Relative Vectoring Using Dual Object Detection for Autonomous Aerial Refueling

PROSPECTUS

Derek B. Worth, Major, USSF

AFIT-ENG-DS-24-S-XXX

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-24-S-XXX

RELATIVE VECTORING USING DUAL OBJECT DETECTION FOR AUTONOMOUS AERIAL REFUELING

PROSPECTUS

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Derek B. Worth, B.S.C.S., M.S.C.E.

Major, USSF

May 2023

AFIT-ENG-DS-24-S-XXX

RELATIVE VECTORING USING DUAL OBJECT DETECTION FOR AUTONOMOUS AERIAL REFUELING

PROSPECTUS

Derek B. Worth, B.S.C.S., M.S.C.E.
Major, USSF

Committee Membership:

Scott. L. Nykl, PhD
Chair

Clark. N. Taylor, PhD
Member

Matthew. C. Fickus, PhD
Member

Table of Contents

Page

List of Figures

List of Tables

RELATIVE VECTORING USING DUAL OBJECT DETECTION
FOR AUTONOMOUS AERIAL REFUELING

I. Introduction

1.1 Background and Motivation

On June 27, 1923, the world witnessed its first successful aerial refueling when Lieutenants Virgil Hine and Frank Seifert passed gasoline through a hose to another DH-4B flying beneath them [1]. After almost a century of development and innovation in the aviation community, Boeing flew a June 2021 flight test in which a prototype for the U.S. Navy's MQ-25A Stringray successfully refueled a F/A-18F Super Hornet, a first-of-its-kind aerial refueling test involving an unmanned tanker [2]. More recently, Airbus' A330 Multi Role Tanker Transport earned the world's first certification to conduct automatic air-to-air refueling boom operations during daylight [3]. These significant milestones clearly reflect the benefits of and growing need to automate the aerial refueling process. In addition to reducing mishaps due to human error and fatigue, automation also has potential as an enabler and force multiplier. Instead of landing at the regular intervals, fully Autonomous Aerial Refueling (AAR) will enable Unmanned Aerial Vehicles (UAVs) to remain aloft with nearly limitless persistence and endurance. For the defense sector, this means unlimited range of its UAVs, ultimately shortening response to time-critical targets, maintaining in-theater presence using fewer assets, and allowing deployment with manned fighters and attack aircraft without the need of forward staging areas [4].

Figure 1. The pilot must *see* both the probe and drogue during aerial refueling operations.

However, before making AAR a reality, autonomous agents must improve their ability to sense the world around them. Aerial refueling requires high precision in a rapidly changing and volatile environment which currently demands talented and highly trained human pilots[1] who can make time sensitive flight decisions with a very tight margin for error. Furthermore, pilots rely heavily on their vision during the refueling process. For example, as depicted in Figure 1, without simultaneously seeing both the probe and drogue, it is nearly impossible for a pilot to guide the two objects together. Thus, the AAR problem fundamentally requires relative aircraft *pose estimation*, e.g., the relative position of the probe in relation to the drogue. In turn, the pose estimation problem requires adequate *sensing* capabilities for extracting pose information from the environment. Finally, both sensing and pose estimation must be *accurate*, *reliable*, and achieved in *real time* to meet the dynamic needs of the aerial refueling process—which subsequently will be used by a higher level autonomous flight control agent to pilot the aircraft and dock for aerial refueling.

Aircraft today can navigate and "sense" the pose of other aircraft around them

[1]Modern aerial refueling is not currently possible without a human operator onboard at least one of the two aircraft to supervise operations.

using global navigation satellite systems such as the global positioning system (GPS), inertial navigation systems composed of IMUs (magnetometers, gyroscopes, and accelerometers), and through the use of various vision navigation techniques. Unfortunately, GPS can be denied [5], IMUs are inherently noisy and drift over time [6], and current state-of-the-art AAR vision algorithms do not simultaneously meet the accuracy, reliability, and execution speed requirements to solve the AAR problem. This work presents a novel computer vision solution, called *relative vectoring using dual object detection*, that consistently converts image data into relative position estimates accurate to less than 3 cm of error (Euclidean distance) at contact and runs in real time (greater than 45 Hz) on a laptop with a Nvidia RTX A5000 GPU.

This technique does not rely on extrinsic camera properties, is resilient to occlusions, and from images containing both the receiver's probe tip and refueling drogue, produces relative position predictions, referred to in this effort as Probe-to-Drogue (PtD) vectors, that tell a receiver aircraft where the drogue is. This proposed computer vision pipeline employs a deep artificial neural network to find image points of two objects, converts those points to pose estimates using solve Perspective-n-Point (PnP), and from the two poses, generates a 3-dimensional (3D) relative position estimate of one of the objects in the reference frame of the other. Only the probe and drogue refueling scenario is explored in this work, but a scenario of a refueling receptacle and tanker boom could be adapted using the same method. Furthermore, only simulated refueling approaches have been explored so far which opens the door to future work in Chapter IV that focuses on bridging the gap between the virtual and real worlds, e.g., through the use of machine transfer learning—an area of research that has already shown great promise [7]. A video summarizing the proposed relative vectoring pipeline can be found at [8].

1.2 Research Objectives

The overall research objective of this effort is to establish an accurate, reliable, and fast computer vision technique for guiding a receiver aircraft on aerial refueling approach to its target, enabling it to synchronize flight with a tanker aircraft for AAR. In this research effort, this objective can be subdivided as follows:

(RO-1) Establish a vision-based relative navigation pipeline aimed at solving the AAR problem that can guide one object to another, namely a probe tip to a drogue (or boom to receptacle).

(RO-2) Enhance the proposed pipeline such that it is accurate (magnitude error less than $7cm$ at contact), reliable (valid predictions at least 99% of the time), and fast (greater than $20fps$) with a total pipeline latency, i.e., time between image capture and pipeline output prediction, less than $250ms$.

(RO-3) Establish a simulation framework for developing and testing the proposed pipeline in a virtual world.

(RO-4) Demonstrate the proposed pipeline in real world applications—perhaps onboard a Boeing test flight.

1.3 Document Outline

This prospectus centers around a proposed monocular computer vision technique aimed at enabling AAR. A brief background is provided in Chapter II to highlight previous attempts at solving this problem. This includes image feature extraction for drogue pose estimation using various analytical approaches, as well as more generic vision-based pose estimation methods using state-of-the-art machine learning (ML) techniques. Chapter III presents the proposed technique in this research effort as

a 5-stage pipeline tailored for AAR that overcomes several limitations encountered in previous work. Although only demonstrated in simulation thus far, Chapter IV details research efforts to be conducted over the next year that will transition this pipeline into the real world. Finally, Chapter V provides a rough timeline of the milestones leading up to my dissertation defense and graduation.

