Annotated Reconstruction of 3D Spaces Using Drones

Suraj Nair
Computer Science
California Institute of Technology
Pasadena, United States
snair@caltech.edu

Anshul Ramachandran
Computer Science
California Institute of Technology
Pasadena, United States
aramacha@caltech.edu

Peter Kundzicz
Computer Science
California Institute of Technology
Pasadena, United States
pkundzic@caltech.edu

***Abstract*—As the fields of robotics and drone technologies are continually advancing, the challenge of teaching these agents to learn and maneuver in the real world becomes increasingly important. A critical component of this is the ability for a robot to map and understand its surrounding unknown environment, both in terms of physical structure and object classification. In this project we tackle the challenge of mapping a 3D space with annotations using only 2D images acquired from a Parrot Drone. In order to make such a system operate efficiently in close to real time, we address a number challenges including (1) creating a optimized version of Faster RCNN that can operate on drone hardware while still being accurate, (2) developing a method to reconstruct 3D spaces from 2D images annotated with bounding boxes, and (3) using generated 3D annotations to complete drone motion planning for unknown space exploration.**

***Index Terms*—Computer Vision, Robotics, Motion Planning**

I. Introduction

As shown in Figure 1, a full real time system for 3D annotated reconstruction contains many moving parts. Throughout the project, we worked both on the vision components, as well as directly with the drone hardware and camera system. The primary vision related components of the project were the optimizations of Faster RCNN [1] for close to real time object detection and classification, and the generation and visualization of a 3D reconstruction from 2D images with bounding box annotations. A large portion of the work related to working with the drone hardware, tracking the drone location using SLAM, and math behind reconstruction can be found in a full length version of the paper, as the focus of this paper is on the vision components of the system outlined in Figure 1.

For fast object detection and classification we make several optimizations to the Faster RCNN [1] algorithm proposed by *Ren et al* in 2015, where we try different ways of decreasing the width and depth of the network to suit our simpler object detection task. The dataset we train on is the City-Scapes dataset [2], which contains over 20,000 images of urban street scenes, with 30 different classes including vegetation, pedestrians, cars, road, etc. While there are several methods for object detection, such as Multibox [3], we chose to base our algorithm off of Faster RCNN due to its speed, since our primary focus is to increase prediction speed to the point that it can run close to real time on a drone.

For 3D reconstruction, we first needed to develop a pipeline that would be able to transform classified 2D bounding boxes to 3D spaces. This involved scaling the bounding boxes to world units and rotating around the camera position to match the orientation vector of the camera. We treated the world as a 3D voxeled grid since fine segmentation of the space does not lead to an efficient and easy-to-use digital representation. Also, we developed a motion planning algorithm that determines the regions of the world that have not been viewed in any frame, and then computes a set of locations and orientations for the drone to visit. This motion planning piece leads the drone to take more images and predict more classified bounding boxes, which in turn improves the 3D world reconstruction in a continuous control loop.

II. Optimized Detection and Classification

A critical component to having the system run in close to real time is having fast object detection and classification that can run on simple hardware. Due to the nature of our task, the number of classes (30) is small compared to the original Faster RCNN paper, so we tried a number of optimizations with the goal of speeding up prediction time while still providing sufficiently good annotations to create a realistic 3D reconstruction.

The standard FRCNN contains 3 main components. First, a base layer, which generates feature rich representations of the image. Second, a Region Proposal Network (RPN), which provides an "attention mechanism", describing what areas to focus on and what anchors to use in each region. Finally, a classifier layer, which takes the proposals from the RPN and the representations from the base layer, and predicts objects and their classes. In our implementation, we use a ResNet50 [4] base. In our experiments we try modifications to different parts of Faster RCNN and create three alternative models: Shallow NN Base FRCNN, Shallow Classifier FRCNN, and Narrow NN Base FRCNN. We build our code-base using Tensorflow and Keras FRCNN [5].

A. Narrow NN Base FRCNN

The idea behind Narrow FRCNN was to shrink the width of several layers in the base network in an attempt to speed up prediction time. More specifically, we reduced then number of

