Ground Up Design of a Multi-modal Object Detection System

Vasu Agrawal

CMU-RI-TR-19-80
December 18, 2019

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Sebastian Scherer, Chair
Kris Kitani
Rogerio Bonatti

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Rapid situational awareness is crucial to enabling a successful response from first responders during an emergency, where time is of the essence. Emergency personnel are often sent into incident scenes to gather information, but this is often a dangerous and slow process. Subterranean environments are particularly challenging due to hazards such as difficult terrain, low visibility, and outdated or incomplete maps. Individual sensors and sensing modalities are unable to reliably identify all categories of objects under these conditions. This work covers the development of a multimodal object detection and localization system to help provide situational awareness in subterranean environments.

We cover the development of two iterations of a modular sensing platform, algorithms to accurately detect and localize objects across multiple sensing modalities, and data transmission techniques to ensure timely updates for human operators from multiple robots in a fleet. Reported information is continually refined with new information from SLAM systems, ensuring global consistency is maintained. We demonstrate that the use of multiple sensors and sensing modalities is advantageous in reporting accurate and timely information. All evaluation is performed with data collected during the DARPA Subterranean Challenge Tunnel Circuit, where the proposed system was used to detect more than twice the number of objects of the next highest performing team, and where we won first place and an award for the most accurate object detected.

Acknowledgments

I want to start by thanking Prof. George Kantor for giving me the chance to get involved with academic research as a freshman, and allowing me chance to experience the full cycle of robotics systems research and development in a single summer. I appreciate his mentorship and guidance and am incredibly grateful for the opportunity, which played a large part in shaping my academic path moving forward.

I'd also like to thank Ratnesh Madaan for introducing me to Basti, the Air Lab, and SubT, and without whom I'd likely still be looking for a thesis advisor. I've had a wonderful experience these past two years and I'm grateful to everyone in the Air Lab who's made my stay a welcome one.

I'd like to thank my entire committee for their invaluable advice throughout this thesis process. It's taken than longer than I (and I'm sure they) would have liked, and I appreciate their patience and support through this endeavor.

Of course, SubT is a team effort, and it wouldn't have been possible without each and every member of the team. I cherish the time I was able to spend building robots and debugging code with everyone, and look forward to being able to work with members of this incredibly talented group of people again. I also appreciate the funding received from DARPA and other team sponsors for allowing me to work on this project and play with some incredibly cool toys.

Finally, I'd like to thank my friends and family for their moral support during thesis writing, and for making sure that I don't take yet another semester to complete it. I particularly appreciate the time that Justin George, Bob Debortoli, and Bill Drozd put towards proofreading drafts and appreciate their comments.

Thanks everyone!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Rapid situational awareness is crucial to enabling a successful response from first responders during an emergency. Incident command uses all information available to them to make decisions and coordinate the first responders. First responders are often sent into the incident scene to gather information, but the nature of emergencies makes this process dangerous and can risk additional lives [10]. Subterranean environments pose a number of additional hazards and challenges, such as outdated maps, dangerous terrain, and low visibility, making them particularly difficult to send humans into safely. For example, emergency personnel cannot safely explore beyond fires in coal mines, and are thus unable to determine the number of people or quality of terrain that may be behind them [43]. Autonomous systems have the potential to significantly improve the quality and timeliness of the situational awareness gained in challenging subterranean environments, and by doing so can reduce risk to human lives.

1.1 The DARPA Subterranean Challenge (SubT)

The DARPA (Defense Advanced Research Projects Agency) Subterranean Challenge was issued to spur the development of new technologies for exploring complex underground environments. Teams are challenged to propose new and innovative solutions for the unique perception, mobility, communication, and autonomy problems present in these environments. Teams compete in a series of 3 separate circuit events, each in a different type of subterranean environment, listed in Figure 1.1. The highest performing teams will be invited to a final event which incorporates elements of each type of environment. In each event, teams are tasked with deploying systems which rapidly provide situational awareness, in the form of map data and locations of predetermined objects placed by DARPA. Each object ("artifact") represents a particular class of objects which are likely to be found in subterranean environments, such as those shown in Figure 1.2.

During each circuit event, teams have 60 minutes to deploy their systems in the environment. The deployment may consist of as many robots as the teams desire, but must be operated by a single human supervisor at the team's base station. Not all areas of the environment will be traversable for all types of robots, so teams are encouraged to develop systems with a variety of mobility capabilities. Autonomous exploration capability is advised to reduce operator load and eliminate a need for constant connectivity. Communication with and coordination of each

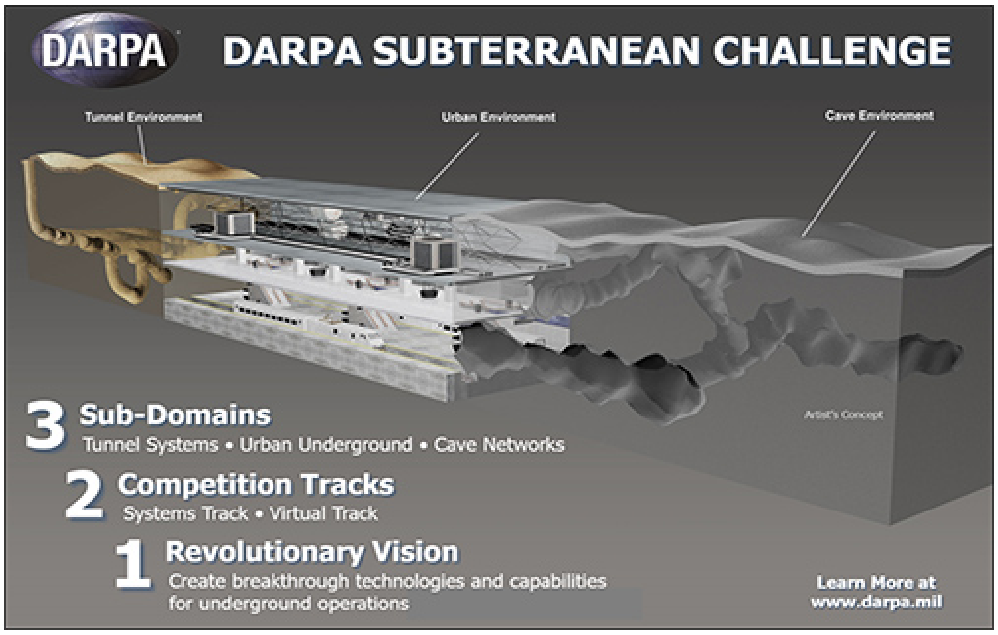

DARPA
DARPA SUBTERRANEAN CHALLENGE
Tunnel Environment
Urban Environment
Cave Environment
Artist's Concept
3 Sub-Domains
Tunnel Systems • Urban Underground • Cave Networks
2 Competition Tracks
Systems Track • Virtual Track
1 Revolutionary Vision
Create breakthrough technologies and capabilities for underground operations
Learn More at www.darpa.mil

Figure 1.1: The Tunnel Circuit focuses on human made tunnel systems. The Urban Circuit focuses on urban environments such as mass transit and municipal infrastructure. The Cave Circuit focuses on naturally occurring cave systems. Image from DARPA [1].

robot in the fleet is a challenge as no communication infrastructure is provided by DARPA. The perception systems used to drive the autonomy should be able to handle a variety of challenging conditions, such as low visibility and variable lighting. This differs from typical perception benchmarks, such as ImageNet [11], which ensure clean and high quality data. Teams may be permitted multiple runs per event.

Before a team's run, DARPA places a predetermined and disclosed number of artifacts in the environment. The locations of specific points on each of these artifacts, referenced to a DARPA-defined frame, are surveyed and treated as ground truth. The location of each artifact and number of each artifact type is unknown to teams. Scoring for each competition event is based entirely on the number of artifacts which are correctly reported to DARPA within the 60 minute period. Teams must report the artifact's category and must report coordinates which are within 5m Euclidean distance of the ground truth for an artifact report to be considered correct. Teams may submit twice the number of artifact reports as there are artifacts in the environment. Artifact reports may be generated in whichever manner a team deems appropriate, such as randomly, by the human supervisor, or fully autonomously by the deployed system. Ties between multiple teams are broken by a number of factors, including the times artifacts were submitted, and the furthest artifacts detected [3].

1.2 Tunnel Circuit Specifics

This work focuses specifically on artifact detection and localization in the context of the Tunnel Circuit. The Tunnel Circuit event took place between at the National Institute for Occupational

Safety and Health Mining (NIOSH) site in Pittsburgh, PA, between August 15 and August 22 2019. DARPA divided the former coal mine into two separate courses, starting at the mine's two separate portals, Safety Research and Experimental. A map of the Safety Research course is shown in Figure 4.1. Each team was allowed 2 runs in each course. The team's final score is a sum of the highest score from each course.

20 artifacts were deployed by DARPA in each course. The placement and distribution of artifacts varied between each course and each run, as indicated in Table 4.1. Five specific categories of artifacts were used at the Tunnel Circuit. A single representative model of each artifact, shown in Figure 1.2, was used. Specific model numbers for each representative object were distributed to teams in advance of the competition [2].

Backpack

The backpack artifact represents a typical adult-sized bag for carrying items. The backpack may be found on the ground, on a wall, or resting on a work surface, such as a table. The backpack contains a sandbag to help keep it in place during the competition. The front backpack is facing outward or upward depending on the initial placement.

Cell Phone

The cell phone artifact represents a radio used for communication, as well as other typical hand-held devices. The cell phone is playing an unspecified full-screen video. Audio is playing from the phone at maximum volume. A 2.4 GHz access point is created by each cell artifact with an SSID of the form "PhoneArtifact##", where "##" is a unique two digit number. The cell phone's Bluetooth radio is on and discoverable.

Drill

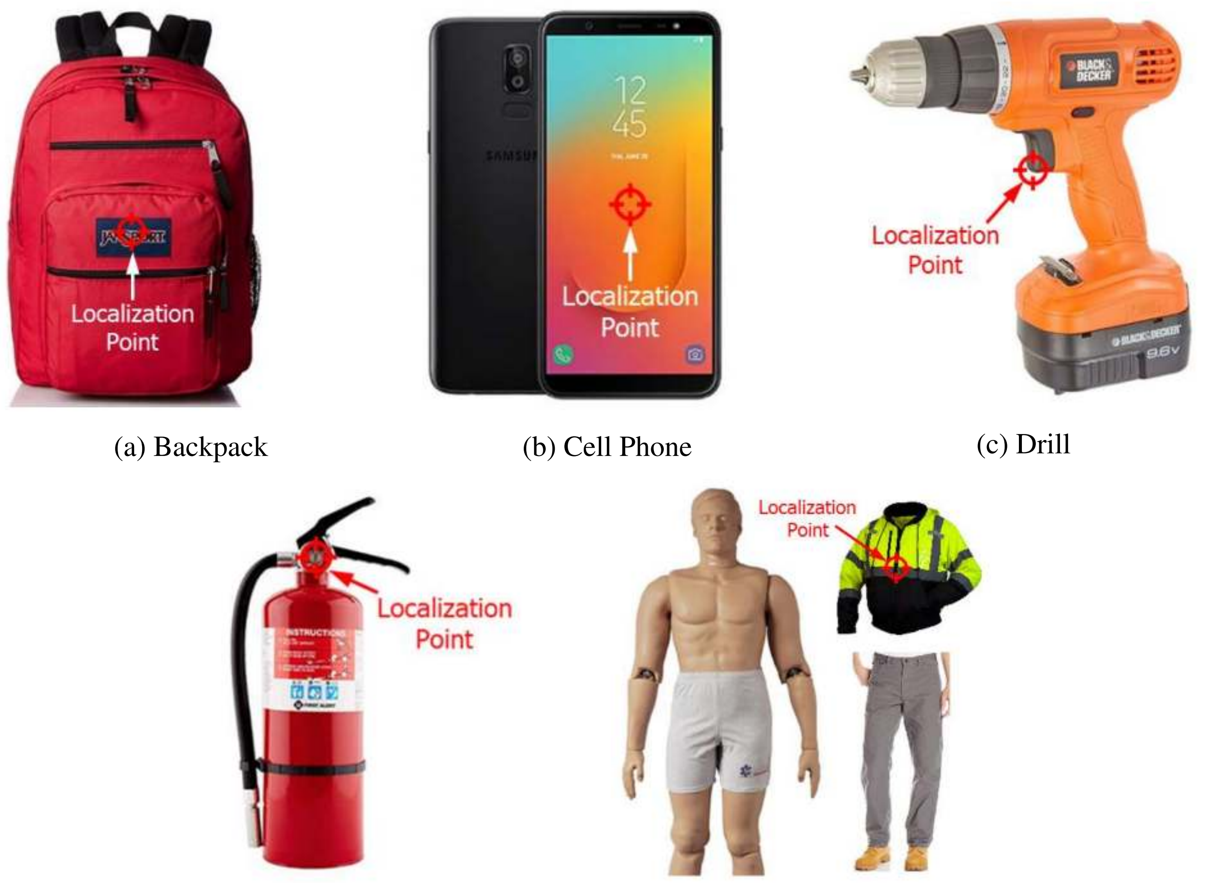
The drill artifact represents a typical handheld tool. The drill has a Philips head driver in its chuck. The artifact may be found on the ground or on work tables. The orientation of the drill is not specified. The drill is not powered during scored runs.

Fire Extinguisher

The fire extinguisher artifact represents a typical common handheld fire extinguisher. This artifact also represents areas where other emergency equipment may be located. The fire extinguisher has its hose in the stored configuration. It may be found on the ground, on a work surface, or hanging from a wall. It will not be used during the scored runs.

Survivor ("Randy")

The survivor artifact represents a human survivor, such as a trapped worker. A thermal manikin is used, and is wearing a high visibility jacket, work pants, and standard steel toed work boots. The manikin has heating elements in its hands and head to partially emulate a human's thermal



(a) Backpack (b) Cell Phone (c) Drill

(d) Fire Exraaktinguisher (e) Survivor ("Randy")

Figure 1.2: The Figure depicts the 5 Tunnel Circuit artifacts. The survivor, cell phone, and backpack artifacts are common to all 3 circuit events. Each circuit consists of an additional 2 circuit-specific artifacts, which were the fire extinguisher and drill for the Tunnel Circuit. All 9 artifacts will be used during the final event. The localization point in the images is the specific point surveyed by DARPA against which error will be measured. Images from DARPA [2].

signature. The manikin is not actuated and does not play auditory clues. The manikin is placed in a static sitting position against walls.

1.3 Our Approach

Our team's overall concept of operations is one of modular autonomy. We aim to develop a set of hardware and software components that can be rapidly reconfigured and adapted to support the needs of any particular environment. Individual modules can be upgraded or replaced as necessary. For mobility, modules such as removable battery packs, wheel assemblies, and drive electronics are used. These mechanical modules combined to form 3 robots during the Tunnel Circuit – R1, a fixed body ground robot, R2, a ground robot with a center pivot body, and D1, a hexcopter. For communication, custom nodes are developed, multiples of which can be stored

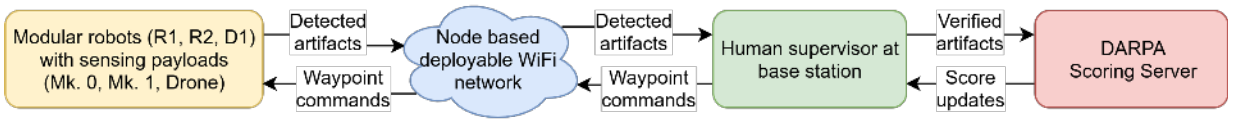

Modular robots (R1, R2, D1) with sensing payloads (Mk. 0, Mk. 1, Drone)
Detected artifacts
Waypoint commands
Node based deployable WiFi network
Detected artifacts
Waypoint commands
Human supervisor at base station
Verified artifacts
Score updates
DARPA Scoring Server

Figure 1.3: Multiple robots explore the mine and deploy communication nodes as they go along. They report artifacts they detect to the human supervisor over the deployed network. The human supervisor verifies the artifacts and sends verified ones to DARPA. DARPA scores the artifacts and returns score updates. The human supervisor uses score updates, as well as other map information received from the robots, to guide autonomous exploration with waypoints.

on modular node dropper assemblies. R1 and R2 each carried a node dropper during the Tunnel Circuit. For perception, this approach means the development of various sensing payloads with integrated software and computation that can be transferred between robots. These sensing payloads run modular state estimation and artifact detection and localization software which can adapt to the capabilities of each system. R1, R2, and D1 each had separate sensing payloads – Mk. 0, Mk. 1, and Drone payload, respectively.

Within each sensing payload, we use a multimodal suite of sensors for artifact detection. Though a single sensor or category of sensors, such as RGB cameras, may be able to detect multiple types of artifacts, it is unlikely to be able to perform well in all conditions. Alternatively, individual sensors may be particularly well suited to detect a specific artifact easily, but may not detect others at all. For example, the survivor and cell phone artifacts emit a thermal signature that may be able to be detected in a thermal camera, leading to the localization of these artifacts even in cases of little to no light. A WiFi radio may be used to detect and localize which is beyond line of sight, but is incapable of detecting any other categories of artifacts. Using a multimodal suite of sensors with intelligent fusion allows us to exploit the strengths of each individual sensor and create a more accurate and robust overall system.

The human supervisor is also considered a module in our system. Though each robot in the system is capable of autonomous operation, the human supervisor is able to combine course and map information reported from multiple robots and direct the exploration patterns of the fleet. Each robot also returns artifact information to the base station, which the human supervisor aggregates and verifies. Artifacts reports which have been approved by the human supervisor are sent to DARPA at the human supervisor's discretion, and are modified and resubmitted as more information becomes available if necessary. An overview of our approach is shown in Figure 1.3.

1.4 Related Work

There has been relatively little recent work which addresses the complete problem of rapid situational awareness in subterranean environments, leading to the issuance of the DARPA SubT Challenge to develop new technologies in this space. However, many smaller components of the overall problem, such as single and multimodal object detection, modular sensing payloads, and object tracking are key research areas and have been well studied in other domains. This work draws inspiration from many of the state of the art techniques and technologies developed for other fields and applies and extends them for use in subterranean environments.

1.4.1 Subterranean Mobile Robots

The early 2000s saw the development of a number of robots intended for use in subterranean environments, such as CMU's Groundhog [16] and CSIRO's Numbat [38]. These robots were built to specifically address mobility and perception challenges presented by underground mines. They demonstrated considerable mobility in a variety of field experiments, but were limited in their autonomous capabilities by the available compute and sensing technologies [32]. They do not have the ability to provide rapid situational awareness to basestation operators, and the approaches used do not scale well to multiple robots.

Groundhog carries two scanning lasers for mapping and navigation, along with a gyroscope, encoders, and tilt sensors for odometry. 2D maps are generated on board and continually optimized to ensure global consistency, but are not reported to the base station due to a lack of communication infrastructure. The maps can be queried from recorded data after Groundhog returns. Though the fidelity of the maps is high enough to be useful for situational awareness, the high latency of the information (potentially multiple hours) can be costly in a disaster response scenario. Even after a base station operator downloads the maps from the robot, searching the point clouds for specific objects is a time consuming and inaccurate process when done manually.

The communication problem faced by Groundhog due to a lack of infrastructure was solved by other subterranean robots with a tether. This enabled robots, such as the MSHA (Mine Safety and Health Administration) Andros V-2, to report video feeds to the base station in real time. The remote camera acts as a source of more rapid situational awareness than the high latency point cloud maps from Groundhog, and provides data in a format more easily understandable by humans. However, tethers proved to be unreliable, being prone to tangling and breakage [33], and thus unsuitable for disaster response scenarios which require robust solutions. Relying on streaming video for situational awareness is also difficult to scale due to the necessity of dividing the human operator's attention, potentially resulting in important data being missed.

The more recent MINBOT-II [44] robot improves upon the tethered communication design with a controlled release fiber line, as well as adds multiple autonomy modes (semiautomatic and fully automatic) to reduce the burden on the human operator. Its tracked mobility design is inherently explosion- and water-proof, allowing for safe and reliable operation in mines. However, though MINBOT-II features more sensors than the Andros V-2 robot, including additional cameras and gas sensors, data is still displayed directly to the human operator without further postprocessing. This does not solve the fundamental issue of divided operator attention and difficult multi-robot scaling.

Our work addresses many of the limitations of previous autonomous systems for providing situational awareness. Each deployed robot in our system autonomously identifies important objects and sends it to the base station over a wireless link. Providing artifacts for situational awareness ensures that the human supervisor receives only high utility information, and reduces the difficulty of operating multiple robots covering a large environment at once. The small size of each artifact also enables transmission over a wireless link even when only low bitrates are available, ensuring real time updates.

1.4.2 Modular Sensing Payloads

A major contribution of our work is a collection of modular sensing payloads for robot mapping, autonomy, and perception. There are a number of academic and commercial projects which have developed payloads which are similar in spirit, though they focus primarily on 3D mapping and autonomy and do not offer on board high level perception algorithms. One such example is the Kaarta Stencil [25], a SLAM-based 3D mapping payload capable of operating on both ground and aerial robots. It provides loop closed maps and pose information to the connected robot, and performs all compute on board. It can also be hand carried if desired, in which case it will store information on board for offline viewing and processing.

A similar system is the Emesent Hovermap [13], a mapping and autonomy payload specifically targeting industrial drones. The payload only uses a rotating LIDAR for its SLAM-based mapping, instead of both (fixed) LIDAR and RGB camera information like the Stencil does, but compensates by offering on board autonomy features instead. It is specifically advertised as being suitable for use in underground mines and other similar GPS-denied environments. Proximity to nearby obstacles is reported to the operator for situational awareness, along with live camera views. The complete 3D maps are only available after the drone has landed and brief offline postprocessing is applied.

The Zebedee Mobile Mapping System [46] is an academic system specifically targeting subterranean mapping in caves. The payload uses a fixed LIDAR mounted on a spring which helps extend the field of view of their selected planar LIDAR. Zebedee requires a separate computer and battery in addition to the payload, both of which are typically carried by a human in a backpack. Data from the LIDAR and IMU is recorded on the attached computer and processed offline to generate a 3D point cloud. Even with the processing time, data can be collected and visualized much more quickly than traditional approaches based on surveying and total stations, but at the cost of map resolution and accuracy. The compact and hand carried nature of the payload makes it usable in tight spaces that would be difficult for robots carrying payloads like the Stencil or Hovermap to access, but the necessity of a human operator prohibits its use in subterranean disaster response scenarios.

1.4.3 2D RGB Object Detection

2D object detection for RGB images has seen significant progress in recent years with the advent of deep learning and high quality datasets such as Pascal VOC [14] and COCO [28]. One of the first frameworks to deliver impressive performance was RCNN [19], which uses 2 distinct stages to detect bounding boxes around objects. RCNN generates multiple region proposals in the first stage, and extracts features from each one using a deep feature extractor (CNN) and uses an SVM to classify the generated features in the second stage. Running the CNN on each proposed region is computationally expensive and was improved upon by Fast RCNN [18], which primarily speeds on RCNN's inference speed by running the deep feature extractor once and then proposing regions from the feature map, instead of the other way around. It also replaces the SVM classifier with a softmax layer to consolidate all training into a single network. Faster RCNN [40] further improves upon the inference speed of Fast RCNN by replacing the selective search based region proposal with a regional proposal network that can be parallelized and run on a GPU, among other improvements.

Even with these improvements, Faster RCNN based detectors are too slow to run in real time on any but the most powerful systems. One stage object detection frameworks such as YOLO and SSD improved inference speed compared to Faster RCNN at the cost of some accuracy. Both YOLO [39] and SSD [30] eliminate the region proposal step and instead used a fixed number of multi scale bounding box priors which are adjusted and classified. The priors are then filtered to retain only ones which are highly likely to be objects. Training with the large number of boxes is difficult due to a class imbalance between positive and negative examples. RetinaNet [29] solves this problem by introducing a new loss function, focal loss, which weights the gradient based on the ease of classification. When combined with an efficient mobile-friendly feature extraction network such as MobileNet [42], each of these object detection frameworks achieves impressive framerates on even modest hardware. Our payloads use object detection networks based on MobileNet and SSD to perform inference at camera framerate for multiple RGB cameras.

1.4.4 RGB + Other Sensor Object Detection

Some prior work exists related to object detection exclusively with non-RGB image-based sensors (e.g. thermal [9, 34]), but is often limited by the lack of large datasets similar to those available for RGB. To overcome this, many approaches choose to utilize existing RGB datasets and networks in some way as a bootstrapping mechanism. [12] uses two branches in its feature extraction network – one operating directly on thermal images, and another operating on pseudo-rgb images generated by an image-to-image translation network with thermal image input. The pseudo-rgb branch is initialized with weights from a pretrained RGB detector. [5] performs joint detection on RGB and thermal image pairs, treating the thermal images as a fourth input channel. The Faster RCNN based network used is initialized with weights from pretraining on COCO. They show that the inclusion of thermal images improves object detection performance over the RGB-only baseline, with notable improvement in nighttime conditions.

Point cloud based object detection methods such as [8, 26] provide ways of utilizing 3D data for object detection, but suffer from the limited resolution available in modern LIDAR sensors. A variety of techniques to fuse LIDAR and RGB data for improved performance, as measured on datasets like KITTI [17], have been proposed. Much of this work has been motivated by the need for self driving cars to accurately perceive their environments in all conditions [15]. [36] performs 2D object detection in RGB images and then projects and refines detections into 3D using LIDAR information. [27] instead jointly extracts features from LIDAR and RGB camera data and performs ROI feature fusion to achieve state of the art performance on KITTI.

