UAV Search & Report
CSE 145/237 Final Report

STANISŁAW STRZELECKI, University of California San Diego, USA
SUBASH KATEL, University of California San Diego, USA
ASHWIN ROHIT ALAGIRI RAJAN, University of California San Diego, USA
EMMANUEL ANAYA GONZÁLEZ, University of California San Diego, USA

Despite their low barrier of entry and versatility, Unmanned Aerial Vehicles (UAVs) still have not seen widespread adoption in some critical applications, such as Search and Rescue (SAR) missions in challenging environments. With this in mind, the primary goal of our team was to develop a proof-of-concept UAV system that can autonomously survey a designated area and accurately detect and report the locations of target objects, using off-the-shelf components and existing software frameworks. Throughout the quarter, we successfully implemented a prototype for this autonomous SAR task, which leverages state-of-the-art computer vision and path planning algorithms to enable robust object detection, localization, and spatial mapping capabilities. Our prototype system can serve as a foundation that our sponsors, TritonAI, can easily extend into a complete, competition-ready solution for the upcoming RobotX Maritime Challenge, showcasing the potential of autonomous UAVs for critical SAR operations in real-world maritime environments. By participating in RobotX, our team aims to push the boundaries of autonomous multi-domain vehicle technologies while gaining invaluable experience in system integration, deployment, and evaluation under realistic contraints - paving the way for future SAR innovations that can save lives.

1 Introduction

Fig. 1. Our completed SAR UAV prototype

The field of Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, has experienced remarkable advancements in recent years, fueled by rapidly declining costs and increased accessibility. While initially perceived as a hobbyist pursuit, the potential applications of UAVs in critical domains such as search and rescue (SAR) operations have gathered significant attention. However, despite their inherent advantages, the adoption of UAVs in SAR efforts has been relatively slow, primarily due to technological limitations and regulatory concerns.

Authors' Contact Information: Stanisław Strzelecki, University of California San Diego, La Jolla, USA, ; Subash Katel, University of California San Diego, La Jolla, USA, ; Ashwin Rohit Alagiri Rajan, University of California San Diego, La Jolla, USA, ; Emmanuel Anaya González, University of California San Diego, La Jolla, USA, fanayagonzalez@ucsd.edu.

Search and rescue operations, particularly in challenging environments such as mountainous regions or vast oceans, pose immense risks to human personnel. Treacherous terrain, unpredictable weather conditions, and the urgency of locating missing individuals or survivors compound the challenges faced by rescue teams. UAVs present a compelling solution, offering a safer and more efficient means of conducting aerial surveys, locating victims, and delivering essential supplies to remote or hazardous areas.

Recognizing the immense potential of UAV technology in SAR scenarios, the RobotX Challenge, a prestigious student competition, has incorporated a Search and Report task as one of its key components. This task simulates a real-world SAR operation, where the contestant UAV must autonomously search a designated area, identify and locate two predetermined objects, and report their precise locations to ground control. The successful execution of this task requires a synergistic integration of various cutting-edge technologies, including computer vision, sensor fusion, and robust object detection and localization.

Our team, in collaboration with the Triton AI lab, has embarked on an ambitious endeavor to develop a UAV system capable of excelling in the RobotX Challenge's Search and Report task. Through meticulous research, innovative engineering, and rigorous testing, we have made significant strides in addressing the multifaceted challenges associated with this undertaking. The key contributions of our work can be summarized as follows:

(1) Development of a highly optimized object detection and localization pipeline tailored for real-time performance on embedded systems, leveraging state-of-the-art computer vision techniques.
(2) Integration of advanced computer vision algorithms to accurately determine the UAV's position and orientation, enabling precise object localization within the designated search area through effective sensor fusion.
(3) Implementation of efficient autonomous navigation strategies to ensure comprehensive area coverage and optimal resource utilization during the search operation.
(4) Establishment of a reliable communication framework for seamless data exchange between the UAV, and companion computer systems.

Our task, as presented in this report, is to build a prototype using off-the-shelf components that demonstrates effective strategies for approaching the Search and Report challenge, enabling the TritonAI team to adapt and scale our solutions to their larger drone for the competition.

By addressing these critical aspects, our work aims to push the boundaries of UAV technology in SAR applications, paving the way for more effective and safer rescue operations. The successful demonstration of our system's capabilities in the RobotX Challenge will not only showcase our team's technical prowess but also the potential of UAVs as invaluable tools in saving lives and mitigating the risks faced by human personnel in high-stakes rescue scenarios.

2 Related Work

In the 2022 RobotX competition [8], TritonAI's drone team participated in the competition for the same Search and Report task and for the 2024 competition we chose to take things in a different route.

We took inspiration from the 2022 competition winners Embry Riddle's pipeline architecture [2] for a few improvements to our pipeline task, specifically pertaining to the pipeline ordering.

We looked for a modular, off the shelf drone and we found the S500 [7]. The S500 allowed for quick prototyping and iterating. We built the S500 using DroneCode's [5] guide since their compute module needs aligned with ours.

