Rapid Global Positioning from Monocular Data

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Joshua Max Holtzman

Spring, 2020.

Tomonari Furukawa, Department of Mechanical Engineering

Introduction

This paper presents a strategy for data manipulation to quickly convert complex video data into simple positional data using prior or learned information about a target. The proposed process is intended for drone use to abstract information either for tracking or for submitting to another autonomous system. For that reason the output data set is minimized to the task and the data processing time is such that the process can run on real-time video streams.

A network including autonomous agents, especially those that also include non-autonomous agents, values greatly the ability to represent data minimally. Semi-Autonomous systems must take into account the limited capacity for human pieces to take in large quantities of data quickly. For that reason, being able to simplify data such as locational data from video footage to simplified graphics such as radar-maps is highly valuable. The proposed algorithm provides one such way to simplify video data into more easily digested features such as global position.

Background

Automation requires, first and foremost, data processing. The natural environment is complex and requires an automated system to process and account for unimaginable variables. For this reason, a sizable amount of the work done in the field of automation is in data processing and manipulation. Especially with the onset of automated vehicles, a focus has arisen to more accurately and more precisely process visual data into features that are usable and trainable in automated systems. Many automated systems today use some combination of algorithmic design paired with machine learning. Drones and low-flying unmanned aircraft rely even more heavily on visual data as nonvisual streams such as radar and lidar require considerably more technical resources to provide similar coverage.

MBZIRC is an automation competition that focuses on using aerial (UAV) and ground (UGV) robots, in cooperation and alone, to complete navigation and manipulation tasks. Virginia Tech and the University of Virginia worked cooperatively under the leadership of Professor Tomonari Furukawa to compete in the MBZIRC competition in 2020. The competition was ranked in terms of three challenges and a grand challenge. The challenges, as described by the official website, are as follows:

1. UAVs will autonomously track and interact with a set of objects following 3D trajectories inside the arena.
2. A team of UAVs and a UGV will collaborate to autonomously locate, pick, transport and assemble different types of brick shaped objects to build predefined structures, in an outdoor environment.
3. A team of UAVs and a UGV will collaborate to autonomously extinguish a series of simulated fires in an urban high rise building fire fighting scenario.
4. The Grand Challenge requires a team of robots (UAVs and UGVs) to compete in a triathlon type event that combines Challenges 1, 2 and 3.

The specific procedures employed to complete these tasks is described below.

Objectives

My research was done as a part of the UAV team with the intent of aiding their work to complete challenge 1. To complete challenge 1, the UAV would have to move throughout an arena (100m by 60m) and pop a series of balloons that were randomly distributed throughout the space, in addition, the UAV would have to secure a target being held by a second UAV that would be flying a figure-8 pattern within the arena. The target ball would detach from the UAV carrying it when a force of fewer than four newtons was applied to it. To complete this task the UAV would have to be able to identify spherical objects of a distinct color in space and approximate their distance.

The primary objective of my research was to manipulate incoming visual data streams to be accessible by a drone mid flight. The subgoals for this task included identifying and tracking objects within the frame, gaining heading information based on location within the frame, and deriving positioning data from known aspects of the object paired with apparent size in the video stream. In the context of the MBZIRC competition, the goal of the research was to be able to autonomously locate and track an object moving through the air via drone. Routing and movement controls for the drone were abstracted upstream in the workflow.

