manipulator sensors (e.g., force-torque sensor) are outfitted with electronics and sensors to enable directed perception and dexterous mobile manipulation. The platform also has a planar Hokuyo line scanner, inertial sensors, and 4 quad-core computers.

3. PERCEPTION TASKS

3.1 Object Detection

The primary perception task needed for the dynamic mission execution scenario is object detection. For this task, the robot must localize and identify objects in the environment using its camera sensor to understand the landmarks and objects mentioned in the commands provided by the human teammate. For this perception task, an object will be localized in the image by placing a tight bounding box around the object's shape outline. The object in this bounding box is then identified by the appropriate object label, e.g., window or vehicle.

A number of open-source implementations of deep learning architectures are available for object detection tasks. We leverage the Pytorch 1.0 implementation of Faster R-CNN within the maskrcnn-benchmark repository.[6] The Faster R-CNN architecture has two networks: a region proposal network (RPN) and a network for detecting objects. A Convolutional Neural Network (CNN) first extracts feature maps from the images. These feature maps are then passed through the RPN which returns the candidate regions of interest (RoI). An RoI

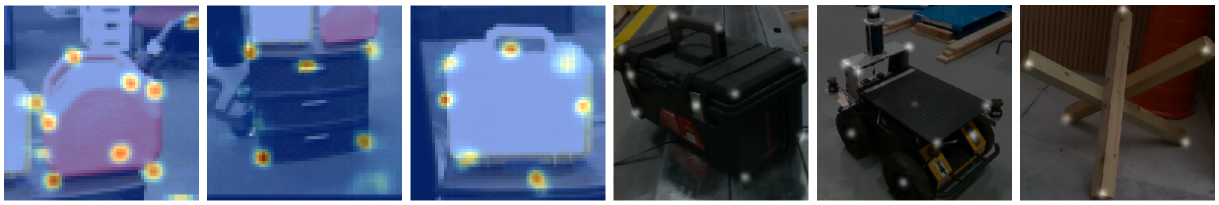

Figure 4: Example of predicted semantic keypoint locations for different object classes. From left to right: gas can, cabinet, briefcase, crate, robot, and Czech Hedgehog.

pooling layer is then applied to bring all the candidates to the same size. Finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects.

We use this architecture as a starting point to learn object detection models capable of identifying objects in our operational environment. Details outlining the data used to train these models and specific learning parameters are provided throughout Section 4.

3.2 Object Pose Estimation

Two-dimensional object detection is insufficient for a robot to precisely interact with three-dimensional objects. To navigate to and manipulate a detected object, the robot must be able to estimate the object's pose and orientation. We leverage three different approaches for the pose estimation block in Figure 1. The specific approach chosen for a task is based on the object of interest, i.e., whether or not there is a trained model for the object, and required quality of the pose estimate needed to successfully complete the manipulation task.

For close range pose estimation, we use the PErception via SeaRCH (PERCH) algorithm.[7] This approach provides accurate pose estimates, but is limited to detecting objects within the range of the RGB-D sensors and requires models of the objects of interest. For unknown object classes, i.e., no object model is available, object position is estimated by averaging the LiDAR readings that fall within the detected bounding box. This method has large errors and cannot predict orientation, but does provide an estimate of the location of an object without any training data.

We focus our discussion on the case for known object classes at longer ranges. In this case, we use semantic keypoint based pose estimation, which gives accurate pose estimates but requires data labeled with keypoint annotations (described further in Section 4.2.2). Point-cloud based object pose estimation methods[7] suffer from the limitations of their sensors, with RGB-D sensors having limited range and LiDAR having limited resolution. In order to detect all objects of interest at any range, we require a solution that relies at least in part on RGB images only.

We utilize a semantic keypoint based pose estimation pipeline, based on work by Pavlakos et al.[8] A stacked hourglass neural network[9] detects semantic keypoints for each object class. Our semantic keypoint detection network was implemented in Pytorch[10] and runs on a NVIDIA Jetson TX2 at 2 Hz. Example detections are shown in Figure 4. The semantic keypoints are matched to a deformable object model, and the pose of the object is optimized to minimize the deformations required to make the model align with the predicted keypoints.[8]

This optimization fails when the object is rotationally symmetric. In order to allow our system to handle objects that have rotational symmetries, we use the camera position to disambiguate the keypoints. For example, on the Czech Hedgehog in Figure 5, the first keypoint is always located on the lower leg that is closest to the camera. This allows the optimization to solve for the pose of the Czech Hedgehog, modulo 120 degrees. Since the Czech Hedgehog is rotationally symmetric, having the orientation modulo 120 degrees is sufficient to accomplish most tasks.

4. PERCEPTION PIPELINE OVERVIEW

In this section, we outline the full pipeline implemented to generate learned perception models. Specifically, we describe the process for collecting raw image data, the annotation process to produce ground truth for the

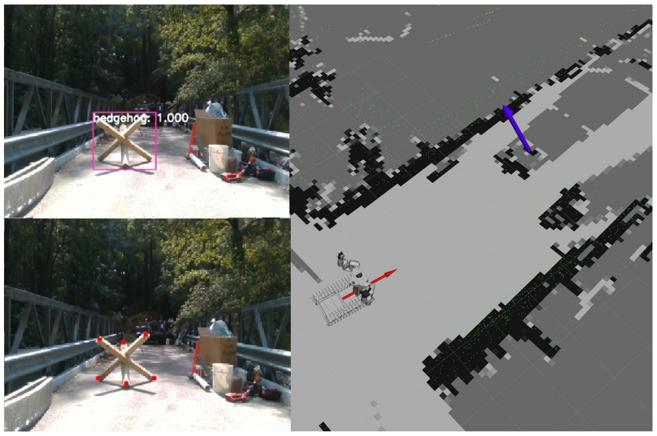

Figure 5: Semantic keypoint example for a rotationally symmetric object. Counter-clockwise from upper left: Object detection, keypoint detection, and pose estimation of the Czech Hedgehog. The blue arrow represents the pose of the Czech Hedgehog while the red arrow shows the pose of the robot platform.

supervised learning techniques, and the training procedure. This pipeline ultimately results in object detection models and semantic keypoint pose estimation models that are deployed on the robots for dynamic mission execution. We also provide a detailed discussion outlining the challenges associated with rapid re-training and deployment of perception models to ensure robust detection in novel domains.

4.1 Data Collection

The first stage of the pipeline is to collect raw imagery that can be used to train the perception models. Collection of raw data is arguably the easiest task in the pipeline, but there are a number of considerations to take into account to ensure the data collected represents the best possible set of training data.

First, the quantity of data matters. As visual perception has made huge advances with deep learning architectures, we have seen that the need for large sets of training data is on the rise. This certainly impacts the second stage of the pipeline, data annotation, but we discuss how this is addressed in the next section. Second, images that matches the operating conditions will provide the most domain specific training data for learning. Specifically, we aim to collect data of the environment using the same hardware and viewing angles that the robot will use to run inference during mission execution.

