3D Localization of Defects in Facility Inspections

Nicholas Califano
University of Central Florida
3D LOCALIZATION OF DEFECTS IN FACILITY INSPECTIONS

by

NICHOLAS CALIFANO
B.S. University of Central Florida, 2019

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2020

Major Professor: Gita Sukthankar
Wind tunnels are crucial facilities that support the aerospace industry. However, these facilities are large, complex, and pose unique maintenance and inspection requirements. Manual inspections to identify defects such as cracks, missing fasteners, leaks, and foreign objects are important but labor and schedule intensive. The goal of this thesis is to utilize small Unmanned Aircraft Systems with onboard cameras and computer vision-based analysis to automate the inspection of the interior and exterior of NASA’s critical wind tunnel facilities. Missing fasteners are detected as the defect class, and existing fasteners are detected to provide potential future missing fastener sites for preventative maintenance. These detections are performed in both 2D on the images and in 3D space to provide a visual reference and real world location to facilitate repairs. A dataset was created consisting of images taken along a grid-like pattern of an interior tunnel section in the AEDC National Full-Scale Aerodynamics Complex at NASA Ames Research Center. To localize the defects, object detection was used to create image level bounding boxes of the fasteners and missing fasteners, and photogrammetry was used to create a correspondence of 3D real world locations and 2D image locations. The image level bounding boxes and the 2D to 3D correspondences are then combined to determine the 3D location of the defects. On the test data, the method was able to successfully localize all of the objects in 3D space with no false positives.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Gita Sukthankar, for her guidance and support with my thesis, her encouragement to pursue higher education, and providing me with phenomenal education during my undergraduate and graduate studies.

I would like to express my sincerest gratitude for my mentor, Dr. David Murakami, for creating the opportunity to work with NASA Ames Research Center and supporting me throughout my graduate studies journey.

I want to thank Joe Adamson for his contributions to the control and position feedback systems of this project. Without you this drone would have literally never gotten off the ground.

A huge thank you goes out to my parents for their love and support throughout my entire life.

I am grateful for the support from Fluid Mechanics Lab at Ames Research Center for sponsoring this project, AEDC for allowing us access to the NFAC, and the NASA Advanced Supercomputing (NAS) Division at Ames Research Center for support with the required computational resources.

This thesis expands on my previous work published in [1].
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. viii

LIST OF TABLES ................................................................. x

CHAPTER 1: INTRODUCTION ...................................................... 1

CHAPTER 2: RELATED WORK .................................................... 3

2.1 Automated Inspections ..................................................... 3

2.2 Object Detection ............................................................ 4

2.3 Photogrammetry ............................................................. 7

2.3.1 3D Object Detection ................................................... 9

CHAPTER 3: DATA COLLECTION ................................................. 11

3.1 Trial 1 ................................................................. 11

3.2 Trial 2 ................................................................. 13

3.3 Prototype Platform ........................................................ 14

3.3.1 Flight Sensors and Flight Controller .............................. 15

3.3.2 Simultaneous Localization and Mapping ........................ 15
# LIST OF FIGURES

2.1 R-CNN Pipeline .................................................. 5

2.2 Object Instance Pixel Size ....................................... 7

2.3 Pinhole Camera Model ............................................. 8

3.1 Grid Point Spacing without Position Feedback ................. 13

3.2 Example Images from First Data Collection ...................... 13

3.3 Example Image from Second Data Collection .................... 14

3.4 Visual Odometry Image vs Data Collection Image .............. 17

3.5 Ground Sampling Distance Visualization ....................... 18

3.6 System for Third Data Collection Trial ......................... 20

3.7 Third Data Collection Trial Full Setup ......................... 21

3.8 Comparison of Grid Point Spacing ............................... 22

3.9 Example of Object Labels ........................................ 25

4.1 Method Pipeline ................................................... 27

4.2 Mask R-CNN Architecture ....................................... 29

4.3 Optimized vs Unoptimized Anchor Boxes ....................... 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Combining Object Detections from Multiple Images</td>
<td>34</td>
</tr>
<tr>
<td>4.5</td>
<td>Ray Tracing to 3D Object Location</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>Object Precision vs Recall</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Instance IoU Example</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Table Title</td>
<td>Page</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>5.1</td>
<td>Frame-AP Performance</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Ablation Experiment</td>
<td>39</td>
</tr>
<tr>
<td>5.3</td>
<td>Instance-AP Performance</td>
<td>42</td>
</tr>
</tbody>
</table>
CHAPTER 1: INTRODUCTION

The National Full-Scale Aerodynamics Complex (NFAC) is the largest wind tunnel in the world, and supports both military and commercial programs in the aerospace industry. Maintenance of the facility requires routine inspections to be performed by technicians and engineers, but it is nearly impossible to perform frequent, thorough, manual inspections of the NFAC as there is more than one million square feet of interior surface area. Additionally, the NFAC surfaces contain complex geometry and ceilings that vary in height from 50’ to 250’, making manual inspections of some areas particularly difficult.

An initial investigation for inspecting the internal surfaces of the facility estimated 800 man hours per inspection. The inspection would need to be performed by specialized technicians and engineers with the aid of temporary structures such as scaffolding and ropes strung from the ceiling. Such an endeavor would be expensive in monetary costs due to the number of hours of specialized labor and potential safety concerns, but also in schedule costs as the tunnel is inoperable during the inspections. Additionally, there are several areas of the tunnel that are extraordinarily difficult for a person to access due to the complex geometry of the facility.

Researchers at the NASA Ames Research Center Fluid Mechanics Lab have proposed the use of small Unmanned Aircraft Systems (drones) to capture images along grid-like patterns of the facility surfaces, and to automate the process of defect detection with computer vision-based analysis. The computer vision systems must be able to provide the 3D location of the defect, as well as image location of the defect for visual aid. This thesis outlines, implements, and evaluates the data collection process and a pipeline to perform that analysis.

There are several potential benefits to using drones rather than manual labor for this application. For one, it reduces the required downtime of the tunnel as processing can be performed offline, after
the tunnel inspection. The only downtime would be during the data collection and maintenance repairs, which are unavoidable. The drones are relatively small and agile, and thus can access many hard to reach areas. The use of drones also reduce the long-term monetary costs as there is little cost after an initial investment for the hardware and software development. Additionally, this drone system could be implemented in other large facilities at NASA with little additional cost.
CHAPTER 2: RELATED WORK

2.1 Automated Inspections

Inspection is a common application for drones and robotics in general [2]. Most inspection work is repetitive and inspection of large or complex structures and facilities is additionally potentially dangerous. Robots typically replace humans in environments and tasks that are dirty, dull, and dangerous. Inspection meets at least two thirds of these criterion.

Recently, drones have become a popular platform for inspection robotics as reductions in cost and advancements in battery technology have made them a more viable platform. Wind turbines are a particularly common application of drone inspections due to their large size and complex geometry. The main goal of wind turbine inspection is to capture images of the structure for later, offline inspection or for real-time inspection with a technician present. The images captured by the drone often serve a dual purpose as it is typical to utilize that data for visual navigation and provide closed-loop feedback of the position of the drone relative to the structure [3, 4, 5].