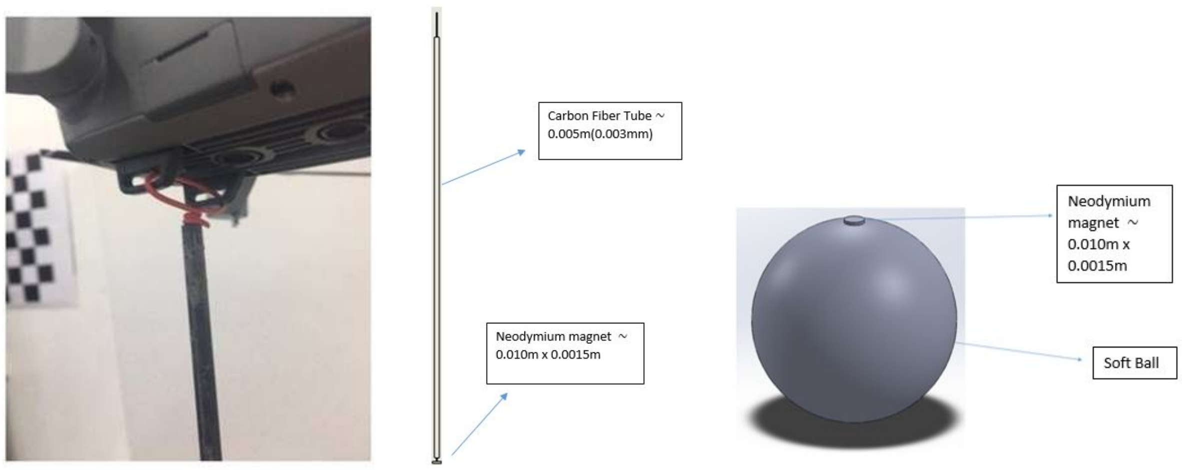
Due to the complexity of visual data processing, this research will stand on the shoulders of giants. Thanks to the work of the developers of OpenCV, the process of identifying and tracking objects in the visual field can be simplified. Part of the research objective is to integrate the open source algorithms made accessible by OpenCV into the real-time drone data processing. The resultant data is intended for use in the control processes of a drone attempting to track and close in on a target.

MBZIRC 2020 and Team VICTOR

MBZIRC 2020

All challenges are intended to be completed autonomously; manual controls are allowed during the competition, although they are penalized.

Figures 1 A-C, below, depict the target ball and way that the ball is mounted to the target drone for challenge 1. The target drone is specified to have a flight speed of less than 10 meters per second and flies in a figure 8 pattern around the arena. For this challenge, teams are allowed up to three UAVs total. The target ball weighs less than 0.15 kg and is attached magnetically to the rod hanging from the target drone shown in the figures. The landing area for drones is a square 100 square meter landing pad on the ground and the UAVs have up to 20 minutes to complete the challenge. The number of target balloons is unspecified. The UAVs must take off and fly autonomously for full scoring and the target ball must be delivered to the landing location.


Carbon Fiber Tube ~ 0.005m(0.003mm)
Neodymium magnet ~ 0.010m x 0.0015m
Neodymium magnet ~ 0.010m x 0.0015m
Soft Ball

Figures 1 A: Drone, rigid attachment link B: Rigid Rod, C: Target ball

Challenge 2 requires the building of a structure using bricks from the arena. There are four brick colors that correspond to sizes and indicate the order in which bricks must be added to the wall. The bricks are cuboids that have the following color-dimension pairings, Red: 0.30m x 0.20m x 0.20m, Green:0.60m x 0.20m x 0.20m, Blue:1.20m x 0.20m x 0.20m, Orange: 1.80m x 0.20m x 0.20m. The bricks must be assembled in a loop of red, green, blue, orange bricks. The bricks weigh between one and two kilograms based on their size and are intended to be gripped magnetically. MBZIRC suggests teams to coordinate UAV and UGV actions, stipulating that some bricks must be carried by UAVs. Teams are given 30 minutes to complete the challenge.