To address these first two considerations, object detection training data is collected by teleoperating the robot in the operating environment and capturing video sequences from the onboard camera sensor. For objects less common in the environment, additional video was captured by moving the camera sensor around the object to capture more viewing angles. The advantage of capturing videos is that a large number of class instances from different distances and perspectives could be quickly collected. Additionally, contiguous frames allow us to leverage semi-automated bounding box propagation during the annotation phase. Yet, annotation of all frames for videos captured at 30 Hz can be time intensive. Thus, the training videos were downsampled to either 10 Hz or 3 Hz (if annotation time was extremely limited) prior to annotation.

Keypoint training images were collected at various viewpoints and distances to the object, both as individual snapshots (which were annotated individually) as well as continuous streams of color and depth data (which were annotated via 3D reconstruction). A hand-held RGB-D camera was used instead of data from a teleoperated robot to ensure smooth camera motion, which is required for accurate 3D reconstruction. While additional keypoint data was collected off-site to evaluate the effectiveness of the annotation approach, the entire collect-annotate-train pipeline was first demonstrated in the previously described environment.

In total, over 60,000 frames (examples seen in Figure 2) were collected for bounding box annotation to train the object detection model. The data came from two data collection efforts in the same environment. One during the Fall and the other during the Summer, which further provide varying viewpoints of the same environment. In the next section we discuss the annotation process used to quickly label these images.

Generator	Crate	Bench
Dumpster	Pelican Case	School Bus
Backpack	Suitcase	Gas Can
Debris	Trash Bin	Gravestone
Wood Pallet	Tower	Barrier
People	Toilet	Police Truck
Light Pole	Barrel	Gas Pump
Control Tower	Tank	Motorcycle
Shop	Window	Electrical Box
Gate	Chair	Bicycle
Table	Traffic Sign	Truss
PVC Pipe	Weapons	Stairs
Door Ground	Ingress Ground	

Table 1: Labeling ontology used for bounding box annotation.

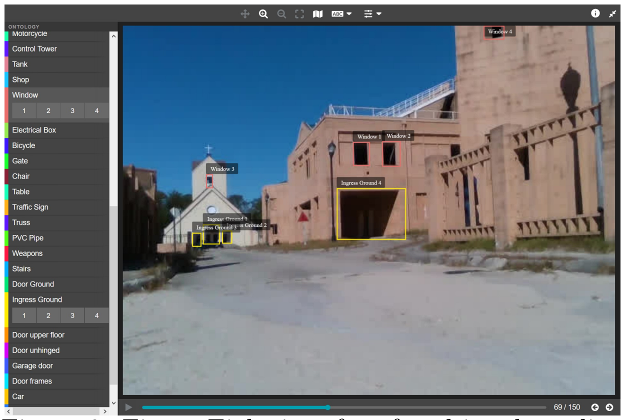

Figure 6: Figure Eight interface for object bounding box annotation.

4.2 Data Annotation

As with any modern machine learning framework, data quality and quantity is as important, if not more so, than the design of a learning algorithm. Yet, collecting and annotating data comes at a cost, which can be mitigated by using advanced annotation tools that leverage existing state-of-the-art in machine learning and geometric reconstruction.

The specific annotation process to collect labeled training data is different for the object detection and object pose estimation tasks. Although each process is unique, they share the underlying goal of making the annotation process as efficient as possible to reduce the total time necessary to curate new training data to learn or update models. We discuss the annotation technology used to produce efficient labeling, the total labeling time, and challenges that still need to be addressed to continue to improve this process.

4.2.1 Bounding box annotation

Accurate data curation using 2D bounding box annotation tools to define regions of interest within an image is extremely important for learning accurate object detection models. This meticulous manual annotation process can be an expensive and labor intensive task, which is recognized as the single greatest bottleneck in developing object detection models.

For the RCTA experimentation, bounding box annotation was outsourced to Figure Eight[*], an Appen company. A total of 38 object classes defined the labeling ontology, shown in Table 1. Annotation was primarily conducted by external contributors via the Figure Eight labeling platform. To improve the accuracy and minimize inconsistencies between annotators, label verification was performed by internal labelers, mainly comprised of RCTA team members. Both initial annotation and verification were performed using the annotation interface seen in Figure 6. This interface had several user friendly features such as interpolation of bounding boxes between frames, shortcuts for the most critical actions, and a dashboard with a list of annotation tasks to help speed up the content curation.

When new operational data was needed during field experimentation, data was collected during the day and annotated by external labelers overnight. Internal verification was conducted the following day so models could be trained and deployed on the robots quickly. Although this approach was feasible for demonstration purposes in the RCTA program, it may not be practical to obtain labeling assistance from remotely located contributors in a real-world battlefield environment. Further, even though the annotation interface had the most advanced state-of-the-art features, it still took several minutes to annotate an image depending upon the number of objects of interest present in each frame. Later in Sections 4.3.1 and 7, we discuss some alternative approaches that were explored during the course of the RCTA program and future work that begin to address these issues.

[*]https://www.figure-eight.com/

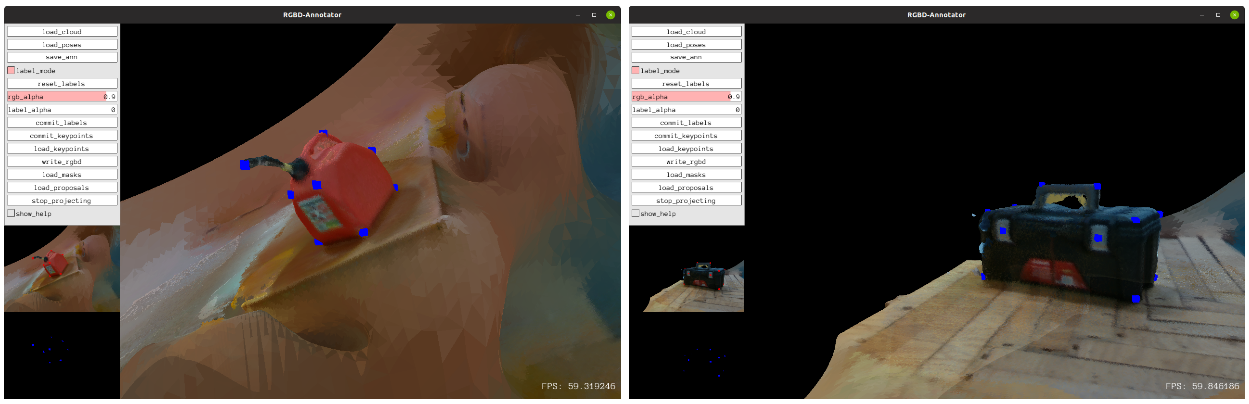

Figure 7: Visualization of 3D keypoint annotations (blue cubes) applied to a scene reconstruction that has been converted from sparse points into a 3D mesh. These keypoints are annotated once in 3D, then automatically reprojected to the constituent images used to create the reconstruction.