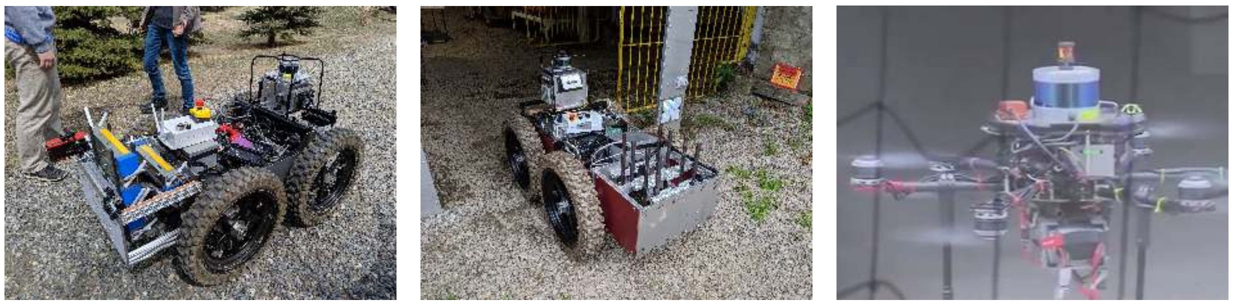
Unfortunately, though they achieve impressive accuracy, state of the art deep fusion techniques do not typically run in real time on constrained hardware. Many of the custom operations used by these networks are unavailable in optimized inference frameworks such as TensorRT and OpenVINO, resulting in either a complete inability to run the network or a fallback to slow reference implementations in other frameworks such as TensorFlow. Even disregarding the compute issues, field of view overlap is required for deep fusion, which is not the case between any pair of RGB and thermal cameras on Mk. 1. Furthermore, the precise sensor synchronization between either RGB and thermal images, or RGB images and LIDAR scans is missing in all of our payloads due to hardware limitations.

Chapter 2

Hardware Development

This chapter covers the development of 3 separate modular sensor payloads, designed in line with our team's concept of operations of modular autonomy. These 3 payloads, Mk. 0, Mk. 1, and Drone, were carried by 3 modular robots, R1, R2, and D1 respectively during the Tunnel Circuit event. A picture of each robot with its payload can be seen in Figure 2.1. The individual payloads can be seen in Figure 2.2. A summary of the components and capabilities of each payload is given in Section 2.7. Each payload is capable of performing all of the high level perception and planning tasks for the robot that carries it, and performs all computation on board.

The Mk. 0 payload was the first sensor payload to be built, and is inspired by the use of the "Blue Payload", a payload similar to the Kaarta Stencil [25] which is capable of state estimation. Mk. 0 proved to be too heavy to carry on D1 and was thus simplified into the Drone payload. Mk. 0 also has certain hardware limitations and proved to be difficult to maintain, both of which were addressed during the design of Mk. 1. These two payloads maintain the same mounting mechanism and similar form factors, allowing for interchangeability between R1 and R2. A component level schematic for Mk. 1, the most capable payload of the 3, is given in Figure 2.3.



(a) R1 with Mk. 0 Payload (b) R2 with Mk. 1 Payload (c) D1 with Drone Payload

Figure 2.1: Tunnel Circuit robots with sensing payloads

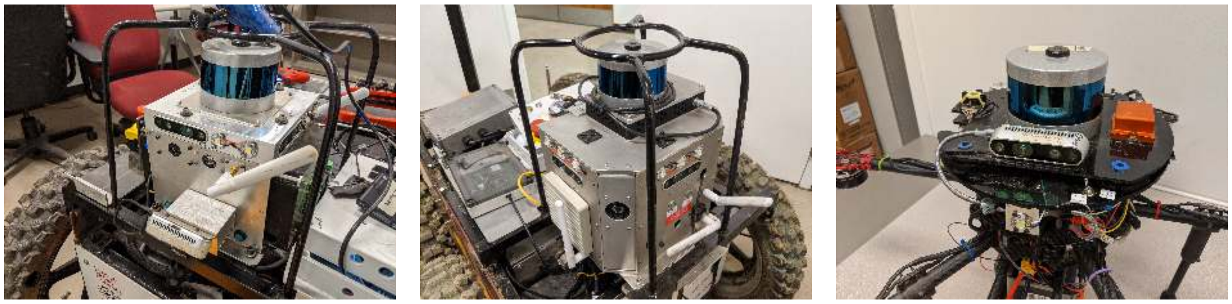

(a) Mk. 0 Payload Close-up (b) Mk. 1 Payload Close-up (c) Drone Payload Close-up

Figure 2.2: Tunnel Circuit payload close-ups

2.1 Payload Requirements

Inspiration for the design and feasibility of a modular sensing payload resulted from using the Blue payload, a prototype state estimation payload similar to the Kaarta Stencil [25]. The payload is self-contained, and can easily be transferred between small and large ground and air robots, as shown in Figure 2.4. It is simple to use and integrate, requiring only power and a network connection to provide state estimation output. Though the Blue payload's sensing and compute capabilities are insufficient for the complete planning and perception challenges of the DARPA Subterranean Challenges, its use served as the basis for the following hardware requirements (HWR) set for Mk. 0, the first payload built. The requirements are also derived from the timeline and obstacles imposed by the SubT Challenge.

HWR1: Rapid Development There were a little under 3 months from the beginning of the competition (September 2018) to the first qualification deadline (December 2018), by which time the payload must be completed. This short timeline heavily incentivizes drawing from previous experience and familiarity, such as with components and sensors, and prefers in-house manufacturing capabilities with short lead times.

HWR2: Self-contained All of the sensing and computation for the high level autonomy features should happen inside the payload itself. Ideally, the payload would only be supplied power and a network connection, just like the Blue Payload, and would output autonomy goals (such as waypoints) and information (such as robot state, maps, and artifact locations) to be relayed to the human supervisor at the base station.

HWR3: Environmental Robustness The field environments for the SubT competition events were expected to be somewhat hostile to the sensors. Specifically, the rules mention that "dust, fog, mist, water, and smoke are within scope" for the Tunnel Circuit [3], and thus the payload should be reasonably protected against these elements. The payload should also be robust to the mechanical loading it would be subject to as a result of rough terrain.

HWR4: Weight Sensitivity The Blue payload demonstrated that the ability to use the same payload between ground and aerial platforms, as in Figure 2.4, was useful in building modular systems, and the same ability was desired for the Mk. 0 payload. The relatively low payload capacity of D1 set a hard upper bound on the weight of Mk. 0, though as low of a weight as possible was preferred due to the inverse relationship between weight and

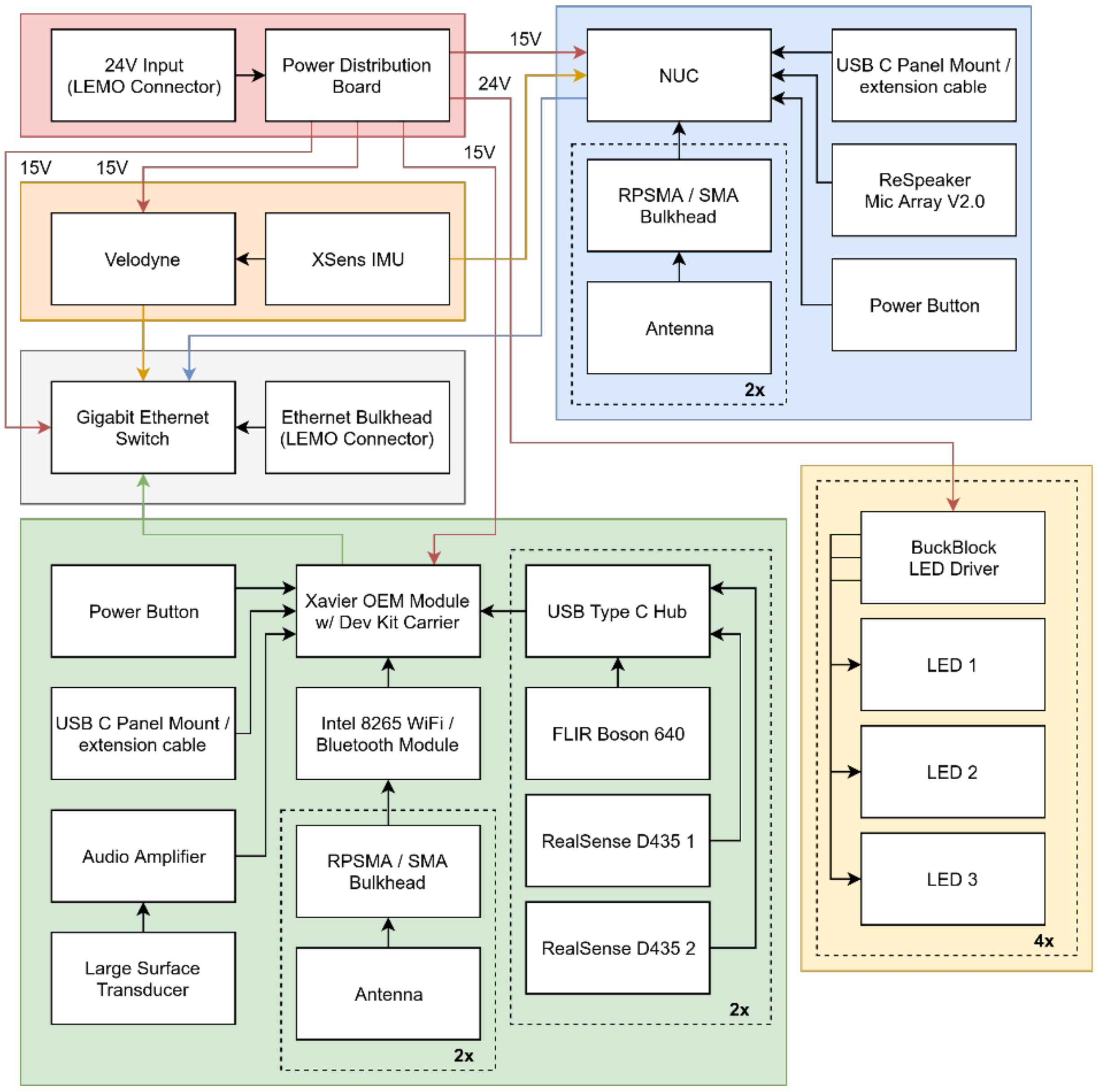
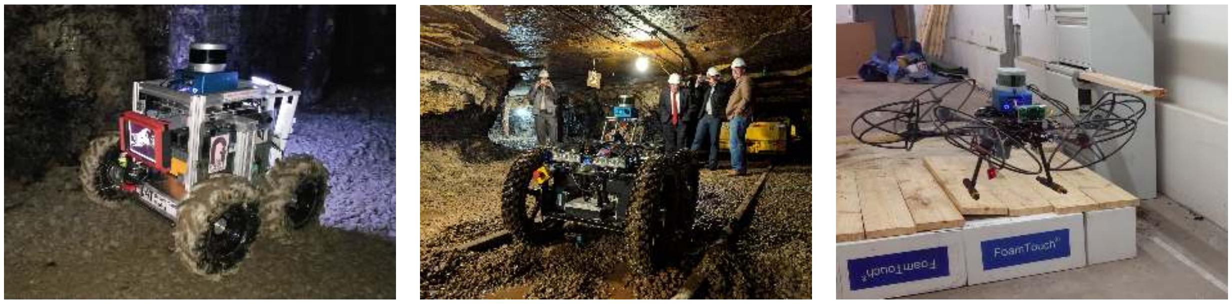


Figure 2.3: Subsystem diagram for the Mk. 1 payload showing the major electrical components present. Payload case fans are not depicted. All state estimation and planning computation happens on the NUC. The Xavier is dedicated solely to artifact detection and localization.



(a) Joeybot with Blue Payload (b) R1 with Blue Payload (c) D1 with Blue Payload

Figure 2.4: The picture shows 3 robots of different sizes each carrying the same Blue payload. Joeybot is a small test platform developed for early testing of autonomy code, but was not used during the Tunnel Circuit. The depicted aerial vehicle is a prototype of D1, featuring tilted rotors instead of level ones.

flight time.

HWR5: Cost Sensitivity Though there are not specific cost restrictions imposed for the robots or payloads, DARPA is interested in identifying cost effective solutions for the SubT Challenge. Minimizing cost is also useful as a design goal to support the creation of multiple copies or iterations of Mk. 0, rather than just a single expensive prototype.

2.2 Mk. 0 Component Selection

The first step in the development of Mk. 0 was the selection of the individual sensors, computers, and supporting components which would be packaged inside the payload. HWR2 (rapid development) dictated that, where possible, familiar components were selected to reduce the time necessary to develop the payload. This was the primary consideration in certain cases, such as with the LIDAR selection. Other components, such as the RGB cameras, were compared against other candidates which significantly exceeded the familiar choice in some metric, such as HWR4 (weight sensitivity), HWR5 (cost sensitivity), or data quality. The unfamiliar choice was chosen if its increased performance or satisfaction of requirements outweighed the decreased familiarity.

It is important to note the context for the development of Mk. 0. During its design, the preliminary planning and navigation stack was being developed and tested on Joeybot, carrying the Blue payload. Work on the artifact detection and localization software pipeline had barely begun. The early stages of these software components meant that sensor processing capabilities and compute requirements were unknown, necessitating educated guesses. Where multiple choices were available, components were selected to allow for the most flexibility in future software algorithms, even at the cost of lower priority requirements.

2.2.1 Computer Selection

Each sensor payload performs 3 broad tasks – state estimation, path planning and navigation, and artifact detection and localization. Though specific resource requirements were not known at this stage, previous experience and available literature could be used to make approximations. Computer selection was considered for each task separately.

State Estimation

The state estimation system used on the Blue payload is LOAM [45], the current state of the art for SLAM as measured on the KITTI dataset [17]. Its performance, as well as the familiarity developed with it while working on Joeybot with the Blue payload, made it an easy selection for the SLAM system for all payloads. LOAM was specifically tuned to run on the hardware inside the Blue payload, which has an Intel NUC with an Intel Core i7 processor inside. To avoid the potentially time consuming process of tuning LOAM to run on a different hardware platform, the NUC was selected as a required component for Mk. 0. Specifically, the NUC8i7BEH model was used as it was the NUC board with the most powerful CPU available at the time. LOAM utilizes the majority of 2 cores, leaving an additional 2 physical cores potentially open for other tasks.

Path Planning and Navigation

The path planning and navigation stack was being actively developed on Joeybot and ran on Joeybot's internal computer, a mid-range industrial-grade PC from Logic Supply. Though the Logic Supply computer was also using an Intel Core i7 processor, the processor itself was underclocked to reduce heat output. Profiling indicated that it should be possible to run the path planning and navigation stack on the 2 available cores of the NUC being used to run LOAM. Significant increases in the compute load were not expected, so it was decided to use a single NUC8i7BEH to run both LOAM and the path planning and navigation stack to avoid increasing system complexity.

Artifact Detection and Localization

At this stage, work on the artifact detection and localization pipeline had just begun. It was unclear what the final approach would be, and thus predicting its computational requirements proved difficult. Convolutional neural networks were shown in the literature to outperform traditional methods in a number of machine perception benchmarks [11, 17, 28], and were thus expected to be a part of the pipeline. However, they require a significant amount of computing resources to run at high framerates (10+ Hz). Multiple neural networks would potentially need to be run in parallel, depending on the quantity of image-based sensors selected. The following hardware platforms were considered, each selected for being small enough and robust enough for inclusion in a modular sensing payload.

1. Intel Movidius Neural Compute Stick
2. Integrated GPU on selected NUC (Intel Iris Plus Graphics 655)
3. Nvidia Jetson TX2
4. Nvidia Jetson AGX Xavier

Of the available options, the Nvidia Jetson AGX Xavier ("Xavier") offers the highest inference performance due to its many CUDA cores, as well as its specialized Tensor Cores and Deep

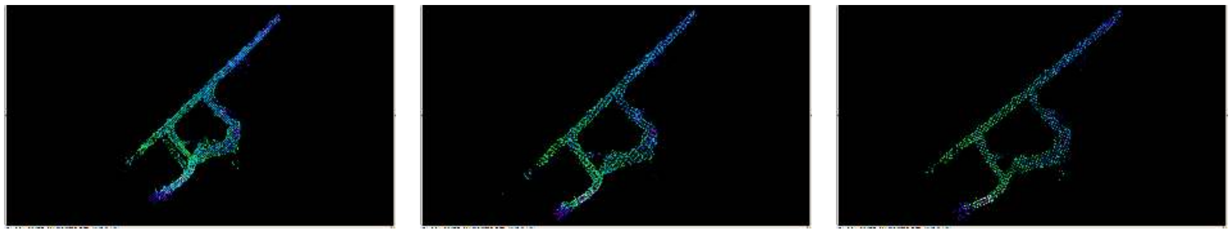

(a) 100% points kept (b) 75% points kept (c) 50% points kept

Figure 2.5: Visualization of LOAM running on an Xavier with different downsampling parameters, using pre-recorded data collected at the Tour-Ed Mine in Tarentum, PA. While the scan matching is correct when 50% of points are discarded due to being able to register scans within 100 ms (fast enough to keep up with the 10 Hz data rate), the number of discarded points was deemed to be too severe.

Learning Accelerators (DLA). Combined, the Xavier is capable of performing more than an order of magnitude more FLOPS (floating point operations per second) than the other platforms, but has the highest cost of all options and weighs the most by a small margin. As no specific compute requirements were determined for the artifact detection and localization stack at this time, the Xavier was selected to provide the highest ceiling for available compute resources.

LOAM on Xavier

The option of using the Xavier to run the entire autonomy stack, consisting of state estimation, path planning and navigation, and artifact detection and localization, was briefly considered but summarily dismissed due to a belief that the Xavier's CPU would be insufficient. After the completion of the Mk. 0 payload, some experiments were performed to determine the validity of this hypothesis. Profiling of the NUC revealed that LOAM was one of the heaviest processes, consuming nearly 100% of a single core in some circumstances, and smaller portions of other cores. Thus, porting LOAM to the Xavier was the first test.

Before benchmarking LOAM on the Xavier, all CPU and GPU cores on the Xavier were enabled (MAX-N mode), and had their frequency maximized. The frequency governor was disabled, and the fan on the Xavier Developer Kit was permanently set to run at the maximum speed to prevent overheating during tests. After setup, LOAM was run in a few different configurations on the Xavier on a playback of pre-recorded data. For each configuration, a downsampling parameter was adjusted which controlled the number of points kept from the laser scan point clouds during a preprocessing step before scan matching inside LOAM. A lower number of points used during scan matching typically results in a lower match accuracy, but requires fewer cpu cycles to run. Additional scans received during scan matching are dropped, requiring that scan matching complete under the output rate of clouds from the LIDAR (10 Hz), and resulting in higher scan matching error otherwise. Figure 2.5 shows a visualization of the results.

In Figure 2.5a, the same configuration as the NUC on Mk. 0 was used. Severe misalignment is visible - the lower tunnel does not align correctly, and a secondary "ghost" tunnel is created. In Figure 2.5b, 75% of the laser scanner's points are kept. This does result in better alignment, though the images of two tunnels are still clearly visible in the figure. Finally, in Figure 2.5c, only 50% of the laser scanner's points are kept. In this environment, the point cloud alignment is successful, suggesting that the state estimate did not drift significantly. However, the requisite 50% downsampling was deemed to be too significant, as it presented a high risk of misalignment under harsh motions or in feature-bare environments due to the low density of points. This experiment confirmed the initial hypothesis that, without significant optimization, the current autonomy stack would be unable to run on a single Xavier.

2.2.2 State Estimation Sensor Selection

Adhering to the requirement of rapid development (HWR1), the first candidates for state estimation sensors were selected to be those already validated to work with LOAM inside the Blue payload – the Velodyne VLP-16 Puck LIDAR and an Xsens IMU. Alternative laser scanners, such as the Ouster OS1, were considered for use with LOAM, but the time required to tune and validate LOAM for these new sensors was deemed to be too much. The specific combination of sensors within the Blue payload has hundreds of hours of use and was proven to be reliable in a variety of environments. Achieving the same with new sensors would take prohibitively long and thus, the exact same sensor complement as the Blue payload was selected for state estimation within Mk. 0.

2.2.3 Sensor Category Selection

The selection of sensors for artifact detection was not as simple as that for state estimation. At this point in the competition, DARPA had not released a specific list of artifacts that would be used in the challenge, only 10 general categories [3]. These categories were compared against the types of sensors which we believed could be integrated into Mk. 0 in a short amount of time. The utility of each type of sensor in detecting artifacts of each category was estimated according to the following guidelines, and is summarized in Table 2.1.

High Artifacts of this category are expected to produce extreme sensor responses and be easy to distinguish.

Medium Artifacts of this category are expected to be found in sensor data with moderate difficulty.

Low Artifacts of this category are expected to be difficult or expensive to separate from sensor noise, perhaps due to a low response, or anticipated interference.

None Artifacts of this category are expected to effect no response in sensors of this category.

Table 2.1 indicates that while there is some redundancy across sensing categories in their speculated ability to detect various artifact categories, most of the sensor types have at least one artifact category that they alone would be highly likely to detect. For example, WiFi / Bluetooth scanning is the only sensing category rated as "High" utility in detecting radios / cell phones. Based on this table, Mk. 0 should contain at least some sort of RGB camera, thermal camera, depth camera, WiFi / Bluetooth scanner, and gas sensor in order to have a high chance of detecting each of the possible artifact categories. A LIDAR is not required for object detection, but will necessarily be included for state estimation as described above. The microphone is

	RGB Camera	Thermal Camera	Depth Camera	LIDAR	Microphone(s)	Wifi / Bluetooth Scanning	Gas Sensor
Survivors	High	High	High	Medium	Low	None	None
Ingress / Egress Points	Medium	Low	High	Medium	Low	None	Low
Electric Pumps	High	High	Low	Medium	Medium	None	None
Backpacks	High	None	Low	Low	None	None	None
Valves	High	None	Low	None	None	None	None
Radios / Cell Phones	Low	Medium	Low	None	Medium	High	None
Tools / Fire Extinguishers	High	None	Low	None	None	None	None
Power Sources	Medium	High	Medium	Medium	None	None	None
Oxygen Level	None	None	None	None	None	None	High
Gas Leaks	None	Medium	None	None	Medium	None	High

Table 2.1: Estimated utility of various sensor categories for DARPA provided artifact categories. These sensor categories were selected for being easy to integrate into the Mk. 0 payload. At this point in the competition, the specific list of artifacts for the Tunnel Circuit had not yet been released, forcing comparison against the overall list of categories instead.

not strictly necessary to detect any particular artifact category, but could be useful for future studies. If possible, it would be included in the payload, but it was considered low priority and little attention was devoted to it. With the list of sensor types decided, experimentation began to determine specific sensors to use from each category in Mk. 0.

2.2.4 RGB Camera Selection

An RGB camera was the first sensor selected as the category offered highest overall utility across a variety of sensor categories. The first selection criteria for the camera was the interface type. Three possibilities were considered - CSI, Ethernet, and USB. CSI, capable of offering very high data rates with low latency, was the preferred interface. However, hardware and driver support for CSI cameras for the Xavier was still developing at this time, eliminating this option. Ethernet was considered for ease of integration. However, the larger size of Ethernet cameras and connectors, as well as the lower available bandwidth for multiple cameras forced this option to be eliminated as well. This left only USB for the camera interface, forcing it to be selected by default. USB 3.0 offers the data rates needed to interface with multiple cameras, but has the potential to suffer from high latency of and jitter in between frames due to multiple levels of hardware and software buffering.

After finalizing USB for the camera interface, two specific USB camera models were considered for use in Mk. 0. The first was the UI-3241LE-C-HQ by IDS Imaging, a bare camera board that we had familiarity with from successful use in other projects. The second was the Intel RealSense D435, selected for its inclusion of a stereo depth pair in a small form factor. When comparing the two cameras, in addition to the overall payload goals, there were a few camera-specific criteria used:

Low light performance The subterranean environments were expected to be dimly lit in general, and occasionally be completely dark. A camera with lower image noise in dark environments was preferred.