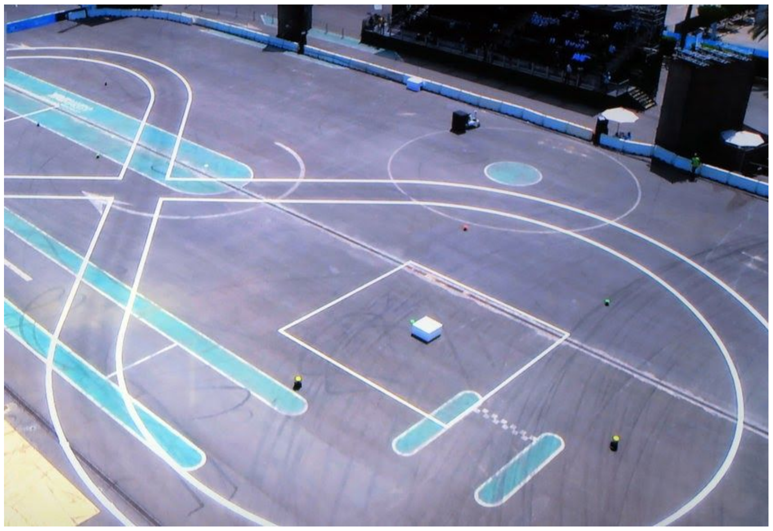


Figure 2: Image of MBZIRC competition arena

Challenge 3 involves the putting out of simulated fires in a tower. The tower is up to 18 meters tall and has large window-like openings on multiple floors. Up to three UAVs and a UGV can all be equipped with water-based extinguishers carrying up to three liters of pressurized water. The challenge duration is 20 minutes.

The grand challenge is a 30 minute window in which up to three UAVs and one UGV are tasked with completing all three of the other challenges. The arena for the grand challenge is 150 meters by 60 meters.

Team VICTOR Contributions

The strategy employed to catch the ball for challenge 1 was to trap the ball in a small net cage above the drone. By moving the drone up and under the target, the ball is positioned between the arms. Once this is achieved, the arms close, trapping the ball in the netted area.

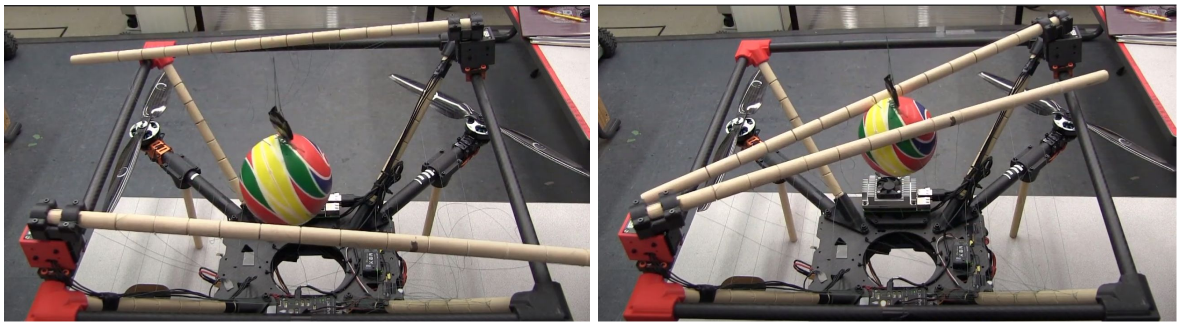


Figure 3: Drone catching target ball using netting and arms

The ball falls then down to the catch-and-release mechanism, which is a small panel that can rotate out of the way of the ball once the drone has moved to the drop location.