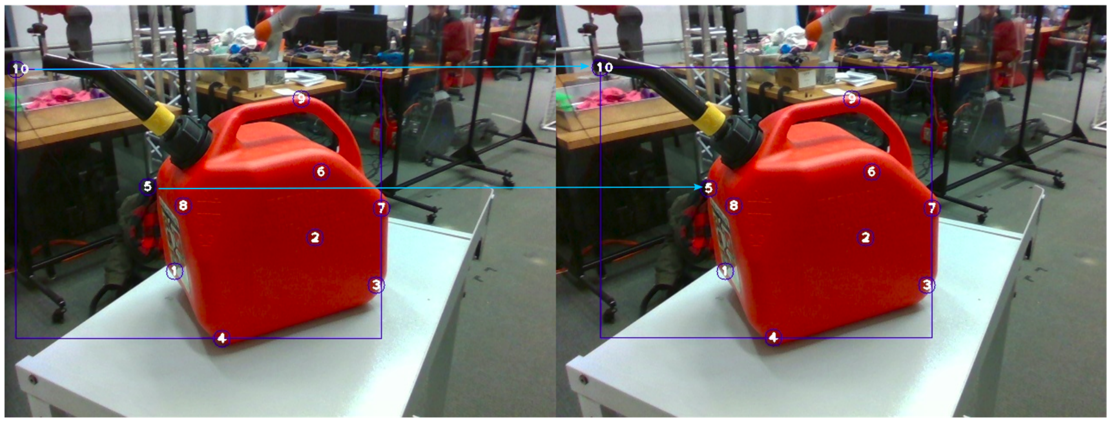

Figure 8: Keypoint projection and refinement. Using estimated camera poses attained from 3D scene reconstruction, keypoints are automatically projected to their constituent images. At left, inaccurate camera pose estimates cause poor keypoint reprojections (keypoints 5 and 10). At right, 3D descriptor matching between real and reprojected depth images produces more accurate keypoints.

4.2.2 Keypoint annotation

Recent works have shown that bottom-up hierarchical segmentation or learned object proposals can be combined with 3D geometric scene reconstructions to provide a multiplying effect of human annotation.[11,12] Here, we extend prior work to facilitate efficiently annotating keypoints for downstream applications such as object pose estimation from color images. With a pre-processing pipeline that includes 3D scene reconstruction[13] and surface reconstruction,[14] an annotation tool[12] facilitates the annotation of relevant keypoints in the 3D model, as shown in Figure 7.

Using camera poses estimated for scene reconstruction, keypoints can then be efficiently reprojected to a stream of images with relatively little human effort. However, drift and other inaccuracies lead to incorrect keypoint projections, in some cases crossing the boundary of the object of interest (as seen on the left in Figure 8). To refine these keypiont inaccuracies, we compare the depth image generated from the 3D reconstruction for each camera pose with the raw depth image. While we could also compare color images, we hypothesize that 3D reconstructions preserve the *shape* of an object better than *appearance* for a given camera pose, which is subject to variable illumination and other effects. Therefore, we derive 3D descriptors[15] at each keypoint in a reprojected depth image, and match to descriptors extracted in the corresponding true depth image to improve keypoint location accuracy. An example of keypoint refinement is shown on the right in Figure 8.

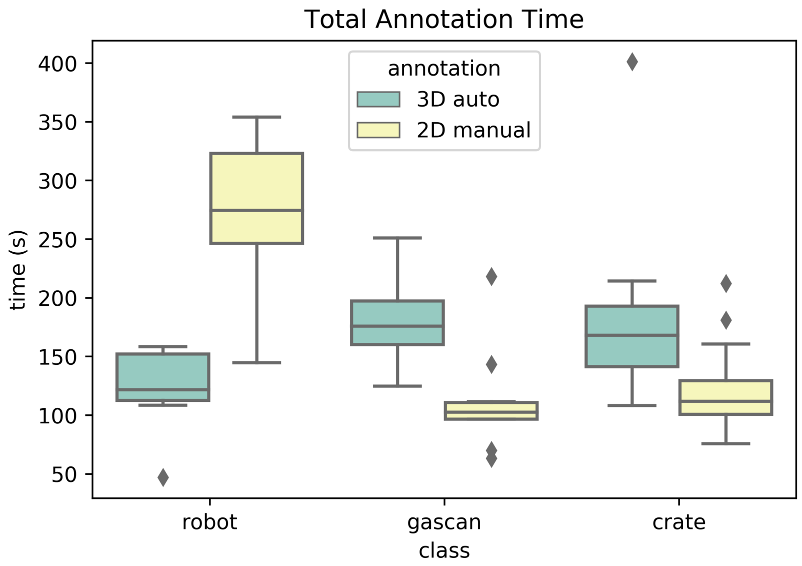

Figure 9: Runtime comparison of keypoint annotations using 2D annotation tool and 3D automatic projection. Here, the 3D approach labels around 1,000 images in approximately the same human annotation time as the 2D method labels 10.

While the 3D scene reconstruction and reprojection approach provides a large multiplying effect of annotation effort, it requires that the object be in range of an RGB-D sensor, which is typically less than 3m for accurate depth estimates. To supplement the data collected with the reconstruction method, we also collected and annotated data of objects at farther distances using direct 2D image annotation. All keypoint annotations were done internally, with three different non-expert users performing 2D data annotation for the object classes robot, gas can, and crate. A single expert-user performed corresponding 3D annotations for the same object classes. The 2D annotators labeled 10 randomly sampled frames from 10 distinct data sequences for each class; the 3D annotator simply annotated the corresponding 3D reconstructions. Figure 9 shows the comparison of the average time to label each 2D sequence (consisting of 10 images each) and the time to perform a single 3D annotation of the same sequence. The runtimes for each are similar, but the 3D annotation can reproject to label all constituent frames, which averaged to about 1000 frames per sequence, demonstrating the efficiency of 3D annotation with automatic reprojection.

4.3 Data Augmentation

When training deep learning object detection algorithms for multi-class tasks, it is critical to have a well balanced dataset that has a) sufficient number of samples for each of the classes and b) similar number of samples for each of the classes. It is difficult to address these requirements at the data collection stage especially during field experimentation in environments with very little a priori knowledge. The data collected for the RCTA program captured thousands of instances of objects such as doors and windows, but very few object instances such as gas pump and chair. Training a deep learning network with such a skewed dataset can negatively impact the accuracy of the detector. We describe two augmentation techniques that were used to address this challenge.

4.3.1 Image Synthesis

The common method of creating training data by labeling many images does not scale well and often is not practical. Besides the large effort needed to label thousands or sometimes hundreds of thousand of images, one does not always have access to that many images. Further, even a data collection of this magnitude may not be possible because of limited access to the relevant environment and objects. An example would be if one wants to train a detector that can find weapons in the territory of the adversary, but only a few images are available for learning.