Synchronization ability Given the expected utility of the RGB cameras, as well as for redun-

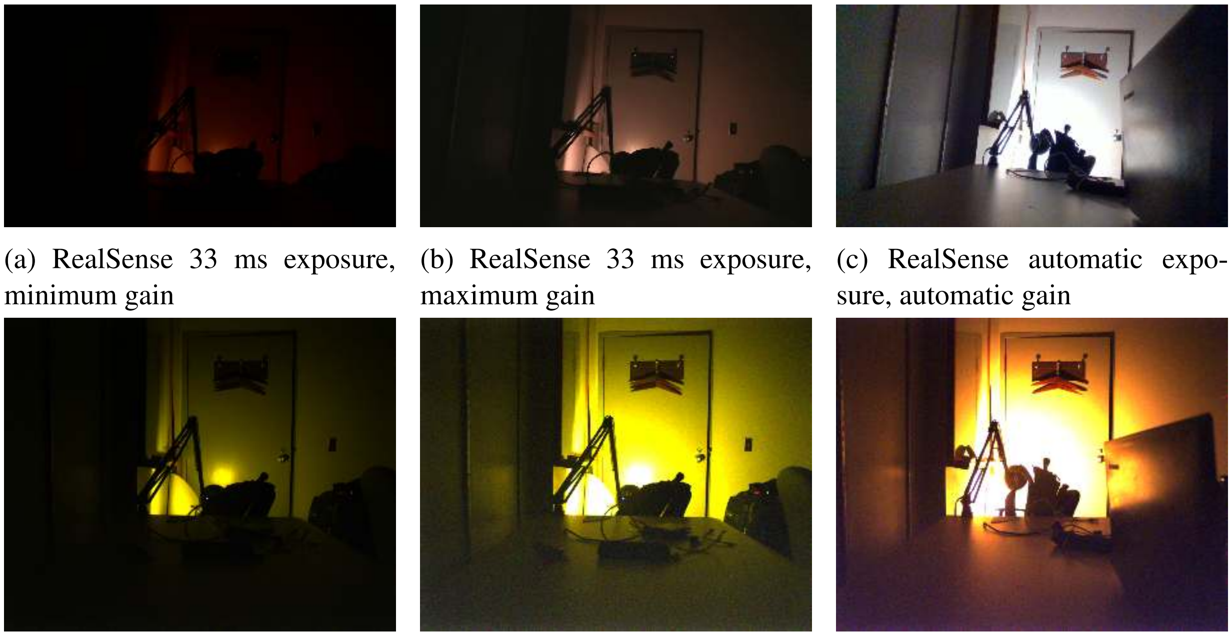

(a) RealSense 33 ms exposure, minimum gain
(b) RealSense 33 ms exposure, maximum gain
(c) RealSense automatic exposure, automatic gain
(d) UI 33 ms exposure, minimum gain
(e) UI 33 ms exposure, maximum gain
(f) UI automatic exposure, automatic gain

Figure 2.6: Comparison of image quality from UI-3241LE-C-HQ and RealSense D435 RGB cameras across a variety of gain values. Automatic exposure was limited to 33 ms due to a specified framerate of 30 fps for both cameras. It should be noted that after this experiment was performed, it was discovered that the RealSense firmware version used had a bug when setting manual gain which lowered the maximum possible gain value. The automatic gain feature did not have this bug, which could explain why the RealSense image with automatic gain in 2.6c appears significantly brighter.

dancy, it was anticipated that multiple cameras would be used on Mk. 0. A hardware synchronization mechanism between the cameras and other payload sensors (e.g. LIDAR) would help increase the accuracy of various software algorithms.

Shutter type A global shutter was preferred over a rolling shutter due to the increased image quality under harsh camera motions, which was expected as a result of rough terrain.

To evaluate low light performance, the provided ROS [37] drivers were used to capture images from cameras at their native or recommended resolutions (1280 x 1024 for the UI camera, and 848 x 480 for the RealSense) across a range of manually selected exposure and gain values with the camera framerates set to 30 fps. The images shown in Figure 2.6 contain images of the same scene from both cameras with 33 ms exposure time and the minimum and maximum gains supported by the drivers. Additionally, images of a similar scene were captured with autoexposure enabled in each camera.

With manual gain enabled on both cameras, the images from the RealSense RGB camera are less bright, but exhibit significantly less noise than the images from the UI camera. Additionally, the UI camera's image at maximum gain 2.6e saturated in the center, while the RealSense image did not, suggesting a lower dynamic range on the UI camera. When autoexposure was enabled on both cameras (2.6c, 2.6f), the RealSense exhibited comparable noise to the UI camera, but produced an image with colors more accurate to the actual scene.

When comparing synchronization ability, it appeared at first glance that both cameras supported external frame triggering, which allows frame capture to be driven by an external clock. This feature had previously been validated on the UI cameras in other lab projects. However, upon closer inspection it appeared that the external trigger for the RealSense module only applied to the depth module, and use of the external trigger would remove the synchronization between the RGB and depth modules on the RealSense camera. Similarly, when comparing shutter types, it appeared at first glance that both cameras had global shutters. This has also been previously validated on the UI cameras in other lab projects. However, further investigation revealed that the RealSense had a global shutter only for the 2 IR cameras used to compute depth, with the RGB camera instead having a rolling shutter.

After comparing the two camera models in these experiments, the RealSense D435 module was selected for use in Mk. 0. It was believed that the dramatic increase in low light image quality would outweigh the RGB camera rolling shutter and lack of external triggering. Additionally, the RealSense contains a depth module which eliminated the need for the selection of a separate depth camera. The RealSense also beat the UI camera in other payload requirements – its price was less than half that of the UI camera (HWR5 cost sensitivity), had shorter lead times (HWR1 rapid development), and had an enclosure which would simplify keeping the payload environmentally robust compared to the bare UI camera board (HWR3 environmental robustness). A total of 4 RealSense modules would be used in Mk. 0, arranged approximately in a square to allow for easy mounting and to provide a nearly 360 degree horizontal field of view with the RGB cameras.

2.2.5 Thermal Camera Selection

The thermal camera selected for the Mk. 0 payload was the FLIR Boson 320 with a 92 degree HFOV. The Boson camera core was newly released at the time Mk. 0 was being developed, and its small form factor, along with ease of integration with the available USB interface, similar to the selected RGB cameras, made it a compelling option. The 92 degree HFOV model was selected specifically as it was believed the larger FOV would allow more artifacts to be visible during a deployment. Additionally, the 92 degree HFOV configuration is the only model to come with a special diamond-like coating which is qualified against harsh abrasion. This allowed us to place the camera in Mk. 0. without a special protective window, which would have been otherwise difficult to do due to the expensive and specific type of glass needed. Other thermal camera options were briefly considered, but the FLIR product was ultimately selected due to familiarity and successful use of some of their other products in other projects. An example highlighting the utility of thermal cameras in fog is shown in Figure 2.7.

2.2.6 Microphone Selection

The microphone was not identified to be a critical sensor in Table 2.1, and thus its selection was not dedicated significant resources. The "Insten VOIP/SKYPE Mini Flexible Microphone for VOIP/SKYPE - Black" was purchased from Amazon.com and was intended to be connected

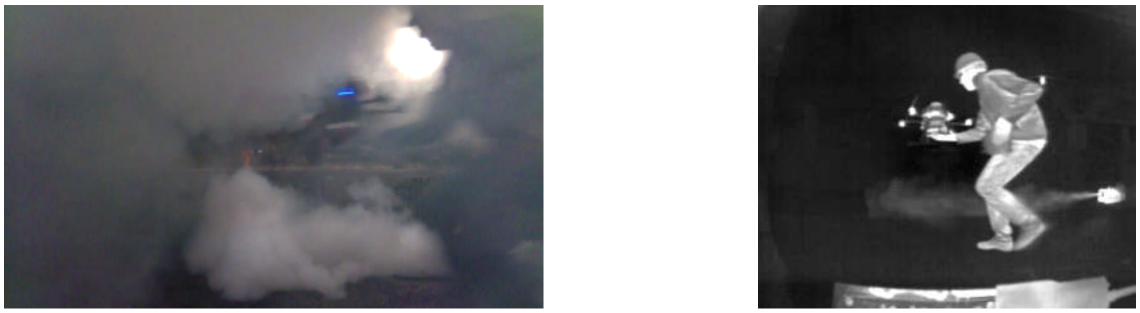

(a) RGB Image (RealSense D435) (b) Thermal Image (FLIR Boson 320)

Figure 2.7: The figure shows a scene containing a drill and a person carrying a drone in heavy fog. The RGB camera is able to see the drill in the parts of the image without fog, but is unable to easily distinguish the human. The thermal camera is able to clearly show the human through the fog but does not register a strong response for the drill.

to NUC. After being unable to record audio information with the microphone connected to the NUC, it was discovered that the NUC's 3.5mm audio jack was a combination headset and microphone jack. A jack splitter was purchased and used to connect the microphone to the NUC successfully. Proof-of-concept driver support was added to record audio from the microphone, but the recorded audio data was never utilized.

2.2.7 WiFi / Bluetooth Scanning Hardware Selection

The NUC used inside the Mk. 0 payload contains an Intel Dual Band Wireless AC + Bluetooth 9560 module, capable of performing both WiFi and Bluetooth scanning simultaneously. The module is not used for other tasks on the robots as their wireless functionality is achieved with other, more specialized mesh hardware. Utilizing the hardware already contained on the NUC meant a reduced component count and easier integration, which was consistent with the payload goals (HWR1 rapid development, HWR5 cost sensitivity), making it a simple choice. Adapters were attached to the 9560 module to convert the dual MHF IV connectors to RP-SMA, allowing multiple antenna configurations to be easily tested without risking damage to the fragile MHF IV connectors on the module.

2.2.8 Gas Sensor Selection

With no specific information provided by DARPA about the gas leak artifact, we chose to focus our preliminary efforts on detecting oxygen levels. The Grove Oxygen Gas Sensor from SeeedStudio was selected for its low price and apparent ease of integration through an analog interface. However, the sensor did not report the expected oxygen concentration values (approximately 21%) using the equations from the provided documentation. The decision was made to continue with Mk. 0 payload development without a gas sensor for the time being, with the intention of exploring alternative options and updating the payload as necessary in the future as more information was released. However, no further work was done with gas sensors as it was revealed soon afterwards that the Tunnel Circuit would not contain any gas artifacts.

2.2.9 Lighting Hardware Selection

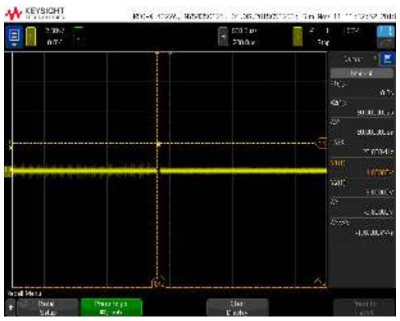
While the low light performance of the RealSense RGB camera was considered acceptable, adding additional lighting to the payload would enable images to be captured with lower noise, as well as allow for lower exposure times which could help decrease motion blur. Additional lighting would also be required for successful object detection in completely dark environments, which were to be expected in subterranean environments. To provide this additional lighting, the 5000K LED from the Opulent XHP70A series of LEDs, mounted on a starboard for ease of integration, was selected. This particular LED was chosen because it has the highest rated luminous flux of all LEDs we were able to find with the starboard form factor at the target voltage of 12V. 2 LEDs would be used per camera for increased brightness, as well as some redundancy in the case of failure of one of the LEDs.

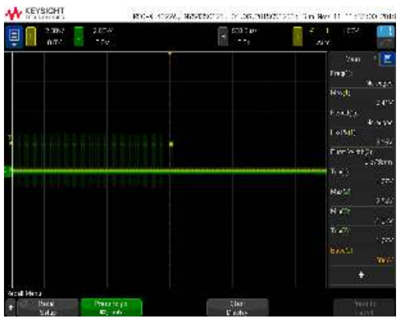
After selecting the LEDs, the original intent was to synchronize and flash each pair of LEDs with the frame exposure of its associated RGB camera. Doing so would require some way of determining when the frame exposure was happening. The RealSense D435 has an expansion header with 9 pins, many of which were undocumented at the time. 1 of these pins was documented as being intended for synchronization of multiple depth modules, but no further details were provided. While probing the other undocumented pins with an oscilliscope did not reveal any signals that could be used to flash the LEDs, it revealed that the depth synchronization pin is simply a 1.8V 50μs pulse. The RealSense documentation was shortly updated to indicate that this pulse occurred slightly after the end of the depth frame exposure. Documentation also revealed that in normal operation, with the RGB camera acting as the master for the entire RealSense module, the start of the depth and RGB frame times should be synchronized. Using this information, the following equation is derived, allowing the time of the start of the next RGB camera exposure to be calculated from the start of the depth trigger:

$$t_{rgb+1} = t_{depth} - t_{delay} - t_{exposure} + t_{frame} \tag{2.1}$$

where t_{rgb+1} is the start of exposure of the next RGB frame, t_{depth} is the time of the start of the depth pulse, t_{delay} is the delay between the end of depth exposure and the start of the depth pulse, $t_{exposure}$ is the exposure time of the depth frame, and t_{frame} is the period of the camera frames. t_{delay} was estimated by looking at the difference between the end of the depth projector control signal (in green) and the start of the depth pulse (in yellow), as shown in Figure 2.8b, and was set to 100 μs. The magnitude of this delay is negligible compared to t_{frame} and $t_{exposure}$, which are in the tens of milliseconds, but it is provided for completeness.

To implement the LED flashing, the depth pulse was wired into an interrupt pin on a microcontroller. The interrupt triggered a timer which would sleep until the start of the next RGB frame exposure, as calculated in 2.1. At the start of the frame, the LEDs would be turned on, and a second timer was initialized to turn the LEDs off after a specified period. The LEDs were controlled with the LED driver's dimming functionality, set to either maximum brightness or off. The LED driver selected was the BuckBlock A009, a constant current LED driver capable of supplying 2.1A at up to 16V, which was sufficient to drive 2 of the selected LEDs at their rated


(a) 1.8V 50μs pulse after depth frame


(b) Measurement used to estimate t_{delay}

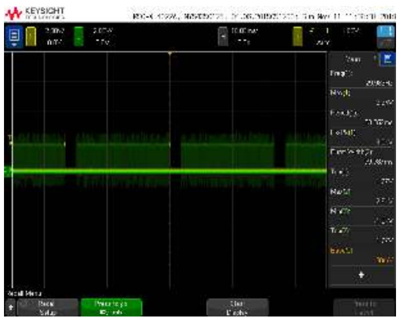

(c) Depth triggers generated for each frame

Figure 2.8: Oscilliscope measurements of RealSense D435 depth trigger. The depth pulse is shown in yellow. The IR projector control signal, active for the duration of the depth frame, is shown in green. The figures depict the depth trigger signal which occurs slightly after the end of exposure of every depth frame, and can be used to synchronize the flashing of LEDs with the RealSense RGB camera image exposure.

brightness when wired in parallel.

Unfortunately, the LED flashing created artifacts in the RGB camera image. The streaking visible in Figure 2.9 is believed to be a result of the rolling shutter on the RGB camera modules. Though each row in the RGB image may be exposing for less time than the frame period, the rows are being scanned sequentially, over the duration of the frame period. This means that if the LEDs are on for less time than the frame period, certain rows (depending on the exposure time set by the autoexposure algorithm) will not expose while the LEDs are on, resulting in a dark streak as seen in the figure. The specific dark rows are a function of the phase offset between the LED pulse and the rolling shutter.

The black streaks visible in Figure 2.9 led to the decision that LED flashing would not be integrated for Mk. 0. The LEDs would instead be left on constantly, powered by a 12V constant voltage source rather than a separate LED driver. This would burn more power and in turn generate more heat on the payload, but would result in higher quality images to be used by other components in the pipeline. The connection of the voltage source to the LEDs was controlled by a switch to allow a user to disable the bright LEDs when not in use.

2.2.10 Networking Hardware Selection

Inside the Mk. 0 payload, the NUC, Xavier, and LIDAR are all connected together via Ethernet. Additionally, an Ethernet port is exposed from the payload to enable data output. In the interest of compactness, a GigEthos Lite board from Gadgetsmyth, which promised to be a very light and small gigabit Ethernet switch, was used originally. The board was successfully used inside the Mk. 0 for a few months after the initial assembly, but suddenly experienced an unexpected failure during normal operation. At that time, the GigEthos Lite board was replaced with a stripped down commodity TP-Link gigabit Ethernet switch. The failure of the GigEthos Lite board is currently believed to be caused due to insufficient cooling inside the payload, and it is

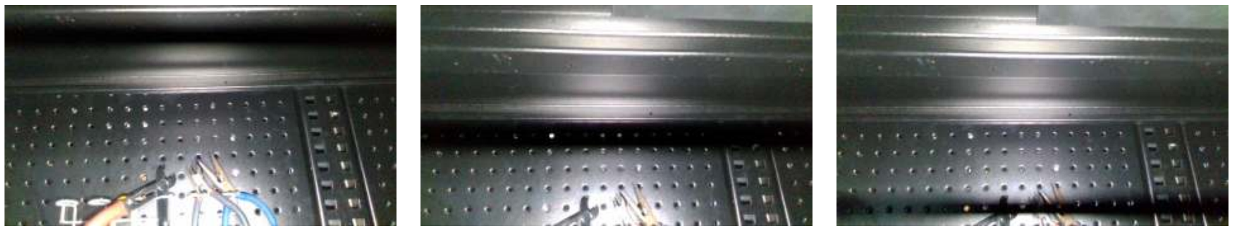

(a) Top dark stripe (b) Middle dark stripe (c) Bottom dark stripe

Figure 2.9: Dark lines are generated in the RealSense RGB images as a result of flashing the LEDs. The different locations of the dark stripes are caused by phase offsets between the LED pulse and the frame exposures. The phase offset was controlled by modifying t_{delay} in Equation 2.1 to be between 0 and the frame period t_{frame}.

believed that the larger surface area of the TP-Link board has helped prevent it from suffering from the same thermal issues.

2.2.11 USB Hardware Selection

The Mk. 0 payload contains a total of 6 USB sensors - 4 RealSense cameras, 1 thermal camera, and 1 IMU. All of the sensors except the IMU plug into the Xavier. However, the Xavier carrier board used (the developer kit carrier board) only has 3 USB ports - 2x 3.1 Gen 2 (10 Gb/s) Type C ports, and 1x 3.1 Gen 1 (5 Gb/s) Type A port. This meant that a USB hub was necessary for Mk. 0. The HB31C4AB hub from StarTech.com was selected as it was the only USB Type C hub with 4x USB Type A inputs capable of 10 Gb/s. Though the RealSense D435 cameras are only USB 3.1 Gen 1 (5 Gb/s) devices, which would limit the bus to 5 Gb/s, this 10 Gb/s capable USB hub was selected to permit future expansion and upgrades.

When thinking about USB devices, it was also decided to expose one of the USB Type C connectors from each of the two computers inside Mk. 0 as a bulkhead connector. This would allow for faster data transfers than would be possible over the existing gigabit Ethernet interface. Additionally, the exposed connectors made it possible to use any commodity USB Type C hub to break out display, keyboard, and mouse interfaces for easier debugging than would be possible via a headless connection.

2.2.12 Power Hardware Selection

The selected components for the Mk. 0 payload have a variety of different input voltage ranges and power requirements. For example, the NUC is rated for supply voltages between 12V and 19V, though was shown in prior experience to be unstable at voltages below 13V. The LEDs could be given no more than 12V without exceeding the maximum current specifications. It was determined that all components in the payload could be powered with one of two supply voltages - 12V and 18V. Unfortunately, neither of these voltages are available on the ground robot platforms. Even if they were, placing such strict requirements on the available voltages reduces the modularity of the payload, which is in direct conflict with one of the major design

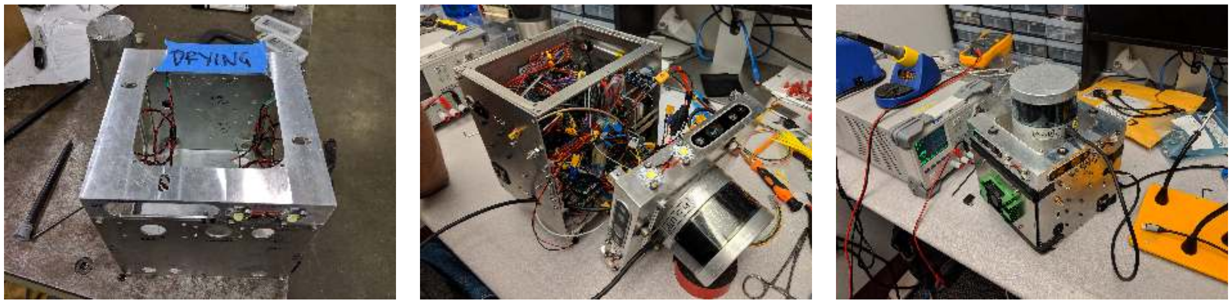

(a) Aluminum body panels (b) Initial wiring structure (c) Preliminary assembly

Figure 2.10: The figure shows various steps from the initial assembly of Mk. 0. The body panels were assembled and adjusted as necessary to ensure a close fit for environmental robustness (HWR3). The electronics were then attached and wired, with enough slack left in the wires to ensure components could be pulled apart for servicing. The initial assembly revealed that everything fit together nicely, but was too large for the drone.

goals. It was instead decided to design the payload to accept a nominal 24V supply, which is available on all three platforms, and internally regulate it down to the desired voltages of 12V and 18V. The ground robots offer a regulated 24V line, while the raw battery voltage from the 6S LiPo (22.2V - 25.2V) can be used on the drone platform.

2.3 Mk. 0 Assembly

The enclosure for Mk. 0 was designed in parallel with the component selection process. Heat dissipation was an important design consideration, as the payload was expected to draw more than 250W at full load. This led to the entire body of the payload being made of aluminum, a metal with relatively high thermal conductivity. Many of the components were then attached directly to the body with thermal paste to ensure optimal heat dissipation. The body itself is a cube constructed of three major pieces - a top plate, a bottom plate, and a single unit comprising all of the side panels. This construction allows for rapid entry into the payload by just removing the top plate. The panels were designed to be simple enough to manufacture in-house or outsourced to a commercial waterjet facility with rapid turnaround. Some pictures from the initial assembly steps are shown in Figure 2.10. After assembling Mk. 0 for the first time, it became apparent that it is too large, at approximately 8" in each dimension without the LIDAR, and too heavy, at nearly 20 lbs, to fly on the drone. A separate, simplified version of Mk. 0 would be needed for the drone.

2.4 Drone Payload Development

The primary constraint for any payload flying on our aerial platforms is weight. Weight is inversely proportional to flight time and dynamics – a lower weight means improved flight time and better handling under turbulent conditions due to higher thrust to weight ratios. After the first assembly of Mk. 0, it was discovered that the approximately 20 lbs weight of the payload would be too high to fly on D1. A separate, slimmed down version of the payload would be necessary. Reusing existing components on D1, even at the expense of modularity, would be crucial to reducing weight.

Prior to the addition of a payload, D1 already contained sensors and compute necessary for LOAM – an Intel NUC NUC8i7BEH, a Velodyne VLP 16 LIDAR, and an Xsens IMU. No additional object detection sensors were present. From Table 2.1, it was clear that the highest utility single sensor would be an RGB camera. The same RGB camera as Mk. 0 was selected for consistency, along with the same lighting solution. Payload weight restrictions prevented adding 4 RGB cameras as was done in Mk. 0. Similarly, weight restrictions prevented adding a separate Nvidia Xavier module to perform object detection, meaning that all object detection code would need to run alongside all other code on the NUC, with adjustments in performance made as necessary to accommodate the resource constraints. The final hardware configuration is pictured in Figure 2.2c.

2.5 Mk. 1 Development

Mk. 0 and the drone payload were initially assembled in early December 2018, in time for the qualification deadline for the DARPA-hosted SubT Integration Exercise (STIX) event to be held in April 2019. After this deadline, DARPA announced the official list of artifacts which would be used in the Tunnel Circuit (see Figure 1.2) and indicated that the same set of artifacts would be used at STIX. While no modifications were made to the Mk. 0 or Drone Payloads as a result of this new information, this information, along with the experience using Mk. 0 and the Drone Payloads through the STIX event revealed a number of potential areas of improvement for the next ground robot payload, Mk. 1. The improvements made to Mk. 1, as compared to Mk. 0, are outlined in the remainder of this section.

2.5.1 RGB Camera Improvements

The experience at STIX indicated that the most useful category of sensor for detecting the Tunnel Circuit artifacts was RGB cameras. The backpack, drill, fire extinguisher, and survivor artifacts could be easily detected in the RGB camera streams, with cell phones being somewhat more difficult but theoretically possible as well. Thus, the first place attention was directed was improving the quality and usability of the data coming from the RealSense D435 cameras. It was decided early in the development cycle that a new RGB camera would not be used on Mk. 1 due to the rapid timeline requirements (less than 3 months of available development time), as well as to minimize component level differences between the payloads to reduce software complexity.

Framerate and Resolution

The most important issue with the RGB cameras on Mk. 0 is the limited framerate and resolution available. Each RealSense D435 module on Mk. 0 streams both RGB (encoded as YUYV) and depth (Z16) at 15 frames per second at a resolution of 640 x 360. The relatively low camera framerate, which allows for long exposure times when using autoexposure, combined with the rolling shutter of the RGB camera frequently causes images to be more blurry than desired. Additionally, the selected resolution is suboptimal for depth accuracy – a resolution of 848 x 480 is recommended [7].

The primary restriction on increasing the framerate or the resolution of the RGB images is the available USB bandwidth. In Mk. 0, all 4 RealSense modules are connected to the Xavier via a single USB 3.0 hub (5 Gbps), a HB31C4AB hub from StarTech.com. A variety of framerates and resolutions were tested using the RealSense Viewer application under this configuration prior to the STIX event. Though configurations at both 640 x 360 at 30 fps per camera (885 Mbps) and 848 x 480 at 15 fps per camera (782 Mbps) both appeared mostly stable, frames would be reported as dropped infrequently. The highest framerate and resolution configuration under which no frame drops were observed is 15 fps at 640 x 360 per camera, and thus this configuration is used on Mk. 0. The results reported by Intel [6] indicate that these configurations should both be well under the USB bandwidth after which frame drops occur. The discrepancy may be due to different cabling, a different USB hub, or a different scheduling algorithm in the USB host controller used on the Xavier.

The solution for Mk. 1 was to split the RealSense depth modules over two separate USB ports, each of which has its own host controller on the Xavier. This increases the maximum theoretical bandwidth available from 5 Gbps to 10 Gbps, as well as decreases the number of devices per bus to 2, decreasing bus contention for each host controller and making the scheduling problem slightly easier. This solution was implemented by using a second HB31C4AB hub attached to the second USB Type C port on the developer kit carrier board. Under this new configuration, 30 fps at 848 x 480 resolution (1563 Mbps) was achieved on both the depth and RGB streams for all 4 RealSense modules, as desired. However, it was discovered that the increased data rate filled up the onboard storage too quickly, and thus an identical configuration to Mk. 0 is used on Mk. 1 until more onboard storage can be added.