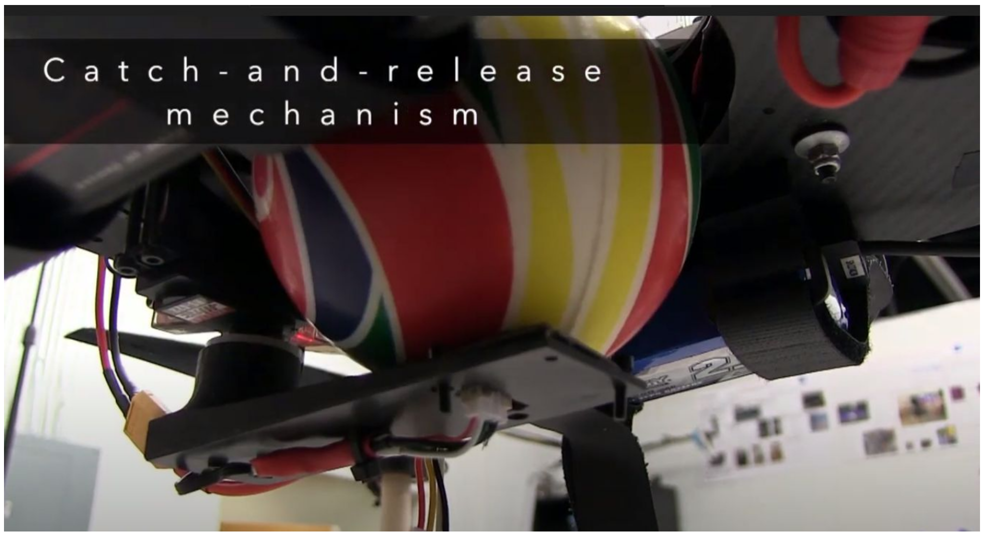
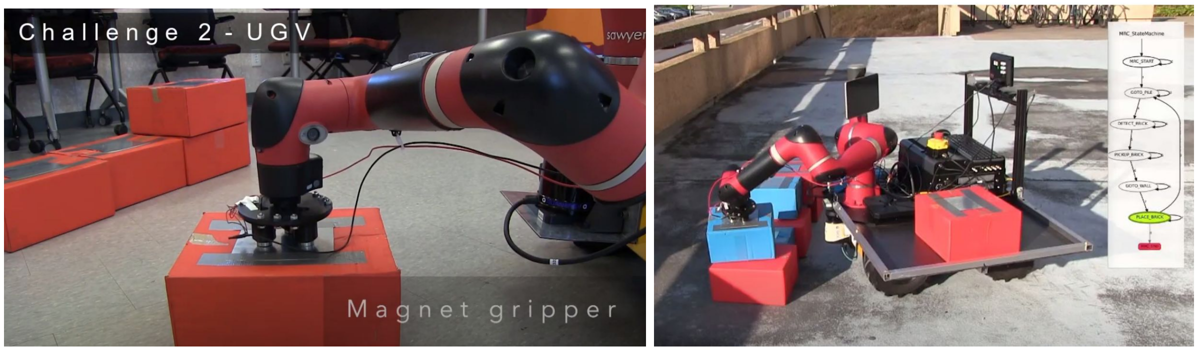


Figure 4: Drone catch-and-release mechanism

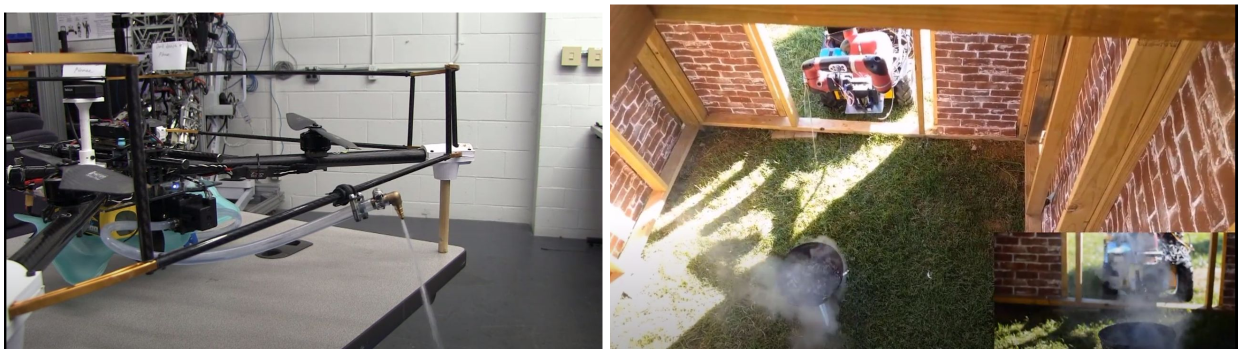
The other task in challenge 1, balloon popping, could be done simply by bringing the blades of the drone close enough to the balloons to make contact. The blades can easily pop inflated balloons, making the majority of this task dependent on the targeting and waypoint algorithms.

Challenge 2 is based primarily on the magnetic gripper shown below. The team focussed here primarily on the UGV automation and using it as the primary base for building the wall. The UGV here used a relatively simple state machine for control here. The UGV would loop through the stages of moving to the pile of bricks, identifying a target, pickup, moving to the droposs location, and dropoff.