II. Background and Related Work

2.1 Vision-Based Autonomous Aerial Refueling

AAR has been an ongoing and thriving center of research for decades in which the literature primarily points to three types of solutions: signals, inertial, and vision based approaches. Several studies have proven centimeter-level precision is possible during formation maneuvers, such as those encountered in aerial refueling operations, when using differential GPS (signals) with INS (inertial) [9, 10]. However, the paradigm continues to shift toward vision based solutions that are impervious to interference, jamming, and drift.

Nearly all AAR vision methods today are implemented as either monocular or stereo configurations and take advantage of the pinhole camera model—which exploits projective geometry of photons striking specific pixels on an image plane after reflecting off objects and traveling straight through a common focal point [11]. This gives pixel information meaning and makes it useful in pose estimation. Thus, most vision approaches rely upon extracting this information from imagery and typically follow some variation of the common four-stage pipeline: image capture, 2-dimensional (2D) feature detection, 2D to 3D feature matching, and pose estimation.

2.2 Analytical Approaches

Early attempts at feature detection involved processing pixels using methods such as infrared LED blob detection [12], Harris corner detection [13], VisNav [14], Speeded Up Robust Features (SURF), Scale-Invariant Feature Transform (SIFT), HSV color segmentation, and active contouring [15]. Each of these methods produced 2D image points that were subsequently matched to corresponding 3D model points using algorithms like mutual nearest point [13], classical assignment model, perspective trans-

formation based matching [16], maximum clique detection [17], and the Munkres algorithm [18]. Finally, the resulting 2D to 3D matches were transformed into 6 degrees of freedom (6DoF) pose estimates using one of three algorithms, namely Gaussian Least Squares Differential Correction (GLSDC), the LHM algorithm (i.e., orthogonal iteration) [19], or perspective-n-point (PnP) [20].

Various combinations of these methods tackle the problem with unique benefits and produce fast predictions with minimal error, but also come with significant drawbacks. For example, LED detection approaches enable operations in low lighting, but also require power to each beacon, are susceptible to over saturation due to light pollution and ambient interference, and most are not resilient to occlusions and missing features. VisNav overcame the saturation problem through the use of feedback control loops to automatically adjust for optimal beacon intensity during flight. However, VisNav was only proven successful under little to no turbulence, is expensive to install, requires complex calibrations, suffers from occlusions, and is thus effectively not in use today [21, 22].

2.2.1 Line, Circle, and Ellipse Detection

Similar techniques were also developed for edge and arc detection of lines, circles, and ellipses. For example, Erkin used Hough transforms to find circles, then solved for an ellipse projection equation to obtain pose estimates [23]. Unfortunately, many of the explored ellipse detection methods rely on unrealistic assumptions. For instance, Fan only tested the AAMED algorithm in his approach on a perfectly round flat red circle [24]. Xufeng used rigid and flat uniformly colored rings that are easy to find in images, but do not exist in any real drogue configurations [25, 26]. Zhao proposed a solution that accounts for drogue deformations by solving for the perspective-three-line problem, but assumes perfect line and circle detections have previously been

obtained [27, 28]. Ellipse detection methods show potential, but currently remain untested in realistic scenarios.

2.2.2 Stereo Vision

Stereo vision approaches, like the one proposed by Parsons [29, 30], take advantage of disparity mapping between image pairs to attain depth information and generate 3D point clouds. Subsequently aligning these point clouds with known 3D models using the iterative closest point (ICP) algorithm can produce accurate pose estimates. Zhang used a similar method, but added some optimization steps and performed the model-to-point cloud alignment with shape fitting [31]. This line of research shows stereo vision can produce estimates with high accuracy, but requires near perfect extrinsic calibrations between the cameras, suffers from occlusions, is typically slower and more complex to implement (and optimize) than monocular approaches, does not always converge to globally optimal solutions, and requires significant computing resources to process large point clouds—even with Delaunay Triangulation as proposed by Anderson [32].

2.2.3 LiDAR and Structured Light

Other vision based depth approaches make use of light detection and ranging (LiDAR) and structured light. Unfortunately, these methods also come with significant drawbacks related to limited operating ranges and sensed object types, such as the challenges Chen et al. encountered while balancing laser power with camera sensitivity [33]. They reported problems with over saturation within close proximity to the drogue and under saturation while far away. Curro pointed out in his AAR research that LiDAR produces widely varied levels of accuracy depending on the surface textures of sensed objects [34]. Similarly, the FFB6D algorithm in [35] used depth from

RGB-D images for accurate real time pose estimation, but the authors also reported certain textures resulted in decreased accuracy.

2.3 Machine Learning

Within the past decade, computer vision has heavily adopted machine learning (ML) techniques to overcome limitations related to the analytical approaches above. More specifically, object recognition—the collection of machine learning tasks associated with identifying objects in digital imagery—offers real time object localization, detection [36], segmentation [37], and tracking [38] despite occlusions, texture changes, and variations in lighting conditions. For example, Guan trained a deep neural network to predict centroids and vertices of 3D bounding boxes surrounding specific objects in RGB images [39]. Dede implemented a convolutional neural network that could easily find semantic key-points [40]. Sun created MPDCNN which consistently and accurately found 16 features along the perimeter of the drogue's inner core and 32 features around the outer drogue canopy [41]. Some researchers, like Garcia [42], even fabricated their own drogues retrofitted with built-in markers specifically designed for effective object detection. Subsequently, analytical techniques were applied to the resulting 2D points from these methods to predict the drogue's pose with high accuracy.

Furthermore and while not directly related to the AAR problem, several other research efforts have proposed a variety of ML models for direct pose estimation from RGB imagery by consolidating the feature detection, feature matching, and pose estimation tasks into individual and independent network architectures. Prominent examples include DFPN-6D [43], YOLO-6D+ [44], SSD-6D [45], SMOPE-net [46], BB8 [47], PoseCNN [48], and a few others [49, 50, 51]. These all performed well on popular open source datasets like LINEMOD, T-LESS, and KITTI-6DoF. However,

some reported difficulties rectifying pose estimates with symmetry and all the ML approaches mentioned above have one significant limitation: they require large accurately labeled training datasets which are costly, or in some cases, impossible to produce. Moreover, every pose estimation approach covered thus far, whether analytical or ML based, makes predictions relative to the camera. This is only useful in AAR applications if the relative pose between the camera and the object of interest (e.g., refueling probe tip) are precisely known and do not change midflight—preconditions that are nearly impossible to maintain in the turbulent scenarios often encountered in aerial refueling.