Mounting

The RealSense cameras in Mk. 0 are mounted to the payload using the single $\frac{1}{4}$-20 threaded hole on the bottom, to which a bolt is attached through a slot in the top of the payload. This slot allows for some translational motion of the cameras, while the single attachment point allows the cameras to rotate. Both problems were addressed to some extent with locking washers, but they did not completely eliminate the problems. As the cameras deviate from their calibrated positions, accuracy of downstream algorithms suffers. Thus, the RealSense modules in Mk. 1 are mounted using the 2 mounting points on the back of the RealSense to multiple attachment points on the payload, eliminating the possibility of any motion.

Orientation

An additional consequence of the mounting mechanism for the RealSense cameras on Mk. 0 is that all images appear to be upside down. This is compensated for in the software running on Mk. 0, but has a slightly computational cost and requires additional bookkeeping. To eliminate

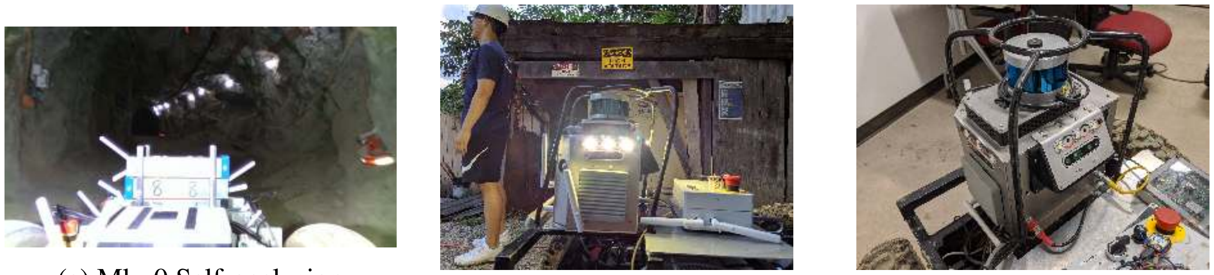

(a) Mk. 0 Self-occlusion (b) Mk. 1 back camera tilt (c) Mk. 1 back camera tilt

Figure 2.11: The back RealSense module on Mk. 1 is tilted to avoid the self occlusion problems of Mk. 0. The tilt also allows for previously unseen views to be captured by the rear camera, making it possible to detect artifacts on or near the ceiling.

the additional CPU burden, the mounting mechanism on Mk. 1 ensures that the cameras are oriented normally.

Occlusion

The back RealSense camera module on Mk. 0 has its field of view partially occluded by the communication node droppers on both R1 and R2. If the possibility of placing an aerial vehicle such as D1 on the back of R1 or R2 were explored, the back camera's field of view would be further obstructed. For Mk. 1, this problem is solved by orienting the back camera upwards slightly so that it no longer sees any part of the ground robot. Having a camera oriented slightly upwards also allows the Mk. 1 payload to detect artifacts which may be on high shelves which were previously out of the field of view of the other cameras, at the cost of some redundancy in field of view. An example of the occlusion on Mk. 0 and tilted orientation on Mk. 1 is shown in Figure 2.11.

2.5.2 Thermal Camera Improvements

The thermal camera on Mk. 0 offers a relatively low native resolution and framerate of 320 x 256 and 8.5 Hz respectively. The low framerate is due to a limitation of the specific camera model selected – a higher framerate version is not readily available. Additionally, with only a single thermal camera, the overall field of view is relatively limited. Artifacts to either side of the robot are often missed, or only captured for a single frame or two by the single front-facing thermal camera on Mk. 0. Finally, there is a consistent dark spot in the center of the thermal images captured from Mk. 0, as well as on those captured from spares, shown in Figure 2.12. Conversations with FLIR indicated that this may be caused by the camera self-heating over time, as well as perhaps a bad lens calibration, bad sensor, or bad flat field correction, though no specific cause was determined.

To resolve all of these issues for Mk. 1, 2x FLIR Boson 640, a higher end sibling of the FLIR Boson 320 used on Mk. 0, were selected. This new thermal camera offers framerates of 30 and 60 fps at a native resolution of 640 x 512, and uses a different lens assembly which offers a 95 degree HFOV. The dark center spot observed in Mk. 0's thermal camera is not observed in the

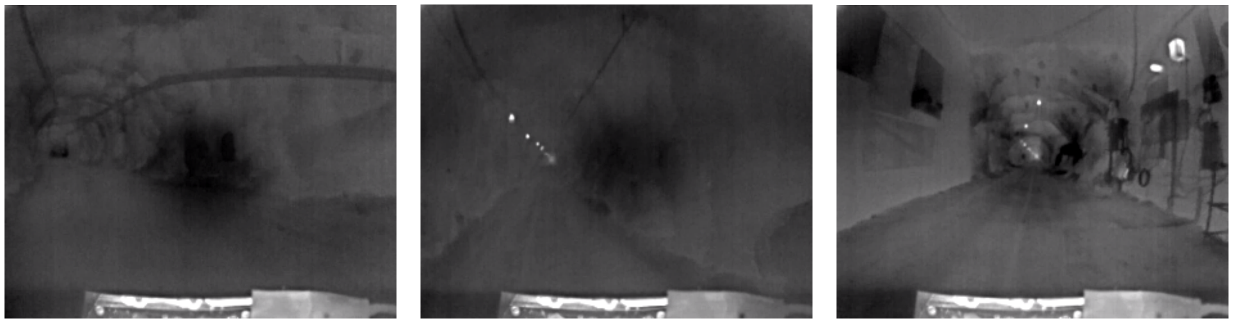


Figure 2.12: The center of each of the 3 images captured from Mk. 0's thermal camera shows an unexpected dark spot. This dark spot is present on images captured from the spares of the same camera model as well, but is not present on the new thermal cameras selected for use in Mk. 1.

Mk. 1 thermal cameras. The two cameras are positioned at the front diagonal corners of Mk. 1, covering the front 185 degrees of field of view.

Unfortunately, the interface for the new thermal cameras is USB 2.0, the same as the interface for the thermal cameras in Mk. 0. While the maximum available bandwidth over USB 2.0 is sufficient to stream 1 of the FLIR Boson 640s at 60 fps (YV12 encoding, 236 Mbps), we were unable to achieve sustained 60 fps streaming on two cameras on the the same USB port. This would require 472 Mbps of throughput over USB 2.0 which, while under the theoretical maximum bitrate, is higher than any practical implementation achieves due to transmission overhead and scheduling inefficiencies. Luckily, the changes to the RGB cameras for Mk. 1 included the addition of a second USB hub, which allows the 2 new thermal cameras to operate on separate USB 2.0 ports. The maximum framerates observed on the RealSenses on Mk. 1 were not affected as a result of this change.

2.5.3 Illumination Improvements

A few problems with the illumination system were observed when working with Mk. 0:

Flickering The Mk. 0 LEDs constantly flicker. This flickering is fast enough that it is not visible in the camera images, but is sufficiently distracting for humans working with and around the robot that it needed to be fixed for Mk. 1.

Switch control The Mk. 0 LEDs are controlled by a physical switch on the payload. Part of the operational procedure for deployments is to ensure that the LEDs have been switched on. However, this step occasionally gets missed, and has resulted in deployments (including once at STIX) without the LEDs activated, limiting the effectiveness of the RGB cameras. For Mk. 1, the LEDs needed to be controlled automatically.

Fixed brightness A byproduct of powering all of the LEDs on Mk. 0 with a single 12V regulator is that the brightness cannot be adjusted (beyond the unintended flickering). While brightness control is not strictly necessary for basic operation, it could serve as a way to manage power consumption and heat production and was a requested feature for Mk. 1 as well.

The first attempt at solving all 3 of these problems was to use a custom PCB to drive the LEDs. Each PCB drives LEDs using the same voltage regulation strategy as in Mk. 0, though with a higher quality regulator intended to reduce flicker. The PCBs also contain a programmable microcontroller which allows another computer, such as the Xavier, to control the brightness of the LEDs by issuing commands over I2C. The PCB has an input for the depth trigger produced by the RealSense depth modules (see Figure 2.8) which can be used to automatically turn on the LEDs, as the trigger is only produced when the camera is broadcasting images. Early results indicated that the PCBs would be able to solve all of the problems with illumination from Mk. 0. The PCB is shown in Figure 2.13.

Unfortunately, while the PCBs were being developed, it was decided that a third LED should be used per side to provide additional illumination, to compensate for the reduced exposure time at the new maximum framerate of 30 fps for RGB images. The current requirements for the third LED proved to be too high for the PCB, which was only designed with the current of 2 LEDs in mind. By the time this issue was discovered, it was too late to upgrade the voltage regulator and fabricate new PCBs. While some of the manufactured PCBs were able to handle the current for 3 LEDs, not all boards were able to do so reliably, which meant that the boards could not be used in a field deployment. The fallback plan, which was implemented for Mk. 1, was to use the BuckBlock A009 LED drivers which had been used for prototyping and the earlier LED flashing experiments.

The BuckBlock drivers solve the flicker problem by providing a constant current source rather than a constant voltage source. They do not offer a programmatic way to control LED brightness, but do support the use of a potentiometer to control brightness manually. The BuckBlocks used in Mk. 1 are also wired directly into power, rather than through a switch, and turn the LEDs on whenever power is applied to the payload. This is a less optimal solution than turning on the LEDs automatically with the image streaming as the PCBs would have been able to do, but it solves the primary problem of forgetting to turn the lights on. We intend to continue development on the custom PCB for use in future iterations of the payload.

2.5.4 Miscellaneous Improvements

A few small but useful improvements were made to the miscellaneous components used on Mk. 0 to improve reliability, robustness, or simplicity as a direct result of problems encountered when working on Mk. 0.

Computer Power Buttons

The computer power buttons used on Mk. 0 are relatively unreliable and do not indicate status of the computers. The power button for the NUC, for example, works very infrequently, and thus it has become standard (and inconvenient) operating procedure to use a small stick to poke the power button directly on the NUC. For Mk. 1, the power buttons have been replaced with rugged and illuminated metal power buttons which indicate computer power status. A blue LED is used for the NUC while a green LED is used for the Xavier.

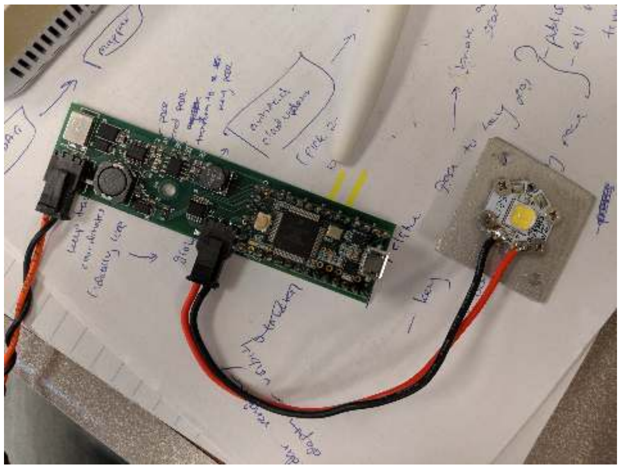


Figure 2.13: This is the custom PCB developed for LED control on Mk. 1. The PCB is capable of brightness control over I2C and Serial for up to 2 LEDs, and can automatically turn them on with a trigger input from the RealSense camera modules. Unfortunately, the current requirements for 3 LEDs proved to be too great for the regulators on this PCB to handle, and there was not enough time left in the development cycle to fabricate new PCBs.

Ethernet Switch

As mentioned previously, the original GigEthos switch in Mk. 0 failed randomly during field testing and was replaced with a commodity TP-Link gigabit Ethernet switch. For Mk. 1, the switch was further upgraded to a robust industrial grade gigabit network switch from Antaira with roughly the same footprint as the commodity one. The Antaira switch comes with screw terminals which make applying power from a regulator significantly easier than the commodity switch's barrel jack.

Ethernet Connector

The RJ-45 cable bulkhead used on Mk. 0 does not lock Ethernet cables very tightly, potentially allowing them to disconnect under strong vibration. Though this failure mode has not yet been experienced during field testing, it is likely to happen as the connector ages. For Mk. 1, an 8 pin connector from LEMO was used to create a custom connection for Ethernet. This solution has the added advantage of being significantly smaller than the RJ-45 bulkhead used on Mk. 0.

LIDAR Cable Connector

The cable for the LIDAR on Mk. 0 is passed through the case and wired directly into other components (e.g. power, network switch) without a connector. This makes servicing in the event of LIDAR failure (as in the case of the robot flipping over) difficult and time consuming. For Mk. 1, another locking connector from LEMO is used to allow easy LIDAR removal and replacement. Backup LIDARS have been prepared to allow replacement in a matter of minutes, rather than hours.

Power Distribution Board

Mk. 0 regulates 24V input down to 12V and 18V rails internally. Each regulator is a separate component from which wires are brought out to individual components. Though functional, this approach uses an abundance of wires which make any servicing inside Mk. 0 quite tedious. Mk. 1 instead uses a custom power distribution board with only a single 15V regulator, eliminating much of the wiring clutter and reducing component count. Use of the 15V regulator was now possible as the LED drivers could directly accept the input voltage to the payload and did not require a separate 12V rail.

2.5.5 New Components

Experience using Mk. 0, as well as the announcement of the actual Tunnel Circuit artifacts indicated that additional sensors and components could be useful for Mk. 1's functionality, as well as add some degree of future-proofing.

Xavier WiFi Scanning

WiFi scanning on Mk. 0 using the chip in the NUC takes an average of 3 seconds for a complete scan. At the ground robots' average speed of 2 m/s, this means the robots will typically move about 5 meters per scan. Assuming that the frequency of each individual scan cannot be improved due to library restrictions, the next best solution is to add a second chip capable of performing scanning. To this end, an Intel 8265 WiFi chip was added to the Xavier used in Mk. 1 to allow a second set of scans to run in parallel. Mk. 1 scans on both the NUC and Xavier simultaneously, and receives scan results at approximately 0.6 - 0.8 Hz.

Microphone Array

Though WiFi and Bluetooth scanning provide reliable presence detection for cell phones, accurate localization with the noisy signals produced from the scans is challenging. DARPA guarantees that the cell phones will be playing video with audio, meaning that audio can potentially be used as a secondary detection and localization mechanism. Mk. 1 includes a microphone array, the ReSpeaker Mic Array v2.0 from SeeedStudio, which provides 4 raw audio channels over USB. The audio streams can be fed into an algorithm such as ODAS [20] to produce 3D direction of arrival estimates, which can be used for localization. This microphone array replaces the single microphone used in Mk. 0.

Speaker

The Mk. 0 payload has no way of communicating with or offering feedback to robot operators. Audio was selected as a viable feedback mechanism for Mk. 1 as it does not require line of sight as a screen would. The Large Surface Transducer from Adafruit was selected, along with an amplifier chip, to provide audio feedback. A transducer, pressed directly against the payload wall, is used instead of a traditional speaker in order to provide higher environmental robustness in the payload.

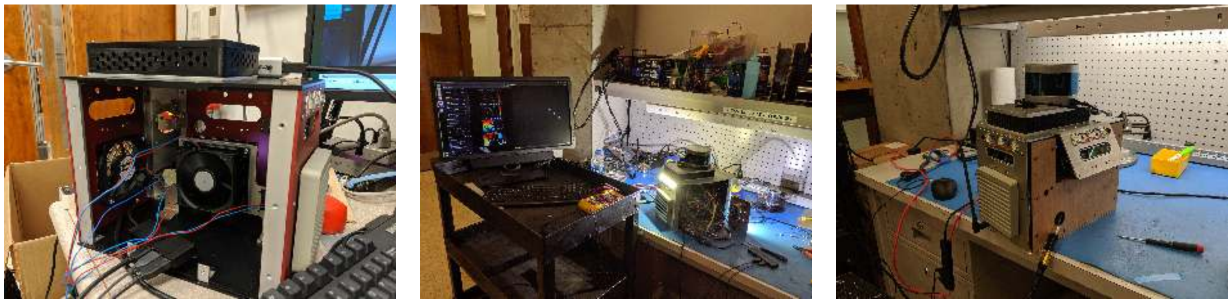

(a) Xavier in interior panel (b) Initial test of LEDs (c) First button-up

Figure 2.14: The panel design used in Mk. 1 makes manufacturing and servicing significantly easier than Mk. 0. Careful consideration was also given to the assembly process, which is notably less painstaking than Mk. 0.

2.6 Mk. 1 Assembly

The assembly for Mk. 1 draws inspiration from Mk. 0, but differs in a few key areas to improve serviceability and reduce manufacturing time. Mk. 1 maintains the same multi-panel design as Mk. 0, but makes it possible to remove each of the 6 main panels from the cube, rather than gluing the 4 center panels together. This allows for a significant improvement in serviceability over Mk. 0. The computers have been moved from the sides of the payload to sheets in the center, with fans added to provide airflow and additional cooling. The fans are guarded by air filters and baffles to prevent dust and debris from entering the payload. The goal of flying Mk. 0 on a drone was entirely abandoned for Mk. 1, allowing thicker sheet metal to be used everywhere, improving the rigidity of the entire structure. Some pictures from the assembly of Mk. 1 are given in Figure 2.14.

2.7 Summary of Payloads

This chapter describes the development of 3 distinct sensing and compute payloads. Each payload is capable of performing state estimation, path planning and navigation, and artifact detection and localization. A summary of the relevant components and capabilities of each payload is provided below. The Drone Payload is a simplified version of Mk. 0 meant for use on aerial platforms, while Mk. 1 is an incremental improvement on Mk. 0 using lessons learned over 6 months of development and use. See Figure 2.2 for pictures of each payload.

Drone Payload

- **Compute**: Intel NUC8i7BEH
- **State Estimation**: Velodyne VLP-16 Puck LIDAR
- **State Estimation**: XSens IMU
- **RGB Camera**: RealSense D435 (front-facing)
- **Illumination**: 1x Opulent XHP70A 5000K LED

Mk. 0 Payload

- **Compute**: Intel NUC8i7BEH
- **Compute**: Nvidia Xavier with Developer Kit Carrier
- **State Estimation**: Velodyne VLP-16 Puck LIDAR
- **State Estimation**: XSens IMU
- **RGB Cameras**: 4x RealSense D435, up to 15 fps at 640 x 360 per camera
- **Thermal Camera**: FLIR Boson 320 w/ 92 degree HFOV, 320 x 256 at 8.5 fps
- **Audio Input**: 3.5mm audio input microphone
- **WiFi Scanning**: NUC builtin 9560 module

Mk. 1 Payload

- **Compute**: Intel NUC8i7BEH
- **Compute**: Nvidia Xavier with Developer Kit Carrier
- **State Estimation**: Velodyne VLP-16 Puck LIDAR
- **State Estimation**: XSens IMU
- **RGB Cameras**: 4x RealSense D435, up to 30 fps at 848 x 480 per camera
- **Thermal Camera**: 2x FLIR Boson 320 w/ 95 degree HFOV, 640 x 512 at 60 fps
- **Audio Input**: ReSpeaker Microphone Array V2.0
- **Audio Output**: Large Surface Transducer from Adafruit
- **WiFi Scanning**: NUC builtin 9560 module, Intel 8265 module in Xavier

Chapter 3

Artifact Detection and Localization

The artifact detection and localization pipeline is responsible for converting the sensor data from the various payloads (Mk. 0, Mk. 1, drone), into a list of artifacts to send to the base station and ultimately report to DARPA. The pipeline was developed to meet a set of requirements, which were derived from the competition rules and our team's concept of operations, shown in Table 3.1. Certain assumptions, given in Table 3.2 were made to constrain the scope of the problem and guide parameter tuning wherever necessary.

An overview of the complete pipeline is given in Figure 3.1. This pipeline runs identically on all 3 payloads with only minor configuration changes, and sensor omissions where necessary (e.g. drone payload does not contain a thermal camera). All robots report artifacts to the GUI independently, and no information is shared between pipelines running on individual payloads to prevent overloading the communication network.

The pipeline consists of 2 major modules - the Signal Localizer and the Object Detection Localizer. Each module takes in various sensor data and produces Artifact Localizations, which are 3D coordinates in the robot's map frame that are believed to correspond to a desired artifact. These Artifact Localizations may contain additional evidence to be displayed to the human supervisor, such as images or point clouds of the artifact and surrounding environment. The complete specification for an Artifact Localization message is given in Table 3.3. Artifact Localizations from both modules are combined inside the Artifact Aggregator and then transmitted to the base station to be displayed on the GUI. The human supervisor inspects artifacts displayed on the GUI and reports valid ones to DARPA.

Reported coordinate of artifact must be within 5m of DARPA-surveyed coordinate
Pipeline must run on-board, either on the Xavier (Mk. 0, Mk. 1), or on part of the NUC (drone)
Artifacts must be transmitted to base station reliably over a lossy wireless link
Pipeline should be capable of detecting all 5 types of artifacts (as shown in Figure 1.2)
All artifacts which the robots pass by (within 5m) should be detected
Pipeline should be identical or nearly identical on all payloads
Artifacts should be detected and reported to human supervisor in real-time

Table 3.1: Software pipeline requirements

State estimation system on all payloads would be LOAM [45]
Artifact detection and localization pipeline results would not feed the robots' waypoint planner
Artifacts are reported in robots' own frames and transformed to a single world frame at base station
Robots will move at approximately 2 m/s
A human supervisor would be available to verify artifact reports, and thus false positives are acceptable

Table 3.2: Software pipeline assumptions

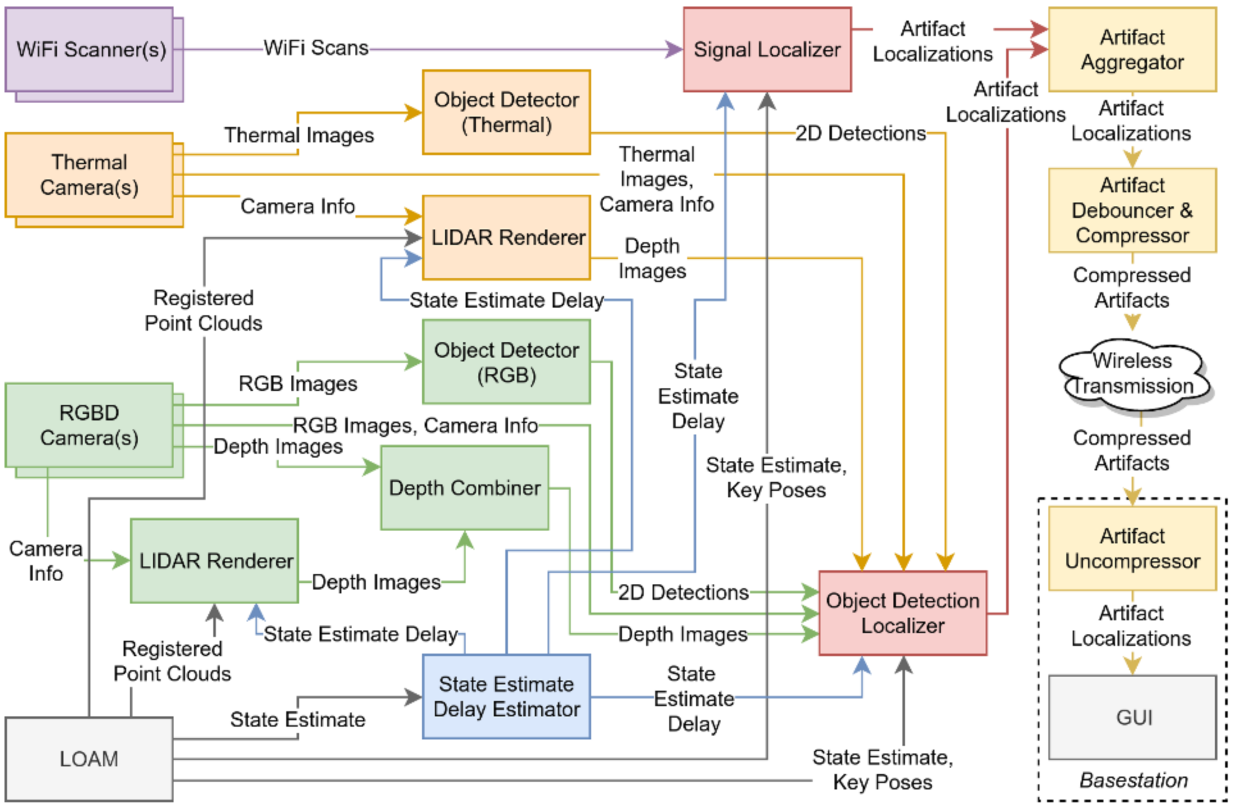


Figure 3.1: Artifact detection and localization software diagram

Datatype	Field Name	Description
Time	stamp	When the artifact was first discovered
bool	valid	Whether the artifact is valid (used for invalidations)
float	x	Robot /map frame x coordinate of localization point
float	y	Robot /map frame y coordinate of localization point
float	z	Robot /map frame z coordinate of localization point
uint	class_id	Machine readable class id

Table 3.3: Composition of an Artifact Localization message

3.1 LOAM Overview

One of the assumptions made during the initial design phases of the software pipeline was that LOAM [45] would be the state estimation system used on all of our payloads. This simplified the development of the artifact detection and localization pipeline as we only needed to develop and test against a single state estimation system. The relevant interface details of LOAM (in the form of ROS [37] frames and topics) are given below:

/sensor This is the robot's local frame, and is coincident with the Velodyne LIDAR's frame.

/sensor_init The fixed frame used as the base for LOAM's odometry.