For the 2022 version the drone team elected to implement a camera on the drone but offload the processing to later [1]. We chose to build in a Jetson Nano compute module for live data acquisition and processing. This is both something we took inspiration from the 2022 winner team and something we found to better fit our tracking goals.

For image capturing and processing we implemented 2 distinct scripts: one leveraging NVIDIA's nvgstcapture tool [4] for offline camera module testing, and another utilizing OpenCV for real-time image capture within the processing pipeline [6]. These scripts facilitated robust image processing capabilities essential for the task.

The control system was based on the Ardupilot software running on a Pixhawk module, configured to emulate the Betaflight rotor setup. Communication with the onboard systems was split into two distinct methods: (1) SSHFS for interfacing with the Jetson Module. The SSHFS was chosen because of interoperability with macOS and windows and live update to the system [10] and (2) Radio control for flight controller.

The previous team had utilized a dual-camera setup with YoloV5 for object detection[1]. In addition to that, we researched current object detection status quo and many include Xin Wu et. all. [21] seemed to suggest Yolo models so decided to try the latest Yolo model for our object detection tasks. However, the Jetson Module proved incapable of processing the data in real time so we elected to use CV algorithms based for quick processing. We analyzed different approaches to this problem and found three suitable algorithms – Scale Invariant Feature Transform (SIFT) [15], Speeded Up Robust Features (SURF) [9] and Oriented FAST and rotated BRIEF (ORB) [16]. After comparing the accuracy and efficiency of the aforementioned algorithms, we decided to proceed using SIFT.

Additionally, we had to enhance the accuracy of our detection, as we needed precise corner coordinates in the image to get a sufficient accuracy in the localization part. We used the Multiple View Geometry in Computer Vision [14] book to choose the most suitable methods for our problem. We decided to use Optimal Corners algorithm [17], enhanced with subpixel accuracy [20] for our corner detection, as well as OpenCV 'findContours' algorithm [18]. We completed those with our authored algorithms, like one to match detected polygons with detected corners.

For object localization, we researched various algorithms, including Single/Two/n-View-Geometry Algorithms from the aforementioned book [14], but also considered using Simultaneous Localization and Mapping algorithm [12]. We settled on and used the Perspective-n-Point (PnP) algorithm, using the IPPE-SQUARE method [11], which, by processing camera pose and pixel coordinates, efficiently generated camera-relative coordinates of targets.

To enhance data reliability, we adopted the Random Sample Consensus (RANSAC) algorithm [13]. RANSAC enables us to remove outlier data and this is applied in addition to the internal RANSAC used in OpenCV's homography computation function [3]. We used these to clean up the signal, which wasn't necessary for the previous teams as they employed machine learning models that ran offline.

In addition to the tasks we've already completed, we are exploring various SLAM algorithms to adapt the drone to "kidnapped robot" scenarios. Preliminary research indicate that EKF-SLAM may be the most promising approach for our needs [19].

3 Technical Details

3.1 Competition Spec

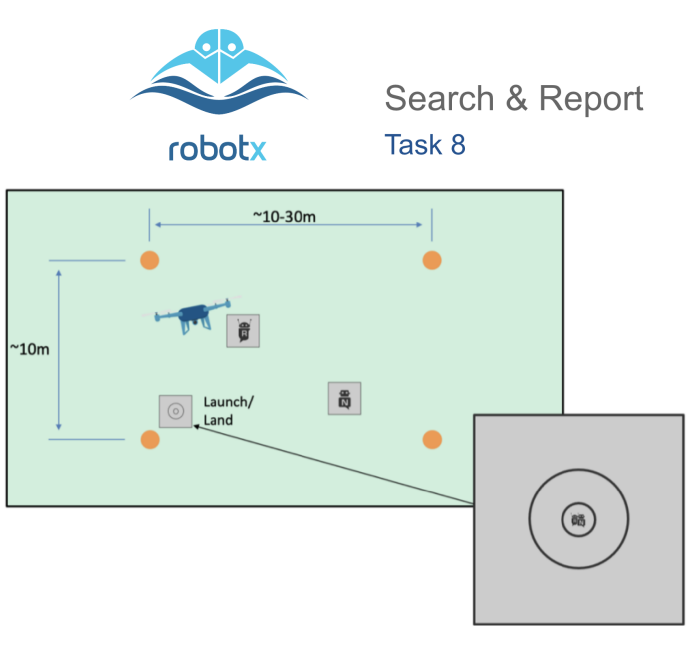

Fig. 2. Challenge Task

The RobotX Maritime Challenge is a biennial competition aimed at expanding the community of researchers and innovators working on autonomous and unmanned multi-domain vehicles. Teams from around the world design, build, and program an Autonomous Maritime System (AMS) to attempt various challenges that test capabilities across the maritime and aerial domains. The competition requires teams to demonstrate their AMS's full autonomy, situational awareness, and ability to seamlessly operate as an integrated multi-domain system.

A key technical focus area is autonomous deployment and control of unmanned aerial vehicles (UAVs) from the unmanned surface vessel (USV). This showcases advancements in areas like UAV launch and recovery, autonomous aerial navigation, object detection/tracking, and coordination between maritime and aerial systems. Specific tasks include the UAV Replenishment challenge where the UAV must locate a floating helipad, collect a colored object, deliver it to another helipad, and return to the USV.