Figures 5 A: UGV magnetic gripper, B: UGV moving bricks onto loading tray

A second, larger, UAV was rigged up with a small hose for use in challenge 3. A UGV was also outfitted in this way. Each of these were used independently and in tandem to put out fires for challenge 3. The strategy employed to complete this stage is to coordinate the two robots to attack found fires from multiple directions. Each system employs its own fire identification algorithm, but can communicate with a central hub.



Figures 6 A: UAV with hose attachment, B: UGV expelling water at a simulated fire

Technical Details

The UAV system combined the computing power of three independent but coordinated systems: a Pixhawk (a set of sensors and a computing system running an autopilot software designed for drones), an Nvidia TX2 (an onboard computer), and a remote computer system. The Pixhawk sends control signals to a ROSnode called mavros, which is a communications node built to bridge autopilot systems. Mavros is then in communications with the core control logic in the TX2, which receives signals from the detection and tracking protocols, object avoidance protocols, the PCL manager, waypoint targeting, and from teleoperation controls if enabled. The goal for my capstone was to integrate a fast-running section between tracking and core logic to simplify tracking data so that it can be used to create a more accurate waypoint. For a visualization of this system, see *figure 7* below.

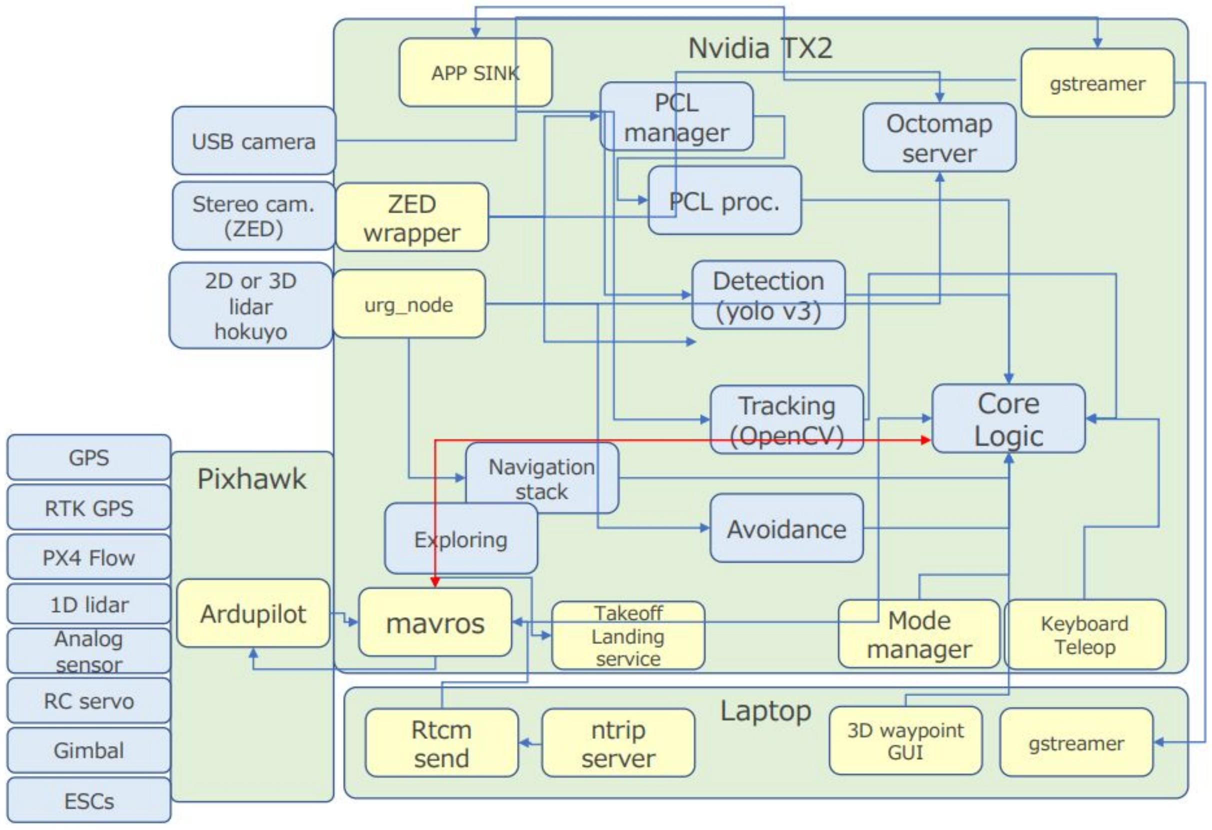


Figure 7. System Diagram[1]