/map The world frame, which is initially coincident with sensor_init but can change after loop closures.

/key_pose_to_map LOAM creates a series of key poses as the robot traverses the environment. These key poses are generated approximately every 2 meters of the robot's path. Each key pose is given a unique ID, starting from 0. The key pose is published relative to the /map frame.

/key_pose_path When LOAM detects a loop closure, it corrects the key poses and publishes a new list of key pose IDs and poses.

/velodyne_cloud_registered The laser scan from the Velodyne LIDAR is aligned to previous scans and published on this topic at 5 Hz. Registered scans are published on this topic even when the robot is stationary. The scans are registered in the /sensor_init frame and accumulate drift over time, but are locally smooth.

/integrated_to_map The 6DOF pose of the /sensor frame is published on this topic at 200 Hz. The pose is corrected by loop closures and thus does not accumulate significant drift, but may be discontinuous locally.

3.2 Object Detector (RGB)

A 2D convolutional neural network based object detector runs on each of the RGB image stream to detect all artifact types except cell phones. To select the network, a variety of network architectures were benchmarked, identifying networks which would be capable of running at camera framerate on the NUC (for 1 RGB image stream) and on the Xavier (for 4 RGB image streams). Pretrained checkpoints for the fastest network types were finetuned using manually collected and annotated data. The best performing networks were selected for use in the pipeline.

The TensorFlow Object Detection API [23] was used for all network training and inference tasks. The API provides checkpoints for a large variety of popular network architectures, and has support from both Nvidia and Intel for optimizing networks trained with the API to run on their respective hardware platforms. The YOLO [39] family of networks was not considered for use in the Artifact Detection and Localization pipeline due to the lack of support for it in the API.

3.2.1 Network Benchmarking and Selection

The TensorFlow Object Detection API was used to evaluate a large number of network architectures and configurations for inference speed on the Nvidia Xavier and Intel NUC. All models available in the model zoo were downloaded, optimized for each hardware platform, and benchmarked for average inference speed.

Xavier Benchmarks

Nvidia provides the TensorRT runtime to accelerate neural network inference on its platforms. Using the TensorRT runtime directly is typically a difficult and involved process. TF-TRT bindings are provided within TensorFlow to simplify the use of TensorRT for certain types of models trained with TensorFlow. After converting a network from TensorFlow to TF-TRT, supported subgraphs of the network will be converted into optimized TensorRT engines, while unsupported operations will execute natively in TensorFlow. Support is provided by Nvidia for models trained with the TensorFlow Object Detection API to ensure many model types can be converted using TF-TRT.

During conversion, all networks were optimized to use 16 bit floating precision, which typically achieves comparable network precision and accuracy as 32 bit precision while allowing for higher inference throughput. Batch sizes of 1 and 4 were used, representing configurations where images from each camera are processed sequentially, or a batch of images (one from each camera) is processed at once. A minimum segment size of 50 was used. Some networks failed to convert to TF-TRT with these parameters and were ignored. Before benchmarking, all CPU and GPU cores on the Xavier were enabled (MAX-N mode) and had their frequency maximized. The fan was set to run at maximum speed to prevent overheating during test runs.

When benchmarking a network, a set of images (equivalent to the network's batch size) was first run through it to allow TensorFlow to optimize runtime parameters. The same set of images were then run through the network 100 times in a loop, and the inference times for each loop iteration were recorded. No limits were placed on the number of cores, percent of GPU, or percent of system memory available to the neural network. The average of the loop iteration times are presented in Table 3.4 for each network architecture which was successfully converted with TF-TRT. Table 3.5 shows the average loop iteration time for all unoptimized networks as a baseline, including networks which were unable to be converted.

The values reported in Table 3.4 were gathered after optimizing and benchmarking each network once. Repeated benchmarking attempts resulted in very similar average frames per second values. However, repeatedly rerunning the optimization process with TF-TRT and then benchmarking each network resulted in some deviation from the observed average frames per second values, up to approximately 25% from those reported in Table 3.4. Thus, the average frames per second values reported in Table 3.4 are not necessarily the maximum possible framerates under the optimization parameters, but we believe they are sufficiently close to inform network selection.

Model Checkpoint Name	Batch Size	Average Batches Per Second	Average Frames Per Second
ssd_mobilenet_v1_coco_2018_01_28	4	20.408	81.633
ssdlite_mobilenet_v2_coco_2018_05_09	4	18.519	74.074
ssd_inception_v2_coco_2018_01_28	4	18.182	72.727
ssd_mobilenet_v1_0.75_depth_300x300_coco14_sync_2018_07_03	4	15.625	62.500
ssd_mobilenet_v1_0.75_depth_quantized_300x300_coco14_sync_2018_07_18	4	14.925	59.701
ssd_mobilenet_v1_quantized_300x300_coco14_sync_2018_07_18	4	12.658	50.633
ssd_mobilenet_v1_coco_2018_01_28	1	47.619	47.619
ssd_mobilenet_v2_quantized_300x300_coco_2018_09_14	4	11.364	45.455
ssdlite_mobilenet_v2_coco_2018_05_09	1	45.455	45.455
ssd_inception_v2_coco_2018_01_28	1	40.000	40.000
ssd_mobilenet_v1_0.75_depth_300x300_coco14_sync_2018_07_03	1	35.714	35.714
ssd_mobilenet_v1_0.75_depth_quantized_300x300_coco14_sync_2018_07_18	1	31.250	31.250
ssd_mobilenet_v1_quantized_300x300_coco14_sync_2018_07_18	1	28.571	28.571
ssd_mobilenet_v2_quantized_300x300_coco_2018_09_14	1	25.000	25.000
ssd_mobilenet_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	4	4.065	16.260
ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	4	3.704	14.815
ssd_mobilenet_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	13.158	13.158
ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	11.905	11.905
faster_rcnn_inception_v2_coco_2018_01_28	1	8.621	8.621
faster_rcnn_resnet50_lowproposals_coco_2018_01_28	1	6.803	6.803
mask_rcnn_inception_v2_coco_2018_01_28	1	6.711	6.711
faster_rcnn_resnet50_coco_2018_01_28	1	4.098	4.098
rfcn_resnet101_coco_2018_01_28	1	3.448	3.448
mask_rcnn_resnet50_atrous_coco_2018_01_28	1	1.835	1.835

Table 3.4: Optimized TF-TRT network inference benchmarks on Xavier

NUC Benchmarks

Intel provides the OpenVINO framework which serves a similar purpose as TensorRT on Nvidia's platforms. While no dedicated integration with TensorFlow is available for OpenVINO, specific examples are provided by Intel on optimizing networks trained with the TensorFlow Object Detection API with the OpenVINO framework. Using the available examples, most models based on SSD, FasterRCNN, and MaskRCNN were able to be converted into the intermediate representation used by OpenVINO. 32 bit floating point weights were used.

The models were benchmarked using a nearly identical process to the Xavier benchmarking. OpenVINO was only allowed to use one CPU core during benchmarking, representing a realistic upper bound on the compute available for network inference when other processes are running on the NUC. CPU frequencies were governed automatically. GPU support was not enabled for OpenVINO. Benchmarking results from successfully converted models are given in Table 3.6.

Network Selection

The primary criteria for selecting a network architecture was the throughput. The relatively low framerate of the RGB cameras (a maximum of 15 Hz on Mk. 0) often causes motion blur, which negatively impacts detection performance. By ensuring that the selected network would be able to run on each frame, we would increase the probability of detecting objects. The SSD MobileNet v1 architecture with COCO 2018/01/28 checkpoint was selected as the base for all networks to run on the Xavier and NUC due to it offering the highest throughput on the Xavier (with a batch size of 4) and offering sufficiently fast throughput (>15 Hz) on the NUC.

Model Checkpoint Name	**Batch Size**	**Average Batches Per Second**	**Average Frames Per Second**
ssd_mobilenet_v1_coco_2018_01_28	4	2.463	9.852
ssdlite_mobilenet_v2_coco_2018_05_09	4	2.445	9.780
ssd_mobilenet_v1_0.75_depth_300x300_coco14_sync_2018_07_03	4	2.364	9.456
ssd_mobilenet_v1_ppn_shared_box_predictor_300x300_coco14_sync_2018_07_03	4	2.347	9.390
ssd_mobilenet_v2_coco_2018_03_29	4	2.336	9.346
ssd_inception_v2_coco_2018_01_28	4	2.252	9.009
ssd_mobilenet_v1_coco_2018_01_28	1	8.065	8.065
ssdlite_mobilenet_v2_coco_2018_05_09	1	7.874	7.874
ssd_mobilenet_v1_ppn_shared_box_predictor_300x300_coco14_sync_2018_07_03	1	7.813	7.813
ssd_mobilenet_v1_0.75_depth_300x300_coco14_sync_2018_07_03	1	7.634	7.634
ssd_mobilenet_v2_coco_2018_03_29	1	7.634	7.634
ssd_inception_v2_coco_2018_01_28	1	7.042	7.042
faster_rcnn_inception_v2_coco_2018_01_28	4	1.377	5.510
ssd_mobilenet_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	4	1.309	5.236
faster_rcnn_resnet50_lowproposals_coco_2018_01_28	4	1.190	4.762
faster_rcnn_inception_v2_coco_2018_01_28	1	4.739	4.739
ssd_mobilenet_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	4.566	4.566
mask_rcnn_inception_v2_coco_2018_01_28	4	1.099	4.396
ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	4	1.056	4.224
faster_rcnn_resnet50_lowproposals_coco_2018_01_28	1	4.132	4.132
mask_rcnn_inception_v2_coco_2018_01_28	1	4.032	4.032
ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	3.663	3.663
faster_rcnn_resnet101_lowproposals_coco_2018_01_28	4	0.895	3.581
faster_rcnn_resnet50_coco_2018_01_28	4	0.842	3.370
faster_rcnn_resnet101_kitti_2018_01_28	4	0.833	3.333
faster_rcnn_resnet101_lowproposals_coco_2018_01_28	1	3.125	3.125
faster_rcnn_resnet101_kitti_2018_01_28	1	2.994	2.994
faster_rcnn_resnet50_coco_2018_01_28	1	2.985	2.985
rfcn_resnet101_coco_2018_01_28	4	0.739	2.956
faster_rcnn_resnet101_ava_v2.1_2018_04_30	4	0.709	2.837
faster_rcnn_resnet101_coco_2018_01_28	4	0.687	2.747
rfcn_resnet101_coco_2018_01_28	1	2.604	2.604
faster_rcnn_resnet101_ava_v2.1_2018_04_30	1	2.500	2.500
faster_rcnn_resnet101_coco_2018_01_28	1	2.445	2.445
mask_rcnn_resnet50_atrous_coco_2018_01_28	4	0.404	1.614
mask_rcnn_resnet50_atrous_coco_2018_01_28	1	1.534	1.534
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco_2018_01_28	4	0.269	1.074
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco_2018_01_28	1	1.004	1.004
mask_rcnn_resnet101_atrous_coco_2018_01_28	4	0.217	0.869
mask_rcnn_resnet101_atrous_coco_2018_01_28	1	0.830	0.830
faster_rcnn_nas_lowproposals_coco_2018_01_28	1	0.808	0.808
faster_rcnn_inception_resnet_v2_atrous_lowproposals_oid_2018_01_28	4	0.177	0.709
faster_rcnn_inception_resnet_v2_atrous_lowproposals_oid_2018_01_28	1	0.676	0.676
mask_rcnn_inception_resnet_v2_atrous_coco_2018_01_28	4	0.132	0.529
faster_rcnn_resnet50_fgvc_2018_07_19	4	0.132	0.528
faster_rcnn_resnet101_fgvc_2018_07_19	4	0.129	0.517
mask_rcnn_inception_resnet_v2_atrous_coco_2018_01_28	1	0.496	0.496
faster_rcnn_inception_resnet_v2_atrous_coco_2018_01_28	1	0.471	0.471
faster_rcnn_resnet50_fgvc_2018_07_19	1	0.446	0.446
faster_rcnn_resnet101_fgvc_2018_07_19	1	0.446	0.446
faster_rcnn_inception_resnet_v2_atrous_oid_2018_01_28	1	0.378	0.378

Table 3.5: Reference network inference benchmarks on Xavier

Model Checkpoint Name	Batch Size	Average Frames Per Second
ssd_mobilenet_v1_0.75_depth_300x300_coco14_sync_2018_07_03	1	62.500
ssdlite_mobilenet_v2_coco_2018_05_09	1	58.824
ssd_mobilenet_v1_coco_2018_01_28	1	41.667
ssd_mobilenet_v1_ppn_shared_box_predictor_300x300_coco14_sync_2018_07_03	1	40.000
ssd_mobilenet_v2_coco_2018_03_29	1	25.641
ssd_inception_v2_coco_2018_01_28	1	11.494
faster_rcnn_inception_v2_coco_2018_01_28	1	2.066
faster_rcnn_resnet50_lowproposals_coco_2018_01_28	1	1.393
ssd_mobilenet_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	0.944
faster_rcnn_resnet101_lowproposals_coco_2018_01_28	1	0.855
ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03	1	0.674
mask_rcnn_inception_v2_coco_2018_01_28	1	0.552
faster_rcnn_resnet50_coco_2018_01_28	1	0.536
faster_rcnn_resnet101_coco_2018_01_28	1	0.446
mask_rcnn_resnet50_atrous_coco_2018_01_28	1	0.063

Table 3.6: OpenVINO optimized network benchmarks on NUC (1 core)

Model Checkpoint Name	Batch Size	Average Batches Per Second	Average Frames Per Second	Relative Speedup
ssd_mobilenet_v1_coco_2018_01_28	16	5.780	92.486	1.942
ssd_mobilenet_v1_coco_2018_01_28	8	10.989	87.912	1.846
ssd_mobilenet_v1_coco_2018_01_28	4	20.408	81.633	1.714
ssd_mobilenet_v1_coco_2018_01_28	1	47.619	47.619	1.000

Table 3.7: Inference throughput with varying batch size on Xavier

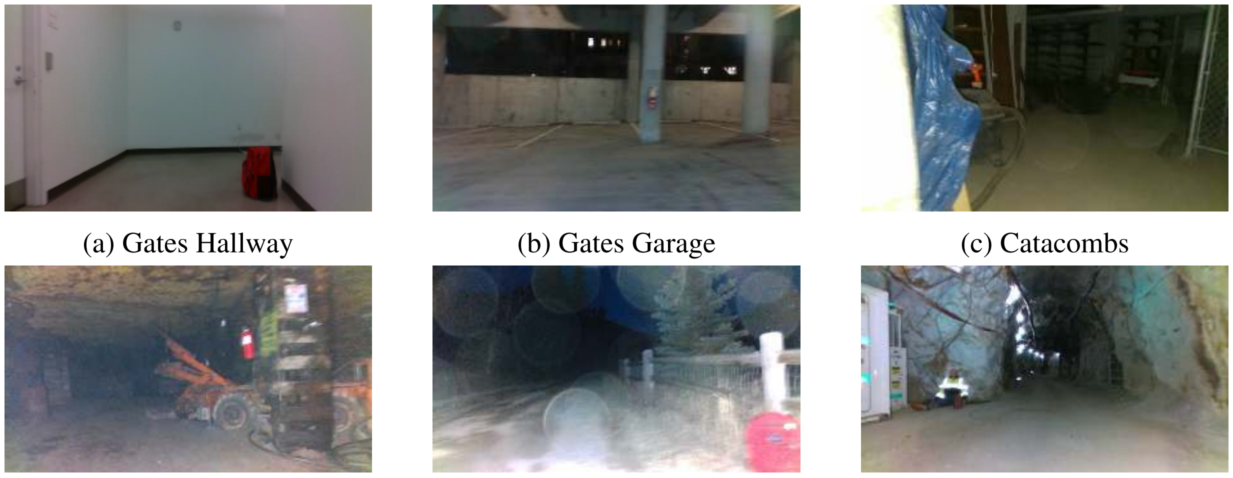
Table 3.4 only gives benchmarks up to a batch size of 4. To investigate if additional throughput gains were possible for the selected network architecture, we benchmarked the network with larger batch sizes as well, as shown in Table 3.7. Batch sizes larger than 4 would require sending more than one image per stream through the network at once. Table 3.7 indicates that this would increase throughput slightly, but it was decided that the added latency and complexity of batching multiple images per stream did not outweigh the small marginal increase in throughput as compared to using a single image per stream (with a batch size of 4).

3.2.2 Data Collection and Labeling

To finetune the selected checkpoint for object detection in the Tunnel Circuit, a large dataset of images and bounding boxes was collected and manually annotated. The dataset contains images sampled from videos of all 5 categories of objects captured from handheld devices, the ground robots, and the drone, in a variety of environments and lighting conditions. Images were labeled using a combination of custom semi-automated and manual tooling.

Environment Name	Environment Description
Gates Hallway	The hallway outside Team Explorer's workspace at CMU in the Gates building
Gates Garage	The parking complex attached to the Gates building (data collected at night)
Catacombs	Dusty underground storage area at CMU
Tour Ed Mine	An old coal mine in Tarentum, PA where the majority of field testing was performed
STIX Airbnb	Trail outside the Airbnb Team Explorer stayed at in Colorado during the STIX event
STIX Mine	The Edgar Experimental Mine where DARPA hosted the STIX event

Table 3.8: List of environments where data was collected for the Tunnel Circuit dataset


(a) Gates Hallway (b) Gates Garage (c) Catacombs

(d) Tour Ed Mine (e) STIX Airbnb (f) STIX Mine

Figure 3.2: A representative image for each environment in the Tunnel Circuit dataset is shown in the Figure. All images except Figure 3.2c were captured from payloads. Figure 3.2c was captured with the handheld setup. Bokeh effects visible in Figure 3.2c and Figure 3.2e are due to dust accumulation on the lens. Noise in Figure 3.2d is due to dust in the air being illuminated by the payload's bright LEDs.

Data Collection

The dataset used to train the network used during the Tunnel Circuit was trained on data collected from the environments listed in Table 3.8. A representative image for each environment can be seen in Figure 3.2. Data was collected in each environment using either a custom handheld setup with a RealSense, thermal camera, and LEDs, or directly from the payloads on each robot. When collecting video of each artifact, only the exact model of artifact specified by DARPA was used as no generalization across different models of artifact was required for the Tunnel Circuit.

Handheld data was collected by placing an artifact in a variety of locations and poses around the environment and recording short video clips while moving the handheld setup around. For each video clip, the cameras started recording when very close to the artifacts, and then moved backwards. The cameras were also moved around to capture a variety of viewpoints while ensuring that the artifact always stayed in frame. Video clips were typically less than 15 seconds long, and typically only had a single artifact visible at a time. It was assumed that this would be the case in the Tunnel Circuit due to the small number of artifacts (20) and large traversal distance

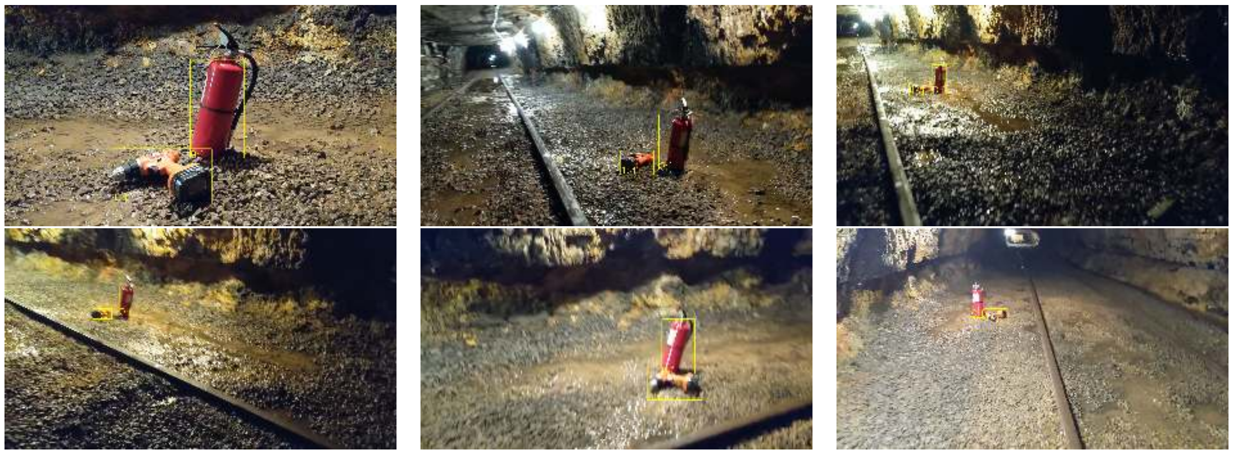


Figure 3.3: Example video labeled with semi-automatic labeling pipeline. The number inside the bounding box indicates the ID the artifact in the image (0-indexed), and the letter is a 1 letter identifier for the classification (d for drill, f for fire extinguisher).

(up to 4km). This assumption was not confirmed by DARPA. For safety, a small number of video clips were gathered with multiple artifacts in close proximity.

Video from the payloads was collected by recording video feeds from all cameras on the robot during both autonomous exploration and teleoperation in the test environments. Teleoperation ensured that all artifacts in the environment would be seen by the robot and at a slow and smooth pace, while autonomous exploration generated data with realistic issues, such as motion blur and rolling shutter effects in a number of frames. Artifacts would be spread through the environment prior to the run and were not moved while data was being gathered. Payload videos typically exceeded 20 minutes each and only had artifacts visible in a small number of frames, but served as valuable source of realistic data and negative examples for the dataset.

Data Labeling

The video clips gathered from the handheld devices were collected in a way to be amenable to a semi-automatic data labeling based on image tracking. Using a custom labeling program, a human labeler draws and labels bounding boxes around each artifact. An image tracker is initialized for each individual bounding box, and is used to update the bounding box for each artifact in every frame. The human labeler presses a key to step through each frame and watches the automatically updated bounding boxes to ensure they are still sufficiently accurate. If the tracker fails for any reason, the labeler adjusts the bounding box to tightly bound the artifact again, and the tracker is reinitialized with the new box. The label assigned to the box is used throughout the entire video clip. Figure 3.3 shows frames labeled with this process sampled evenly from a 30 second video clip.

A few parameters were added to the labeling program to trade off between labeling accuracy and speed. The tracker performance could be modified with a flag to change the scale of the image used for tracking. Downsampling the image resulted in a faster tracker framerate, but at occasionally reduced quality. The tracker type could also be changed for each video clip. We determined that, of the trackers we considered (those available in the OpenCV contrib library [35]), the CSRT tracker [31] offered the best tracking accuracy while maintaining a reasonable framerate (typically around 30 fps on a laptop) at native resolution, and thus it was used for all labeled video clips. Only every 5th labeled video frame was added to the dataset to reduce redundancy between samples.

The semi-automated labeling pipeline requires the handheld video to be recorded in a specific way to ensure a good tracker initialization, as described in the previous section. Videos recorded from the payloads on each robot do not have the all of the artifacts visible in the beginning and throughout the entire video, and thus and must be labeled manually. To speed up the manual labeling process, a second custom labeling program was developed. Labelers watch the video at an accelerated speed until they see an artifact. They then step backwards in the video until the artifact first entered the frame, and begin labeling every 5th frame. After drawing a box around an artifact (two clicks), the class can be assigned by pressing a single key (unique per class) on the keyboard. The last assigned class is preserved between frames to simplify the common case of only a single artifact being visible in frame. A tracker was not used to propagate labels between frames due to time constraints, but was the labelers' most requested feature. All labeled images, from both the semi-automated and manual labeling programs, were validated by two additional labelers to ensure consistent, high quality labels.

During the labeling process, we discovered the cell phone artifact was particularly difficult to label due to its small size, dark color which did not offer significant contrast to the environment, and large appearance variation due to the video being played on it. We speculated that a neural network would have similar issues when attempting to detect the cell phone, and thus did not label a significant number of cell phones in RGB images during the labeling process.

Dataset Generation and Statistics

Not all labeled images were used to train networks for the Tunnel Circuit. The sparse distribution of artifacts in the environment means that the video collected from the robots' payloads will have a large percentage of frames without artifacts, typically larger than 90% per video. Using such a large number of negative examples negatively impacts training performance and thus many must be filtered out. The train and eval datasets were generated such that there could be no more than a set ratio (15%) of negative examples to overall examples. Additionally, since all training images were sampled from videos, randomly distributing the videos into a train and eval set would not be sufficient as videos are highly correlated and the resulting sets would be very similar. Instead, entire video clips were manually assigned to be a part of either the train or eval datasets. All images were preserved at their native resolution (640 x 360) and losslessly encoded when added to the dataset. Any labels for cell phone artifacts were discarded. Statistics on the RGB train and eval datasets are presented in Table 3.9.

3.2.3 Training

All neural network training was performed using the TensorFlow Object Detection API. The benchmarks in Table 3.4 and Table 3.6 indicated that the MobileNet + SSD family of networks would offer the highest framerates, so a number of configurations using MobileNet were trained.