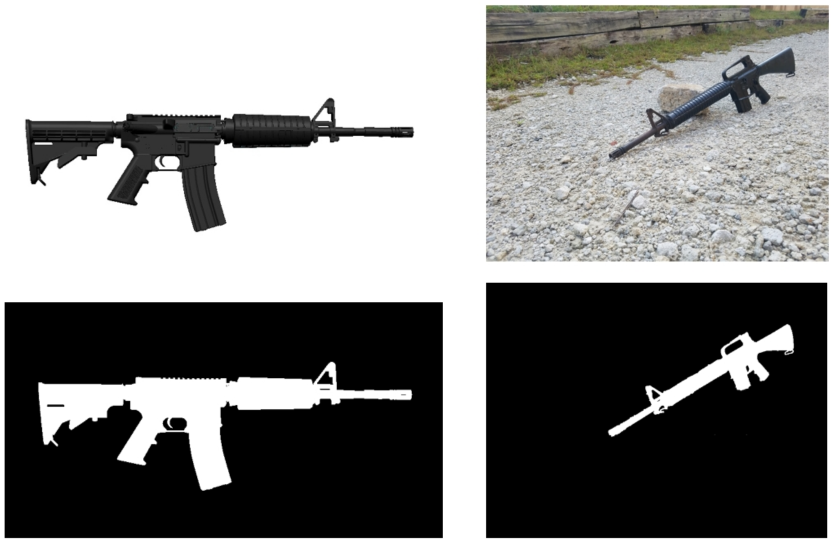

Figure 10: Cutouts of rifles: Upper left is a CAD model of a rifle and on the upper right is a real image of a rifle. Below are the corresponding masks.

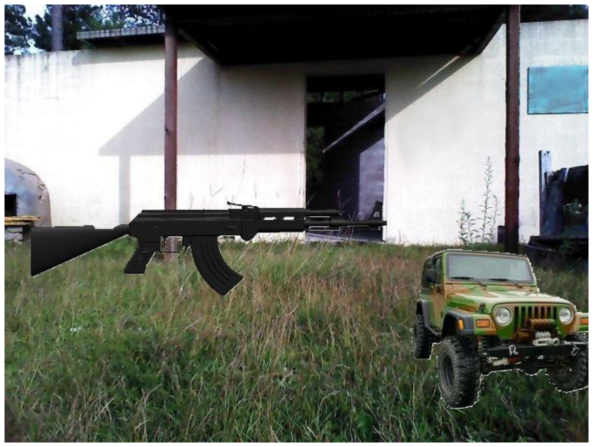

Figure 11: Augmented image: A rifle and a distractor object, vehicle, are pasted on a background image of the operationally relevant environment.

One method to overcome this limitation is to create augmented data using an image synthesis technique.[16] Annotated training images are synthesized for instance detection using a cut and paste approach to achieve patch level realism. We use the code that is made publicly available on Github.[17] The core idea is to cut out the object from the available images, manipulate it by scaling, rotating, blurring, and changing brightness and contrast, and finally paste it on different backgrounds.

We illustrate the cut and paste image synthesis approach for a *rifle* object in Figure 10. We start with real images of the rifle, an example can be seen in the upper right, and synthetic images, like the CAD snapshot shown in the upper left. Next we create the corresponding masks that indicate which pixels belong to the rifle as seen in the bottom of Figure 10. The pixel masks are used to manipulate the objects so they are in different orientations, scales, etc. from the original versions, and then these rifles are pasted on top of different operationally relevant backgrounds as seen in Figure 11.

This cut and paste technique can be performed to generate large amounts of additional training data at the cost of simply defining the pixel-wise mask for the original objects. Although this can be tedious, there

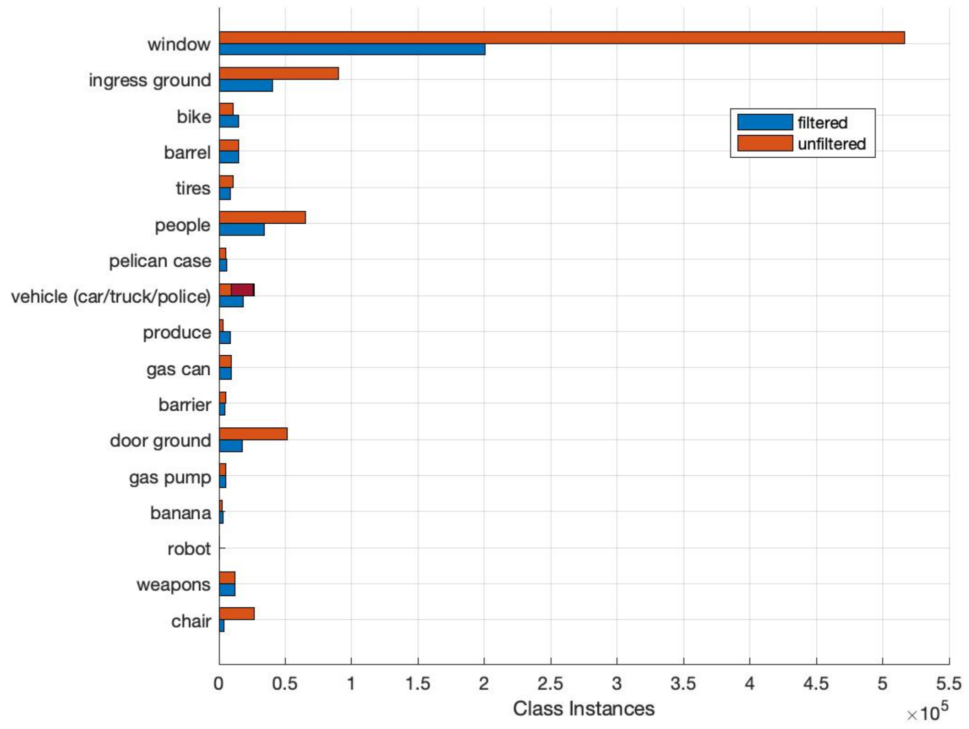

Figure 12: Number of training instances of classes for the unfiltered dataset, and after filtering out high-frequency classes and grouping related objects into superclasses.

are usually relatively few examples of the object of interest available to us to actually perform this task. So the annotation effort is extremely low in comparison to the thousands of synthesized images we can create for training.

4.3.2 Sampling Approach

Using video for training data facilitated quicker collection of new object instances and allowed use of the Figure Eight labeling platform that automated inter-frame tracking to speed up human annotation. However, video also contributed to significant class imbalances due to an unequal distribution of instances across all classes throughout the scene. To mitigate this issue, we identified high-frequency objects such as windows and doors, and then filtered the data by removing any images from the training data that contained only labels of objects in the high-frequency list.