[1] *Masaki Shibuya, April 2020.*

OpenCV was used to track the target objects based on the USB camera datastream. The algorithm for doing this was to perform a Gaussian blur on the input frame to account for flickering and randomness. The model is trained using the target object in a variety of lightings and manual data highlighting to identify the normal expected color range of the target. Using these color filters, the data is then cropped to isolate areas that could contain the target object. OpenCV has fast contour-identifying algorithms; these are then used to identify the largest closed region of highlighted space. This region should well approximate the target using the following assumptions: the object is a singular color, the object is easily differentiable from the background, the background contains no other large objects of the same color.

Once an enclosing circle is created within the image, the pertinent information can be extracted. An effort was made here to correct for the fisheying of the camera lens. This line of effort can be fruitful, but most modern cameras come with built-in rectification as a low-level process. This rectification makes the visual data much simpler to process algorithmically. There is some amount of stretching that occurs as the object is moved away from center-frame. This stretching is minimal for most cameras, so it can be ignored, although it can also be corrected for relatively easily using a transformation matrix that needs to be tuned to the camera.

Trigonometric corrections are then used to adjust for the fact that the fixed-distance from the camera is a curved surface. This correction converts the spherical data into the cartesian coordinate system for spatial mapping. The image is adjusted based on the camera angle differences between the x and y axes. Temporal blending is used to ensure that occasionally erratic data does not affect the output beyond an acceptable range. That data is then normalized with data that came from images that were taken at a similar time and passed to the ROS system to be converted to global space and used to guide the movement of the UAV. Local to global data conversion is done via affine transform using a matrix that is maintained by the drone using gyrometer data.

Original Contributions

While my work does not propose a new methodology for data processing, I designed and developed an algorithmic solution to combine disparate data into concise and usable data. A simple data flow controls the processing of camera data, see *figure 8*.

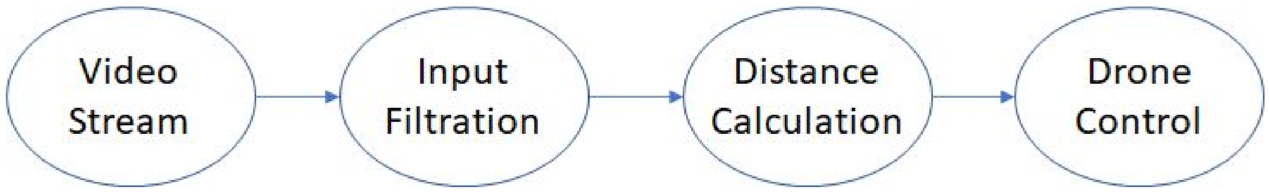


Figure 8. Data flow

My methodology abstracts the data as much as possible before processing. By removing large regions of the input data stream and abstracting the object identifier within the image into the bounding box location, the distance calculation step can be done extremely quickly. As described in the technical details section, the process works by isolating the object within the frame and pairing known information regarding the object's size and the intrinsic parameters of the camera with the object's respective size and location within the frame to produce positioning data that can then be converted from local camera space into global coordinates or the local coordinate system of the drone. My contribution to this research was the development of this process and the algorithm that defines the distance calculation metric.