Statistic	**Train**	**Eval**
Examples with 0 bounding boxes	1408	71
Examples with 1 bounding boxes	7595	382
Examples with 2 bounding boxes	240	13
Examples with 3 bounding boxes	3	0
Positive examples (1+ bbox)	7838	395
Negative examples (0 bboxes)	1408	71
Overall examples (0+ bboxes)	9246	466
Ratio of positive examples to total	0.848	0.848
Ratio of negative examples to total	0.152	0.152
Backpack bounding boxes	1634	82
Drill bounding boxes	1626	88
Fire extinguisher bounding boxes	1978	100
Survivor bounding boxes	2846	138
Small bounding boxes	1970	106
Medium bounding boxes	3779	186
Large bounding boxes	2335	116
Ratio of small bboxes to total	0.244	0.26
Ratio of medium bboxes to total	0.467	0.456
Ratio of large bboxes to total	0.289	0.284

Table 3.9: Dataset statistics for Tunnel Circuit RGB Dataset

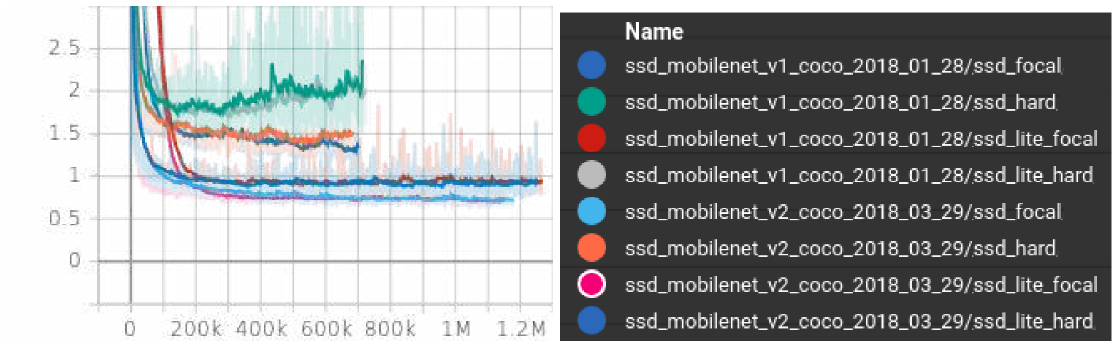


Figure 3.4: Loss values for all 8 configurations, with a smoothing of 0.95 applied in TensorBoard. X axis indicates number of steps. Y axis indicates overall loss value.

All network configuration files were derived from checkpoint files available in the TensorFlow Object Detection API, with all values except those mentioned left as provided. Each configuration used a batch size of 12, an input image size of 640 x 360, and the random horizontal flip, random brightness adjustment, random contrast adjustment, and ssd random crop data augmentations provided by the API with their default parameters. The RMSProp optimizer was used to train all networks, with a decay of 0.9, a momentum value of 0.9, an epsilon value of 0.9, and an exponentially decaying learning rate with an initial learning rate of 0.004, 500000 decay steps, and a decay factor of 0.95. A score threshold of 0.4 was used for non maximum suppression.

Networks using either MobileNet V1 or Mobilenet V2 as a feature extractor, SSD or SSD Lite as an object detection head, and focal loss or hard example mining as a loss function were trained, resulting in 8 separate configurations. Each network was initialized using checkpoints provided by the TensorFlow Object Detection API, which are pretrained for classification on ImageNet and then trained for object detection on COCO. All configurations were trained for a week with a single GPU for each network on an AWS p3.16xlarge instance. As no limit was placed on the number of steps for each network, checkpoints were kept every 4 hours of training to allow a checkpoint to be selected during evaluation from any point in the training.

3.2.4 Evaluation

COCO evaluation metrics were used to quantify the performance of each network. The overall loss plot for each network (in Figure 3.4) indicates that each network reached convergence by approximately 300k steps. The COCO metrics for each network configuration at 300k steps are given in Table 3.10. The table indicates that while the SSD Lite MNV2 Focal configuration had the best overall performance, all of the networks performed comparably in the majority of the evaluation metrics. As inference speed was a priority, using guidance from the previous benchmarking results we decided to use the SSD MNV1 Hard configuration for the Tunnel Circuit.

3.2.5 Inference

The selected RGB network is used for inference on each platform by first optimizing it as described in the benchmarking section with either TensorRT or OpenVINO, and then using a cus-

Configuration	**mAP**	**mAP**$_{large}$	**mAP**$_{med}$	**mAP**$_{small}$	**mAP**$_{0.5IOU}$	**mAP**$_{0.75IOU}$	**AR**$_1$	**AR**$_{10}$	**AR**$_{100}$	**AR**$_{large}$	**AR**$_{med}$	**AR**$_{small}$
SSD MNV1 Focal	0.6113	0.762	0.6601	0.3408	0.9353	0.6748	0.6687	0.6713	0.6713	0.805	0.716	0.4173
SSD MNV1 Hard	0.5799	0.7616	0.6395	0.3769	0.9017	0.6237	0.6611	0.6611	0.6611	0.7839	0.6949	0.4327
SSD Lite MNV1 Focal	0.6047	0.7627	0.631	0.3649	0.9094	0.674	0.6498	0.6502	0.6502	0.7886	0.6742	0.4223
SSD Lite MNV1 Hard	0.5505	0.7328	0.6008	0.3687	0.9026	0.5959	0.6432	0.6437	0.6437	0.7796	0.6627	0.4414
SSD MNV2 Focal	0.6288	**0.7909**	0.658	0.3485	0.91	0.6979	0.6744	0.6748	0.6748	**0.8163**	0.7123	0.4275
SSD MNV2 Hard	0.5747	0.753	0.6237	0.3687	0.9115	0.6539	0.669	0.669	0.669	0.7862	0.6902	0.4437
SSD Lite MNV2 Focal	**0.6365**	**0.7909**	**0.686**	0.3873	**0.9514**	**0.7202**	**0.7036**	**0.7056**	**0.7056**	0.8139	**0.7356**	0.4685
SSD Lite MNV2 Hard	0.5683	0.7756	0.6307	**0.4316**	0.9316	0.6528	0.6845	0.6845	0.6845	0.8053	0.6945	**0.5004**

Table 3.10: COCO evaluation metrics for RGB networks at 300k steps. MNV1 and MNV2 configurations use MobileNet v1 and v2 feature extractors respectively.

tom inference node to feed images through the network. When inferring on images from multiple cameras, batch inference is done on a set of images containing one image from each camera. A list of detected objects is published for each frame, containing bounding boxes, a classification, and a classification confidence. An empty list is published if no objects are detected.

3.3 Object Detector (Thermal)

The development of the thermal object detector closely mirrored that of the RGB object detector. The same data collection and labeling pipeline was used to label thermal images. Thermal images proved more difficult to label, but labelers found that viewing RGB images of the scene alongside the thermal images increased labeling speed. Backpacks, drills, and fire extinguishers were discovered to have minimal thermal signatures and were excluded from labeling. Cell phones were visible in thermal images but proved too small to label accurately and were thus also excluded from labeling. Only the survivor artifact was labeled. All data was kept at the native camera resolution of 640 x 512 and losslessly encoded. The dataset was generated in a similar fashion to the RGB dataset, though no negative examples were used. Statistics for it are given in Table 3.11.

Thermal object detection network training was performed after the RGB network had been trained. Using the insight from training the various RGB network configurations, we decided to only train a single configuration of the thermal network, identical to that of the selected RGB network (SSD MNV1 Hard). All configuration parameters were reused, with the only difference being the dataset. The process for inference with the thermal network is also identical to that of the RGB network.

3.4 LIDAR Renderer

The LIDAR renderer uses camera information (intrinsics and extrinsics) and registered point clouds from LOAM to render depth images. For RGB images, the LIDAR renderer provides an alternative source for depth information as depth images are already produced by the RealSense cameras. However, the LIDAR renderer serves as the only source of depth information for the thermal cameras, as they do not otherwise have associated depth information.

Statistic	Train	Eval
Examples with 0 bounding boxes	0	0
Examples with 1 bounding boxes	2031	391
Examples with 2 bounding boxes	1	0
Positive examples (1+ bbox)	2032	391
Negative examples (0 bboxes)	0	0
Overall examples (0+ bboxes)	2032	391
Ratio of positive examples to total	1	1
Ratio of negative examples to total	0	0
Survivor bounding boxes	2033	391
Small bounding boxes	1	0
Medium bounding boxes	160	59
Large bounding boxes	1872	332
Ratio of small bboxes to total	0	0
Ratio of medium bboxes to total	0.079	0.151
Ratio of large bboxes to total	0.921	0.849

Table 3.11: Dataset statistics for Tunnel Circuit Thermal Dataset

3.4.1 Method

Two implementations of the LIDAR renderer were written - a reference implementation which runs on CPU, and an optimized one which runs on a GPU using CUDA. The optimized implementation is used on both Mk. 0 and Mk. 1 and runs on the Xavier, where it renders depth images for either 5 or 6 image streams (4x RGB and 1 or 2x thermal). The reference implementation was used to validate the GPU implementation for correctness. The reference implementation also runs on the NUC on the drone payload as it does not have a GPU that supports CUDA. The slower performance of the reference implementation is acceptable as renders only need to be produced for a single RGB stream. Both implementations share the same overall algorithm, comprising two separate tasks – cloud aggregation and rendering.

Cloud Aggregation

The renderer aggregates the registered point clouds from LOAM. A rolling buffer of these clouds is maintained, whose size is proportional to the time it takes to render an image. 10 clouds are stored in the reference implementation, and 30 clouds are stored in the GPU implementation. The clouds are already transformed into a common frame and form a locally smooth point cloud. Global drift is present, but can be ignored since the rendered depth images only see a small local portion of the map.

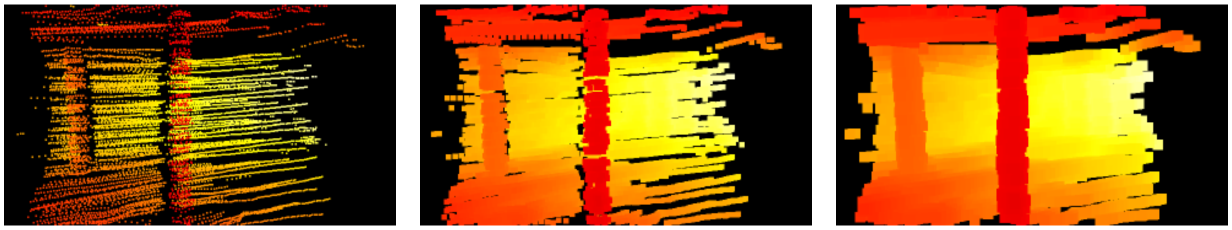

(a) 1 pixel inflation (b) 4 pixel inflation (c) 8 pixel inflation

Figure 3.5: Rendered images of the same scene with different inflation values. An inflation value of 4 was used for all payloads.

Rendering

For each rendered image, the renderer computes the position of the camera in the same frame as the aggregated point clouds. The renderer then uses the camera intrinsics to construct a pinhole camera model and projects each point through it to obtain a location in image coordinates. This coordinate, as well as pixels around this coordinate (based on a configurable inflation parameter) are updated based on the distance to the point, keeping the closer point. The projected coordinates are inflated (as shown in Figure 3.5) to provide a denser output image to ensure that depth values are not lost in any potential future downsampling. An inflation value of 4 was used for all payloads.

3.4.2 Results

A selection of outputs from the LIDAR renderer running on Mk. 1 is given in Figure 3.6, along with the depth image output from the RealSense camera and the associated RGB image for reference. In the first row, with Mk. 1's camera looking down a long tunnel, the rendered image is significantly sharper and more consistent than that of the RealSense. In the second row, with the left camera looking at a nearby wall, both depth images are similar. In the final row, with a scene from Mk. 1's back camera (which is tilted upwards), the RealSense depth image has significantly fewer holes than the LIDAR's due to the LIDAR's narrow vertical field of view.

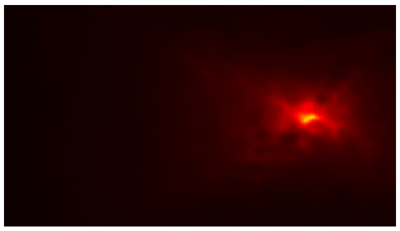
These results show that the output from the LIDAR renderer is either on par with, or better than that from the RealSense cameras, assuming the camera's full field of view is captured by the LIDAR. By fusing both depth images in the Depth Combiner, we obtain better depth images than either source provides individually.

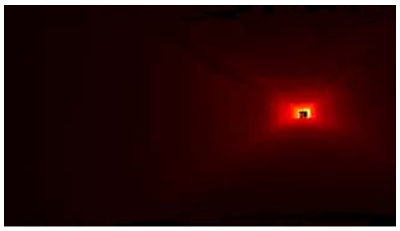
3.5 Depth Combiner

The depth combiner fuses depth images from the RealSense depth camera and LIDAR renderer into a single depth image to be used in the object detection localizer. The two image streams are aligned, and can thus be fused per-pixel. The following equation was used (shown as C++ code):

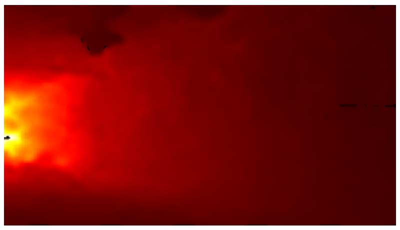
```
fused = (realsense > threshold || realsense == 0) ? lidar : realsense;
```

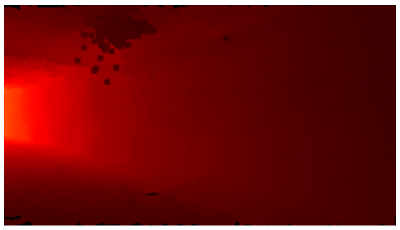
(a) Mk. 1 Front Color


(b) Mk. 1 Front Depth

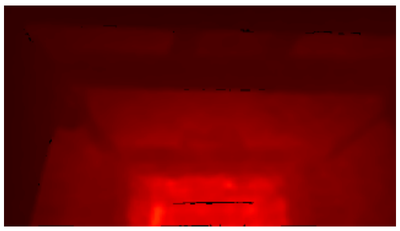

(c) Mk. 1 Front Rendered

(d) Mk. 1 Left Color


(e) Mk. 1 Left Depth


(f) Mk. 1 Left Rendered

(g) Mk. 1 Back Color


(h) Mk. 1 Back Depth

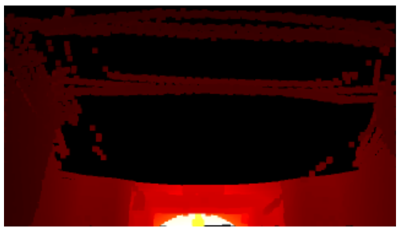

(i) Mk. 1 Back Rendered

Figure 3.6: Comparison of depth images generated by the LIDAR renderer to depth images from the RealSense cameras. Color images, captured simultaneously by the RealSense cameras, are provided for reference.

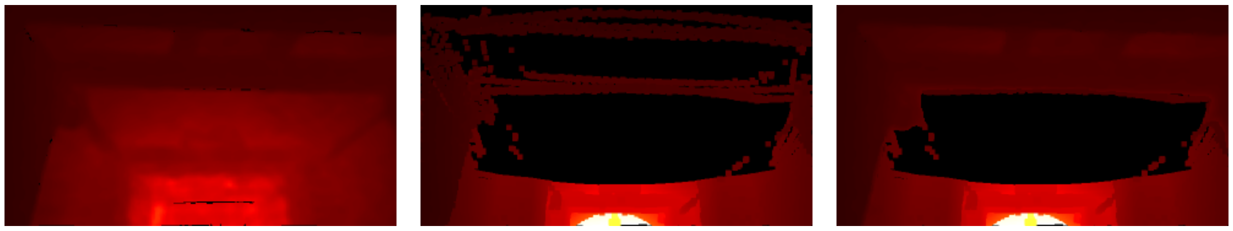

(a) Mk. 1 Back Depth (b) Mk. 1 Back Rendered (c) Mk. 1 Back Combined

Figure 3.7: Example of the output from the depth combiner, using the same scene as in 3.6g.

This fusion keeps LIDAR data whenever the reported RealSense data is either not present (`realsense == 0`) or exceeds some threshold (`realsense > threshold`). The threshold was empirically selected to be 2.5m, which is the approximate distance after which we observed significant variance in reported depth values between frames. This fusion allows us to use the high density RealSense depth information whenever it is available and sufficiently accurate (under the 2.5m threshold), and utilize the sparser LIDAR information otherwise. An example output is given in Figure 3.7.

3.6 Object Detection Localizer

The object detection localizer aggregates multiple 2d object detections for the RGB and thermal images and produces candidate Artifact Localizations. The localizer is able to combine detections of the same object from different camera streams together into a single Artifact Localization. Once sufficient evidence has been obtained for an Artifact Localization, the localizer will publish it to the Artifact Aggregator. Artifact Localizations are updated at 1 Hz using key pose information from LOAM to ensure global correctness for the duration of the robots' exploration.

For any potential artifact, evidence is only accumulated in the form of positive detections from the object detectors. Negative (empty) detections do not reduce the likelihood that an artifact is present at a given location. This biases the object detection localizer to return a large number of false positives, as most spurious detections from the object detectors will get reported as Artifact Localizations to the base station. This approach is consistent with our requirements and assumptions - as the human supervisor was able to validate artifact reports, even a large number of false positives is acceptable if it means that no artifacts are missed.

The object detection localizer consists of 3 main operations, separated into two threads. The first thread, running the key pose maintenance and detection backprojection tasks, is responsible for creating and maintaining a data structure which stores a 3d point cloud associated with each 2d detection with respect to a key pose. The second thread, running the clustering task, is responsible for extracting new Artifact Localizations from the data structure, updating existing ones, and publishing changes. The clustering task's execution time scales with the size of the data structure and thus runs in a separate thread, allowing all callbacks in the primary thread to be serviced in time.

Detection Key Pose Maintenance

The key pose maintenance task is responsible for populating and updating a data structure with key pose information from LOAM. When key poses are initially received, they contain a timestamp and an ID in addition to the pose information. Both pieces of metadata are stored alongside the key pose. The ID is used to update the corresponding key poses when a list of updated key poses is received from LOAM after a loop closure. The timestamp is made available to other tasks to be able to associate other sensor measurements to a key pose.

Detection Backprojection

The detection backprojection task runs as a callback upon receiving a set of 2d object detections, a color image, an associated depth image, and camera intrinsics information. For each detection, the depth information and camera intrinsics are used to project all pixels which fall inside the bounding box through a pinhole model into a 3d point cloud in the camera frame. The point cloud is colored by the RGB image, and each point in the cloud is assigned an ID corresponding to the classification of the detection. The timestamp of the image frame is then used to look up the immediately preceding key pose in the structure built by the key pose maintenance task. The point cloud is then transformed to be in the key pose's frame, and is added to a list of point clouds associated with that particular key pose. The color image has the bounding box drawn on it, and is stored alongside the point cloud. The centroid of the point cloud is also computed and stored, to enable more efficient clustering. This backprojection process is identical for both RGB and thermal images.

It is important that the backprojected point clouds are stored relative to a particular key pose rather than relative to the robot's /map frame to maintain global consistency and handle loop closure events. The transform necessary to convert the point cloud into the key pose frame is obtained in two steps. First, the static transform from the camera to the LOAM sensor frame is combined with the /integrated_to_map frame at the image timestamp, yielding a transform from the camera frame to the /map frame. Then, the pose of the corresponding key pose (which is already in the /map frame) is used to create a transform from the camera frame to the key pose frame. Storing the backprojected point clouds relative to a key pose rather than directly in the /map frame allows the clouds to be more accurately transformed into the /map frame as the key poses are updated. By using the /integrated_to_map topic, we ensure that the error the eventual transformation into the /map frame is due to small IMU integration error between key poses, rather than a large global drift.

Some additional filtering is also performed during this step to improve the quality of data being stored. The robot's pose estimate must move by at least a small amount (7.5cm) between frames, or clouds are not generated at all. This threshold ensures that redundant information does not get stored if the robot is sitting still or moving very slowly. Additionally, depending on the source of depth information being used, different downsampling and depth cutoff parameters are used in the backprojection. Downsampling reduces the density of the backprojected point cloud by skipping certain pixels when backprojecting. Every 4th row and column is kept when utilizing dense RealSense depth information, while every 2nd row and column is kept when using the comparatively sparse LIDAR rendered depth information. The depth cutoff is used to discard any points which have depth values beyond a certain point, and is used to compensate for low accuracy in the depth values. A depth cutoff of 10m was selected, intended to only act as a cutoff for the LIDAR depth values as a lower cutoff of 2.5m was used for the RealSense depth data in the Depth Combiner.

Clustering

The clustering task continually (at 1 Hz) creates Artifact Localizations from the data structure populated by the backprojection task, and publishes updates for downstream nodes to use. First, the centroid of each backprojected point cloud is transformed into the /map frame using the updated pose of the corresponding key pose. This centroid cloud is then clustered using Euclidean clustering from PCL, with a cluster tolerance of 0.5m and a minimum cluster size of 3. These parameters group objects which have been detected a sufficient number of times (3) and are close enough (0.5m) together. The clustering is purely geometric, and does not account for the classification of the centroid. This choice enables the clustering process to be more robust to misclassification from the network, but in exchange sacrifices the ability to distinguish multiple objects of different types near each other.

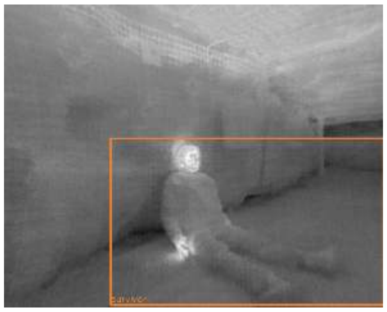
The centroid of each cluster can be treated as an approximation for the centroid of each artifact that the robot has seen. However, the computed centroid of centroids is imprecise, and can be refined by integrating the complete point clouds for each detection. First, the clouds are transformed into the /map frame and aggregated into a single cloud for the cluster. The centroid of this cloud is then found, and used as the refined position of an Artifact Localization. This centroid differs from the one computed using the centroid of each point cloud as it will weight the centroids based on the number of points in the cloud, rather than weighting all centroids equally. While finding the centroid of the cluster point cloud, a majority vote is performed over the classifications to determine a label for the Artifact Localization. Finally, the color images (with bounding boxes) for each detection in the cluster are added to the Artifact Localization as evidence.

Once the complete list of Artifact Localizations has been created from the available data, it is compared against existing valid Artifact Localizations generated by previous iterations. New Artifact Localizations are matched pairwise against previous ones, again only considering Euclidean distance, using a threshold of 0.25m. Previous Artifact Localizations which are matched to new ones have their parameters updated with the ones of their match. Previous ones which were not matched are now considered invalid, and are marked as such, but are not forgotten. New Artifact Localizations which did not match any previous ones are given a new ID and stored as new valid Artifact Localizations. All changes to Artifact Localizations (updates, invalidations, and new ones) are published, along with the IDs of each Artifact Localization. This process of publishing deltas in the list of Artifact Localizations is used to increase memory efficiency and reduce transmission bandwidth, and is used everywhere Artifact Localizations are published.

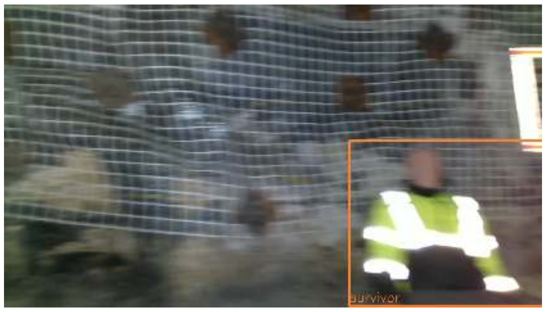
When publishing Artifact Localizations, only a small number of images are kept. Typically, an Artifact Localization will amass 50 or more detections from different cameras as the robot drives by an artifact. These images are highly redundant, and it is thus inefficient to transmit them all. Instead, only 4 selected images are published. The images are sorted by their capture timestamp and then evenly distributed into 4 chunks. The first image from each chunk is kept,

(a) Survivor Image 1

(b) Survivor Image 2



(c) Survivor Image 3 (thermal)



(d) Survivor Image 4

Figure 3.8: Each published Artifact Localization contains only a few images automatically selected from the many detections which combine together to form a single Artifact Localization. These images can be selected across different cameras and different types of cameras, as shown above.

resulting in images which show multiple perspectives of an artifact as the robot travels by. An example of 4 automatically selected images for an artifact is shown in Figure 3.8.

3.7 WiFi Scanner

The WiFi scanner uses the WiFi modules on the NUC (Mk. 0, Mk. 1, drone) and the Xavier (Mk. 1) to search for nearby devices acting as access points. Scans are run using wpa_supplicant and are run continuously. Each scan produces a list of nearby access points with a MAC address, SSID (if available), and an RSSI value. The scan results are timestamped with the completion time of the scan, are assigned a frame ID coincident with the LOAM sensor, and are published immediately. Each scan takes approximately 3 seconds to complete, and thus Mk. 0 and the drone payloads publish scan results at approximately .35 Hz while Mk. 1 is able to publish at approximately double the rate, between 0.6 - 0.8 Hz since it uses two WiFi chips to scan instead of just one. Other WiFi interfaces are present on each robot, but are used for wireless communication. The process of WiFi scanning significantly reduces the available bandwidth for wireless communication and thus the WiFi scanner is only allowed to use interfaces which are otherwise unused.

3.8 Signal Localizer

The signal localizer produces Artifact Localizations for cell phone artifacts using the scan results from the WiFi scanner. A Bluetooth scanner was also implemented that could feed into the signal localizer, but was disabled due to low reliability. The structure of the signal localizer is quite similar to that of the object detection localizer. Artifact Localizations are generated continually at 1 Hz, and are updated with new key pose information when available. When changes to the Artifact Localizations are significant, they are published to be used downstream.

Like the object detection localizer, the signal localizer consists of 3 main operations, this time running in a single thread. The first two tasks maintain a data structure with key pose information and wifi scan results. The third task continually solves for positions of cell phone artifacts based on the scan results, and publishes Artifact Localizations for new detections or for significant position changes.