A second approach to address the class imbalance issue was to combine multiple fine-grained object classes into a superclasses so long as it did not severely restrict the types of mission execution commands the robot could a receive. For example, the annotation labels car, truck, and police truck were combined to represent the class *vehicle*. Figure 12 shows that this superclass combination and filtering technique significantly reduced the skew seen in the class distribution.

4.4 Model Training and Evaluation

With the annotated objects in our operating environment, we trained a ResNet-50-FPN model using the maskrcnn-benchmark.[6] The final model used for RCTA experiments was trained with 51,864 annotated training images. We used the default constant learning rate of 0.0025 and weight decay of 0.0001 to train for 6,000 iterations. The stopping point was determined by running inference with checkpoint models generated every

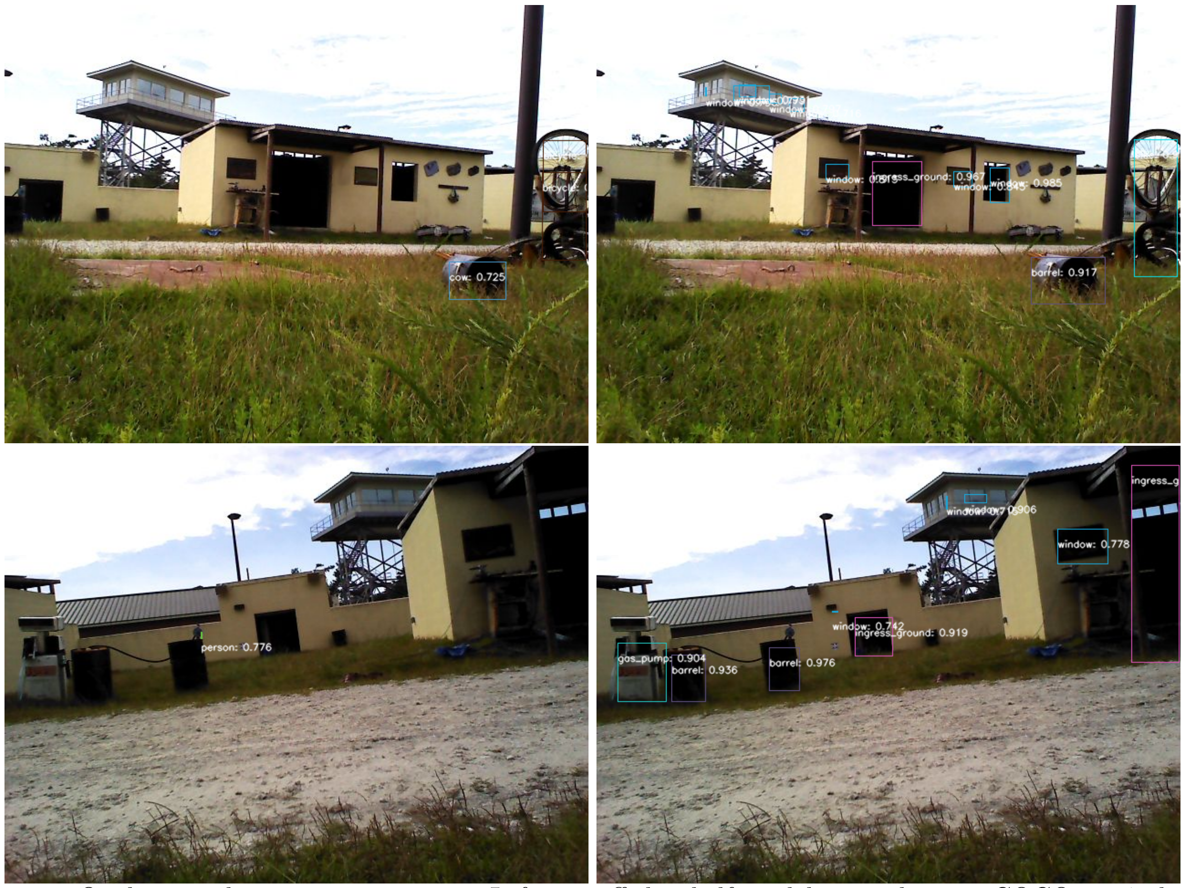

Figure 13: Qualitative detection comparison. Left: an off-the-shelf model trained on 81 COCO 2014 objects[3] Right: a customized model trained on scenario-relevant objects.

1,000th iteration on 9,047 annotated validation images not used for training. The network weights at the checkpoint where average precision peaks for most classes is selected for deployment. Training took ∼4 hours with 4 NVIDIA GeForce GTX TITAN X GPUs.

4.4.1 Comparison to off-the-shelf models

Up until this point we have assumed that an off-the-shelf object detection model would be insufficient for the perception needed in operationally relevant environments for dynamic mission execution. This is because the military relevant operating environment used for human-robot mission execution is quite different than the benchmark datasets commonly used in the research community. To demonstrate this, we run the COCO 2014 model[3] on images from the operating environment described in Section 2.

As seen on the left in Figure 13, the COCO 2014 model fails to detect objects such as barrels, gas pumps, windows, and ingresses, and produces false positives for environment-irrelevant objects like cow. Using our trained model, we can see the detection improvements on the right hand side of Figure 13. The use of domain specific annotations clearly improves detection performance and provides the robot with the perception cues is needs to execute mission commands.

To further quantify the improvements on object detection using our trained model versus a COCO model, we calculated the mean Average Precision (mAP) for both models. The average precision of each object of interest is found by first calculating the Intersection over Union (IoU) which is defined as the area of overlap divided by the area of union. The mAP values were calculated by running inference on 3,317 frames from videos not used for training but still containing a representative distribution of class instances.

COCO labels	remapped labels
person	people
car	vehicle
truck	vehicle
bicycle	bike
motorcycle	bike

Table 2: COCO label re-mappings to facilitate a fair comparison to our trained model.

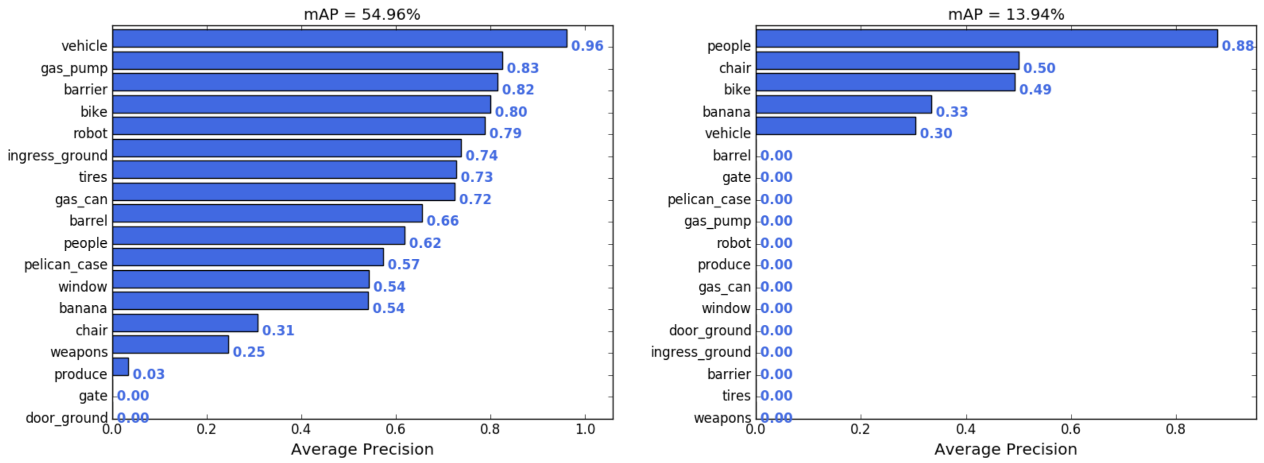

Figure 14: Comparison of mean Average Precision (mAP) results on test data from the operating environment. Left: results using a model trained with images from the operating environment. Right: results for an off-the-shelf model trained on 81 COCO 2014 objects.[3]