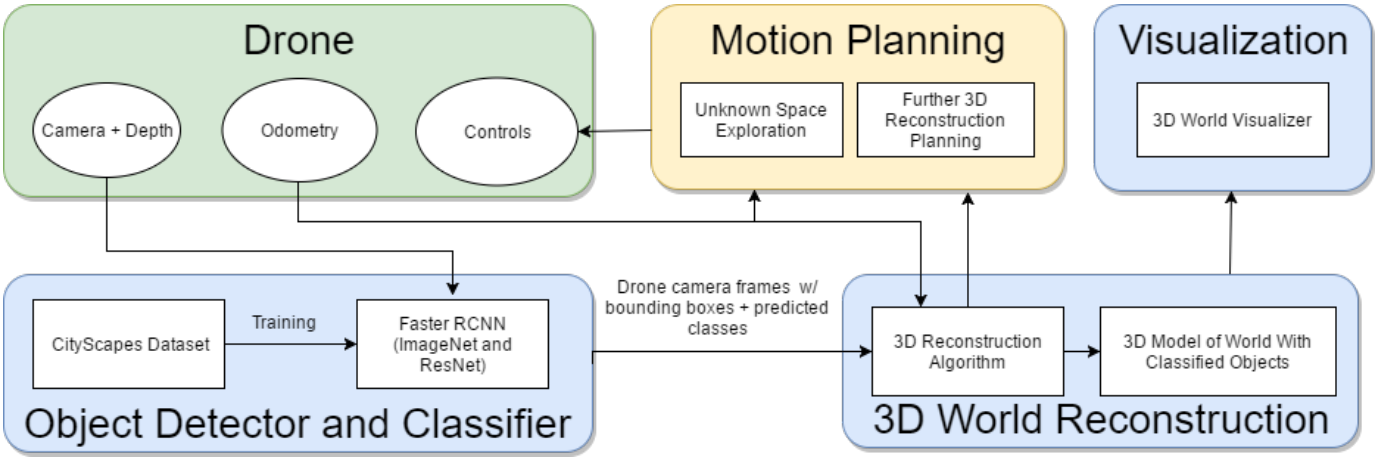


Fig. 1: **3D Annotated Reconstruction Flowchart:** The drone collects pictures and relays them to an image classification system. The object detector and classifier is trained on a cityscapes dataset, so that it can efficiently determine bounding boxes from the drone's feed. These bounding boxes are then used for 3D world reconstruction and further motion planning for the drone.

filters in all of the convolutional blocks in stage 2 and 3 by a factor of 2 compared to the ResNet50 base.

B. Shallow Classifier FRCNN

The idea of Shallow Classifier FRCNN was simply to remove several layers completely from the classifier portion of the network. Specifically, the base classifier portion of the network, which originally has 9 convolutional blocks and 3 pooling blocks, was reduced to 3 convolutional blocks and one pooling block.The logic behind this change was that since our data has a relatively small number of classes (30) compared to the data used in the original FRCNN paper, the classifier portion of the network really does not need to be nearly as complicated, and for our classes we should get comparable accuracy even after removing these layers for a speedup.

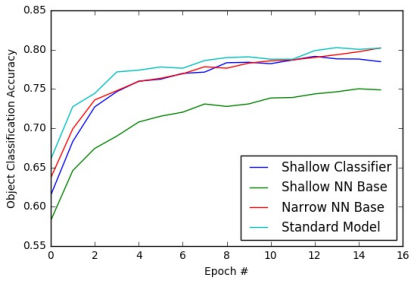
C. Shallow NN Base FRCNN

In this modification, we throw out a large number of layers from the base component of the standard FRCNN. Specifically, we use only 6 convolutional layers out of the original 9 from the first stage of the base, we use only 6 of the 12 convolutional layers from the second stage, and we only use 6 of the 15 convolutional layers from the third stage. Generally, we expect drone images to be lower resolution than those in most image detection datasets. Our logic was that we therefore expect that the large depth of ResNet 50 is unnecessary for extracting the important features and we should be able to get good results with fewer layers.

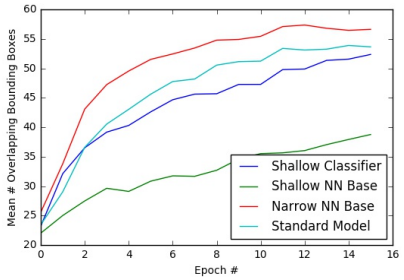
D. Accuracy and Loss Analysis

The primary criteria of our optimizations is efficiency without compromising performance accuracy.

The class accuracies and the bounding box overlap on the dataset provides idea of model performance (Figure 2). The bounding box overlap gives an idea of how well the RPN is working, and the class accuracies gives an idea of how well the classifier component of the network is working.



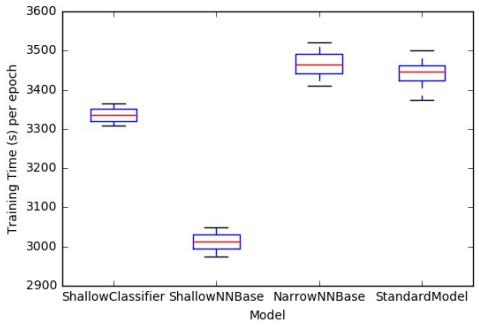
(a) Object Class Accuracy



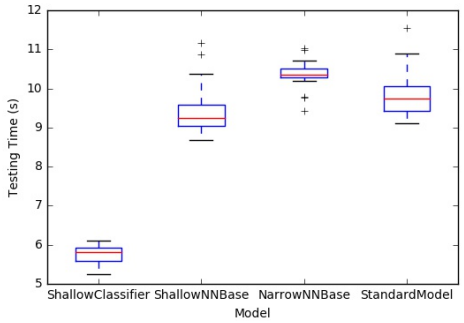
(b) Mean Overlapping Bounding Boxes

Fig. 2: **Bounding Box and Class Accuracy:** Shown are the percent accuracy for class predictions and the mean number of overlapping bounding boxes, where overlapping is defined as the proposed and actual bounding boxes having an IOU $\geq$ 0.3.