For these robotics applications the main focus is designing a tool that allows remote operators to inspect the facility or structure, but not to create a completely automated inspection. In particular the goal of [5] was to eliminate the risks involved for a human to climb these structures by replacing the human with a drone. The goal of this thesis is to not only eliminate the human in the data collection process, but also the human in the inspection of the data as well.

Distinct from the use of drones, computer vision is also frequently used for inspections at large scales. A common application for computer vision methods is in manufacturing for defect and anomaly detection [6, 7, 8]. Ideally, every unit manufactured should be identical and so it can be relatively simple to identify anomalies and thus defects.
Computer vision has seen a rise in popularity in civil and structural engineering as the performance of detection methods has improved [9, 10, 11, 12, 13]. Improved computer vision methods have allowed for defect and anomaly detection on materials with non-uniform surface texture such as concrete and asphalt that were previously difficult to perform computer vision analysis upon.

A similar work, [10], used Faster R-CNN to detect cracks, rust, and other defects in civil and structural engineering environments. Similar to the application of this thesis, several modifications were made to the network in order to boost performance. The anchor boxes sizes and ratios and network depth being the most important. However, the training and test images used in [10] were still acquired by a human, and so several of the associated benefits such as alignment with the object and large object pixel size were present. Additionally, detections were made only in the 2D images and had no concept of 3D localization.

2.2 Object Detection

Within computer vision, recognition is the class of problems that determine information of a relevant object, feature, or activity in an image. The main sub-tasks of recognition are classification, object detection, semantic segmentation, and instance segmentation. The ultimate goal of this thesis is to determine the 3D position of the defects, which will inevitably require the image location of each defect. The two sub-tasks that determine the pixel location of each object instance in an image are object detection and instance segmentation. They key difference is that object detection outputs a bounding box for each object instance while instance segmentation outputs a pixel-wise mask for each object instance.

Since the application of wind tunnel facility inspections is relatively niche, a custom dataset was required. Manually labeling a dataset is expensive in terms of cost, time, and effort [14, 15].
Bounding box labels can be created using less resources than pixel-wise masks. Additionally, bounding box level detections are more than sufficient for visual reference and to perform 3D localization. As such, object detection was selected as the preferred computer vision sub-task.

Object detection methods are typically composed of three main components: a region proposer, a feature extractor, and a classifier. Figure 2.1 shows the R-CNN detection pipeline, which is representative of object detectors in general. The region proposer produces potential object locations in an image in the form of regions of interest. This method can ignore contextual information and procedurally generate bounding boxes based on the image size [16, 17], or use contextual information to generate bounding boxes [18, 19, 20, 21, 22]. The feature extractor creates high density information related to the features of the image. Classical feature extraction methods are based on hand-crafted features [23, 24, 25, 26] while deep learning object detectors typically make use of convolutional neural networks that learn features [21, 16, 17]. The classifier determines the class of the object based on the extracted features. Commonly used classical methods for classification include Bayes classification [27], k-nearest-neighbors [28], support vector machines [29], and decision trees [30]. Fully-connected neural networks have gained popularity recently [31, 32] and most state-of-the-art methods [33, 21] employ them for classification. Object detectors can be further divided between region proposal detectors, and regression detectors [31, 32].

Figure 2.1: R-CNN Pipeline
Region proposal detectors have some method that creates regions of interest to be classified. This method can use classical computer vision [19] such as graph based image segmentation [18, 34] or a deep learning method such as a Region Proposal Network (RPN) [21, 22]. Feature extraction and classification is then performed for each region to determine the class of the object captured by the region. Regression methods overlay a regular grid on the image rather than having a distinct region proposal step. Feature extraction is performed on the entire image, and each grid space is classified [16, 17]. For both types of classifiers, bounding box regression is performed to improve localization performance, and Non-Maximum Suppression (NMS) is performed to remove redundant classifications [19, 20, 21, 16, 17].

Region proposal methods are typically slower than regression methods due to the extra step of region evaluation. However, region proposal methods typically have better performance than regression methods, especially on small objects [35]. Due to the nature of facility inspections, almost all of the objects are considered to be small to medium sized since they are smaller than 128 by 128 pixels. A plot of the pixel size of all object instances is shown in Figure 2.2 for reference. Additionally, there are no real-time requirements or other speed requirements for this application. In fact, improved detection performance would be preferred at the expense of a reasonable increase in computation time. As such, the implemented architecture for this method was based on the R-CNN family of detectors [19, 20, 21, 22].

The method used in this thesis is most similar to the object detection portion of the Mask R-CNN detector [22]. First, the Mask R-CNN architecture creates a set of feature maps of the entire image. These feature maps are then passed to an RPN that filters out background regions and refines the locations of the potentially non-background regions known as Regions of Interest (ROIs). The original feature maps and ROIs are then passed to an ROI align layer that crops the feature maps to the regions, then interpolates the cropped regions to a fixed size. The fixed size feature maps are then passed to a classifier network, which further filters out background regions and further refines
the location of the regions. For the method used in this thesis, the outputs of the classifier network are all that is required. The original Mask R-CNN detector goes further to additionally pass the feature maps to a segmentation network that produces a pixel-wise mask for the network output.

![Figure 2.2: Object Instance Pixel Size](image)

2.3 Photogrammetry

Photogrammetry is a well-studied and accurate method to extract 3D spatial dimensions from a set of 2D images taken at various locations [36]. Using a pinhole camera model, a ray can be defined starting from the camera position and passing through the pixel position on the image plane [37] as shown in Figure 2.3. This ray defines all 3D locations that could have projected onto the image at that location. The image plane is located at a distance equal to the focal length (f) from the camera, and the optical axis of the camera passes through the plane at the location \((c_x, c_y)\).
When additionally accounting for lens distortion, the ray from the pinhole camera model visualized in Figure 2.3 follows the path subject to equations 2.1 through 2.8. Where terms $k_1$, $k_2$, and $k_3$ are the radial distortion coefficients and terms $p_1$ and $p_2$ are tangential distortion coefficients.

Through the use of computer vision methods, object recognition can be used to match local features between images. These local features are known as keypoints, and are found with keypoint detectors. Typically classical methods for keypoint detection include SIFT, SURF, and ORB [23, 24, 25]. Based on the keypoint descriptor, the similar keypoints from two different images can be matched. Given the camera poses, camera parameters, and optical parameters, and matched keypoints from two or more camera locations, the 3D location of the corresponding point can be determined with triangulation [37].
\[
\begin{align*}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} &= R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \\
x' &= x/z \\
y' &= y/z \\
x'' &= x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x'y' + p_2 (r^2 + 2x^2) \\
y'' &= y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 (r^2 + 2y^2) + p_2 x'y' \\
\text{where} & \quad r^2 = x'^2 + y'^2 \\
u &= f \cdot x'' + c_x \\
v &= f \cdot y'' + c_y 
\end{align*}
\]

Equations for Projection with Lens Distortion

For the purposes of this thesis, photogrammetry can be thought of as creating a sparse mapping between 2D image points and 3D locations. However, since photogrammetry keypoints are based on object recognition and not object detection, it does not classify the points that it identifies. It simply identifies a persistent object between frames.