Signal Key Pose Maintenance

This task is functionally equivalent to the key pose maintenance task in the object detection localizer. Key poses are stored in a data structure alongside their timestamp and ID, and are updated with new poses from loop closures when available.

Signal Reading Storage

This task serves a similar role to the object detection localizer's detection backpropagation task - to populate a data structure with sensor data to be used by a solver. DARPA guarantees that each cell phone artifact's access point will have an SSID of the form "PhoneArtifact##", where ## is non-negative 2 digit integer. The scan result is filtered to remove all SSIDs which do not match the provided pattern. All remaining results are guaranteed to belong to a distinct cell phone artifact. Using the timestamp of the scan result, the robot's sensor frame position is determined relative to the nearest key pose, in a manner similar to that in the object detection localizer. The transformation between the robot's sensor frame and the antennas is ignored. Each remaining scan result, consisting of a MAC address, SSID, and RSSI value, is stored with the robot's computed position relative to the corresponding key pose in the data structure.

Signal Localization

Each cell phone artifact has a unique MAC address, which allows multiple readings for the same cell phone artifact to be aggregated together. Readings are gathered by MAC address and each group is processed sequentially in an identical manner. First, the stored position (which is relative to a key pose) is transformed to the robot's /map frame using updated key pose information. This creates a list of RSSI values at different locations, which is the setup for typical cell phone trilateration problems. However, treating the cell phone localization problem as one of trilateration requires a model to convert from RSSI to distance [24]. While these models work well with line of sight between the transmitter and receiver, we were unable to obtain accurate distance estimates around multiple corners and through thick rock walls, and thus an alternate solution was implemented.

The alternate solution relies on an important insight about the structure of the problem. We realized that the corridors in the tunnel circuit are quite narrow, meaning that the robot would, at some point, get within a meter or two of each artifact as it drove by them. Since the maximum allowable error for artifact positions was 5m, simply reporting the position of the robot with the highest RSSI value for each cell phone could be sufficient, using the assumption that RSSI was inversely proportional to the robot's distance from the artifact. This is the case for RSSI values from WiFi scanning, but is not true for Bluetooth. Results from Bluetooth scans would saturate the RSSI value when the robot was within approximately 20m of the cell phone, requiring a slightly more complex solution than simply finding the largest RSSI value.

First, the received RSSI values and corresponding robot poses are sorted by timestamp. Then, the highest RSSI value is found, and the list of RSSI values is segmented into contiguous groups where the RSSI values equal the maximum found. For WiFi, the maximum RSSI value will typically only occur for one reading, so the corresponding pose can be returned immediately. For Bluetooth, the groups are typically a set of values for each time the robot drove by the cell phone (e.g. going into and coming out of the tunnel) since the RSSI readings saturate when within 10m to 20m of the cell phone. The midpoint of the arc formed by the robot poses in each set is then found. The midpoints for each set are averaged together and reported as the position of the cell phone artifact. After positions are determined for each cell phone artifact, they are compared against those from the previous iteration, and updates are published in a manner similar to the object detection localizer if any new cell phones are present, or if any have moved significantly.

3.9 State Estimate Delay Estimator

An implementation detail of LOAM is that the timestamps for the state estimates are based on a separate clock than other connected components. Subsecond precision for this clock comes from the IMU's internal clock, which starts counting from 0 at the IMU's startup, while the seconds count is derived from the NUC's system time. This scheme results in an offset between the two clocks of between 0 and 1 seconds, with the NUC's system clock always being ahead. This offset is different with each initialization of the state estimation system, but is consistent for the duration of a run (1 - 2 hours). All other clocks on a robot are synchronized to the NUC's system clock using Chrony, which results in an error on the order of hundreds of microseconds between clocks and is considered negligible.

The state estimate delay estimator estimates the offset between the NUC's system clock and the state estimate clock and publishes the offset to be used by other nodes which use data from both the state estimate and other sensors. Subscribers subtract the published estimate from the timestamp of received sensor data when querying the robot or sensor pose corresponding to sensor data. The estimate is obtained by computing a moving average of the offset between the published timestamp and the NUC's system time for every twentieth state estimate received on the 200 Hz /integrated_to_map topic. The estimate is published every update, resulting in a rate of 10 Hz. The following equation is used to update the estimated delay:

$$d_{now} = 0.9\, d_{prev} + 0.1\, (t_{now} - t_{state}) \tag{3.1}$$

where d_{now} is the current delay estimate, d_{prev} is the previous estimate (initialized to 0), t_{now}

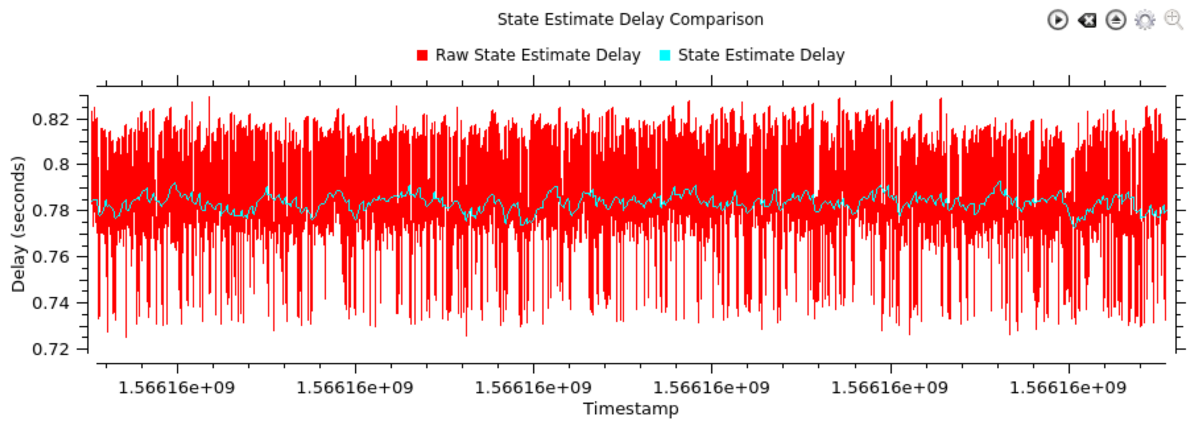


Figure 3.9: The plot shows delay estimates over a period of 60 seconds from a single run using the Mk. 1 payload. The red line shows the raw delay between the state estimate timestamp and the system time, $t_{now} - t_{state}$, at 200 Hz. The cyan line shows the filtered output using Equation 3.1, published at 10 Hz.

is the current system time, and t_{state} is the timestamp contained in the state estimate. An example output from this filter is shown in Figure 3.9.

3.10 Artifact Aggregator

The artifact aggregator combines Artifact Localizations from multiple sources into a single stream using two tasks.

Delta Listener

For efficiency, each source publishes Artifact Localizations as a series of deltas, rather than publishing the entire list of Artifact Localizations at each update. This task listens to the deltas from each source and accumulates them into a complete list of Artifact Localizations for each source. Unlike previous steps in the pipeline, no transformations between frames happen here as the individual sources are expected to publish artifacts in the /map frame and update via deltas when necessary.

Filtering

Every second, the accumulated Artifact Localizations from each stream are combined using a similar clustering mechanism to the object detection localizer. First, the position of each Artifact Localization will be added to a point cloud. No transformation is necessary as all points are already in the /map frame. Then, the point cloud is clustered using PCL's Euclidean clustering with a cluster tolerance of 0.5m and a minimum cluster size of 1. The cluster tolerance was kept the same as the object detection localizer, but the minimum cluster size was reduced to 1 to not ignore any artifacts reported by any source.

Artifact Localizations are combined based on their geometric clustering results. For each cluster, the classification of the new Artifact Localization is determined by summing the reported confidence from each constituent Artifact Localization and selecting the class with the highest confidence. The images for the new Artifact Localization are selected identically to the object detection localizer – all images are sorted, and then 4 images are selected by choosing the first image in each fourth of the list. Individual images may come from different sensing modalities. An example is shown in Figure 3.8. The list of new Artifact Localizations is compared to the previous list of Artifact Localizations to determine a delta (consisting of updates, invalidations, and new Artifact Localizations), which is published and transmitted to the base station.

3.11 Artifact Debouncer & Compressor

The artifact debouncer & compressor is responsible for providing an efficient and reliable transmission mechanism for Artifact Localizations over the wireless communication link between each robot and the base station. Artifact Localization deltas from the artifact aggregator are used to generate a complete list of Artifact Localizations, each of which is initially marked as dirty. A number of strategies, outlined below, are used to determine which dirty Artifact Localizations should be transmitted to the base station. The bandwidth usage with the various techniques over an hour long competition run from the Tunnel Circuit for Mk. 1 can be seen in Figure 3.10.

Compression

The images associated with each Artifact Localization are compressed before transmission. JPEG compression with a quality parameter of 95 is used for each image, resulting in images which are typically between 5 and 10 percent of the original image size in bytes. No other data associated with the Artifact Localization is compressed as its size (10s of bytes) is negligible compared to even the compressed images (each 50-100 kilobytes).

Retransmission

The lossy nature of the wireless communication link means that it is not guaranteed that transmitted packets will arrive at the other end. Thus, simply publishing a dirty Artifact Localization is insufficient to guarantee receipt at the base station. This is solved by attaching a timestamp to each transmission and having the receiving node at the base station publish an acknowledgment indicate that it received the Artifact Localization(s) sent at a particular timestamp. Upon receiving this acknowledgment, this node clears the dirty bit from each transmitted Artifact Localization if it was not updated since transmission. This acknowledgment may be dropped when sent via the wireless link as well. This is counteracted by retransmitting dirty Artifact Localizations from each robot until an acknowledgment is received.

Backoff

One reason for packets not reaching the base station may be that the robot is simply out of wireless communication range. Continually retrying transmission of dirty Artifact Localizations while out of communication range simply pollutes the wireless spectrum for other robots which may still be in range. Thus, linear backoff is employed. After each transmission attempt, an increasing predetermined time must pass before the dirty Artifact Localization can be transmitted again. The delay values are 5, 10, 15, 20, 25, and 30 seconds. Each transmission attempt increases the delay before the next transmission, up to a maximum of 30 seconds.

Debounce

As a robot drives by an artifact, the artifact will be detected in multiple consecutive camera frames. The object detection localizer will continually update the position of the artifact, and will publish updates every second while the robot drives by. The signal localizer does the same thing as the robot approaches a cell phone. These updates propagate through the artifact aggregator and result in a large number of updates published as the system converges on an estimate of the artifact's location and class. Forwarding each of these updates directly to the base station would be wasteful as another update would follow a second later. This node debounces artifact updates by only marking Artifact Localizations as dirty once an update has not been received for at least 3 seconds. This typically happens after the robot has finished driving by the artifact and can no longer sense it, and is unlikely to update the Artifact Localization again.

Grouping

Various factors, such as a loop closure event, can result in a large number of dirty Artifact Localizations that are ready to be transmitted. Towards the end of a robot's run, this can be close to 200 Artifact Localizations to transmit at once, or approximately 50 MB assuming an individual compressed Artifact Localization is 250 KB. Sending this much data in a single transmission is infeasible over a wireless link. Instead, a maximum of 10 dirty Artifact Localizations are transmitted at once.

3.12 Artifact Uncompressor

The artifact uncompressor is responsible for receiving and uncompressing Artifact Localization messages sent by the artifact compressor nodes on each robot. Upon receipt of a message, the uncompressor will publish an acknowledgement with the transmission timestamp contained inside the received Artifact Localization message. The acknowledgement is only published once. If the acknowledgement is dropped in transit over the wireless link, the compressor will simply publish the artifact again at a later time. The uncompressor does not manipulate or store the messages it receives in any way - each compressed Artifact Localization received from the compressor node is simply uncompressed and broadcast on the base station for other nodes, such as the GUI, to use.

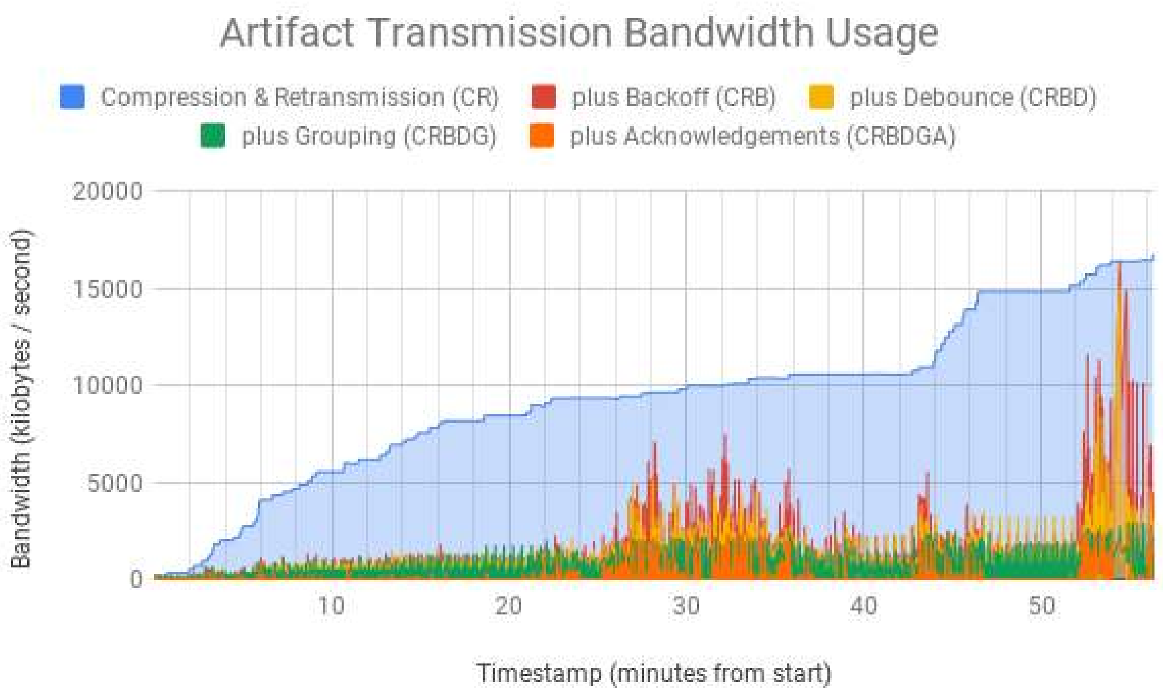
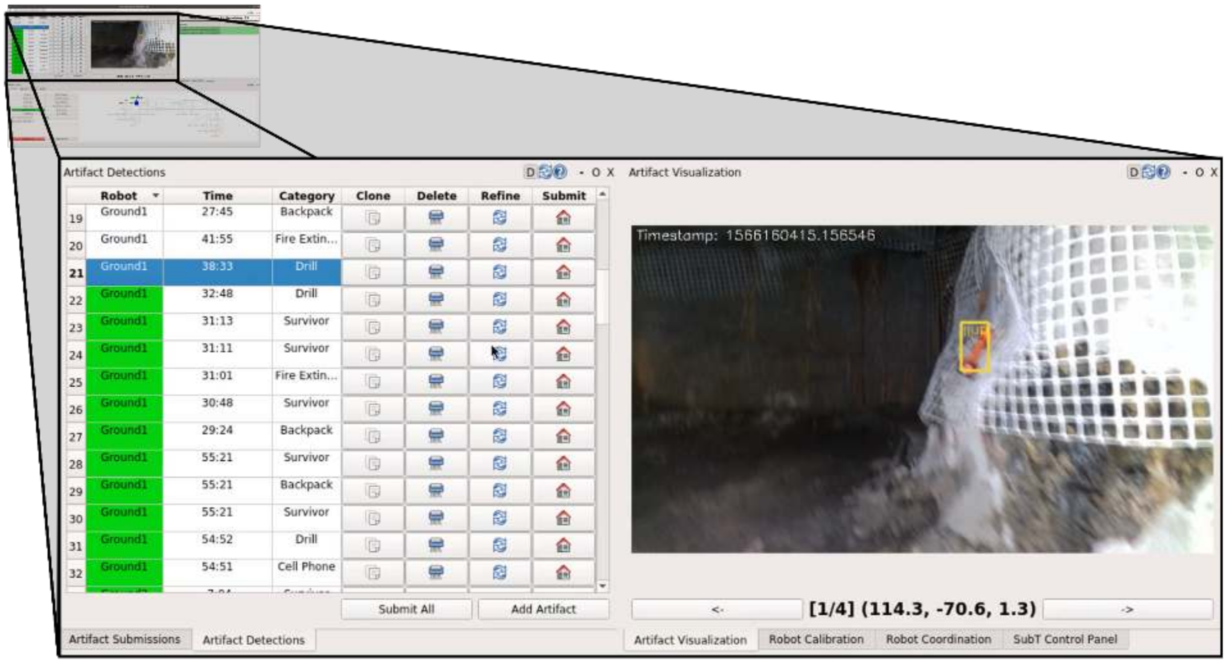


Figure 3.10: The graph shows the bandwidth used by the compressor to try to transmit artifacts over an hour long competition run from Mk. 1 during the Tunnel Circuit. Features are incrementally added, starting with compression and retransmission, and then backoff, debounce, and grouping. None of these configurations have acknowledgements coming back from the base station, which is common when the robot is not within communication range of the base station. The final configuration (CRBDGA, orange) shows the effect of simulating guaranteed acknowledgements.



	Robot	Time	Category	Clone	Delete	Refine	Submit
19	Ground1	27:45	Backpack				
20	Ground1	41:55	Fire Extin...				
21	Ground1	38:33	Drill				
22	Ground1	32:48	Drill				
23	Ground1	31:13	Survivor				
24	Ground1	31:11	Survivor				
25	Ground1	31:01	Fire Extin...				
26	Ground1	30:48	Survivor				
27	Ground1	29:24	Backpack				
28	Ground1	55:21	Survivor				
29	Ground1	55:21	Backpack				
30	Ground1	55:21	Survivor				
31	Ground1	54:52	Drill				
32	Ground1	54:51	Cell Phone				

Timestamp: 1566160415.156546

[1/4] (114.3, -70.6, 1.3)

Figure 3.11: The top left portion of the GUI is used for Artifact Localization verification and consists of two separate panes. The left side of the image shows a queue for Artifact Localizations which still need to be reviewed by the human supervisor, along with the classification and which robot reported the Artifact Localization and when it was detected relative to the start of the run. The right side of the image shows the images contained in the Artifact Localization report, if any, one at a time. The images can be cycled through using the arrows. The coordinates in the Artifact Localization displayed underneath the images.

3.13 GUI

The GUI is used by the human supervisor at the base station to perform a large variety of tasks related to initializing and controlling the fleet of robots. One of these tasks is to validate and, if necessary, refine Artifact Localizations returned from robots in the fleet. Validated Artifact Localizations are reported to DARPA for scoring. An example of the validation GUI, populated with Artifact Localizations, is shown in Figure 3.11. All other tasks performed at the base station are unrelated to the scope of this work.

Uncompressed Artifact Localizations sent by the artifact uncompressor are received by the GUI and have their coordinates transformed into the DARPA frame using the corresponding robot's calibration. The ID of each Artifact Localization is used to determine whether the Artifact Localization is already being displayed or if a new Artifact Localization has been received. New ones are added to a queue of Artifact Localizations which still need to be inspected by the human supervisor and are highlighted in green. Updates to existing Artifact Localizations either cause them to be removed from the queue, in the case of an invalidation, or simply updated, in which case they will again be highlighted in green. The human supervisor can click on any pending Artifact Localization in the queue, which will render it in white and display the first contained image, if any, and perform one of the following actions:

1. **Submit** – After verifying that the images, if any, contain an object, that the bounding box is correct, and that the classification matches the object, the Artifact Localization can be submitted to DARPA. If the artifact is within 5m of DARPA's surveyed coordinates, the displayed score in the GUI will increase.
2. **Refine** – If the human supervisor has verified that images, bounding box, and classification are correct and submitted the artifact to DARPA but did not receive a score increase, the artifact position can be manually refined. This can be used to correct for map drift or a poor initial calibration between the robot's world frame and the DARPA frame. The Artifact Localization's class can also be updated if necessary.
3. **Clone** – Any Artifact Localization can be duplicated, with the copy being placed in the queue as a new artifact. This is useful if two artifacts have accidentally been clustered together. The duplicated artifact can be amended manually to match the second of the two artifacts seen in the images.
4. **Delete** – The human supervisor can choose to delete any artifact. Typically, they will delete false positive Artifact Localizations, duplicates from multiple robots, and updates to artifacts which have already been successfully scored.

During the Tunnel Circuit competition event, the human supervisor for Team Explorer was able to inspect approximately 200 Artifact Localizations in less than 5 minutes. Many of these Artifact Localizations could be rejected in less than 1 second due to the displayed image being wrong. Valid Artifact Localizations were submitted quickly in the beginning as no duplicates were possible. Towards the bottom of the queue, additional care was taken to ensure duplicates did not get submitted, increasing the time spent per artifact. Cell phone Artifact Localization coordinates were occasionally refined due to the high error of the signal localizer. No other artifact types were refined unless the human supervisor suspected large map drift had occurred. There were no instances under which the classification of the artifact needed to be updated manually.

Chapter 4

Experiments and Results

During the Tunnel Circuit, we deployed our complete system, comprising 2 ground robots (R1, R2) and a drone (D1) a total of 4 times over 4 days. For the 1st and 4th days, we operated in the Experimental mine, while on the 2nd and 3rd days we operated in the Safety Research Mine. The second run for each mine had a different set of artifacts and locations than the first. The final scores for each deployment, as recorded by DARPA, are shown in Table 4.1. Each configuration had 20 artifacts deployed.

4.1 Experimental Setup

After the Tunnel Circuit, a number of experiments were performed to determine the utility and performance of each part of the system. All experiments were performed by playing back data recorded on R2 (which carried the Mk. 1 payload) on day 2, in the Safety Research mine with artifact configuration A. Ground truth artifact locations for this configuration were obtained by transforming the ground truth provided by DARPA after the competition into R2's /map frame using the calibration determined at competition. The calibration was not updated between experiments. This transformation allowed all experiments to take place in R2's /map frame, rather than requiring a transformation of artifact coordinates during experiments. A map of the Safety Research mine, artifact locations, as well as the areas traversed by R2 on day 2 is given in Figure 4.1.

During each experiment, the system reported artifacts directly to a scoring server, rather than reporting them to the base station for human verification. While this setup is slightly different than that used during competitions, it enables more consistent and repeatable experiments and

Date	**Course Name**	**Artifact Configuration**	**Official Score**
August 17 2019	Experimental	A	10
August 18 2019	Safety Research	A	13
August 20 2019	Safety Research	B	12
August 21 2019	Experimental	B	12

Table 4.1: Official scores from Tunnel Circuit

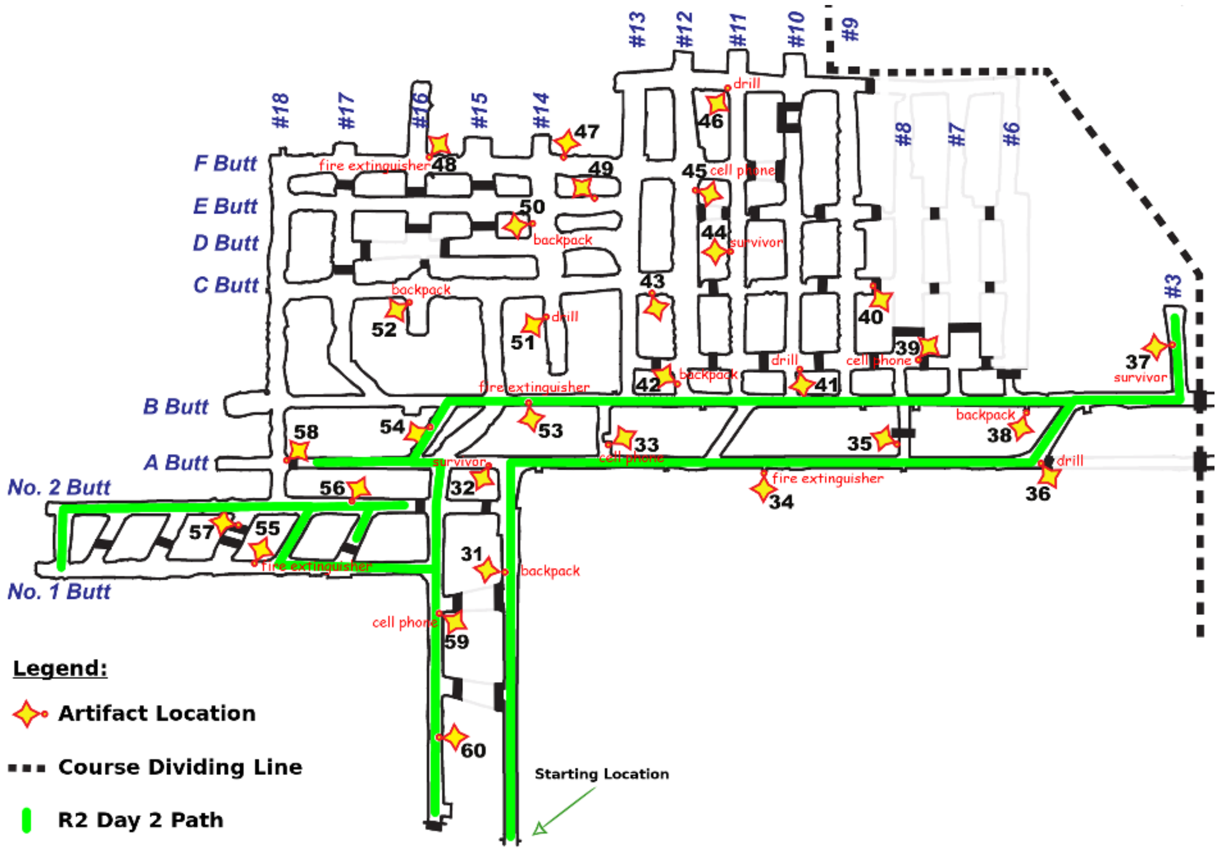


Figure 4.1: Ground truth locations for artifacts in the Safety Research mine during day 2 of the Tunnel Circuit. Numbers with text represent locations where artifacts were actually present. Locations without text did not contain artifacts. The cell phone at location 33 was not broadcasting an access point during this day due to a confirmed error by DARPA. The robot started at the right tunnel in the center of the figure and turned around after reaching No. 1 Butt after approximately 20 minutes.

allows multiple experiments to be run in parallel. The scoring server aggregates all artifacts reported by the system similarly to other nodes, and scores the entire list of artifacts whenever an update is received. Artifacts are scored by matching valid artifacts to ground truth artifacts, ensuring that the reported artifact is within 5m Euclidean distance of the ground truth artifact and shares the same class. Precision is determined according to Equation 4.1. Accuracy is measured by counting the number of correctly scored artifacts. The results are appended to a file on disk, allowing the system's performance over time to be analyzed.

$$precision = \frac{\#\ correct\ artifacts}{\#\ valid\ artifacts} \tag{4.1}$$

Streaming results directly to the server also removes the effect of communication network, assuming instead that the robot is always able to communicate with the base station. An analysis of the actual bandwidth usage under different configurations of the artifact compressor for the R2 day 2 dataset is given in Figure 3.10.