2.4 Summary

Vision-based approaches to AAR offer significant benefits over that of signals and inertial-based approaches. Namely, they are not susceptible to interference, jamming, or drift. The literature offers several vision solutions, each with varied levels of accuracy. Unfortunately, many of these approaches (both analytical and machine learning) also come with significant limitations and all focus on estimating drogue poses relative to the vision sensor. The main contributions of this work address these challenges by reframing the AAR problem of "drogue pose estimation relative to the vision sensor" to that of "drogue position estimation relative to the probe." As explained in the Chapter III, this subtle difference allows the proposed pipeline to ignore extrinsic camera properties, mitigate challenges related to detection and tracking of a symmetric object (i.e., the drogue), and dynamically adapt to changes in real world situations.

III. Relative Vectoring (In Simulation)

3.1 Proposed 5-Stage Pipeline

Relative vectoring, as proposed in this work, overcomes dependencies on extrinsic camera calibrations by providing the receiver aircraft direction and distance, computed in its own local reference frame, to its target, i.e., to the drogue, without any knowledge of extrinsic camera properties. This effort performs relative vectoring by exploiting dual object detection (DOD) and simple reference frame transformations. DOD is defined here as the application of solve PnP on features of two separate objects in the same image, as shown in Figure 2. This produces a bonded pair of 6DoF pose estimates in the camera's local reference frame corresponding to the two objects. The resulting poses were used to compute a vector between the two objects, transformed into the local reference frame of the estimated receiver pose, see Figure 3.

Note that performing solve PnP on two objects in the same image guarantees the two resulting pose estimates are computed for them at precisely the same moment, making time alignment unnecessary. Subsequently, transforming the vector between the two poses from camera reference frame to receiver reference frame obviates any need for extrinsic camera calibrations. The camera could be flipped upside down and positioned anywhere on either aircraft, and as long as both objects are present in the image, this method will produce the same vector! Furthermore, solve PnP makes accurate predictions, even with missing points. The algorithm can make pose predictions with as few as three 2D to 3D matches, making this technique resilient to occlusion (e.g., the probe blocking parts of the drogue from view). The end result is a PtD vector that provides accurate direction and distance from the probe tip to drogue center from the perspective of the receiver. This vector is exactly what an autonomous agent needs to pilot the probe into the drogue during aerial refueling.

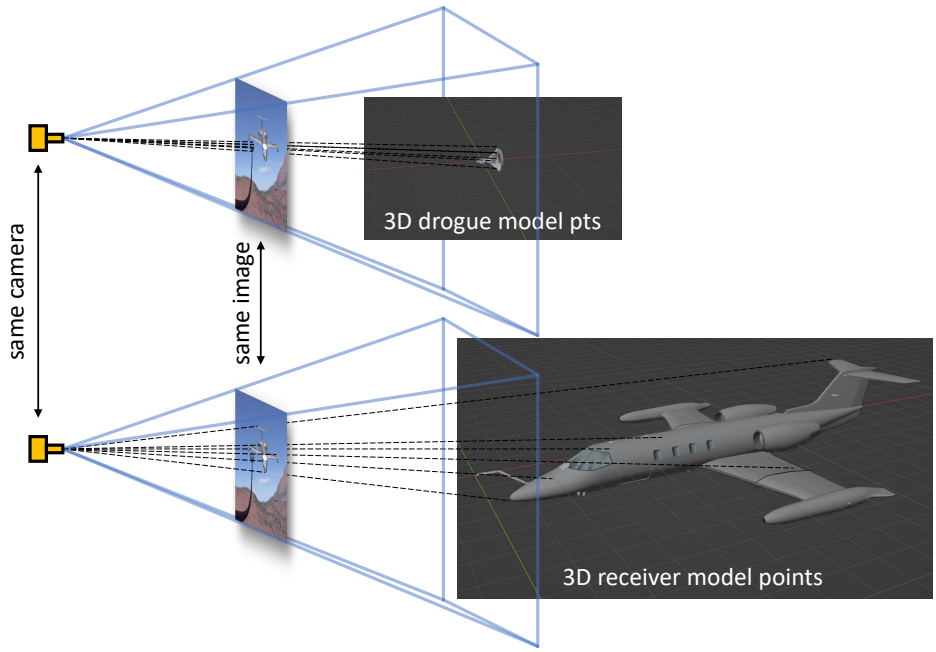


Figure 2. Dual object detection uses machine learning to find 2D features of two separate objects in the same image and matches them to corresponding 3D model points before passing them to solve PnP for estimating relative poses of the objects.

The stages for computing the PtD vector are summarized as follows:

1. Capture image containing both receiver aircraft and drogue basket.

2. Find 2D image points of known receiver and drogue features using machine learning, e.g., an object detector.

3. If at least 3 features are found for each object, match detected 2D image points of each feature to corresponding 3D object frame points.

4. Estimate camera frame poses for the receiver and drogue separately, but from the same image.

5. Transform the two receiver and drogue poses into a single probe-to-drogue vector.

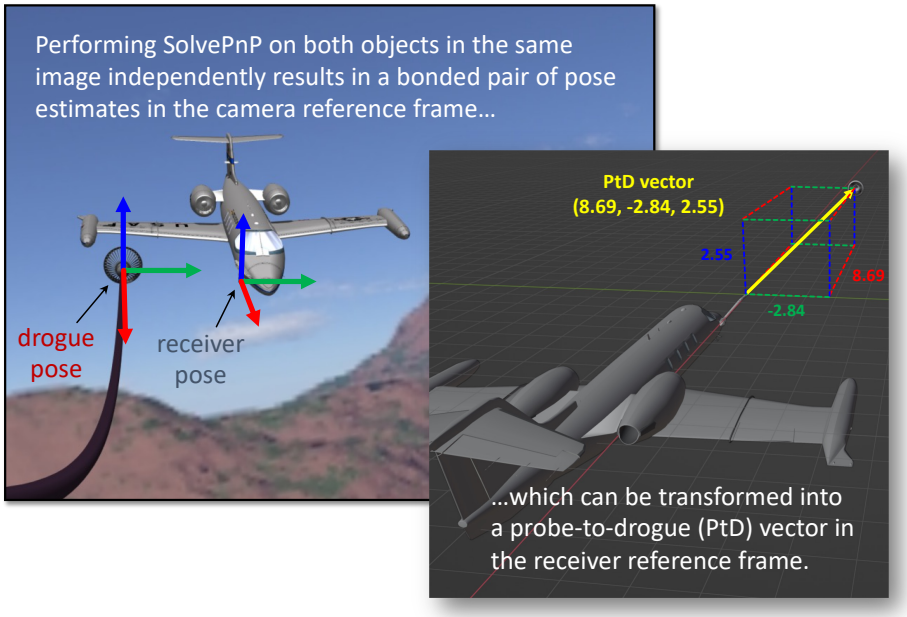


Figure 3. From two poses (generated by dual object detection), a relative probe-to-drogue vector is computed.