2.3.1 3D Object Detection

The desired task to be performed for this thesis is 3D object detection. Current 3D object detection methods are not suited for the task of detecting small objects in industrial facilities from only a set of images. Most object detection is intended for relatively large objects with salient features and often relies on some other associated sensor data such as LiDAR, stereo camera data, etc.

Self-driving car research has been interested in 3D object detection since the 1980’s [38], and recent advancements in computer vision have made self-driving cars and other autonomous vehicles viable for commercial use [39, 40, 41]. Datasets with 3D labels have been created to benchmark the performance of these methods [42, 43]. However, the application of autonomous vehicles is
distinct from the intended application of this thesis. As mentioned these systems typically detect large objects with salient features and utilize some additional sensor.

Often times computer vision systems for autonomous vehicles rely on a LiDAR for point cloud generation. Due to the weight constraints of an aerial vehicle, and cost constraints of the project, the system of this thesis does not implement a LiDAR. Additionally, the application of facility inspection is distinct from autonomous vehicles because autonomous vehicles typically capture images at locations along a one dimensional parametric line navigated by the vehicle. Typical autonomous vehicle systems then process the input images similarly to a video stream, and propagate information between sequential frames. For the application of this thesis, image capture locations are positioned along a two dimensional parametric surface that follows the inspection surface rather than a one dimensional parametric line.

Additionally, the typical classes of objects in 3D object detection such as vehicles, pedestrians, or other large objects have very salient features. The object classes for the application of this thesis, missing fasteners and fasteners, as well as intended future object classes, cracks, spalling, etc, do not have such salient features. The object classes for this thesis are nearly flush with the facility surfaces and barely perceptible in the point clouds.

For instance, [40] implemented a successful method that was able to fuse data from multiple LiDAR devices and an RGB camera. Their method was able to create 3D bounding boxes by creating feature maps from the LiDAR point clouds and RGB images independently, then combined those feature maps to detect the objects. This method is reliant on a dense point cloud from a LiDAR, as well as easily perceptible objects in the point cloud. Luxuries that the application of this thesis does not share.
CHAPTER 3: DATA COLLECTION

The intended application of this thesis is for NASA’s large facilities. As a case study and proof of concept, data was collected in the NFAC. There were several features that make the NFAC a difficult environment to inspect. The inside of NFAC has ceilings ranging from 50’ to 250’, complex geometry, no source of light, and surfaces marked with oil stains, scratches, dents, etc. Additionally, the inside of the NFAC is GPS denied and has large metal structures making the use of a magnetic compass troublesome.

There were several benefits to creating a custom dataset from images taken in the NFAC. First and foremost since the dataset is collected from the same environment that the full-scale system will be deployed, there will be no reduction in performance between the test data and full-scale implementation due to differences in data distribution. Secondly a custom dataset allowed for complete control of the imaging system. All images were taken on the same camera and the same camera rig, so all images have consistent lighting, scale, etc. This causes lower variability in the dataset and should improve the performance of the model.

3.1 Trial 1

The first attempt at data collection was performed with a crude setup. A Basler Dart dA1280-54um camera was attached to the flight computer, an Intel NUC Mini PC, via USB. The camera was mounted to a small frame with 4 LEDs driven with 3 watts each to provide light. There was a laser distance indicator to provide visual feedback on the working distance between the camera and target surface. A small circuit managed the timing between the camera shutter, LEDs, and laser distance indicator. The setup was powered by a portable lithium-polymer battery.
The computer and camera setup were each held by a user for the data collection. The camera was set to capture frames at a constant rate, and the operator holding the camera was to walk at a constant rate so that the capture locations were evenly spaced. One pass with the camera consisted of traversing the desired distance with the camera held at a fixed height from the ground and a fixed distance from the surface. Once the appropriate distance was traversed, the user lifted the camera and captured images on the pass back to the starting location. The camera was then raised again for another forward and backward pass. This process repeated until the appropriate number of passes were complete.

There were several issues with this setup. The most obvious is that the spacing between frames was very uneven. The distance between points did not provide enough overlap in some areas for sufficient 3D correspondences. Figure 3.1 visualizes uneven spacing between grid points when the data collection is performed without position feedback and control. The next issue was that the light provided by the LEDs used was not sufficient. Figure 3.2b shows an example of an unusable image due to lighting. Additionally, due to the low light level the camera ISO had to be increased causing significant noise in the images. Finally, the last issue was the camera. The camera had a resolution of only 1.2 megapixels (MP). Typically object detectors have the best performance on objects with large pixel sizes. In order to have the appropriate image scale and desired number of image instances per real world object, over 1000 images would have to be collected per square meter. This amount is unreasonable since it would take an extraordinary amount of time to position the drone in 1000 locations for just a single square meter. A further minor issue was that the camera used was overly sensitive to blue light causing the images to have a noticeable blue tint. Some of the data was usable but was of overall poor quality due to lighting and camera resolution. An example image of this is shown in Figure 3.2a. Additionally, several other image examples are available in APPENDIX A. These issues prompted two further trials.
Figure 3.1: Grid Point Spacing without Position Feedback

Figure 3.2: Example Images from First Data Collection.
Image (a) was relatively bright as it was near the tunnel inlet and allowed for some outside light. Image (b) had no natural light and has barely visible features.

3.2 Trial 2

The second trial addressed the issues of camera resolution and bad lighting. The 1.2 MP camera was replaced with the Basler Ace acA4112-20uc, a 12 MP camera. This allowed for sufficient ground sampling distance and only requires roughly 50 images per square meter. The new camera also remedied the blue tint issue. The LEDs were driven with 6 watts each to increase the brightness, and lenses were added to direct the light in a 60° cone rather than the undirected 180° spread. The focused lighting allowed for the camera ISO to be reduced, which resulted in significantly reduced image noise. An example image with the new camera taken near the same areas as Figure 3.2a is shown in Figure 3.3. The images with the new camera and focused light were brighter, but had a
slight vignette. With a square camera sensor and circular lenses, there will always be the trade-off between a brighter circle of illumination and utilizing the edges of the camera sensor. The only way to have the same brightness but less vignetting would be the use of more LEDs. The lenses used on the 12 MP camera had more vignetting than those used on the 1.2 MP camera, but it was tolerable.

Figure 3.3: Example Image from Second Data Collection

3.3 Prototype Platform

Drones are the intended platform to use for data collection. Drones are small, agile, relatively cheap, and can access all areas of the facility for inspection. For the purposes of this thesis, a drone has 4 main systems: flight sensors, flight controller and computer, payload, and propulsion mechanism. Trials 1 and 2 were performed with only the payload (the camera and camera rig), and the flight computer. In order to remedy the issue of even grid-spacing, the flight sensors and flight controller would need to be implemented.
3.3.1 Flight Sensors and Flight Controller

A Pixhawk flight controller was used to handle the low-level processes of flight. The controller contains an inertial measurement unit (IMU) that measures the linear acceleration and angular velocity of the vehicle with orthogonal accelerometers and gyroscopes respectively. The flight controller also contains a magnetic compass, but this was unusable due to the amount of metal structures in the NFAC.