For a fair comparison, some label re-mapping was done for the COCO model. These re-mappings are described in Table 2. This remapping was done after COCO ran inference using its own classes and did not affect its accuracy. The only overlapping classes between our model and the COCO 2014 model were people, chair, bike, banana, and vehicle. This highlights the domain gap between academic research benchmarks and our real-world application. Figure 14 shows the mAP comparison for the two models. The COCO model did have a higher AP value in the people and chair classes but did slightly worse in the other classes.

4.4.2 Label filtering evaluation

The high frequency class filtering and superclass combination played a significant role in the performance of our trained model. Table 3 shows the mAP for our model trained on the unfilted raw annotations versus a model trained on data that filtered the high frequency objects and combined related classes into superclasses. These significant improvements lead us to deploy the model trained with the filtered data, and was also the model used in the COCO comparisons provided previously.

Even with the filtering, however, there is still room for improvement. Figure 15 provided a more in-depth analysis of our model by showing the number of true and false positive detections. The false positive rate is still quite high for many classes, most notably for window, ingress ground, and the produce classes.

4.4.3 Image synthesis evaluation

We also provide an analysis of learning with the cut and paste image synthesis technique previously discussed. We perform this evaluation for a single object class, *rifle*. Again, we have very few training images with a rifle present so additional data is collected via image synthesis. We train two Faster R-CNN models, one which is trained using only real-world image examples of the rifle, and one using only synthesized images of the rifle.

Figure 16 (left) typical correct, missed, partial, and false detections are shown. To evaluate the performance we compared the precision-recall to the normal method of hand-labeled training data (Figure 16 right). As one might expect, the model trained with augmented data does not perform as well as the regular one, but it still

Class	w/ High Freq. Data	Filtered Data
banana	0.00%	54.04%
barrel	35.42%	65.04%
barrier	0.00%	78.19%
bike	0.00%	74.80%
chair	16.11%	14.59%
electrical_box	3.66%	n/a
gas_can	11.68%	70.78%
gas_pump	42.36%	75.95%
gate	3.21%	n/a
ingress_ground	31.77%	72.77%
pelican_case	0.00%	54.92%
people	56.76%	60.98%
produce	0.00%	2.80%
robot	0.00%	77.44%
tires	18.70%	68.50%
vehicle	71.85%	96.09%
weapons	0.00%	1.78%
window	28.25%	52.76%
mAP	**16.83%**	**51.19%**

Table 3: Comparison of mean average precision for model trained with all data vs. filtered data with some high frequency object instances removed

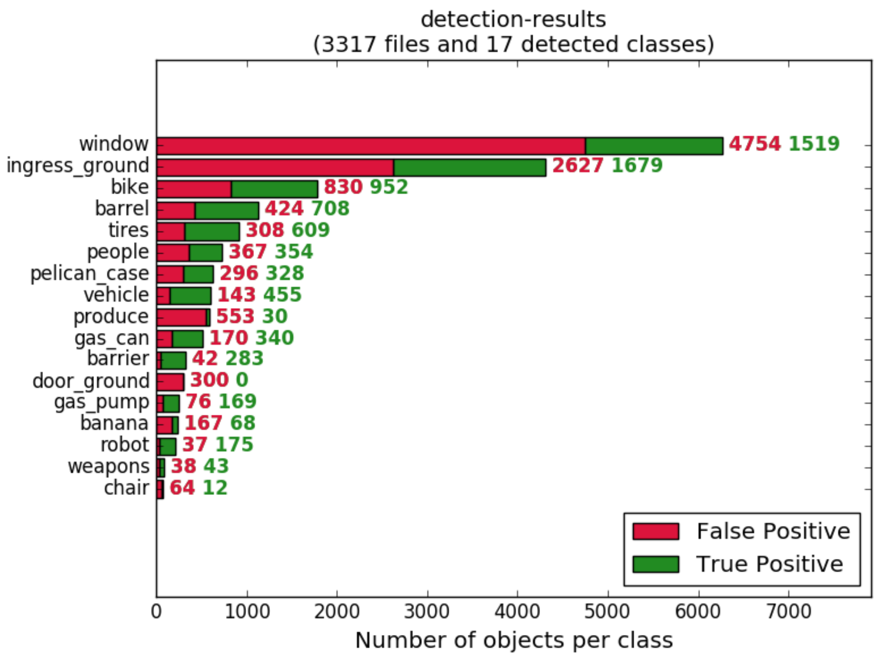

Figure 15: Evaluation of the model trained with images from the robot's operating environment.

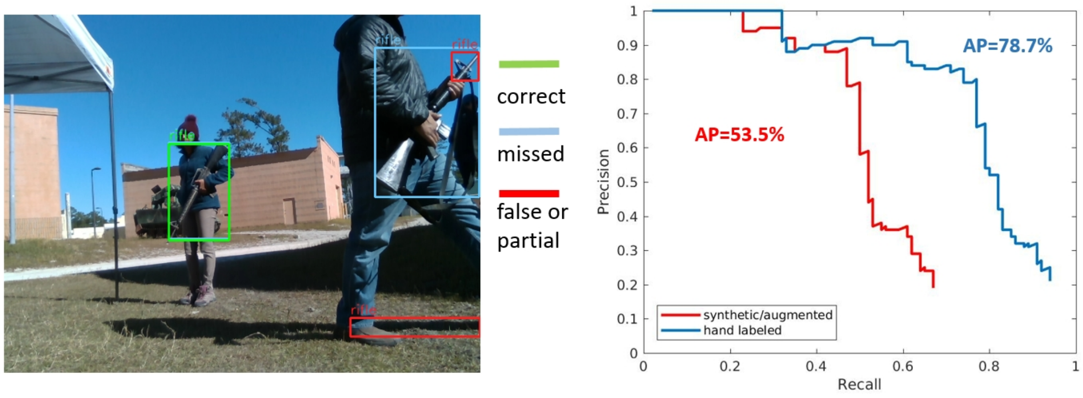

Figure 16: Left: Qualitative evaluation of rifle detection trained with synthesized images only. Right: precision-recall curves comparison for a model trained with synthesized images only and a model trained with regular hand-labeled images.

	normal scenarios	**difficult scenarios**
hand-labeled	78.7%	33.2%
synthesized/augmented	53.5%	16.8%

Table 4: Average precision of rifle detection for a model trained with synthesized images only and a model trained with regular hand-labeled images for difficult and normal rifle scenarios.

has an acceptable performance. In particular, it has about 100% precision at 20% recall. For our application, a 20% recall is acceptable, because we see an individual rifle many times.