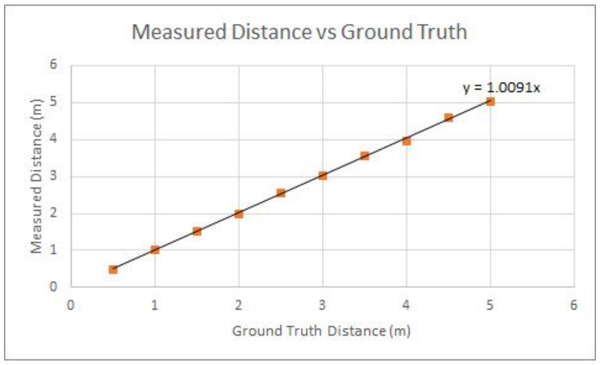
Validation

Validation for this software was done using standard four-level testing. Design and first-level (unit) testing were interspersed as a variety of algorithmic adjustments needed to be made for code consistency. These changes included the shift away from using rectification in the algorithm and adjusting the correction matrix. These unit tests were used for small data-sets composed of real data in a controlled environment.

Simulated integration tests were done using Unity to simulate the camera frame on targets of known distance and size. This stage of testing proved to be useful, although it was not necessarily representative of the final results. The testing drove design towards a more modular base, as the rectification was considerably different in the simulated environment as compared to the real one.

When the algorithm was pushed to stage two and third level of testing, real world integrated tests, there were a few features that appeared to have been unaccounted for. Simple off-by-a-constant corrections helped, but there was some skewing with data nearing the edges of the range. I attempted to correct this in a number of ways; adding and removing variables in my equations, but what ended up working was bypassing a section of code that was intended to correct for the spherical nature of the constant distance plane (as that was being done as a part of the rectification step internally in the camera). I noticed that on the camera that we were using, unlike similar cameras, the ball's shape was generally maintained throughout the entire area of view. Many fisheye cameras rectify in such a way as to skew the lengths of target in their images, whereas this one elected to focus the skew primarily on curvature rather than length, vastly simplifying the image processing problem. Once this realization was made and some data normalization was added, the algorithm functioned accurately to within 2% of the correct absolute distance and direction vector when tested on a 10cm ball between 1m and 5m away from the camera.

Figures 9,10 below show the validation of the system using varied distances. The former shows the measured distance against the ground truth (measured by hand), while the latter shows the measured distance divided by the ground truth.



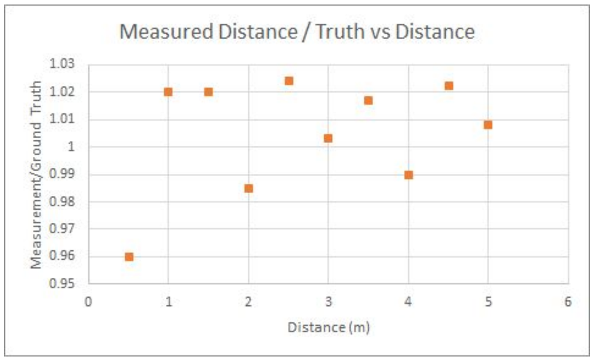
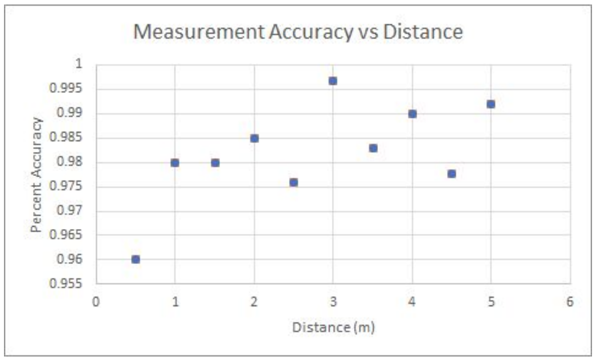


*Figure 9. Measured Distance vs Ground Truth** *Figure 10. Measured Distance / Ground Truth vs Distance**

Validation was performed against the ground truth using measurements made by a tape measure from a 78 degree FOV USB camera. Validation was performed across two axes. First, the camera was pointed directly at the target, in this case a 10cm green foam ball and steady-state measurements were taken from the system measuring the distance to the ball at regular intervals; a second measurement was made to identify the time between movement stoppage and steady-state values being output from the system. Steady-state was defined as data being consistent to within 3% of previous values for 5 consecutive frames. For *figure 11*, percent accuracy is calculated as 1-□, where □ (percent error) is the absolute value of the measurement divided by the ground truth minus 1. See *figure 11, figure 12.*