From Figure 2 we can conclude that the Shallow NN Base architecture is substandard compared to the other architectures. However, we can also see that the Shallow Classifier architecture and Narrow NN Base architecture are comparable in performance to the Standard Model. With this established,



(a) Training Time Per Epoch



(b) Time To Predict A Single Image on CPU

Fig. 3: **Training and Prediction Speed:** We examine the training time per epoch and prediction time per image for each of the Faster RCNN modifications. Note the prediction time is on a CPU, more realistic to the sort of hardware one might expect on a drone. We see that during training time, every architecture except the Shallow NN Base model has roughly the same training time per epoch. More interesting is the testing time, where we see that the Shallow Classifier is noticeably faster per image than the other models.

we look into the speed improvement provided by the new architectures.

E. Efficiency Analysis

In Figure 3 we look at both the training time per epoch and testing time per image for each architecture.

The results from the testing plot show significant promise in reducing prediction time. The time to predict a single image on a CPU drops from around 10 seconds to less than 6 seconds using the Shallow Classifier architecture, which showed similar results to the standard model in terms of accuracy and loss.

F. Robustness Analysis

After analyzing the accuracy, loss, and other quantitative metrics, we also qualitatively assessed each models ability to adapt to differences seen in the real world. We first look at robustness of the different models to changes in height. We varied the height of the drone from 0 to 20 meters pointing in the same direction and compared the bounding

(a) Height Robustness (a) Lighting Robustness

Fig. 4: **Model Robustness:** For the optimized Shallow Classifier, the left images show bounding boxes and confidences images of the same scene at different heights, and the right images show these for images at different times.

box classifications and accuracies. We also looked qualitatively at robustness to lighting changes by taking images from the same point of view at different times from 1 PM to 10 PM. We found that all models performed at roughly invariant levels until the amount of sunlight dropped significantly (late evening to night). Some images of the Shallow Classifier's behavior under robustness testing is shown in Figure 4.

III. 3D World Reconstruction

A. Overall approach

The world is treated as a 3D rectangular grid of voxels, initialized with dimensions and a resolution value (which is the edge length of a singular cubic voxel). Each voxel contains an array of confidence values, of size $(\text{num classes}+1)$, where all but the last corresponds to a confidence that we believe that the voxel corresponds to the corresponding object class. Values are updated whenever a bounding box in an image frame that contains that voxel is labeled with that object class. The last value in the array is a confidence that no object is in that voxel, and is updated for all voxels that are visible in a frame but do not correspond to any bounding boxes. This allows us to both triangulate the rectangular pyramidal views and to remove incorrect detections of objects.

For each frame, we use the trained, modified FRCNN to predict the object classes and bounding boxes. From the stereoscopic camera on the drone unit, we can get a rough estimate of how far the front of the object is, d_{front}, by taking the mean distance to the pixels within the bounding box. Since we have no idea how deep the object is, we calculate the voxels that the bounding box planar surface would occupy