The UAV Search and Report task specifically evaluates Search and Rescue (SAR) capabilities crucial for maritime operations. The UAV must autonomously launch, conduct an aerial search over a

designated field area, detect and determine the precise locations of two distinct objects simulating SAR targets, and accurately report those locations back to the judges. Successfully completing this showcases integrated maritime/aerial autonomous systems with robust perception, navigation, and situational awareness for time-critical SAR scenarios. Teams must pass all mandatory inspections and demos before attempting this capstone SAR task in the competition rounds.

3.2 UAV Assembly

Building the physical drone was a challenging yet rewarding process. We started by carefully following the instructions provided by the manufacturer to assemble the frame and mount the various components. Particular attention was given to ensuring a secure and stable placement of the critical parts like the flight controller, motors, and electronic speed controllers.

One area that required some extra effort was the soldering of connections. With multiple wires and delicate circuits involved, we had to exercise caution and precision to avoid any short circuits or loose joints. Emmanuel's prior experience with soldering came in handy here, and he guided the rest of us through this intricate task. Once the core assembly was complete, we methodically integrated the remaining peripherals such as the telemetry module, GPS, and camera. Proper cable management and routing were crucial to prevent any interference or disconnections during flight.

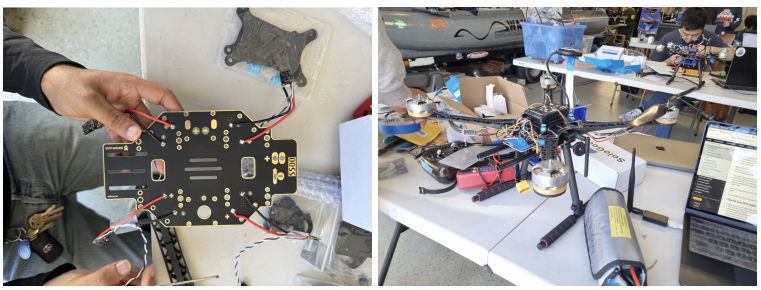

Fig. 3. First stage of drone build

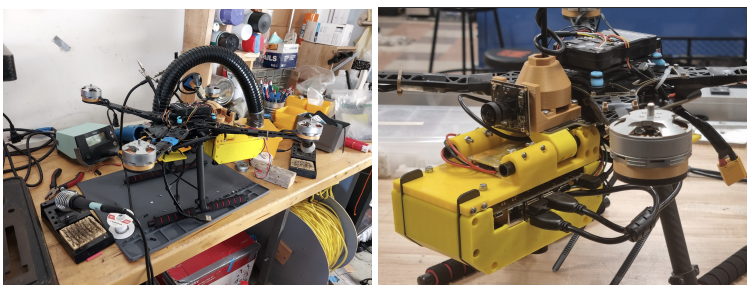

Fig. 4. Second stage of drone build

Throughout the assembly process, we consulted the expertise of the TritonAI team and incorporated their valuable feedback. Their recommendations on hardware selection and configuration were invaluable in ensuring a robust and airworthy drone.

3.3 Hardware

At the heart of our UAV system lies the Pixhawk flight controller, a powerful and versatile autopilot module. Pixhawk not only governs the drone's movement and stability but also serves as a central hub for gathering data from various onboard sensors.

The Inertial Measurement Unit (IMU) plays a pivotal role in tracking the drone's orientation and movement. By continuously monitoring acceleration and rotational rates, the IMU provides crucial inputs for maintaining stable flight and accurate navigation. For precise location tracking, we rely on the Global Positioning System (GPS) module. This module receives signals from orbiting satellites, enabling us to pinpoint the drone's coordinates with a high degree of accuracy – a critical requirement for our object localization task.

Capturing visual data is the responsibility of the wide-angle camera mounted on the drone. This camera feeds real-time imagery to our object detection and localization algorithms running on the Jetson Nano companion computer. The Jetson Nano itself is a remarkable piece of hardware, packing significant computing power into a compact and energy-efficient form factor. Its also has parallel processing capabilities which will allow us to run our computationally intensive computer vision algorithms with minimal latency, ensuring smooth and responsive operation.

Establishing reliable communication between the various components was a challenging process. We leveraged the MAVLink protocol to enable bidirectional data exchange between the Pixhawk and the Jetson Nano. This link not only allowed us to retrieve telemetry data but also provided a channel for sending navigation data. With this robust hardware infrastructure in place, our UAV system is well-equipped to undertake the demanding tasks of autonomous object detection, localization, and ultimately, participation in the RobotX Challenge.