The rest of this section describes implementation details of the proposed relative vectoring pipeline above (also see Figure 4) and experimentally justified design decisions.

3.1.1 Capturing Imagery

This may seem like the most simple and trivial stage—*just point the camera and snap a picture, right?* However, as any professional photographer or cinematographer will attest, images come in many different shapes and forms. The domain has several dimensions that effect computer vision, including pixel density, aspect ratio, distortion effects, fields of view, ISO, shutter speed, aperture, exposure, lighting conditions, and vantage point to name a few. Each of these comes with a trade space. For example, higher resolution images may reveal more information about the scene to potentially increase pipeline accuracy and reliability, but also requires more computational resources and ultimately detracts from real time execution. Thus, these

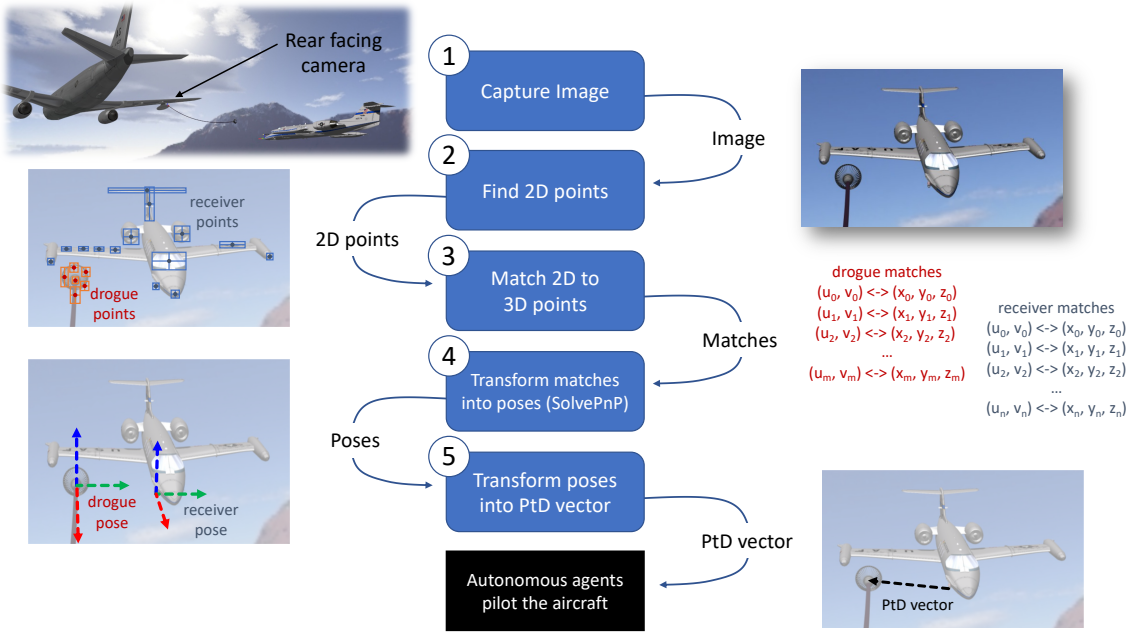


Figure 4. The full relative vectoring pipeline using dual object detection includes 1) capturing an image, 2) using machine learning to find 2D points in the image corresponding to known 3D features on both the probe/receiver and drogue/tanker, 3) matching the 2D and 3D points, 4) converting the matches to 6DoF poses using solve PnP, and 5) computing a probe-to-drogue vector using reference frame transformations.

trade spaces influenced the camera properties chosen in this chapter.

Since the scope of this chapter is limited to proving the viability of the proposed relative vectoring technique in simulation—leaving real world implementation for future work—the problem is simplified by assuming no lens distortion, i.e., perfect intrinsic calibrations in which all subjects in the image are in focus regardless of distance from the camera. The camera fields of view are also varied during different phases of the aerial refueling approach to maximize the spread of features across the pixel space. Because real cameras with variable zoom have infinitely many zoom levels, each requiring an independent intrinsic camera calibration, e.g., using Zhang's method [52], the simulated cameras were restricted to static discrete horizontal fields of view (hFOV). This could easily be implemented in real world with separate cameras operating in parallel, each with their own fixed intrinsic parameters. Since the

main objects detected in the image frame (i.e., aircraft) are mostly short and wide, rather than tall and narrow, a 2K resolution with a relatively wide aspect ratio of 1.90:1 for all cameras was used. Table 1 summarizes the camera parameters used in this chapter.

Table 1. Intrinsic camera parameters.

Resolution	2048×1080 pixels
Horiz. FOV (hFOV)	15, 25, 55, and 75 degrees
Dist. coefficients	zeroes (no distortion)
Optical center	c_x, c_y = (1024.0, 540.0) pixels
Focal length	$f_x = f_y = \frac{c_x}{\tan(\text{hFOV}/2)}$ pixels

Additionally, current aircraft designs motivated the camera positions tested. For example, cameras could easily be mounted inside the cockpit or on detachable static wing pods in the real world, while other locations such as those difficult to route power and data links to or those on moving parts and flight critical aerodynamic surfaces were less viable. Thus, the two primary camera locations on the receiver were forward facing on the probe side wing pod (see right side of Figure 6) and inside the cockpit. A rear facing camera mounted on the tanker buddy pod next to the drogue hose feed port was also experimented with.

Software Simulation. Implementing these cameras to model realistic aerial refueling scenerios and render corresponding camera imagery needed for testing the relative vectoring pipeline required a high fidelity simulator. As previously alluded to, the simulator must produce realistic and undistorted 2D camera imagery of 3D objects with corresponding truth data for pipeline validation. The *AftrBurner* computer graphics engine [54] was chosen in this chapter for its integrated camera modeling and OpenGL-based rasterizer, which takes advantage of ray casting and 3D rendering on 2D image spaces using traditional coordinate system transformations, as described in

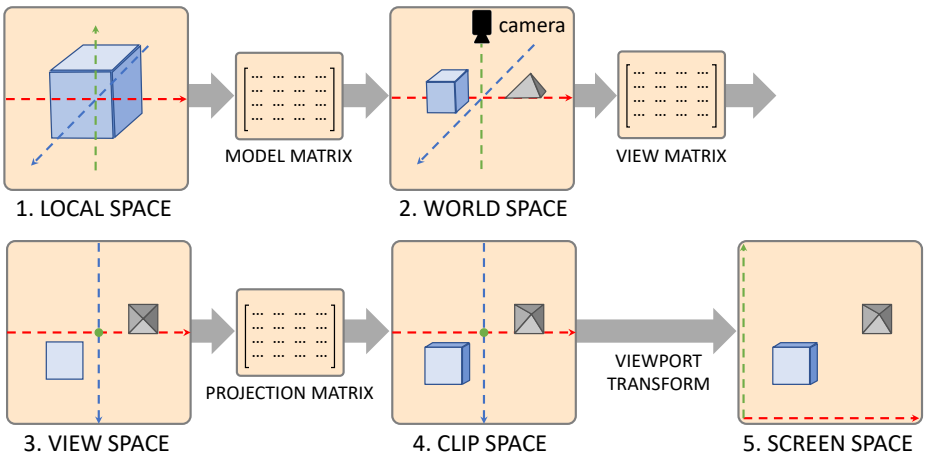


Figure 5. Local to screen space transformation [53].