The results of each experiment are compared to the results of the configuration used during the Tunnel Circuit, described in Figure 3.1. Values in plots indicating the number of artifacts detected have been slightly altered by adding an offset of between -3% and 3% to all values to improve legibility. Values in all other plots have not been altered.

4.2 Single Sensor Usage

Our core hypothesis for this work is that the use of multiple sensors and sensing modalities is necessary and advantageous in detecting and localizing artifacts. To test this hypothesis, as well as to understand the utility of each individual sensor, the artifact detection and localization pipeline was run with only a single sensor active at once. Each of the 4 RGB cameras and 2 thermal cameras were run separately, while both radios feeding the signal localizer were run simultaneously. The results are shown alongside the full system with all sensors active simultaneously in Figure 4.2.

When combining all of the sensors, the robot was able to score all 10 artifacts in its path by approximately 15 minutes, which was a few minutes before it turned around at No. 1 Butt. No single sensor was able to achieve the same score, even at the end of the competition run. A total of 9 artifacts could have been detected by each RGB camera, which is all artifacts except the cell phone at location 59 in Figure 4.1. Each RGB camera scored most, but not all, of the artifacts in the robot's path, and required the robot to turn around and back to the entrance of the tunnel, which happened at approximately 20 minutes into the run. Using the full system, the robot would instead be able to keep exploring and continue to report information for situational awareness.

Two survivor artifacts were present along the robot's path which could have been detected by the thermal cameras. When using only the left thermal camera, only 1 survivor artifact was scored. None were scored by the right thermal camera. This may be attributed to a lower quality thermal object detector as a result of fewer training examples, or the low framerate of thermal detection (5 Hz) preventing sufficient confidence from accumulating to consider a detected object an artifact.

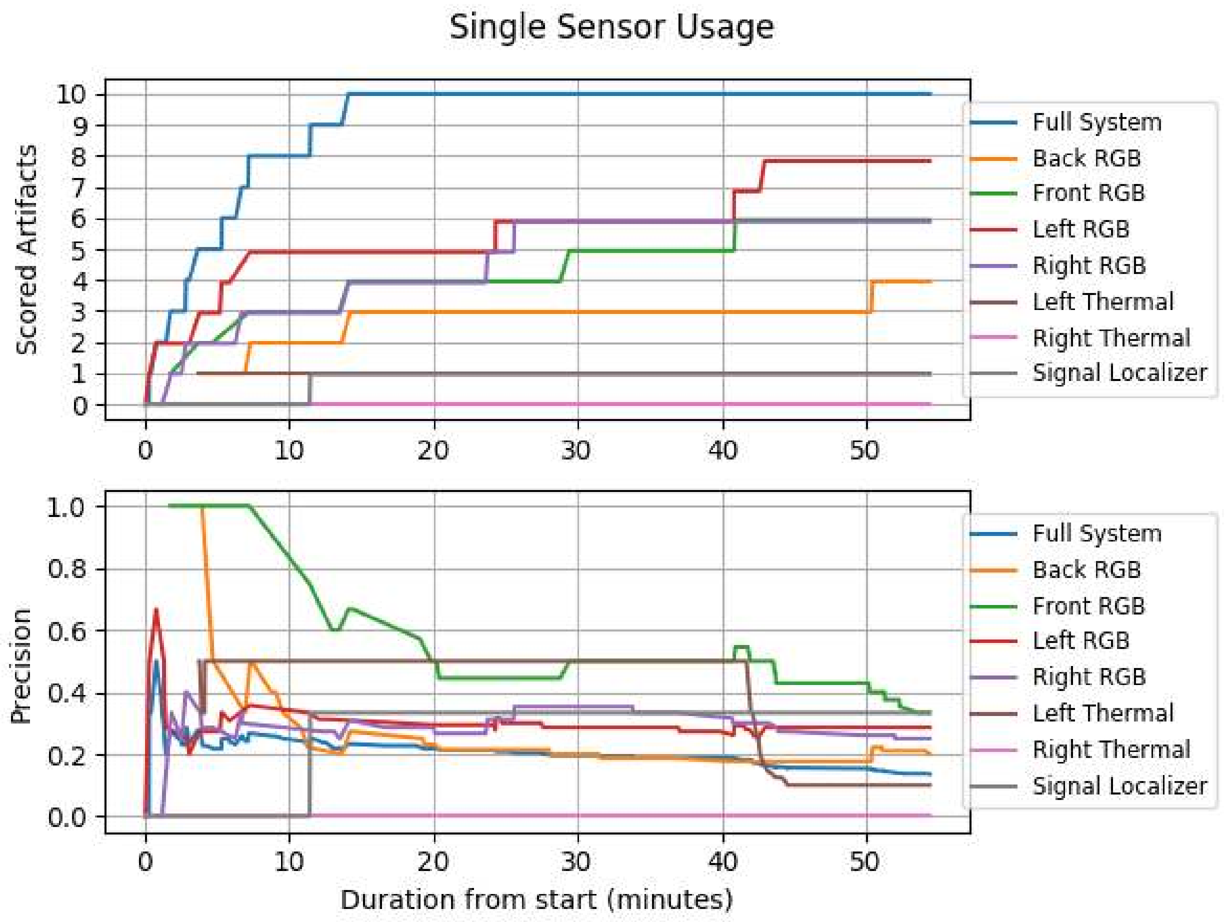


Figure 4.2: The plot shows the results from configurations in which only a single sensor was active. Both radios were used simultaneously for the signal localizer. No single sensor performs as well as the complete system, which detected all artifacts in its path. Individual sensor configurations do typically offer higher precision (fewer false positives) than the complete system.

The signal localizer is the only module capable of localizing cell phones, and was able to score the single cell phone artifact acting as a WiFi access point along its path (59). The other two broadcasting cell phone artifacts (39, 45) were detected by the WiFi scanner but could not be localized sufficiently accurately by the signal localizer since the robot did not traverse the areas near the artifacts.

While each individual sensor had a lower overall score than the complete system, each single sensor configuration except those with the thermal cameras resulted in a higher precision than the overall system. This may be due to the fact that all configurations used the same confidence thresholds for determining when to publish an Artifact Localization. When only a single sensor is used, more false positive detections are required per sensor to create a false positive Artifact Localization than when all sensors are fused together. The higher false positive rate for the full system is acceptable based on the team's concept of operations, but this experiment may indicate an opportunity to lower the false positive rate by dynamically adapting confidence thresholds based on the number of sensors being fused.

4.3 RealSense Depth Only

During the development process, we discovered that the RealSense camera's depth accuracy drops significantly after approximately 2.5m. As a result, depth information from the RealSense is discarded after 2.5m and replaced with depth information from the LIDAR in the full system. This experiment quantifies the utility of the fusion of LIDAR data by examining the system performance with only a single RealSense activated and limited to using only RealSense depth information under 2.5m. The results are shown in Figure 4.3.

The front and back cameras have the largest decrease in accuracy when utilizing only RealSense depth information, with each scoring only half of the artifacts scored when fusing LIDAR information. This is likely due to the topology of the environment itself. The Tunnel Circuit environment consists of many narrow corridors. When the robot drives by artifacts and they are detected by the left and right RGB cameras, the artifact is typically very close to the robot. In contrast, the front and back cameras are typically looking down the long corridors and thus artifacts are often beyond the 2.5m threshold. The precision of each of the RGB cameras with both types of depth information is comparable and significantly higher than that of the full system.

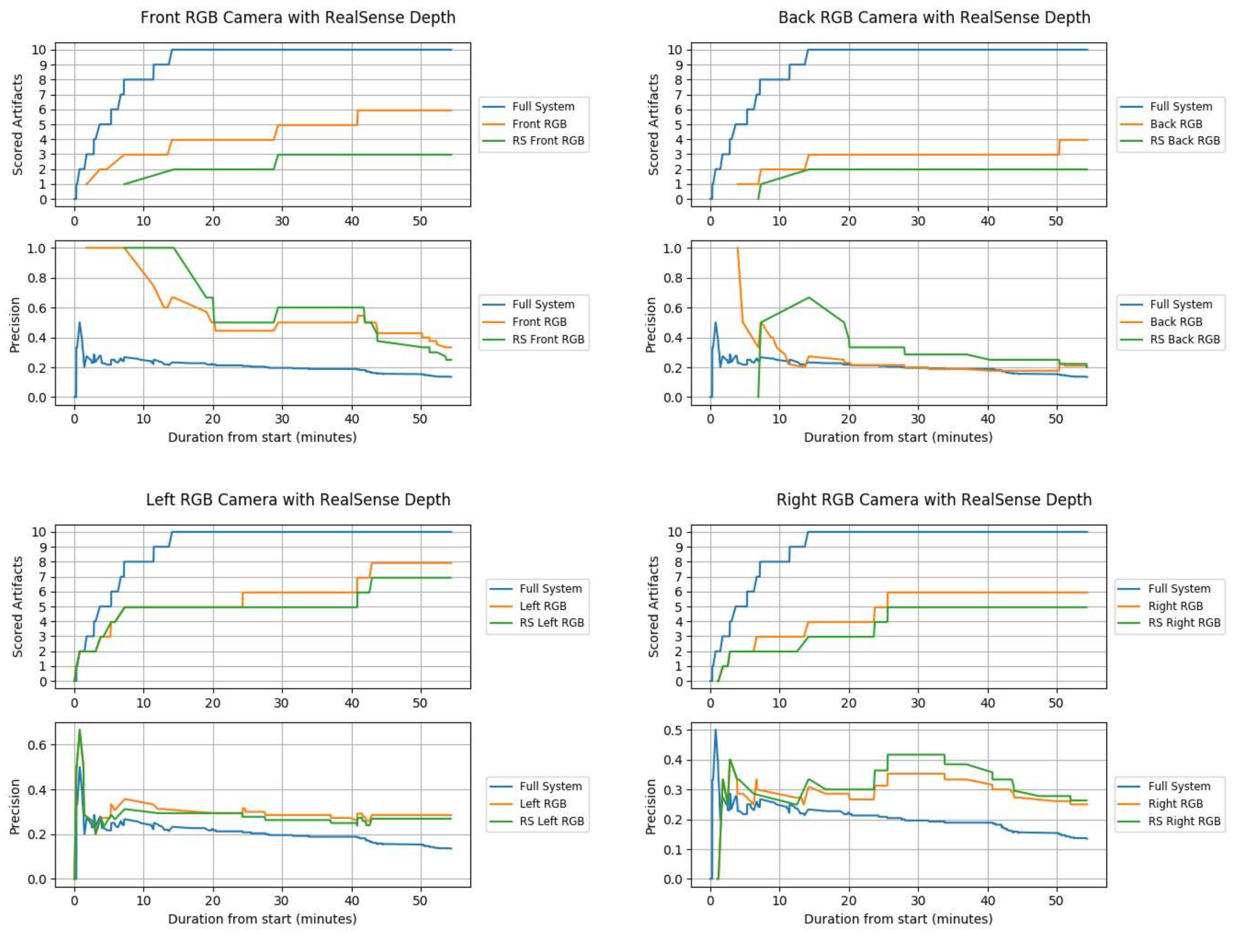


Figure 4.3: Only the RealSense depth information was used to localize artifacts in this test, resulting in an effective detection range of 2.5m for the RGB cameras.

Chapter 5

Conclusions and Future Work

This work comprises the development of three separate payloads and a scalable and flexible software system running on each payload capable of providing rapid situational awareness in the form of Artifact Localizations for the DARPA Subterranean Challenge. Each of the three payloads (drone, Mk. 0, and Mk. 1) contains a variety of sensors and sensing modalities which are fused to provide a single, globally consistent list of artifacts to a human operator at the base station in a robust and timely manner. This simulates information being reported first responders and emergency personnel during a disaster scenario, where rapid and accurate information about the environment is critical to saving lives and mitigating damage.

The complete system was tested during a number of field experiments, culminating in a series of deployments during the Tunnel Circuit of the DARPA Subterranean Challenge. We reported 25 out of 40 artifacts correctly between the two portals (Safety Research, Experimental), and won first place out of the 11 teams at the event. We also won an award for reporting the most accurate artifact, a backpack artifact reported during the second deployment in the Safety Research portal, with an error of 0.18m. Using the data collected at the Tunnel Circuit, we demonstrate the advantage of the fusion of multiple sensors and sensing modalities, which results in more artifacts being found, and being found more quickly, than is possible with any single sensor.

5.1 Future Work

Our results can be improved by addressing a number of problems with the current stack, described below. These changes, and many others, would improve the Artifact Localizations reported by the system and thus the quality of the situational awareness provided to future base station operators, including first responders and emergency personnel. We hope that our work may some day be used to reduce risk to human lives in dangerous situations underground.

Hardware Synchronization

Timestamps for individual sensor measurements are currently not synchronized to a single unified clock. Hardware synchronization for timestamps between all sensors in the payloads would result in a more consistent registration of sensor data to state estimates, eliminating the need for the State Estimate Delay Estimator and removing its inaccuracy. Jitter in timestamps for camera frames sent over USB would also be removed, since the exact time each frame was taken would be known, and would not need to be measured by arrival time. For the current sensor suite, full hardware synchronization can be added by timestamping trigger pulses from the RealSense cameras and PPS pulses from the IMU on the Xavier (or a separate microcontroller for the drone payload) via GPIO interrupts, and feeding PPS and camera trigger pulses into the Velodyne and thermal cameras respectively. Custom driver development is required to perform the timestamping and send the pulses, and to correlate the timestamps with data received over USB. The driver should be robust to frame drops over USB, which may be difficult without control over the firmware of the camera devices.

Artifact 3D Modeling

Coordinates for artifacts generated by the object detection localizer are found by taking the centroid of all points in the point clouds corresponding to each artifact. Even under the assumption that the generated point cloud is perfectly accurate and in the right position globally, this method will have an inherent error of between 10cm and 1m depending on the class of artifact since the point surveyed by DARPA is not exactly the centroid (see Figure 1.2). Fitting 3D models of each artifact to the generated point clouds and using the models to determine the coordinates of the specified localization points could help remove this inherent error.

Reduced False Positives

The majority of the false positive artifacts reported to the base station stem from false positive bounding boxes coming from the object detectors. The detectors currently detect some real artifacts with low confidence, requiring a low threshold to ensure no artifacts are missed. However, this also lets through many false positive detections which get propagated through the pipeline and appear at the base station. Improving the quality of the dataset by having tighter bounding boxes and more background and viewpoint variation should help reduce false positives. Other object detection network architectures could also be explored provided they are able to run efficiently on the Xavier and NUC, such as YOLO [39] or newer versions of MobileNet [22]. Larger object detection networks such as FasterRCNN [40], networks which perform semantic segmentation, such as U-Net [41], or those which perform instance segmentation, such as MaskRCNN [21], may be able to offer more accurate and precise labels at the expense of slower than real time framerates. These labels could be propagated with a tracker to maintain real time framerates.

Cell Phone Trilateration

The current cell phone localization strategy is only able to localize cell phones which the robot directly drives past. This generates correct localizations in the tunnel circuit if the robot traverses the entire tunnel, but will fail if that is not the case. The experiment in Figure 4.2 shows the signal localizer had a final precision of $\frac{1}{3}$, indicating that it detected 3 distinct cell phones (39, 45, 59 in Figure 4.1) but was only able to localize one of them (59). The localization accuracy can be improved by implementing cell phone trilateration utilizing either Bluetooth or WiFi

RSSI values, such as in [24]. The underlying RSSI to distance models work well in open spaces where line of sight is possible, but become more inaccurate as additional layers of structure (such as rock) are introduced. An environment aware algorithm which is able to vary the model parameters based on the surrounding environment parameters is likely to be necessary.

Additional Sensors

The sensor suite on the current payloads was selected specifically to detect artifacts for the Tunnel Circuit. The upcoming Urban Circuit introduces a gas artifact which cannot be detected with the current sensors [4]. To handle the gas artifact, new sensors will need to be integrated into the payloads, and a new software module will be required to localize the gas artifact. The new module would produce Artifact Localization messages and feed into the Artifact Aggregator, just as the current Object Detection Localizer and Signal Localizer modules currently do. Other software modules could also be written to offer alternative ways to detect and localize artifacts with the existing sensor data, such as one which utilizes multi view stereo to localize artifacts detected in images without any depth information, or one which fuses audio information with signal readings to refine cell phone localizations.

5.2 Lessons Learned

A number of lessons were learned during the ground-up development and field testing of the artifact detection and localization system described in this work. While it would be impossible to enumerate everything, some of the more interesting and non-obvious highlights are listed below, in no particular order.

Time synchronization is important (and difficult!) A significant amount of time throughout the year, over multiple attempts and by multiple people was devoted to attempting to achieve proper time synchronization between all of the sensors and computers. Multiple synchronization schemes were attempted, including using GPS, a separate external clock, and the internal clock of the IMU, none of which proved successful in the time allotted. Many bugs were encountered as a result of improper synchronization, and hacks such as the State Estimate Delay Estimator are still needed to account for its absence.

Convincing people to label data is hard It turns out that even the promise of free food is not enough to motivate grad students to label thousands of images for hours on end. Labeling quality also deteriorates as the night drags on – validation of labels is equally, if not more, important to ensure a high quality dataset.

Mines are very cold You would think that temperatures increase when you go underground. That might be true many miles beneath the Earth's surface, but the mines we tested in, even in the spring and summer, required more layers than expected to stay warm. This is also one of the few places it is beneficial to be short, as the shorter members of the team did not hit their heads against the mine ceiling *nearly* as often as the vertically gifted.

Don't let people know you can design PCBs PCB design is not something which is explicitly taught at CMU, which makes the skill difficult to come by and thus in high demand. Once someone has figured out that you can design PCBs, it becomes extraordinarily challenging to be assigned something else. The same can be said for web and GUI development, which is a skill that roboticists may pick up unwillingly and eventually be stuck with.

Frequent testing is crucial Though life in Pittsburgh can have its disadvantages, being next to a multitude of coal mines turns out to be a blessing when building robots for the Tunnel Circuit. This enabled us to perform very frequent (multiple times a week) testing, which helped rapidly identify issues and prepare our field team for the stress of competition. It is also a great way to make friends with and learn from seasoned coal miners!

Bibliography

[1] Defense Advanced Research Projects Agency. Darpa subterranean challenge, 2017. URL `https://www.darpa.mil/news-events/2017-12-21`. 1.1

[2] Defense Advanced Research Projects Agency. Artifacts specification tunnel circuit event, 2019. URL `https://subtchallenge.com/resources/SubT_Tunnel_Artifacts_Specification.pdf`. 1.2, 1.2

[3] Defense Advanced Research Projects Agency. Competition rules tunnel circuit, 2019. URL `https://subtchallenge.com/resources/SubT_Challenge_Tunnel_Rules.pdf`. 1.1, 2.1, 2.2.3

[4] Defense Advanced Research Projects Agency. Artifacts specification tunnel circuit event, 2019. URL `https://subtchallenge.com/resources/SubT_Urban_Artifacts_Specification.pdf`. 5.1

[5] Kshitij Agrawal and Anbumani Subramanian. Enhancing object detection in adverse conditions using thermal imaging. *arXiv preprint arXiv:1909.13551*, 2019. 1.4.4

[6] Aki Takagi John Sweetser Kevin Zhao Tri Khuong Dan Nie John Woodfill Anders Grunnet-Jepsen, Paul Winer. Using the realsense d4xx depth sensors in multi-camera configurations, 2018. URL `https://simplecore.intel.com/realsensehub/wp-content/uploads/sites/63/Multiple_Camera_WhitePaper04.pdf`. 2.5.1

[7] John Woodfill Anders Grunnet-Jepsen, John N. Sweetser. Best-known-methods for tuning intelÂő realsenseâĎć d400 depth cameras for best performance, 2018. URL `https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/BKMs_Tuning_RealSense_D4xx_Cam.pdf`. 2.5.1

[8] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo De La Escalera. Birdnet: a 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3517–3523. IEEE, 2018. 1.4.4

[9] Amanda Berg. *Detection and Tracking in Thermal Infrared Imagery*. PhD thesis, Linköping University Electronic Press, 2016. 1.4.4

[10] DARPAtv. Darpa subterranean challenge: Why it matters, 2018. URL `https://www.youtube.com/watch?v=4I4J67jxODE`. 1

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1.1, 2.2.1

[12] Chaitanya Devaguptapu, Ninad Akolekar, Manuj M Sharma, and Vineeth N Balasubramanian. Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1.4.4

[13] Emesent. Hovermap, 2019. URL `https://emesent.io/products/hovermap/`. 1.4.2

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 1.4.3

[15] Di Feng, Christian Haase-Schuetz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges, 2019. 1.4.4

[16] David Ferguson, Aaron Morris, Dirk Haehnel, Christopher Baker, Zachary Omohundro, Carlos Reverte, Scott Thayer, Charles Whittaker, William Whittaker, Wolfram Burgard, et al. An autonomous robotic system for mapping abandoned mines. In *Advances in Neural Information Processing Systems*, pages 587–594, 2004. 1.4.1

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1.4.4, 2.2.1, 2.2.1

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1.4.3

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1.4.3

[20] François Grondin and François Michaud. Lightweight and optimized sound source localization and tracking methods for open and closed microphone array configurations. *Robotics and Autonomous Systems*, 113:63–80, 2019. 2.5.5

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 5.1

[22] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019. 5.1

[23] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE confer-*

ence on computer vision and pattern recognition, pages 7310–7311, 2017. 3.2

[24] Héctor José Pérez Iglesias, Valentín Barral, and Carlos J Escudero. Indoor person localization system through rssi bluetooth fingerprinting. In *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 40–43. IEEE, 2012. 3.8, 5.1

[25] KAARTA. High fidelity, rapid, long-range mobile mapping and generation with stencil, 2019. URL `https://www.kaarta.com/products/stencil-2-for-rapid-long-range-mobile-mapping/`. 1.4.2, 2, 2.1

[26] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016. 1.4.4

[27] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019. 1.4.4

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1.4.3, 2.2.1

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1.4.3

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1.4.3

[31] Alan Lukezic, Tomas Vojir, Luka ËĞCehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017. 3.2.2

[32] Aaron Morris, Dave Ferguson, Zachary Omohundro, David Bradley, David Silver, Chris Baker, Scott Thayer, Chuck Whittaker, and William Whittaker. Recent developments in subterranean robotics. *Journal of Field Robotics*, 23(1):35–57, 2006. 1.4.1

[33] Robin R Murphy, Jeffery Kravitz, Samuel L Stover, and Rahmat Shoureshi. Mobile robots in mine rescue and recovery. *IEEE Robotics & Automation Magazine*, 16(2):91–103, 2009. 1.4.1

[34] Jing Nan and Lei Bo. Infrared object image instance segmentation based on improved mask-rcnn. In *Optoelectronic Imaging and Multimedia Technology VI*, volume 11187, page 111871E. International Society for Optics and Photonics, 2019. 1.4.4

[35] OpenCV. Repository for opencv's extra modules, 2019. URL `https://github.com/opencv/opencv_contrib`. 3.2.2

[36] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 1.4.4

[37] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob

Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009. 2.2.4, 3.1

[38] Jonathon C Ralston and David W Hainsworth. The numbat: A remotely controlled mine emergency response vehicle. In *Field and Service Robotics*, pages 53–59. Springer, 1998. 1.4.1

[39] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1.4.3, 3.2, 5.1

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1.4.3, 5.1

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5.1

[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1.4.3

[43] Jr. Stephen G. Sawyer. Potential use cases for robotics in mine incidents, 2019. URL `https://youtu.be/rMW-wZTecqE?t=9782`. 1

[44] Weidong Wang, Wei Dong, Yanyu Su, Dongmei Wu, and Zhijiang Du. Development of search-and-rescue robots for underground coal mine applications. *Journal of Field Robotics*, 31(3):386–407, 2014. 1.4.1

[45] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014. 2.2.1, **??**, 3.1

[46] Robert Zlot and Michael Bosse. Three-dimensional mobile mapping of caves. *Journal of Cave & Karst Studies*, 76(3), 2014. 1.4.2