One shortcoming of our augmentation method is that we do not model how a rifle is occluded by a person. The left image of Figure 16 qualitatively shows the performance of the rifle detection model that is trained only using synthesized images. The plot to the right compares the precision-recall curves of both models. The model trained with real world annotations does outperform the synthesized images only model, yet the trade-off comes at the cost of much greater annotation effort. Table 4 breaks down this comparison into two types of detection scenarios. Difficult scenarios are defined as those where most of the rifle is occluded by the body of a person or another object. In these difficult cases the performance of both detection models is much worse.

We use this analysis to suggest that the use of only synthesized images, which requires little to no human labeling effort, has potential to train high performing models. We continue to work on how to make the synthesized images appear more photo-realistic and how to train models for the more difficult rifle detection scenarios.

5. PERCEPTION DURING MISSION EXECUTION

The overall motivation of the perception pipeline is to generate models that provide environmental context and identify landmarks necessary for a robot teammate to perform higher level mission execution. Providing a quantitative evaluation of this higher level task in beyond the scope of this paper. Instead, we provide some qualitative evidence of the perception stack in action that demonstrates the utility of the pipeline we have described.

The figures and descriptions in this section are extracted from a mission execution demonstration performed in an environment different from the location in which training data was collected. Several other similar missions were also performed at the training site location (described in Section 2).

Figure 17 highlights some of the successful object detection results throughout the mission execution. On the left, a *person* is identified. This identification connects with a tracking system that allows the robot to successfully execute the command, “Follow me,” which is given by the human teammate. On the right, a *barrel* is identified, which allowed the robot to successfully execute the command, “Go behind the nearest barrel.”

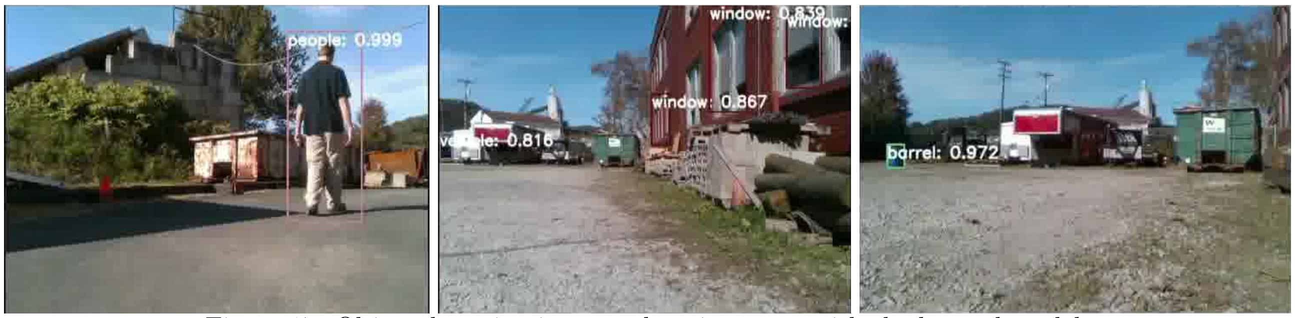

Figure 17: Object detection in a novel environment with the learned model.

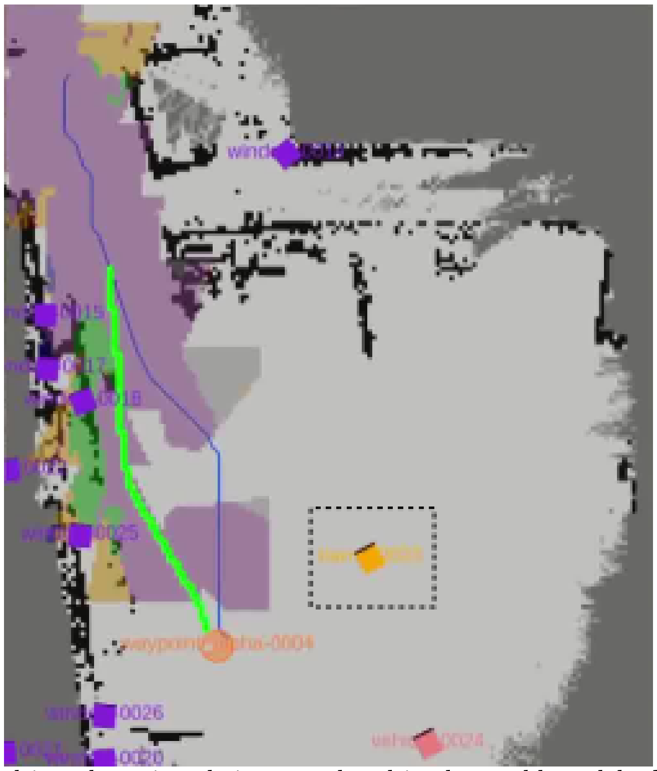

Figure 18: Visualization of object detections being populated in the world model which is used by the robot to execute higher level commands.

Figure 18 is a visualization of several components of the entire autonomy architecture used to execute the mission. The perception and planning information is displayed on top of the map generating by a SLAM algorithm. As objects are detected they are placed into a world model, and these entities can be seen as square blocks in the figure. The *barrel* identified in the right image of Figure 17 is outlined by a dashed line in Figure 18 (not part of the true visualization) for easy reference. The blue and green lines represent different planned paths for commands given to the robot.

Without the detections provided by the perception stack, the execution of the mission would be significantly harder. The placement of these objects in the world model to use as landmarks has been shown in many environments to allow the robot to execute a variety of commands. This includes environments are not represented in the training data used to learn the object detection model. Although we do not provide a quantitative evaluation the successful mission execution in these environments suggests at least some robustness of the model to novel environments. Yet we note there are still significant improvements that could be made and we touch on those in our discussion of future work in Section 7.

6. RELATED WORK

Visual perception has made huge advancements in the last decade in part because of the effort the research community has gone through to collect large annotated sets of imagery to use for training. The emergence of ImageNet[5] allowed deep learning techniques to become feasible, and ultimately resulted in state-of-the-art performance in image classification.[18,19] By leveraging pre-trained models with this large dataset, object detection and segmentation networks[20–22] have been generated by fine-tuning with other more densely labeled datasets such as MS COCO,[3] Pascal VOC,[4] Cityscapes,[1] KITTI,[2] and others.