[53] and depicted in Figure 5.

AftrBurner also provides precise control over position and orientation of world objects, enabling implementation of complex and dynamic scenarios. The five primary objects implemented in each of my experiments included the receiver aircraft (with attached refueling probe), camera (implemented as a frame buffer object with the intrinsic parameters listed in Table 1), tanker, and drogue basket—all imported as static models from OBJ files. The fifth object was a flexible hose with dynamic indexed geometry connecting the drogue to the tanker. This allowed us to implement an accurate drogue dynamics model from actual flight test data. In other words, the simulated drogue flopped around with turbulence and other wind effects in simulation as it would during a real refueling scenario. These objects were scaled in simulation to real world dimensions. In the end, AftrBurner generates a digital twin by rendering realistic scenes from imported textures and OBJ files, projecting true 3D points corresponding to object features as 2D points in synthetic imagery, and precisely modeling object orientation and movement within a common world reference frame. For example, the receiver aircraft approaching the drogue basket extended behind a tanker flying straight and level (and while performing banking maneuvers) was modeled, as shown in Figure 6. This allowed for validation of the relative vectoring pipeline with

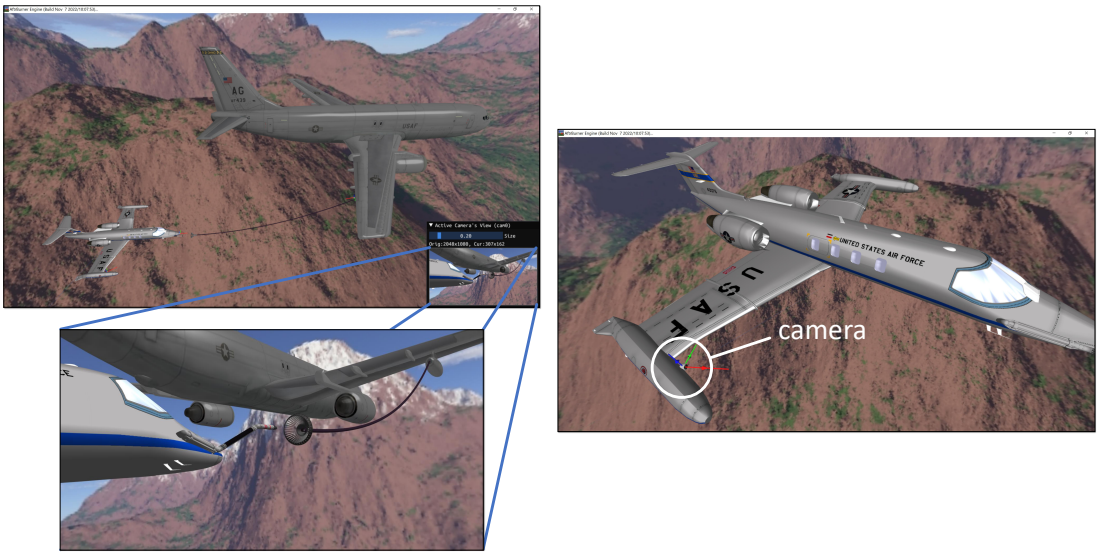


Figure 6. The top left image shows the AftrBurner simulation with the tanker and receiver flying in an aerial refueling echelon formation. A camera is mounted on the right wing of the receiver such that it can capture realistic imagery, as shown in the bottom left image, containing both receiver and tanker features during approach.

3D truth in multiple reference frames.

Monte Carlo Approaches. During real aerial refueling operations, the receiver aircraft typically begins its approach behind and from below, gradually climbing to match the tanker's altitude, all the while chasing a centerline approach that aligns its probe tip laterally and vertically with the drogue basket center. Therefore, similar approaches were modeled in this effort. Rather than implementing a complex flight dynamics model to replicate this behavior, a simple relative motion model was applied in which the receiver aircraft is always positioned at a dynamic 6DoF pose offset from the origin of a local reference frame centered at the average drogue location, with the x, y, and z components of this reference frame pointing in the tanker forward, left, and up directions respectively (see Figure 7). In other words, if the drogue was stationary relative to the tanker, this origin would coincide with the drogue center. The receiver "flies" by updating this dynamic offset with PtD vectoring and incremental rotations.

Thus, each approach was modeled as follows: The receiver aircraft was initialized with a random pose—the dynamic pose offset set to a random position within a

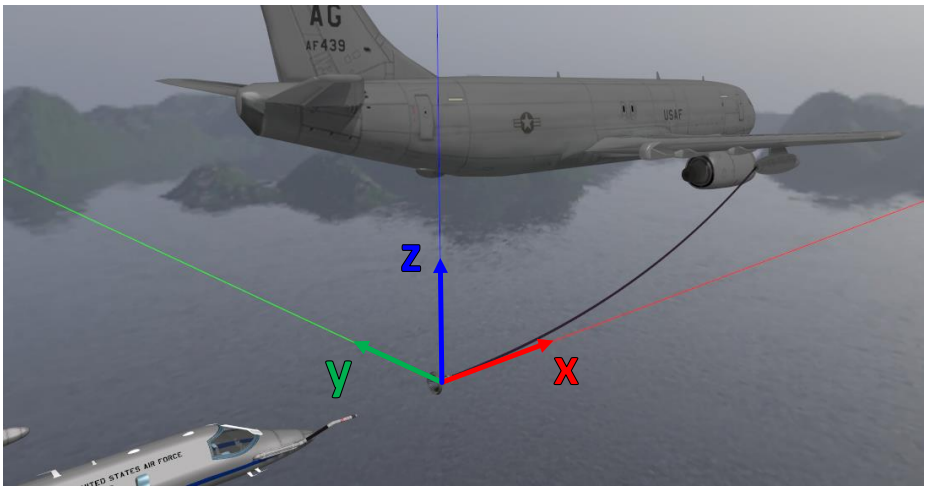


Figure 7. Drogue-centered local reference frame.

specific range behind the drogue and an orientation set to match that of the tanker. Then, the pose was randomly perturbed between ± 10, ± 15, and ± 45 degrees yaw, pitch, and roll respectively. Once initialized, the approach would proceed by following a true PtD vector computed in the receiver's local reference frame such the the receiver moved in the direction of the vector with a convergence along the y (lateral) and z (vertical) components approximately twice as fast as that of the x (forward) component. Simultaneously, spherical linear interpolation was applied over quaterions to the pose offset to model the receiver's controlled rotation corrections, which gradually converged to match the tanker's orientation. This process produced simple, yet convincingly realistic aerial refueling approaches that imitate the behavior of an actual receiver aircraft chasing a drogue as it "flops" around behind the tanker. This process continued until contact was made, which was defined as less than $1cm$ Euclidean distance between the probe tip and drogue center. After contact, the receive would reinitialize and start over with a new random position and orientation. This process was repeated continuously throughout all experiments and data collects.