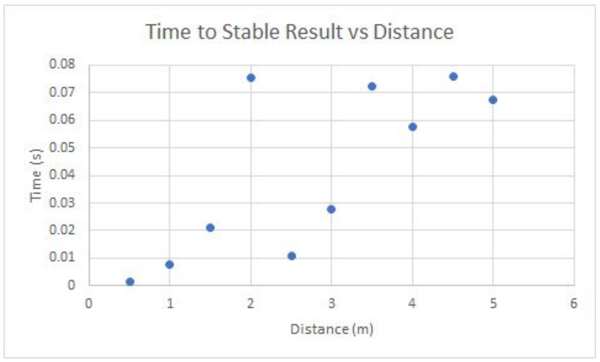


*Figure 11. Measurement Accuracy vs Distance**

*Figure 12. Time to Stabilize Result vs Distance**

A followup test was performed in an effort to identify bias in the visual field. This test was performed by positioning the same 10cm ball at 3 meters away from the camera and moving it about the camera's visual field, taking measurements at regular intervals. *Figure 13* shows the accuracy of the measurements with respect to its location within the visual field.

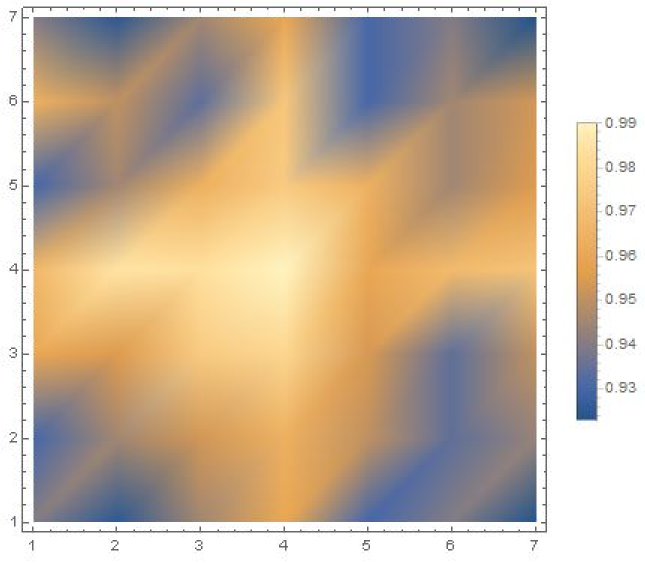


*Figure 13. Plot of Prediction Accuracy Against Region of Frame**

*Data in these figures stands as a placeholder for real testing values

Conclusion

This paper has presented a strategy for data manipulation to quickly convert complex video data into simple positional data using prior or learned information about a target. By leveraging known information, automated systems can quickly assess their environments and ignore the vastness of data represented by a live video feed. This is more likely to produce good results when paired with a machine learning algorithm as a considerable reduction in degrees of freedom enables a much more comprehensive and accurate response map in the input space.

The research goals were achieved. Validation in controlled settings on spherical targets was successful. For spherical targets of a unique color, the algorithm was able to identify the location of the target in global space to within 2% up to a tested distance of five meters. The accuracy of this identification should diminish slowly for larger distances; the upper limits of the protocol were not found during validation. Results demonstrated the effectiveness and applicability of the proposed approach. As a proof-of-concept, this model of data transformation was effective and fast enough to be deployed for live processes during drone tracking flights.

Further research goals include the incorporation of correction features for complex geometry and for less easily recognizable targets in less clear visibility environments. Most target shapes will not be represented at distinct distance/orientation pairings identically in video footage, this feature of irregular objects can be exploited by simulating orientations of the target and matching a theoretical orientation with the video footage, once this is completed, the proposed process can be used to identify its location in global space. This process can also be applied to non-uniform spheroids or objects of known relative orientation to the camera leveraging standard computer vision processes.