The measurements from the IMU provide local updates as to the state of the drone but these values are subject to drift, especially in regards to the position of the vehicle. Typically the position of the drone in the world must be updated with some driftless sensor such as GPS. However, the NFAC is GPS denied as so visual odometry is used to update the position instead.

Visual odometry is the process of using visual landmarks to determine the position and orientation of a camera relative to the world. If the 3D location of keypoints is known, then based on the projected image position of those keypoints, the 3D position and orientation of the camera can be determined [44]. The camera is always at some fixed location on the drone, and so the 3D position and orientation of the drone can be determined from the position of the camera.

3.3.2 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a class of methods in robotics that constructs and updates a map of an unknown environment and simultaneously tracks the robot’s location within that map. The map in this case is a 3D point cloud of keypoints. When a new image is captured, the visual SLAM algorithm finds and matches keypoints between the image and the existing point cloud to determine the location and orientation of the robot in the global map of the facility. This allowed for closed feedback of the system during data collection which allowed
for the image capture locations to be better controlled. ORB SLAM was implemented by a fellow researcher, Joe Adamson, and for the purposes of this thesis was largely treated as a black box.

The drone payload features the previously described 12 MP camera for data collection with a camera rig containing LEDs for lighting and a timing circuit so the LEDs are powered during the exposure of the camera. Unfortunately, it was not practical to use the 12 MP camera for both data collection and visual odometry. The 12 MP camera had relatively low perspective distortion to better capture information in the images of the tunnel. As such, the field of view is relatively small. Additionally, the 12 MP images are relatively large and would require significant processing power to use for visual odometry, which needs to be performed in real time. Instead, the previously described 1.2 MP camera from the first data collection trial was given a wider angle lens and repurposed for visual odometry. A comparison of the images taken from the cameras is shown in Figure 3.4. The images of that figure are centered on roughly the same real world location. It was relatively simple to allow for the LEDs to flash for the shutter of both cameras, and the additional camera added little weight to the system. The 12 MP camera will be referred to as the data camera, and the 1.2 MP camera will be referred to as the SLAM camera.

### 3.3.3 Sensor Fusion

The position and orientation from the SLAM subprocess is then feed into the flight controller which implements sensor fusion and visual inertial odometry to provide a high quality state estimate of the vehicle. This process of visual odometry is relatively slow since it is limited by the camera frame rate and computational power of the flight computer. In principle, the position of the drone is updated using visual odometry on a relatively slow, yet minimally drifting basis and quick intermediate updates are provided with dead reckoning from the IMU measurements. The output of the flight controller was taken as the most reliable state for the system.
3.3.4 Setpoint Spacing

With SLAM implemented the setpoints could be defined on a grid pattern. When the system is within some tolerance distance and orientation of a setpoint an image is captured by the data camera. To create the setpoints, the desired horizontal and vertical distance between points would need to be determined, as well as the distance from the camera to the surface.

The first quantity of interest is the working distance, the distance between the camera and the surface. For simplicity, it was assumed the wall was flat. In reality the wall was corrugated, but this had little effect on the performance of the system. To determine the working distance, the desired ground sampling distance (GSD) would need to be determined. GSD is a measure of the area represented by a pixel. A GSD of 2 cm means the pixel represents an area of 2 cm by 2 cm on the imaged surface. The GSD can be calculated based on the similar triangles shown in Figure 3.5. GSD is a function of the physical camera sensor size, camera sensor resolution, focal length, and working distance. By tuning the working distance, the desired GSD can be achieved.
For reasons elaborated upon in Section 4.2.2, due to the stride length of the object detection model the smallest object should be no smaller than 16 by 16 pixels. The missing fasteners were the smaller object class, with a diameter of about 0.4 cm (0.16 inches). Based on the focal length of 16 mm, and pixel size of 3.75 µm, the working distance was set to 1 meter. Based on the theoretical calculations, the smallest object size should be roughly 18 by 18 pixels. These calculations were validated with the experiment as shown in Figure 2.2. One meter was also close enough to the surface that the LEDs could provide sufficient light.
The next quantity to be determined is the horizontal and vertical spacing of the points. It was desired to have each point appear in roughly 25 images to perform 3D reconstruction from the images as described in Section 4.1. This number was empirically determined with a number of small scale tests, but is not highly optimized.

The camera resolution was 4096 by 3000 pixels, and the field of view shared that same aspect ratio of 1.365:1. With that in mind, it was most convenient to have each point visible in 6 images per horizontal pass and in 4 passes. The camera frame was oriented such that the larger image dimension was aligned with horizontal passes, and the smaller image dimension was aligned with successive vertical passes. This meant that each point appeared in 24 images, which was close to the desired number of 25. With the camera aligned with the world as previously described, the width of the surface visible was 91.8 cm, and the height was 67.3 cm.

Since the objects have an area and are not point objects, it is possible for the object to be partially imaged along the border of the image. In that case, the partially imaged instance does not count toward the number of appearances for that object. Based on the physical size of the fasteners (the larger object class), the focal length, physical sensor size, and the working distance, the spacing between images was determined to be 13.9 cm in the horizontal direction, and roughly 14.8 cm in the vertical direction. These step sizes meant that each horizontal step had 620 columns of pixels with new information, and each vertical step had 660 rows of pixels with new information. The image would have completely new information with 6.5 horizontal steps, or 4.5 vertical steps.

Based on the working distance, the fasteners were expected to be up to roughly 128 by 128 pixels. An image contains 6.5 horizontal steps or 4.5 vertical steps depending on the direction of the steps. Since the fastener is smaller than the remainder after 6 horizontal steps and 4 vertical steps, it is guaranteed that the fastener will be visible in at least 6 horizontal images per pass, and in 4 vertical passes.
The system was operated manually for Trial 3, so the step size was reduced to by 2 cm in each direction to add a safety factor and ensure that the objects were visible the desired number of times. The tolerance on position was 2 cm per axis, and the tolerance on orientation was 10° per axis, as this was approaching the tightest tolerance achievable with reasonable effort.

3.4 Trial 3

The setup for the third data collection featured all systems that the final drone will implement, except for those related to propulsion. For hardware, this trial added the implementation of the flight sensors and flight controller to provide the localization feedback. For software, this trial added SLAM, and other small peripheral processes. Figure 3.6 shows the system used with key components labeled. Several examples of images captured in the third trial are available in APPENDIX B. This trial served as a test of all systems related to 3D localization and provided enough data for qualitative results of the pipeline.

![System for Third Data Collection Trial](image)

Figure 3.6: System for Third Data Collection Trial
The data collection system was mounted to an adjustable frame and the frame attached to a cart, as shown in Figure 3.7. The frame allowed for easy adjustment of the height of the system, and the cart allowed for movement on the ground plane. A monitor was on the cart, which displayed the position and orientation of the system.