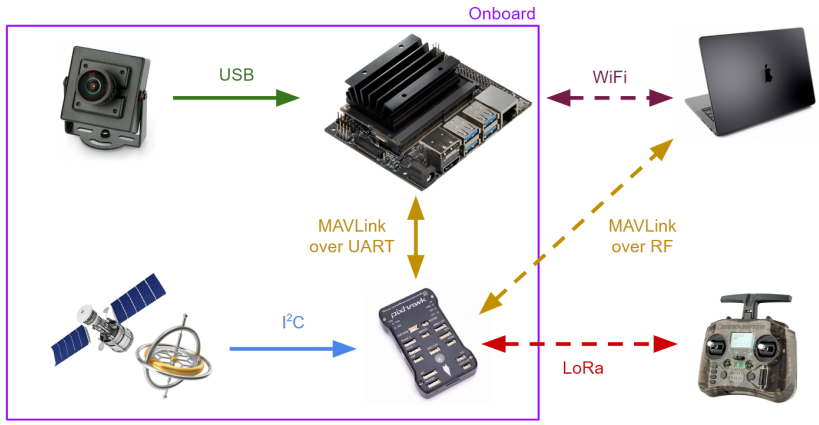

Fig. 5. Hardware communication model

3.4 Software

The goal is to periodically capture the readings from the onboard sensors, perform object detection to find the targets, estimate their real-world coordinates, and then finally report them to the ground station.

Capturing Environment

The main sensor sources that we use are the onboard cameras and the sensor readings from the autopilot module, in particular the GPS, IMU and orientation information. In order to achieve low-latency, we keep maintain a separate thread for both the camera and the autopilot and save the latest reading to memory. We make sure that the camera and autopilot reading happen ideally at the same time for better accuracy. We save this information for later use and the captured frame is then sent to the target detection section.

Target Detection

When our pipeline starts, we first load the target reference images for target detection. This allows us to load them once and keep them in memory which gives us a massive performance boost. Our target detection CV algorithm is an image matching algorithm, broken down into two steps:

a) Feature Extraction
b) Feature Matching

For feature extraction, we use the SIFT algorithm. We tested SURF, SIFT, and ORB algorithms. We found out that SIFT offers a nice balance of ease of use, speed, and transformational invariance. For improved efficiency, we extract features only once per reference image and per camera image, since it's a computationally expensive operation. We then reuse those features for every reference-camera image pair.

For feature matching, we use KNN, making sure that the best match is significantly better than the second-best match. This helps us remove false positives, which were a big source of confusion between the two (pretty similar) reference images. Our previous method was based on a different matching algorithm which we had to abandon specifically for this reason.

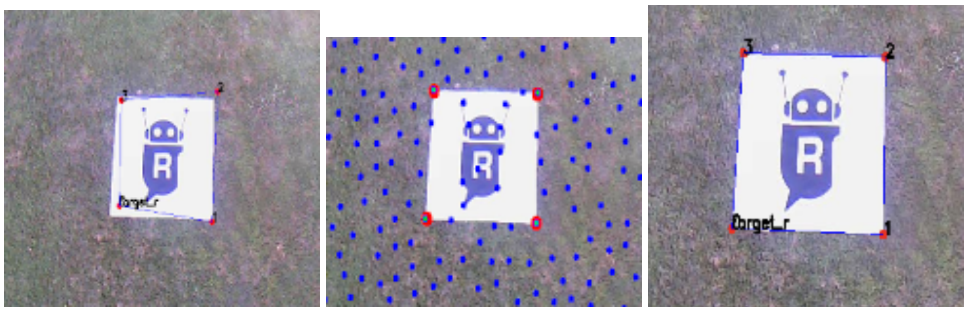

Fig. 6. Steps of object localization

Once the features were extracted and the objects were detected, the target detector would return the image coordinates of the detected targets' corners. However, these results are still noisy, so the next step is to detect potential target outlines in the image, by finding the contours in the image and limiting them to realistically sized quadrilaterals. We then match every detected target to the closest polygon in the image, fixing any inaccuracies in the corner localization. Then, we detect

corners with a subpixel accuracy to even more enhance our precision.

This data, having a near-perfect precision, is then sent to the localization pipeline.

Target Localisation

Using the drone pose from the IMU, the GPS coordinates of the images, the GPS coordinates of the boundary, and the image coordinates of the detected images, we're able to localize the actual GPS coordinates of the targets using 3D view reconstruction.

We use the Perspective-n-Point algorithm to get the camera position relative to the target 3D position (since we know the target dimensions, we set the target in the origin of the 3D space). Then, knowing the real camera pose, we calculate the real 3D pose of the target (specifically, its center).

We then send this information to our aggregation module.

Data Aggregation and Reporting

We collect a lot of data points, but we still have to average them. Since not all the readings are correct (sometimes we detect or localize targets incorrectly), we have to remove the outliers. We use the RANSAC algorithm, assuming a reasonable error for our readings and removing anything above that (as since all the false detections appear in random places, they get localized very far away).

Fig. 7. Steps of results aggregation

After removing the outliers, we can average the results to get a precise estimate of the real-world location. We can then finally send it to the base.

4 Milestones

4.1 Hardware

Drone Assembly

The initial milestone was to assemble the drone, which we completed successfully. We installed, connected and secured all components, including the fight controller, motors, and other electronics. We also organized the cables really well to prevent any potential interference or disconnection during flight. This was foundational step as it ensured that the drone was in really good structural condition and ready for further integration. One of the crucial part during drone assembly was

soldering connections given the delicate nature of circuits we had to make sure we connected everything properly and securely. Emmanuel's prior experience with soldering came in handy during this phase. As his guidance helped us ensure that all the connection were robust, minimizing any risk of short circuits or loose joints, making the drone more reliable during operating it. Due to the critical nature of this part it really made this process time consuming and led to multiple retries to achieve really good connections.