3.1.2 Finding 2D Image Points

Using the camera pinhole model [11], the images captured in the previous stage serve as the mapping between 2D image points and corresponding 3D world points. Automating accurate 2D image point detection thus became the next logical stage in the relative vectoring pipeline. To keep it accurate, reliable, and fast, machine learning was used.

Object Detection with YOLO. Instead of creating or modifying an existing machine learning algorithm to perform pose estimation directly, I exploit what modern object detectors already can do with high precision, recall, and speed—namely, predict 2D bounding boxes. Specifically, they excel at simultaneous localization and categorization of multiple objects in an image, and many state-of-the-art algorithms, such as You Only Look Once (YOLO), can do this in real time [55, 56, 57]. As originally proposed by Lynch [58] and Worth [59], YOLO can find the 2D image points needed for this effort.

Many different versions and modifications to the algorithm exist, including YOLO MDE [60] and YOLO-6D+ [44] which both perform depth and pose estimation directly. However, the one used in this effort was the unaltered Ultralytics YOLOv5 PyTorch implementation found at [61]. This open source version provides a simple interface with easy to follow tutorials that guide users through the training process. Similar to other versions, YOLOv5 trains on labeled images, and after enough training, a YOLOv5 model can find just about any distinct feature for which it was trained. Unfortunately, YOLOv5 outputs predictions in the form of bounding boxes surrounding 2D objects—not exactly the 2D points corresponding to 3D points needed in the proposed relative vectoring pipeline, i.e., the centers of these bounding boxes do not align with any particular 3D points. To overcome this misalignment, models were trained with labeled images generated with *bounding box corrections* specifically de-

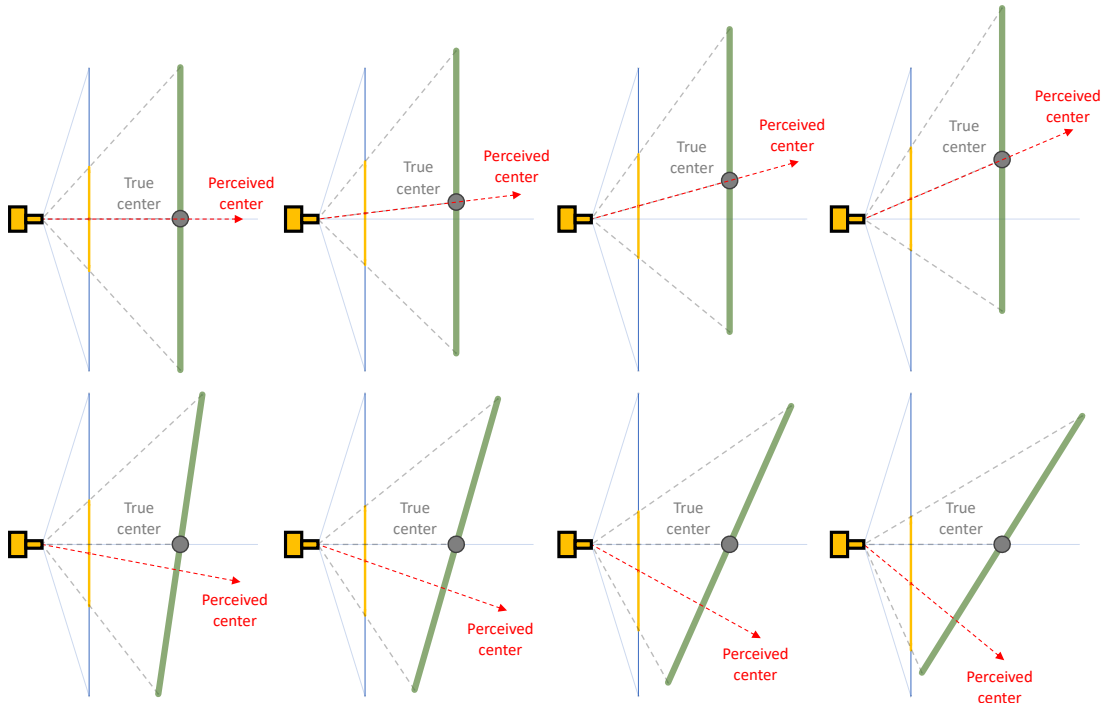


Figure 8. The top row shows that translation does not effect the alignment of the true to perceived object center, as long as the object surface remains parallel to the camera's sensor plane. However, rotation (bottom row) has a parallax effect, forcing the perceived center to diverge from the truth.

signed to align YOLOv5's predictions with the 3D geometric centers of strategically chosen features.

Parallax Effect. So, why are bounding box corrections necessary? The center of an object in an image projects to that same center in the real world, right? Actually, no! The camera pinhole model tells us that the entire surface of an object projected into an image is subject to skewing, even in cameras with no lens distortion [11]. In this effort, this phenomenon is referred to as the *parallax effect*, which is due to the disparity in projective geometry between points closer to the camera's image plane and those further away. This same effect is what makes a picture frame hanging on the wall look rectangular when viewed from the front and trapezoidal when viewed from the side, causing its dimensions to appear disproportionate. Figure 8 demonstrates how the parallax effect causes the perceived object center, as viewed in the image, to