The setpoints were created, and the cart was moved manually until the system position was within tolerance of the setpoint and an image was captured. The system was configured such that the setpoints needed to be captured sequentially. The monitor displayed the current position, and the difference between the current position and the desired position ($\Delta$). The user knew when a setpoint was captured because $\Delta$ would change abruptly and significantly since the new desired position would be one grid space step from the current position. Once an entire horizontal pass was finished, the system was raised vertically on the frame for another horizontal pass to be completed. This process was repeated until the desired area of the tunnel section was covered.
The addition of SLAM allowed for the image capture locations to be spaced more evenly, as shown in Figure 3.8. This figure shows half of the total grid points of the third data collection since showing all of them would be cumbersome. The majority of the discrepancies that caused points to be misaligned with the grid are due to the generous tolerance in the capture position to allow for human error. It is expected that the alignment of the points to the grid will improve drastically when the autonomous vehicle is used and the tolerances can be tightened.

![Grid Point Spacing](image)

(a) Grid Point Spacing without Position Feedback

![Grid Point Spacing](image)

(b) Grid Point Spacing with Position Feedback

Figure 3.8: Comparison of Grid Point Spacing with and without Position Feedback

3.5 Full-Scale Systems

3.5.1 Vehicle

After the success of the prototype platform, the system was to be converted to a proper UAV platform. A quadcopter frame from a previous NASA project was repurposed to be used for this thesis. Several modifications and 3D printed mounting fixtures were made, but the transfer of components from the cart system to the drone was trivial in the scope of this thesis.
At the start of an inspection, the setpoints are generated in the pattern of successive horizontal passes. The flight computer monitors the position of the vehicle and captures an image when the camera is within tolerance of a setpoint. The flight computer communicates high level position and orientation setpoint commands to the flight controller, which commands the electronic speed controllers to keep the vehicle airborne and move it to the desired positions.

3.6 Trial 4 - Full System Test

A full systems test of the vehicle was performed in an indoor drone test facility. This tested the vehicle’s out-of-the-box flight performance and tuned the controller PID gains. The next intended step was to test that the vehicle was able to capture setpoints in a reasonable amount of time and with good position and orientation accuracy. After that, the final step would be a full system deployment and data collection in the NFAC. The goal of this final deployment would be to gather a large amount of data to rigorously test the 3D localization pipeline and demonstrate the capabilities of the physical system in the real test environment.

The setpoint capture test was scheduled for late March 2020, and the full system deployment was scheduled for June 2020. Unfortunately, due to COVID-19, NASA Ames and the NFAC have been closed since early March 2020. The project has been put on an indefinite hold until the facility is able to open again safely.

3.7 Data Labeling and Partitioning

Since the full system test was postponed due to COVID-19, the only data available to benchmark the method was from the third trial of data collection. The third trial collected images from a section of the interior NFAC walls over a 20 meters by 2 meters area using the system shown in
Figure 3.6. The images were collected at a distance of 1 meter from the surface, and captured on a grid-like pattern along the wall. The grid was spaced such that each object appears in at least 6 images per row, and in at least 4 rows of images. The image capture locations were supervised by a SLAM algorithm to control the spacing between capture locations, distance between the cameras to the wall, and orientation of the system.

While the amount of data is relatively small, it is enough to provide a qualitative measure of the performance of the method. The area of the tunnel imaged contained 148 real world fasteners, and 6 real world missing fasteners. Each real world object was captured in multiple images. Due to lighting and perspective changes, each image sample of an object was distinct enough to train a machine learning method as if they were different objects. A total of 1834 images were taken, and there were 4015 image instances of fasteners and 259 image instances of missing fasteners.

3.7.1 Object Labels

Two types of labels were created for the dataset. First, the image labels were created in the PASCAL VOC [45] format. These labels provided information regarding the pixel position of an object in an image. Fastener and missing fastener image instances were given a bounding box and class label. An illustration of this type of label is shown in Figure 3.9. The second type of label created was the object set label. This label was a text file that contained the image name and bounding box index for all images associated with the same real world object.

3.7.2 Data Partitioning

Information in the training, validation, and testing partitions must be distinct. A real world point cannot appear on more than one of these partitions, as it would cause data leakage and unfairly
increase the performance of the model. To ensure that each object appears in no more than one partition, division of objects was performed at the object sets level. All bounding boxes related to an object set appeared in exactly one partition.

The object sets were divided in an 64/16/20 split between the training, validation, and testing partitions respectfully. This was done to give an 80/20 split of the data between the training/validation partition and test partition, then an 80/20 split of the training/validation partition between the training and validation partitions. The split was done such that each partition contained its percentage of the object image instances, and each real world object could only have image instances in one partition. This was solved approximately since splitting exactly 64/16/20 was not possible.

The collection process to meet the desired goal of each object captured in 24 images is only valid for an infinite area, and does not hold for finite collection. In particular, the collection process collects less than 24 images when the object is on the edge of the collection area. In that case, the objects along the edge of the inspection area will appear in fewer images per row and/or fewer rows of images. In a full-scale test, the border of the inspection area would be a relatively small percentage of the total area. However, for this small scale test it was a significant portion of the total area.
These images along the edge can still be used in the training data partition, but it was avoided using them in the validation and test partition since the evaluation of the method would not be representative of the performance.

The training set contained 113 fastener object sets and 4 missing fastener object sets. Many of the training object sets were on the border of the inspection and so they did not contain the full 24 images per object set. The validation set contained 15 fastener object sets and 1 missing fastener object set. The test set contained 20 fastener object sets and 1 missing fastener object set. All object sets in both the validation and test set contained at least 24 images per object set.

Images containing multiple objects were cropped into sub-images such that each sub-image contains objects of only one partition. Additionally, the background of the sub-images was given a best effort to be unique to that partition by ensuring that the keypoints visible in the sub-image were unique. The keypoints are not a dense mapping, so it is possible that there is a small amount of overlap between background points of sub-images in different partitions.
CHAPTER 4: 3D LOCALIZATION METHODOLOGY

The 3D localization pipeline consists of three stages as shown in Figure 4.1. The 3D reconstruction stage uses the set of input images to generate a 3D point cloud. The point cloud consists of keypoints which map a 3D point to the 2D pixel points in the images in which that 3D point is visible. The image level detection stage uses the set of input images and generates object detections in the form of image level bounding boxes with an associated class. The 3D object detection stage combines the image level detections and the 3D point cloud to localize the objects in 3D space.

4.1 Photogrammetry

Pix4D [46] was used for the photogrammetry purposes of our method. Pix4D is able to process a set of images with no prior knowledge of the camera positions. This is highly desirable as the recorded position of the camera when the image is captured is subject to noise and drift, or may be completely unavailable in the worst cases. For the setpoint spacing described in Section 3.3.4, Pix4D generated an average of almost 3 thousand 3D keypoints per image for our dataset, and each of the keypoints was visible in an average of 3.7 images.
4.2 Computer Vision

Object detection was the selected computer vision task to determine the pixel locations of the defects in the images. Our method uses a deep learning method for object detection due to the significant increase in performance of deep learning methods observed in recent years [31, 32]. The current state-of-the-art methods use convolutional neural networks, and can be divided between two main groups, the region proposal methods and regression methods. Region proposal methods are typically slower than regression methods due to the evaluation of each region individually. However, region proposal methods typically have better performance than regression methods, especially on small objects [35]. Within reason, performance is more important than computation time for this application. Due to the nature of facility inspections, almost all of the objects are considered to be small to medium sized since they are smaller than 128 by 128 pixels. As such, the implemented architecture for our method was based on the R-CNN family of detectors [19, 20, 21, 22], a family of region proposal object detectors.