What we presumed to take us a day, took us four weeks to complete, proving that assembling a working drone even under the support of experts isn't an easy task. We were promised a working drone, so having to assemble one put us strongly off track, and therefore, we had to give up on some goals we wanted to achieve, such as autonomous flight (in case of any serious crash, we would no more have a drone to work on) and border detection.

During our testing, we had a minor crash, but we worked around that. One of our wings is still temporarily fixed, proving how good the drone is, as even with such a flaw, it is flying very well. This incident showcases the robustness and reliability of our drone design and the effectiveness of our assembly process.

Another issue we faced after our minor crash was that the battery started to die halfway through our testing sessions, forcing us to work with the other team's battery. We are really grateful for their help and support, which allowed us to continue testing without major interruptions. This experience highlights the importance of collaboration and mutual support within the robotics community.

Camera & Companion Computer Integration

We mounted a wide-angle camera on the drone to capture visual data. This camera played a key role in our project as it fed real-time image data to our object detection and localization algorithms running on the Jetson Nano. Ensuring that the camera was securely installed and didn't have any unnecessary movement was a crucial step towards enabling accurate data capture.

Integrating the Jetson Nano companion computer with the drone was another significant milestone. We successfully connected the Jetson Nano to the drone, enabling seamless communication between the two systems. The Jetson Nano can now gather essential data from the drone, such as GPS coordinates and IMU readings, powering to make error free decisions.

Putting it all together:

(1) With all the components installed, connected, and tested, we now have a fully functional and flight-ready drone. We can fly the drone using our remote control without any issues, showcasing the success of our assembly process.
(2) The successful integration of the Jetson Nano with the drone enables bi-directional communication. The Jetson Nano can retrieve crucial data from the drone's sensors, such as GPS coordinates and IMU readings, while also sending commands to control various aspects of the drone's operation.

(3) By establishing control over the camera through the Jetson Nano, we have completed the hardware pipeline necessary for our MVP. We can now capture images during flight, which is essential for gathering the data required to localize objects in the real world.
(4) To ensure that our hardware setup meets the requirements and standards set by the TritonAI team, we have confirmed the final build with their mentors. This validation from experts in the field gives us confidence in the robustness and reliability of our drone platform.

These hardware milestones show the progress we've made in terms of assembling a fully functional drone, integrating the Jetson Nano companion computer, enabling camera control, and gathering crucial data for object detection and localization. The collaborative efforts of our team, along with the valuable input from the TritonAI mentors, have been really helpful in achieving these milestones.

4.2 Software

Hardware support

We fully completed all milestones related to hardware support. We have software that can communicate with hardware in all matters, starting with reading the GPS values and ending with capturing images with the camera. Also all our software is running on the compation computer (Jetson Nano) without any issues

Object detection

Object detection was a tricky part, as it was tempting to overkill the problems with a big powerful ML model as many other teams did. We also initially thought of doing that, so we adapted a simple Yolo model to detect objects. We started with detecting some pretrained objects from real life and quickly ran into a problem where running this algorithm on Jetson Nano would be slow. Our computing unit would freeze for around 5 seconds to perform detection on a single photo. As we wanted to compute way more than a single detection and needed a decent efficiency, we started looking for alternative solution. We looked at computer vision and found three potential algorithms to use – SIFT, SURF and ORB. Out of these free, SIFT became our choice, as a very fast, robust and rotation invariant algorithm – exactly what we needed. ORB calculated results pretty fast as well, however its results were hard to interpret in the way we needed to, however SURF would have licensing problems if we ever wanted to use it for more serious purposes. And this way we ended with SIFT as our main object detection algorithm.

However SIFT, while detecting the target images with pretty good accuracy, did not know how to place the target corners properly in the image – the thing we needed the most. Therefore it became clear that we have to detect corners separately, and then match the corners closest to the SIFT detection as the output ones. It turned out that detecting corners in the image is way more difficult that one might have imagined. Simple corner detection algorithms were very noise prone and worked with different efficiency between different photos, or even different parts of the same photo. Finally, our best result was finding optimal features for corner matching, which is an algorithm that for each window of a given size selects the best corner in it, allowing for a fair distribution of the corners throughout the picture. Unfortunately, while yielding decent results, this algorithm would still miss a corner in place of some noisy bit of grass for around 20% of targets, so we needed