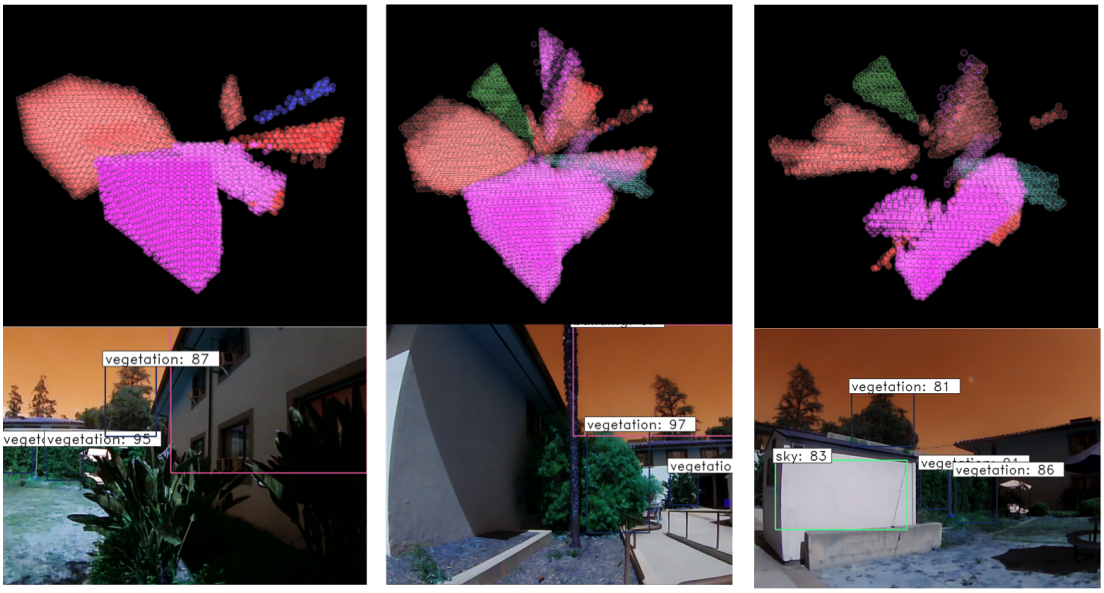


Fig. 5: **Full Courtyard Reconstruction:** Demonstration that as we get more frames from different positions and orientations, we go from large rectangular pyramidal blocks of where we believe objects could be to more clear boundaries and shapes of objects.

for each camera-to-frame world distance starting from d_{front}. This forms a truncated rectangular pyramidal volume where the object could occupy, with vertex at the drone position in the direction of the camera's orientation, clipped by the faces of the rectangular world. We then label a voxel with an object if the confidence value of the particular object label, aggregated over several image frames holding that voxel, is above a set threshold.

B. Results

We used VisPy [6] with an OpenGL backend to visualize our reconstructed 3D worlds. We were able to create an interactive interface for the world, where a user can rotate, translate, and zoom in and out to explore different parts of the reconstruction, screenshots of which are presented in Figure 5.

Figure 5 shows the progression of a complex reconstruction of a courtyard over 50 frames with associated bounding boxes and classification. Note that the drone stayed in the center of the courtyard area. It did not go 'behind' obstacles to completely truncate the rectangular pyramidal object predictions.

IV. Motion Planning

We first determine the voxels that have not been viewed in any frame. Using K-means clustering, we computed cluster centers for the unseen voxels. We then find points within each unseen cluster of voxels that are near the fringe of the cluster, not too close to categorized objects (to prevent crashing), and not too far from the cluster center in the z-direction (to prevent drone roll/tipover when taking images). Examples of position and path determination of the drone are shown in Figure 6.

Using the provided locations, the ROS script then directs the drone to travel to those locations. Specifically, it slowly applies the necessary linear movement and rotation in the x y and z directions to match the desired position and orientation. It is applied in small steps to prevent drift.

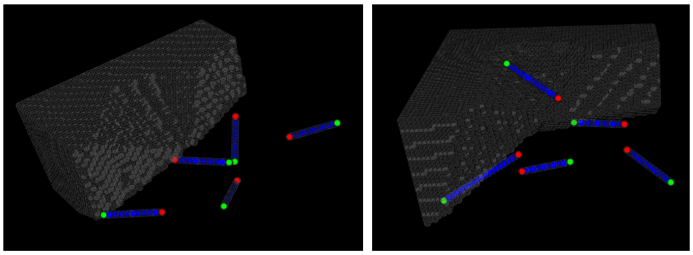


Fig. 6: **Two Examples of Position/Orientation Determination for Drone Motion Planning:** The transparent white voxels correspond to voxels that have been seen from at least one processed frame. The green dots correspond to positions the drone should visit, and taking images in the direction of the corresponding blue vector would maximize viewing of unseen voxels. The red points are cluster centers calculated from K-means clustering on the unseen voxels.

V. Conclusion

Ultimately, we found that with the CityScapes dataset and a modified Faster RCNN, one can train a drone to detect and classify common outdoor objects. Then using our 3D Reconstruction algorithm, we can create an annotated 3D representation of the world and use motion planning to navigate this world.

Future work can focus on further speeding up the image detection and classification, and predicting shape to refine the 3D reconstruction.

REFERENCES

[1] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. CoRR, 2015

[2] Cordts, Marius and Omran, Mohamed and Ramos, Sebastian and Rehfeld, Timo and Enzweiler, Markus and Benenson, Rodrigo and Franke, Uwe and Roth, Stefan and Schiele, Bernt. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. CVPR, 2016

[3] Christian Szegedy and Scott E. Reed and Dumitru Erhan and Dragomir Anguelov *Scalable, High-Quality Object Detection*. CoRR, 2014

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition*. CoRR, 2015

[5] Yann Henon. *keras-frcnn*. GitHub repository, https://github.com/yhenon/keras-frcnn , 2017

[6] *Vispy*. GitHub repository, https://github.com/vispy/vispy, 2017