For object detection, we use the object detection portion of the Mask R-CNN architecture [22]. The Mask R-CNN architecture is shown in Figure 4.2. The final step of mask generation in Mask R-CNN was removed, so the output is only the object bounding box and classification. This could also be viewed as using the Faster R-CNN architecture, but with an ROI Align layer rather than an ROI Pooling layer.

The R-CNN family was implemented for use on datasets of object detection challenges such as MS COCO [47] and PASCAL VOC [45]. Our dataset is distinct in several key ways that required modifications to the training procedure and configuration of the original R-CNN detectors. Our dataset has objects much smaller than MS COCO, has well under 1% the number of images and labels as MS COCO, and has a severe class imbalance. Each of these complications required some enhancement to improve the performance.
4.2.1 Anchor Box Optimization

The anchor box sizes and aspect ratios used in the original Mask R-CNN implementation, Faster R-CNN implementation, and in most object detectors are too large for our dataset. New anchor box priors were selected using k-means clustering similarly to YOLOv3 [17]. We randomly initialized two cluster centers per class, then using k-means clustering iteratively modify the center location to maximize the average Intersection over Union (IoU) across all samples. IoU refers to the intersection and union of the set of ground truth pixels and the set of prediction pixels. IoU is the number of pixels in the intersection of those sets divided by the number of pixels in the union of those sets. However, in general IoU can refer to the intersection over union of any two sets.

The optimized anchor boxes found, in (width, height) format, were (28.10, 34.65), (42.85, 34.02), (74.10, 77.27), and (93.71, 100.55). Figure 4.3 shows the effect of anchor box optimization against unoptimized boxes. For the unoptimized anchor boxes squares of 16, 32, 64, and 128 pixels were used. Mask R-CNN and Faster R-CNN originally only implemented 128 square anchor boxes or larger, so the addition of the smaller anchor boxes is already tailoring the anchor boxes to this dataset. From the figure, the optimized anchor boxes clearly better represent the data.
4.2.2 Network Architecture

The ResNet50 [48] model was used as the network for object detection. A ResNet model contains five stages each made up of several residual blocks. The first residual block of each stage downsamples the input by a factor of two and adds convolution layers. Blocks after the first add convolution layers but do not downsample their input. As was done in [48] and [22], the layers up through and including the fourth stage were used for the backbone. The backbone network had a stride of 16, which set the minimum reasonable object size to 16 by 16 pixels.
The same network as used in Faster R-CNN was used for the RPN, with the number of outputs adjusted for our number of object classes. The fifth stage of the ResNet model was then used for the classifier, with the appropriate number of output channels for the object class and regression terms.

### 4.2.3 Training Scheme Modifications

During training the only form of data augmentation used by Mask R-CNN and Faster R-CNN was horizontal flipping since other forms of data augmentation would produce unrealistic images. i.e. an image of a person rotated 90° or flipped vertically would look drastically different from the original. Whereas an image of a person flipped horizontally is still realistic. Our dataset is composed of facility surfaces which can be augmented more aggressively and still produce realistic images. As such, we implemented random horizontal flipping with probability of 50%, and random rotation in 90° increments, each with 25% likelihood. Vertical flipping was not implemented, because it would not add new transformations to the set of transformations possible with horizontal flipping and rotations.

Our training procedure was more similar to that of Faster R-CNN than Mask R-CNN. Mask R-CNN used 16 images per batch during training, but we used 1 image per batch as was done in Faster R-CNN [21] on the PASCAL VOC dataset. The model was trained with stochastic gradient descent with a momentum of 0.9 and a decay of 1e-4. Cross-entropy was used as the loss function for classification, smooth L1 was used as the loss function for bounding box regression, and they were weighted equally. The model was initialized using weights pre-trained on ImageNet.

Since our dataset is significantly smaller than that of object detection challenges, our training schedule was shorter than that of Faster R-CNN. We trained for 26k iterations with a learning rate of 1e-3, then for 2k iterations with a learning rate of 1e-4. These were the learning rates for the
respective coarse and fine trainings of Faster R-CNN. This training schedule was based on the performance on the validation data. There was no distinct plateau of performance but past 26k and 2k iterations for the coarse and fine trainings the performance had no major improvements on the validation data.

Faster R-CNN uses 0.7 as the IoU threshold to classify an anchor box as a positive sample and an RPN batch size of 256 samples. Due to the small object size and lower number of objects per image, the NFAC dataset produced a lower number and lower concentration of positive anchor box samples than images of MS COCO or PASCAL VOC. To better preserve the intended positive to negative ratio of the RPN training, an IoU threshold of 0.5 for anchor boxes was used, and the RPN batch size was reduced to 128 per image.

The classifier was only trained when the RPN produced at least one positive region for the image. Positive for the classifier was also defined as an IoU of 0.5. The NMS IoU threshold between the RPN and classifier was reduced from 0.7 as in Faster R-CNN to 0.5 in an attempt to increase the number of positive samples as well. The batch size for the classifier was dynamic, and 6 negative samples were used for every positive sample. The batch size was capped at 64 samples, and the positive to negative ratio was allowed to increase up to 1:1 should a very high number of positive samples be present. In practice, the largest number of positive samples per image was 8, so this cap was never reached.

4.2.4 Class Imbalance Remediation

Our dataset featured a severe class imbalance between the fasteners and missing fasteners. The number of missing fastener samples was roughly one sixteenth that of the number of fastener samples, and so we oversampled the images with missing fasteners sixteen times. This allowed for the same number of training iterations to be performed on fasteners and missing fasteners. For
each copy, we randomly cropped up to 15 pixels from the border of the image so that the missing fastener location was not always at the exact same position of the model stride.

Additionally, since the fasteners were much larger than the missing fasteners, the fasteners had 6.76 times the number of anchor boxes above the objectness IoU threshold per instance than missing fasteners. To combat this, during training we weighed the missing fasteners to be 6.76 times as important as fasteners.

4.3 3D Localization

Once the point cloud and image level detections are created, the objects can be localized in 3D space. At first, the 2D pixel location of each keypoint is considered. A keypoint is associated with an object detection if it lies within the detection’s bounding box. This will also associate a small number of keypoints in the background with the object, but that is acceptable.

Each keypoint has a unique ID and is associated with several images. When two image level object detections contain the same keypoint, they are grouped together into an object set. By iteratively combining these sets, all detections related to the same real world object are grouped into the same set. This iteration stops when no detections outside the set share a keypoint with detections inside the set. An illustration of two detections being combined is shown in Figure 4.4.

If there were a large number of keypoints per image such that each bounding box contained many keypoints, only the bounding box of the detection would need to be used for this set combining process. However, the bounding boxes were relatively small compared to the concentration of keypoints so some of the detections were not combined properly. To remedy this, all bounding boxes were padded by 25 pixels. This amount was somewhat arbitrary, but was based on the performance on the validation data.
The blue star was detected in both images. The keypoint of the bottom point is present in both images, which allows the algorithm discern that two images contain the same real world object.