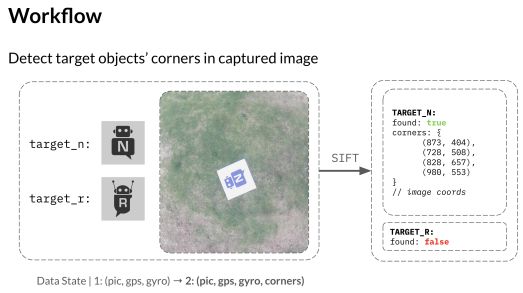

Fig. 8. SIFT Workflow

an even better solution. We found an OpenCV algorithm for contour detection, that would help us detect polygons. This algorithm is way less prone to noise, as there are very few reasonable polygons in the image. The only problem with this algorithm is that it requires image thresholding, which can stop working for different lighting conditions, so requires tuning to the background and sun level. However, it turned out to be working very well for most of our test flights. Our final solution finds optimal corners as well as polygons. It is possible to turn off the polygon part, which makes the results worse but works no matter what backgorund and lighting condition we are in. It uses optimal corners if any matches the polygon corner (up to some error), otherwise assume that the polygon corner (which is less accurate then the optimal corner) is the ground truth. Finally, it calculates the corners up to sub-pixel accuracy to just slightly even more push the precision. This way we get very accurate target corners in the image.

Last part worth noting is the efficiency. Some of the used algorithms, such as key feature detection and contour detection, are pretty slow. Therefore, we run those once for every photo, and reuse for every target searched. Additionally, for target images, we load and preprocess them at the beginning of the pipeline. This way we can calculate their features very accurately and not worry about the time loss. Combining all these optimization gives us the speed of one image per 4 seconds, which is a pretty good result, taking into the account how small Jetson Nano is and that we did not install OpenCV with cuda.

Localization

For localization, we knew that we have the exact drone pose and the exact target corners in the image, however coming with a way of getting the exact target location wasn't easy. We were initially thinking of using two view geometry, but this felt like an overkill, taking into the account that we knew the drone pose and exact target shape. Then, we got inspired, that actually single view geometry is enough to solve the problem, since we know the exact target dimensions. Our solution was to place the object in 3D in the origin and using Perspective-n-Point algorithm, calculate the relative position of the camera. This algorithm needs 3 points to find up to 4 solutions, with 4 points guaranteeing a unique solution. This way, knowing the transformation from the drone to the target, and the drones pose, we can calculate the target's real-world position.

An important part was that all the algorithms operate on meters, however any coordinates are in angles. This required a conversion algorithm, which, while not being difficult, took some time to test

and tune to a required accuracy. Thanks to this tool, we were able to retrieve the results in coordinates, which enabled us to test our algorithm on real-world data gathered throughout our test flights.

The main difficulty in this part was that there is very little good documentation for the Perspective-n-Point algorithm implemented in OpenCV. Specifically, the axis were very problematic, as OpenCV sometimes (but not always) uses the Y axis pointing down instead of up. With this in mind, it was difficult to interpret the results of the algorithm and took a bit of trial-and-error to find out the correct transformation algorithm. But finally, after hours of debugging, we succeeded with the transformation getting errors of up to 1 meter, a satisfactory result!

Aggregation

In the beginning, we did not even assume this would be a separate part. However, it quickly became clear that not only we will have errors which have to get averaged out, but also achieving a 100% of successful and correct image detections is almost impossible (and is so, would not detect many less certain but still valuable targets locations in the images). Therefore, we decided to accept more detections with a risk of false matches. To deal with this, we added an outlier rejection algorithm – RANSAC. This ensured, that any serious false detections will be immediately discarded. After applying that, we could safely average the rest of the results not worrying about singular very incorrect detection visibly affecting our prediction. Thanks to that, we could leave our object detection algorithm with a sub-100% accuracy and be calm that any mistakes made will not impact out performance.

4.3 Abandoned

(1) One of the efficiency improvement we wanted to make was reinstalling OpenCV with CUDA – making a better use of the GPU in our companion computer. However efficiency was not our priority, especially hardware efficiency (since TritonAI would use a different drone for the competition anyway). Because of that, we kept postponing this task as we had more important things to work on, finally deciding to not do it at all. While it made us spend more time for field testing (since we needed to fly the drone longer to get a similar amount of photos), we believe investing the time we would spend reinstalling the libraries and fixing CUDA problems into other tasks was a wise decision overall.
(2) We decided to abandon the autonomous drone flight. It is a complex part of the drone system, which is fully managed by another TritonAI team. Our algorithm does not require autonomous flight to work properly, but just a simple coverage algorithm, which can be easily implemented by anyone. An important sidenote: We did implement the automation routines for the control aspect of the drone, meaning we're able to give commands such as "hold altitude", "move forward", etc. The only autonomous task that we haven't implemented is the autonomous navigation task. This is being focused by another TritonAI team, so we elected to focus on making our detection and localisation far more robust that asked for.
(3) Communication with the base (Autonomous Maritime System) AMS system using radio was a task that we initially had in mind as a stretch target, but we abandoned later. While trying to implement the radio controls for the drone we discovered that the test radio platform we had would differ significantly from the actual platform that would be present in the competition. We felt that it would be a fruitless task to implement this for the sake of testing. Since the AMS' communication routines are not different from the previous system, the TritonAI team could just plug that into the current system with little modification.