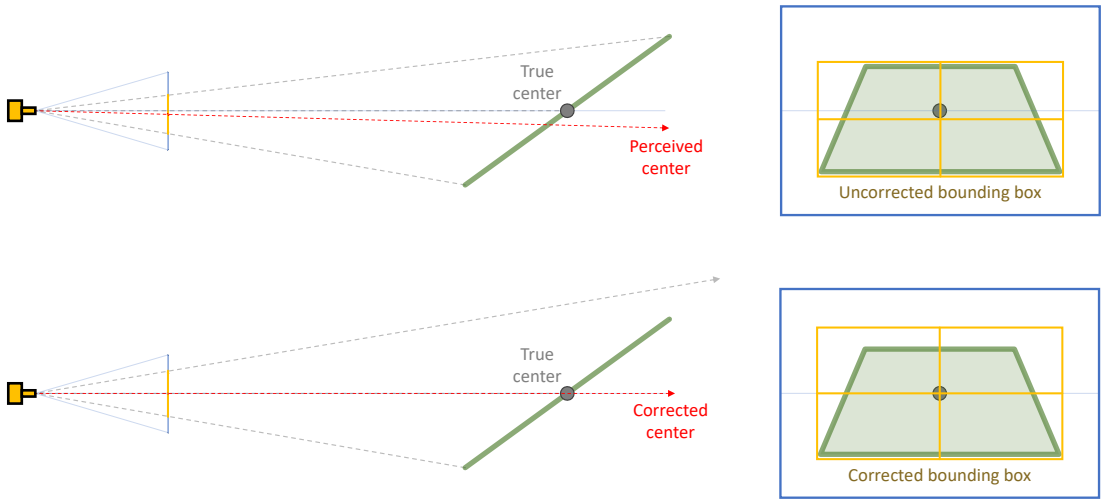


Figure 9. Bounding box corrections align the perceived 2D center (i.e., original uncorrected bounding box center) with the true 3D center. Corrected image labels, as depicted on the bottom right, allow YOLO to learn these corrections and accurately predict 2D points corresponding to 3D feature centers.

diverge from the true 3D object center. Notice that mere translation, as depicted in the top row, has no divergent effect since all points across the object's surface remain equidistant from the camera's image plane. In contrast, rotation of the object has a dramatic effect, causing the perceived center to quickly diverge with even small distance-to-image plane disparities.

Bounding Box Corrections. To overcome the parallax effect and successfully train YOLO to find 3D center points in 2D images rather than perceived center points, we must feed it corrected training data in the form of labeled images, as shown in Figure 9. Furthermore, for this to be a viable option, this process must be automated and accurate. Manually labeling images is a tedious process. For example, the 328K images from the popular MS COCO dataset required over 30K worker hours to produce [62]. Even more concerning, manually labeled images are prone to human error and are often laced with missing annotations, misclassifications, and imprecise bounding boxes [63]; all of which will most certainly decrease the accuracy of the proposed relative vectoring pipeline—thus, the need for automation.

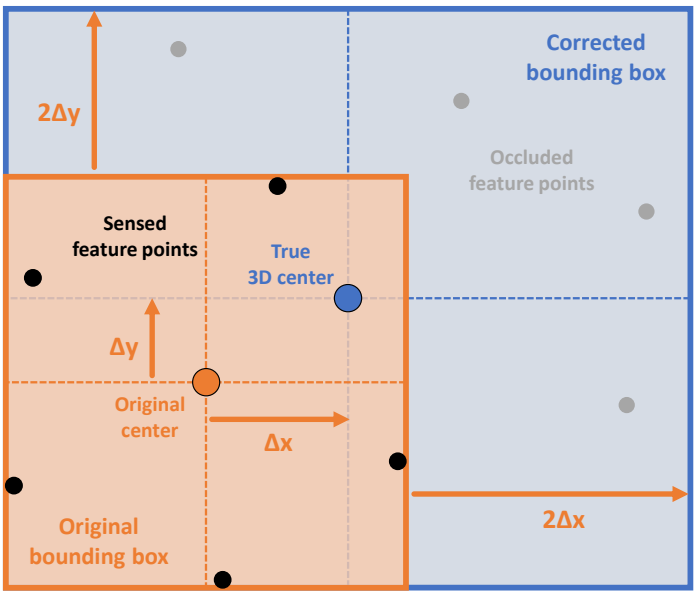


Figure 10. Two sides grow outward until the bounding box center aligns with the true 3D feature center.

Fortunately, accurate and reliable label generation in simulation is relatively easy to automate. Simply establish a 3D feature by selecting 3D points in local model space surrounding such feature, then transform the points to camera screen space (see Figure 5). Next, surround the corresponding projected pixel coordinates with the tightest fitting bounding box. This is accomplished by finding the maximum and minimum x and y values among all sensed points. Finally, grow the original bounding box by extending two adjacent sides outward such that the new bounding box center aligns with the feature's true 3D geometric center projected into the image. This is done mathematically by computing the differences in x and y components between the original and true 2D centers, $(\Delta x, \Delta y)$, then expanding the box toward the true 3D center projection by $2\Delta x$ and $2\Delta y$ along the corresponding adjacent sides respectively, see Figure 10. An example of applying this process to synthetic imagery is depicted in Figure 11.

Also, corrected bounding boxes are excluded from image labels that extended near (a threshold of 10 pixels was chosen) or off the edge of the image frame. YOLO

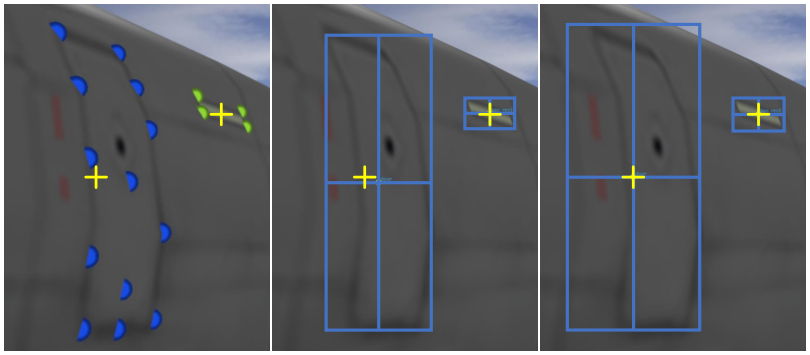


Figure 11. The yellow crosshairs coincide with image projections of true 3D feature centers—these 2D center points are what we want YOLO to learn. On the left, 3D model points were selected representing tanker features. The middle image shows the tightest fitting bounding boxes surrounding the 2D image projections of those points—notice that the bounding box centers do not align perfectly with the crosshairs. The final corrected bounding boxes are shown on the right—these are the labels YOLO was trained with.

should not learn partial features near the edge, since this would lead to bounding box predictions with centers misaligned with true 3D center projections. For example, it is impossible for YOLO to predict a bounding box with a 2D center outside the image frame, even if it is trained with partial bounding boxes containing centers outside the image frame. Instead, YOLO would interpret, learn, and later predict such partial features as whole features, resulting in incorrect bounding box centers and decreasing pipeline accuracy when features appear near the image edges.