Although these datasets and associated challenge problems are becoming more sophisticated, e.g., from bounding box detection to object segment masks to pixel-wise semantic segmentation, these datasets are largely representative of commercial applications. In other words, training data is representative of objects and scenes that are found in highly populated urban environments. Even tasks such as keypoint detection focus largely on keypoints of humans only.

This presents a challenge when the goal is to deploy a perception pipeline capable of detecting objects and keypoints in highly unstructured environments. The domain gap between what currently exists in the research community, and military relevant operating conditions can be vast. Curation of datasets that are more representative of unstructured environments are beginning to emerge. Donlon et al.[23] discuss the benefits of having diverse training data when learning visual perception models for robot applications in unstructured environments. They lay out how heterogeneous onboard sensor suites and a set of heterogeneous platforms could provide this type of data collection. The Robot Unstructured Ground Driving (RUGD) dataset[24] is a collection of videos captured in unstructured environments, where the robot used to collect the data is primarily operating in off-road conditions. The focus of RUGD, however, is on establishing semantic segmentation ground truth for terrain identification as a perception need for autonomous navigation tasks. As a result, this data includes very few object class labels and does not differentiate instances of objects in the ground truth.

Directly deploying an off-the-shelf model that has been trained on the previously mentioned publicly available data will not produce results reliable enough to correctly execute missions defined by the objects in the environment (as seen in Figure 13). Thus, establishing a pipeline to collect and annotate data from the operating environment to train and deploy models is essential for reliable mission execution.

7. CONCLUSION AND FUTURE WORK

Within the RCTA program we were able to integrate several perception modules and develop a perception pipeline that allowed for rapid data collection, annotation, and model training for ground robot applications. The perception modules provided localization and identification of environmental landmarks and object keypoints for a robot to use in dynamic mission execution alongside a human teammate. The development of such a pipeline was necessary for RCTA experimentation as the robot's operating environment was highly unstructured and included several objects not commonly found in academic and commercial benchmark datasets. Although our described perception pipeline helped adapt the robot's perception modules quickly to new operating environments there are still many improvements we will look at in future work.

Transitioning manned-unmanned teaming capabilities to the field, specifically battlefield environments for military applications will require developing algorithms and approaches that can learn on the fly to improve the performance of perception models in real time and respond intelligently to situations never encountered before. Additionally, in many military applications, learning and inference has to done at the battlefield edge, i.e., without reliance on cloud or high processing resources. In this work, we rely on off-line training with multiple GPUs and on-line inference with a single GPU. Our next steps will include new techniques to learn and infer using edge resources without significantly degrading performance. Several research areas in computer vision are showing promise in this area including end-to-end incremental learning where new data is introduced over time, while maintaining a small exemplar set of samples from older data. Other areas to consider are human-in-the-loop learning, semi-supervised and unsupervised learning.

Finally, we will explore a more extensive study on how to use simulated environments and synthetic data for training purposes. High fidelity simulation has the potential to close the domain gap between synthetic and real data. As this allows us to generate large amounts of labeled training data without human intervention we see this as essential research to explore to help with challenges previously mentioned, i.e., lack of labeled training data. By exploring approaches that transfer models learned in simulated environments to real-world environments we should be able to address the shortcomings we saw in our evaluation, such as the inability to identify rifles that are occluded by humans (discussed in Section 4.3.1).

REFERENCES

[1] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B., "The cityscapes dataset for semantic urban scene understanding," in [*Proceedings of the Conference on Computer Vision and Pattern Recognition*], 3213–3223, IEEE (2016).

[2] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for autonomous driving? the kitti vision benchmark suite," in [*Proceedings of the Conference on Computer Vision and Pattern Recognition*], 3354–3361, IEEE (2012).

[3] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., "Microsoft coco: Common objects in context," in [*Proceedings of the European Conference on Computer Vision*], 740–755, Springer (2014).

[4] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision* **88**, 303–338 (June 2010).

[5] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision* **115**(3), 211–252 (2015).

[6] Massa, F. and Girshick, R., "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch." `https://github.com/facebookresearch/maskrcnn-benchmark` (2018). Accessed: July 2019.

[7] Narayanan, V. and Likhachev, M., "Perch: Perception via search for multi-object recognition and localization," in [*Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*], (May 2016).

[8] Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., and Daniilidis, K., "6-DoF object pose from semantic keypoints," *Proceedings - IEEE International Conference on Robotics and Automation* , 2011–2018 (mar 2017).

[9] Newell, A., Yang, K., and Deng, J., "Stacked hourglass networks for human pose estimation," in [*European conference on computer vision*], 483–499, Springer (2016).

[10] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "Pytorch: An imperative style, high-performance deep learning library," in [*Advances in Neural Information Processing Systems 32*], Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., eds., 8024–8035, Curran Associates, Inc. (2019).

[11] Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., and Yeung, S.-K., "Scenenn: A scene meshes dataset with annotations," in [*International Conference on 3D Vision (3DV)*], (2016).

[12] Osteen, P. R., Owens, J. L., and Kaukeinen, B., "Reducing the cost of visual DL datasets," in [*Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*], Pham, T., ed., **11006**, 121 – 139, International Society for Optics and Photonics, SPIE (2019).

[13] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J., "ElasticFusion: Dense SLAM without a pose graph," in [*Robotics: Science and Systems (RSS)*], (July 2015).

[14] Kazhdan, M. and Hoppe, H., "Screened poisson surface reconstruction," *ACM Trans. Graph.* **32** (July 2013).

[15] Salti, S., Tombari, F., and di Stefano, L., "Shot: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Underst.* **125**, 251–264 (2014).

[16] Dwibedi, D., Misra, I., and Hebert, M., "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in [*The IEEE International Conference on Computer Vision (ICCV)*], (Oct 2017).

[17] https://github.com/debidatta/syndata-generation.

[18] Simonyan, K. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," in [*Proceedings of the International Conference on Learning Representations*], (2015).

[19] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [*Proceedings of the Conference on Neural Information Processing Systems*], 1097–1105 (2012).

[20] Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," in [*Proceedings of the Conference on Computer Vision and Pattern Recognition*], 580–587, IEEE (2014).

[21] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *transactions on pattern analysis and machine intelligence* **40**(4), 834–848 (2018).

[22] Lin, G., Milan, A., Shen, C., and Reid, I. D., "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.," in [*Proceedings of the Conference on Computer Vision and Pattern Recognition*], **1**(2), 5 (2017).

[23] Donlon, J., Young, M., Wigness, M., and Hayes, C., "An analysis on data curation using mobile robots for learning tasks in complex environments," in [*Proceedings of Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*], **11006**, SPIE (2019).

[24] Wigness, M., Eum, S., Rogers, J. G., Han, D., and Kwon, H., "A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments," in [*Proceedings of the International Conference on Intelligent Robots and Systems*], 5000–5007, IEEE (2019).