5 Experiments

5.1 Field Tests

Fig. 9. Field Testing

We performed our tests in two flight sessions. We performed both sessions by first measuring the ground truths (taking 10 gps readings with the drone standing on each target and averaging them), and then performing a flight with full pipeline activated. We then analyzed the saved results (potentially rerunning the pipeline locally to polish our algorithms). Since all the images and readings were saved locally, we could use those to rerun the pipeline without the drone, therefore being able to gather detailed statistics (which we will present later) that would take computational time or were only possible to gather manually. During each flight, we took around 100-150 photos, gathering around 40-50 inlier measurements of each target. We performed both sessions in different parks and in different lighting conditions, achieving similar results, proving that our algorithm is not tuned to a very specific environment, but generalizes well. Additional tests in water environment (so in the competition setting) are to be made, but they will be concluded with the competition drone by the TritonAI team.

5.2 Results

For hardware results, we managed to achieve a fully functional flying drone. The flight was very stable, requiring little to no flight experience to pilot the drone.

We were able to get sensor reading in a speed of 30FPS, a result sufficient enough for any further development. Our images are in a high (1960x1080) resolution and are not blurred even if the drone is in flight.

Our object detection is very good, 90% or targets in the images. Most of the missed detections are due to the target being very close to the image edge (and thus very distorted), sun glare on the target or too high flight. Only 7% of all detections are false detections, most of them eliminated during the image localization phase (as PnP algorithm claims that such a point configuration is impossible), and the rest during the outlier rejection. Less than 3% if images are detected correctly, but their corners are not placed correctly. This is the only really harmful case, but since it appears so rarely, it can get easily averaged out.

Overall, we get a 1-meter accuracy when detecting locations, which is a result of taking 50 photos. It's worth noting that the GPS has a 1 meter error and IMU also has errors which we didn't measure.

Therefore, we believe that improving the efficiency and therefore taking more images of the targets can dramatically improve the accuracy (for comparison, taking only 20 images pushed the error up to 2-3 meters. Also, with more precise instruments in the competition drone, the results can get even better. However, for the prototype we were meant to build, we are happy with this result.

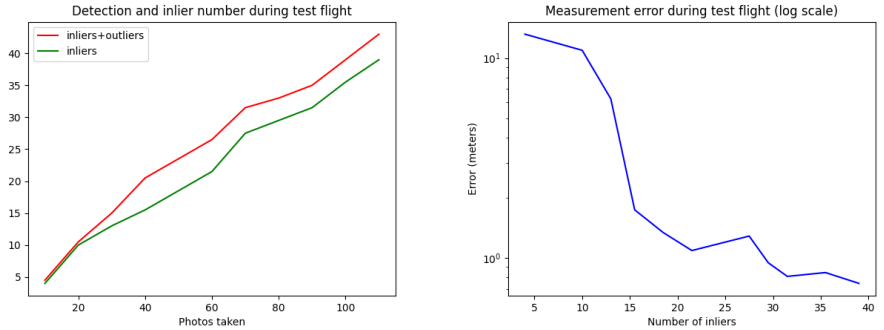

Fig. 10. Results

5.3 Future work

Here we cover next steps we would take as a team to even further improve our solution

(1) **Better localization** – We used Single-View Geometry to solve the localization problem. This completely ignores the fact that we not only have multiple measurements, but we know our position changes between them. Assuming only multiple measurements, we can use n-View Geometry to more precisely localize the targets in the real-world, since this approach uses a more robust algorithm that minimizes a specialized error, this way achieving better accuracy then averaging a lot of individual measurements. However, since we also know how we change the position between measurements, we can use Simultaneous Localization and Mapping (SLAM) algorithm to achieve even better results. SLAM will not only take all the measurements into the account when localizing the targets, but also is build to help mitigate GPS and IMU measurement errors, this way achieving a great way to get a very precise measurement that by design handles measurement errors, not only errors in our algorithms, but also errors in drone positioning and movement.

(2) **ROS Node** – While we communicate with our drone through MAVROS, it isn't a very scalable solution. When running a few algorithms like our (for example when working on few tasks during the competition), our latency will increase and we will have problems with parallelization. A great solution for that is using Robot Operating System (ROS) – converting our algorithm into a ROS node. This will make our algorithm run in parallel, easily access all the necessary readings in real-time and not interfere with other potential algorithms running on the drone.

(3) **Coverage algorithm** – While we did not implement autonomous flight, it is an important thing that should be addressed sooner than later. Specifically, we need our drone to cover a designated area, so a suitable coverage algorithm is needed. As a step further, an adaptive algorithm, that after detecting targets in given locations focuses on nearby area to gather more inlier detections and get a better estimate, rather than flying above area that we know contains no targets. As for the coverage algorithm itself, it can be further improved by flying high with small detection accuracy and then further investigating only smaller areas

of interest. Depending on competition specs different measures of efficiency and accuracy should be implemented in order to give our object detection and localization algorithms as much data to work with as possible.

(4) **General object detection** – For now we can only detect square images with characteristic drawings on them. It would be great to detect and localize arbitrary objects. The first step is detecting orange markers showing the borders of the search zone. This task, while an intermediate step between our current work and general detection, is important to detect the search area in our task and therefore make the coverage more efficient. While ML object detection algorithms should work, the localization part can get tricky as our algorithms assume the targets are squares. A small redesign should be made, preferably placing simplicity over accuracy, as we don't need precise locations of the orange markers, just a general estimate.