Object sets can be filtered to remove potential false positives by examining the number of detections associated with the object set. With the setpoint spacing as described in Section 3.3.4, it is expected that each object appears in at least 24 images. An object detector with a recall rate of 75% would find roughly 18 of these. Assuming that to be the case, then any object set with less than 18 detections can be flagged as a false positive. In practice, the minimum detection threshold could be set to 12 for both object classes without reducing the pipeline’s recall rate. We noticed that false positive detections seemed to come in two flavors, random and object-similar.

A random false positive was effectively noise in an image that passes as an object, and the pixel locations of these were randomly distributed throughout all images. So long as the image locations
corresponded to different 3D locations, the random false positives did not share keypoints and were not grouped together. Usually each random false positive created its own object set with only one instance. Because random false positives have very few image detections, they can be filtered without reducing the overall pipeline’s recall rate. Additionally, these random false positives also typically had low confidence compared to the true positive detections. On the test data there were 21 random false positive object set detections. Twenty of them had only one image detections, and one had three image detections. Additionally, all but one were classified as missing fasteners. This is likely because missing fasteners have less salient features than the fasteners. All of the random false positives were filtered without reducing the overall pipeline’s recall rate.

The object-similar false positives are instances where the background has features similar to one of the object classes. It can also be the case that an object instance has features similar to another object class causing a misclassification. However the former was the only object-similar false positive observed. These are significantly more difficult to filter as the object detection model has high confidence for all detections, and a relatively large number of detections are grouped into the same object set. The output of the pipeline for the object-similar false positives is effectively identical to the real objects. The only way to reduce the number of object-similar false positives is use a more powerful object detector that can discriminate between real objects and these hard negative samples. On the test data there were 2 object-similar false positive object set detections. Both object-similar false positive detections contained less image detections than the true positive detections and were filtered without reducing the overall recall rate.

Once the object sets are created and filtered, the 3D location of the keypoints is considered. The 3D locations of all keypoints associated with an object set are then clustered with mean shift clustering [49] to determine an average 3D location of the keypoints. The largest cluster is selected, which makes this method robust to outliers and spurious keypoints of the background. Mean shift was selected as it requires no priors as to the number of clusters, and since the center of each cluster is
the maxima of the spatial density. A maxima in spatial density indicates that many keypoints are located in that position. Figure 4.5 shows an example of the ray tracing from the camera position, through the image plane at the position of the fastener, to the 3D location of a fastener. In practice many rays are traced per image, in order to better localize the object.

Figure 4.5: Ray Tracing to 3D Object Location
CHAPTER 5: EXPERIMENTAL RESULTS

5.1 Photogrammetry

No independent analysis of the accuracy of the photogrammetry was performed as Pix4D provides its own analysis. Additionally, it was not within the scope of this thesis to do so. For the Trial 3 data, Pix4D claims a mean reprojection error of 0.174 pixels. It can be assumed that the 3D reconstruction was accurate within bounds allowable for the application.

5.2 Computer Vision

Precision vs recall, and particularly average precision (AP), is typically used to evaluate object detectors. Average precision is the average value of the interpolated precision over all recall values, or equivalently the area under the interpolated precision vs recall curve. Recall is the percentage of objects detected, i.e. true positive detections divided by the number of positive samples. Precision is the percentage of detections that are correct, i.e. true positive detections divided by the total number of detections. By raising or lowering the confidence threshold, the detector returns respectively less or more detections. In general, lowering the confidence threshold increases the recall, but decreases the precision.

Object detectors are evaluated with interpolated precision rather than traditional precision, because it better reflects the actual performance of the detector. Interpolated precision for recall value $r$ is the maximum precision found for any recall greater than or equal to $r$. This better reflects the performance of detector because if there is a dip in precision only for it to rise at a higher recall rate, the confidence threshold that has the high recall and high precision can be used to cover for the
dip. To do this, the detector is used with the confidence threshold to create the high recall and high precision output, then a random subset of the output are removed. With a large enough sample size, by randomly removing some of the outputs the precision is persevered but the recall will decrease. This simulates having a high precision and low recall, which covers for the dip.

We use frame-AP as defined by [50] to evaluate the object detector. Frame-AP measures the average precision on a per image instance basis. An object image instance is deemed detected if the IoU between the ground truth label and object detector bounding box is above a threshold and the class label is correct. We follow the rules of the PASCAL VOC 2012 object detection challenge to calculate our score. Figure 5.1 plots the precision vs recall for the fasteners and missing fasteners corresponding to a 0.5 IoU threshold. The frame-AP values are given in Table 5.1.

![Precision vs Recall](image1.png) ![Precision vs Recall](image2.png)

Figure 5.1: Object Precision vs Recall (IoU Threshold 0.5)

<table>
<thead>
<tr>
<th>Class</th>
<th>IoU Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>Fastener</td>
<td>77.4</td>
</tr>
<tr>
<td>Missing Fastener</td>
<td>64.0</td>
</tr>
<tr>
<td>Mean</td>
<td>70.7</td>
</tr>
</tbody>
</table>

Table 5.1: Frame-AP Performance (%)
5.3 Object Detector Ablation Experiment

An ablation study was performed on the implemented enhancements to the object detector pipeline and training scheme. This study verifies that the enhancements improved the performance of the detector. The items tested were the optimization of the anchor boxes and the use of imbalance remediation. The anchor box optimization selected anchor boxes using k-means clustering by randomly initializing two clusters per object class, then optimizing for maximum IoU across all samples. For unoptimized anchor boxes, to keep an equal comparison, four anchor boxes were used. The unoptimized anchor boxes were squares with side lengths of 16, 32, 64, and 128. The imbalance remediation consisted of both oversampling and class weighting. For training, images containing missing fastener instance(s) were oversampled so that the number of training iterations with each of the classes was equal. Additionally, due to their larger pixel size, on average the fasteners had 6.76 times as many anchor boxes above the objectness IoU threshold compared to missing fasteners. Class weighting was also implemented to weigh the importance of missing fasteners to be 6.76 times that of the fasteners.

As expected, the trial with both optimized anchor boxes and imbalance remediation performed the best. The results from all trials are given in Table 5.2.

Table 5.2: Ablation Experiment on Object Detector (Frame-mAP, 0.5 IoU Threshold)

<table>
<thead>
<tr>
<th>Anchor Box Optimization</th>
<th>Imbalance Remediation</th>
<th>Fastener AP</th>
<th>Missing Fastener AP</th>
<th>Mean AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>33.6</td>
<td>2.8</td>
<td>18.2</td>
</tr>
<tr>
<td>k-means</td>
<td>None</td>
<td>50.9</td>
<td>19.4</td>
<td>35.2</td>
</tr>
<tr>
<td>None</td>
<td>CW and OS</td>
<td>42.0</td>
<td>35.9</td>
<td>38.9</td>
</tr>
<tr>
<td>k-means</td>
<td>CW and OS</td>
<td>74.9</td>
<td>62.9</td>
<td>68.9</td>
</tr>
</tbody>
</table>

CW - Class Weighting, OS - Over Sampling
With all else equal, anchor boxes optimization improved the performance for both classes substantially, with the increase being roughly equal in magnitude for both classes. Without imbalance remediation, the anchor box optimization increased the AP by 17.3 points for fasteners and 16.6 points for missing fasteners. With imbalance remediation, the anchor box optimization increased the AP by 32.9 points for fasteners and 27.0 points for missing fasteners.