6 Conclusion

In summary, we built a prototype drone to detect and localize target objects. The drone is constructed from off-the-shelf and 3D-printed components in an affordable manner, and it utilizes GPS, IMU, and a down-facing camera as sensors to complete its tasks. We use a combination of computer vision algorithms to first detect the targets, then precisely place them in the camera image, and finally localize them in the real world by calculating their precise GPS coordinates.

Our work can now be utilized by the TritonAI team, with our prototype drone being fully functional and ready for further field tests. The drone can be used for this specific competition task and potentially other tasks as well. Moreover, our algorithm can be migrated to the competition drone and be run to complete the Search and Report task. With our algorithm demonstrating high reliability and a precision of 1 meter, it can be used as-is to achieve a high score in the competition, supporting our team in their pursuit of victory.

The next steps, as discussed earlier, would involve improving the object localization accuracy and integrating the algorithm into the entire codebase ecosystem. This will further enhance the performance and seamless integration of our solution with the overall system.

Fig. 11. Team Picture

Acknowledgments

Thanks to TritonAI for all the help provided.

References

[1] 2023. ECE191 Team B Pipeline Instruction. https://github.com/ece191-team-b/.github/blob/main/profile/pipeline_instruction.md. Accessed: 2023-04-14.
[2] 2023. Embry-Riddle Aeronautical University RobotX Team. https://www.eraurobotx.org/. Accessed: 2023-04-14.
[3] 2023. Features2D + Homography to find a known object - OpenCV Tutorial. https://docs.opencv.org/3.4/d1/de0/tutorial_py_feature_homography.html. Accessed: 2023-04-14.
[4] 2023. First Picture with CSI or USB Camera - NVIDIA Developer Tutorial. https://developer.nvidia.com/embedded/learn/tutorials/first-picture-csi-usb-camera. Accessed: 2023-04-14.
[5] 2023. Holybro S500 V2 Pixhawk4 Tuning - PX4 Documentation. https://docs.px4.io/main/en/frames_multicopter/holybro_s500_v2_pixhawk4.html. Accessed: 2023-04-14.
[6] 2023. How to capture image from webcam on click using OpenCV - Stack Overflow. https://stackoverflow.com/questions/34588464/python-how-to-capture-image-from-webcam-on-click-using-opencv. Accessed: 2023-04-14.
[7] 2023. Reference Frames Holybro S500 - ArduPilot Documentation. https://ardupilot.org/copter/docs/reference-frames-holybro-s500.html. Accessed: 2023-04-14.
[8] 2023. RobotX Challenge 2022. https://robotx.org/programs/robotx-challenge-2022/. Accessed: 2023-04-14.
[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In *Computer Vision – ECCV 2006*, Aleš Leonardis, Horst Bischof, and Axel Pinz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 404–417.
[10] Brismuth. 2022. *Working from Home: The Awesomeness That is SSHFS.* https://brismuth.com/working-from-home-the-awesomeness-that-is-sshfs-22b41019fb20 Accessed: 2023-04-14.
[11] Toby Collins and Adrien Bartoli. 2014. Infinitesimal Plane-Based Pose Estimation. *International Journal of Computer Vision* 109 (2014), 252–286. https://api.semanticscholar.org/CorpusID:8674118
[12] H. Durrant-Whyte and T. Bailey. 2006. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine* 13, 2 (2006), 99–110. https://doi.org/10.1109/MRA.2006.1638022
[13] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (jun 1981), 381–395. https://doi.org/10.1145/358669.358692
[14] Richard Hartley and Andrew Zisserman. 2004. *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge University Press.
[15] Tony Lindeberg. 2012. *Scale Invariant Feature Transform.* Vol. 7. https://doi.org/10.4249/scholarpedia.10491
[16] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*. 2564–2571. https://doi.org/10.1109/ICCV.2011.6126544
[17] Jianbo Shi and Tomasi. 1994. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 593–600. https://doi.org/10.1109/CVPR.1994.323794
[18] Satoshi Suzuki and KeiichiA be. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30, 1 (1985), 32–46. https://doi.org/10.1016/0734-189X(85)90016-7
[19] J. Weingarten and R. Siegwart. 2005. EKF-based 3D SLAM for structured environment reconstruction. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3834–3839. https://doi.org/10.1109/IROS.2005.1545285
[20] Eberhard Gülch Wolfgang Förstner. 1987. A fast operator for detection and precise location of distincs points, corners and center of circular features. *Proc. ISPRS intercommission conference on fast processing of photogrammetric data* (1987). https://scholar.google.com/citations?view_op=view_citation&hl=en&user=7oW2GNYAAAAJ&citation_for_view=7oW2GNYAAAAJ:u5HHmVD_uO8C
[21] Xin Wu, Wei Li, Danfeng Hong, Ran Tao, and Qian Du. 2021. Deep Learning for UAV-based Object Detection and Tracking: A Survey. *arXiv preprint arXiv:2110.12638* (2021). https://arxiv.org/abs/2110.12638 Accessed: 2023-04-14.