The improvement in performance from anchor box optimization can only be due to the increase in the number of anchor boxes above the objectness threshold. Both the unoptimized and optimized trials have the same stride length, and same number of anchor box priors, but the optimized anchor box trials have more anchor boxes above the objectness threshold. This improves the detector’s performance for two reasons. First, the amount of regression that needs to be performed by the RPN to produce positive samples for the classifier is reduced. This lead to more positive samples for the classifier to train upon. Second, the RPN also had more positive samples for training since more anchor boxes were above the objectness threshold. This created an appearance of more data and a substantial increase in performance for both classes.

With all else equal, imbalance remediation also improved the performance for both classes substantially, with the increase being considerably more for the missing fasteners. Without anchor box optimization, the imbalance remediation increased the AP by 8.4 points for fasteners and 33.1 points for missing fasteners. With anchor box optimization, the imbalance remediation increased the AP by 24.0 points for fasteners and 43.5 points for missing fasteners.

The improvement in the missing fastener performance due to imbalance remediation was expected as more training iterations were performed with missing fasteners and their importance was weighted higher. The improvement in fastener performance seemed to be because the training was more stable with imbalance remediation. With no remediation the network would optimize on the fasteners for several successive iteration since missing fastener samples were sparse. Then when training on an
image with a missing fastener, the network would have a relatively high loss which would cause the network to change abruptly and effectively undo some of the previous optimization on the fasteners. So without imbalance remediation the performance was overall worse for both classes.

5.4 3D Localization

For videos, video-AP is the metric used to evaluate video detection methods. The video-AP is defined by [50] as the temporal IoU multiplied by the average IoU per frame in the temporal intersection. Similarly to frame-AP, detections with a video IoU above some threshold are counted as true positives. Video-AP is then the area under the interpolated precision-recall curve. For the application of this thesis, instances are the item of interest rather than videos.

To evaluate detections and score the method, we define new metrics: instance-AP and instance IoU. An intermediate quantity, object set IoU, is defined to be the IoU between the set of images in the ground truth and the set of images in the detection. This is similar to the temporal IoU used for videos. We then define the instance IoU to be the object set IoU multiplied by the average frame IoU of the images in the intersection of those two sets. A visual illustration of this is given in Figure 5.2. A detection is a true positive if the instance IoU is above some threshold. The instance-AP score is then the area under the interpolated precision-recall curve.

Typically, video detection methods are evaluated with IoU thresholds on an interval from 0.2 to 0.7. An evaluation of our method using the instance-AP metric over the same interval is given in Table 5.3. For maintenance applications we believe the 0.2 threshold would be sufficient to visually interpret the severity of the defect. From experience with the data, as long as about half of the frames are detected and the average IoU is roughly 0.5, the object can be interpreted by a human. A detection with that level of performance would have an IoU of 0.25. An example of a detection
with roughly that quality is given in APPENDIX C for reference.

![Diagram showing object set intersection and frame intersections of object set intersection.]

**Figure 5.2: Instance IoU Example**

<table>
<thead>
<tr>
<th>Class</th>
<th>IoU Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>Fastener</td>
<td>100.0</td>
</tr>
<tr>
<td>Missing Fastener</td>
<td>100.0</td>
</tr>
<tr>
<td>Mean</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Table 5.3: Instance-AP Performance (%)

Individual precision-recall curves are not given because for most cases the curve was simply a step function with 100% precision up to the max recall rate where it then dropped to 0% precision. This was because for most cases the true positive detections had more image instances than the false positives. This allowed the false positives to be filtered without reducing the overall recall rate. The area under the precision-recall curve was then equal to the maximum recall rate. The only exception was the 0.7 IoU threshold of fasteners, but this is the least applicable case for 3D defect localization since that threshold is unreasonably high.
With only one real world missing fastener object the only possible instance-AP scores for that class were variants of $1 / (1 + n)$ where $n$ is the number of false positives. If the missing fastener IoU was below the IoU threshold, $n$ could be taken as infinity and an instance-AP score of 0% would be given. Since the false positive missing fasteners could be filtered, the instance-AP score was 100% until the IoU threshold surpassed the missing fastener IoU score of 52.8%.

5.5 Overall Performance

The method performed well, especially when considering the 0.2 instance IoU threshold. The pipeline was able to detect all objects with no false positives. From these detections, the 3D position of the objects could be determined. The 3D localization method was able to find a correct 3D position for the defect within the point cloud generated by Pix4D. However there was no ground truth 3D location for the object measured independent from Pix4D, so the instance-AP score will need to be used as the metric for the pipeline. Overall, it is a fair claim that the method localized all of the object in 3D space with no false positives.
CHAPTER 6: CONCLUSION AND FUTURE WORK

This thesis describes a method to collect a set of images from a large, indoor facility using a drone equipped with a camera, and process those images to detect defects in the 3D space of that facility. This method has the potential to enable quick and inexpensive inspection of large wind tunnel facilities. We were able to create a 3D point cloud of keypoints and defect detections in 2D images, then combine the detections and point cloud to localize the defects in 3D space. The method performed well and the results are promising. In the test data, all of the objects were successfully localized with no false positives. The dataset was small, and so the results simply demonstrate that the method has potential, but this is by no means a rigorous test of the method.

The most obvious future work is to properly test the method when more data is available. More data would also allow for a deeper backbone network and more rigorous testing of the backbone to find a more optimal network. More data may also allow for the image object detection to utilize information from multiple images simultaneously rather than a single frame at a time. Video object detection methods such as Sequence Level Semantics Aggregation [51] treat the video as a set of frames rather than a temporal list and could be applied if more real world object instances are available. Overall, more data would allow for major improvements to the image object detection portion of the method.
APPENDIX A: FIRST DATA COLLECTION IMAGE EXAMPLES
Examples of images collected in the first trial of data collection. These images are 1280 by 960 and had the maximum ISO setting making them noisy when viewed at magnified scale. There is also a noticeable blue tint to the images.
APPENDIX B: THIRD DATA COLLECTION IMAGE EXAMPLES
Examples of images collected in the third trial of data collection. These images are 4096 by 3000 and were capture with the lowest ISO setting on the camera. There is a slight vignette to these images, but no noise due to high ISO, and no blue tint.
APPENDIX C: INSTANCE IOU EXAMPLE
In this example, 50\% of the frames of a fastener were found (16/32), with an average per frame IoU of 62.5\%. The instance IoU was therefore 31.3\%. While the score was relatively low, this object could be easily discerned by a human from the following images and associated 3D location. The detection generated by the model is highlighted in blue. Other objects may have been detected but are not shown as they would be part of a different object set.
Supplementary documentation.
LIST OF REFERENCES