Note that choosing to either include or exclude unseen (i.e., visually occluded) feature points during label generation has potential to form different bounding box sizes and shapes, ultimately inducing variations in what the trained YOLO model learns. For the YOLOv5 models trained throughout this effort, all projected feature points were included regardless of their visibility in the image. For example, all drogue feature points were included when forming corrected bounding boxes around drogue features, even when the receiver's probe occluded such points from view. Theoretically, this has potential to give YOLO object permanence or an "x-ray vision" like ability to detect a feature behind another based on surrounding spatial information

in the image. As later reported in Section 3.3, this was observed in some of the experiments. However, this effect was not extensively tested in this chapter and serves as a good research area in the next chapter.

Other Machine Learning Decisions. With the above image labeling method and previously mentioned Monte Carlo simulated approaches, precise error-free labeling of thousands of high-resolution synthetic images were automated in a relatively short amount of time (approximately 30K images, each with at least 80 features, per hour). This included data augmentation—an important process the Google Research Brain Team highlights as a critical component of training deep learning models [64]. In simulation, this comprises an assortment of different lighting effects, backgrounds, orientations, vantage points, and occlusions. These labeled images were further augmented (scale, mirror, crop, mosaic, etc.) using default settings within the Ultralytics YOLOv5 training implementation. The limited memory of the platform used in this effort, listed in Table 2, limited the amount of 2K images cached and stored in RAM. Thus, each dataset was limited to between 8,000 and 10,000 images, each labeled with at most 40 receiver/probe and 40 tanker/drogue features, i.e., 80 features total per image. An 80/20 training and validation data split was chosen and each model was trained with 300 epochs and a batch size of 16.

For all experiments, a *small* YOLOv5 model[1] was used with a relatively low resolution input size of 864×864, forcing me to resize and pad the 2K images generated by the virtual cameras prior to YOLO accepting the them as input during inference time. This configuration resulted in sufficient model performance while maintaining real time execution—a more optimal configuration likely exists, but was not heavily sought out in this chapter. All simulations, image labeling, training, and experimentation took place on a laptop with the configurations listed in Table 2.

[1]YOLOv5 model sizes include nano, small, medium, large, and xlarge with a performance (mean average precision) vs speed trade space outlined at [61].

Table 2. Platform configuration.

Environment setting	Version/specification
Computer (laptop)	Lenova ThinkPad P15
Operating system	Windows 10 (64-bit)
Processor	Intel Core i9-11950H
Memory (RAM)	128GB
Disk storage	1TB
GPU	Nvidia RTX A5000
Nvidia CUDA	v11.7
OpenCV (with cuDNN)	v4.6.0
OpenGL	v4.3
YOLOv5 model format	.onnx, exported from .pt

3.1.3 Matching 2D to 3D Points

YOLO models fully trained with the method described above can find corrected 2D points corresponding to 3D geometric centers of distinct features. However, before using these 2D points for pose estimation in the relative vectoring pipeline, they must be paired with correct corresponding 3D object model points. Fortunately, YOLO not only finds features, but also assigns each a class ID.

The internal complexity of YOLO and how it makes its predictions is beyond the scope of this work. However, to effectively use it, we must at least understand the structure of its output and what that output represents. When given a square input image, YOLO makes predictions by dividing the image into grid cells at three different scales by dividing each side by 32, 16, and 8—this enables scale invariant learning in which the network can predict bounding boxes surrounding small, medium, and large objects respectively. Subsequently, the network applies three different anchor boxes to each grid cell and outputs a prediction for each. Thus, YOLO makes P predictions on a square image with s pixel side lengths, where:

$$P = 3\left[\left(\frac{s}{32}\right)^2 + \left(\frac{s}{16}\right)^2 + \left(\frac{s}{8}\right)^2\right] = \frac{63s^2}{1024} \tag{1}$$

In turn, each prediction defines a bounding box comprising x, y, w, h, and c corresponding to its 2D center coordinate, width, height, and "objectness" (i.e., confidence) score respectively. Additionally, each prediction also includes a set of class probabilities, one for each learned object class. Hence, YOLO outputs 45,927 predictions for each of the 864×864 input images. Objectness, class probability, and non-maximum suppression thresholds of 0.200, 0.250, and 0.200 respectively are applied to obtain the predictions corresponding to the 80 trained features. This quickly narrows the search by filtering out any predictions with values below such thresholds. Of the remaining predictions, feature assignment is performed based on highest class probability and only the highest objectness scoring prediction per class is retained. Finally, the bounding box (x, y) center coordinates are stored as 2D feature image points and matched by class ID to their corresponding 3D model points. Any missing feature predictions are simply omitted from the list of 2D to 3D matches. The final output of this stage of the pipeline are two lists of 2D to 3D matches, one for each of the two objects (e.g., probe and drogue) observed in the image.

3.1.4 Estimating Dual Object Poses

Before computing poses from the above matches, it is important to note that the 3D local model reference frame points can be represented in a variety of different ways. To minimize the number of reference frame transformations needed in this pipeline, the receiver and probe are considered a single object, i.e., any point on the receiver is considered a probe point and vice versa. Furthermore, the object origins are defined as the probe tip, drogue center, and average drogue center (relative to the tanker) for the receiver, drogue, and tanker respectively. The axes of the three

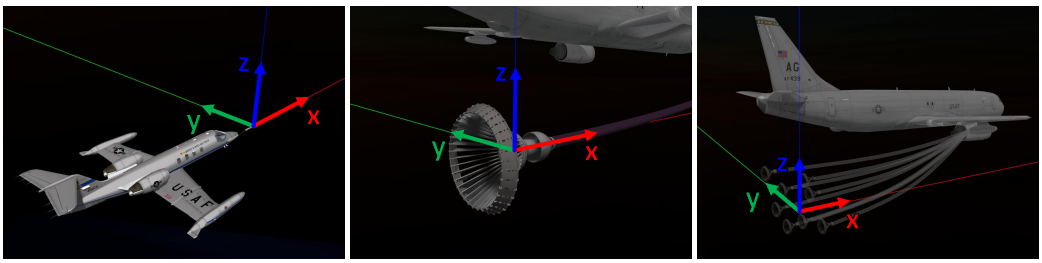


Figure 12. Local reference frames. Note that the tanker reference frame origin is located at a fixed offset from the tanker at the average drogue center.

reference frames are also aligned similar to the previously described drogue-centered local reference frame (recall from Figure 7) where the x, y, and z components of each reference frame point in the object forward, left, and up directions respectively, see Figure 12.

With 3D points defined in their corresponding local reference frames, the two lists of 2D to 3D matches can be transform directly into probe tip and drogue centric pose estimates using one of the three analytical algorithms mentioned in Section 2.2. In this effort, PnP was chosen—namely, OpenCV's RANSAC enabled *cv::solvePnPRansac* method [65]. In addition to the feature matches from the previous pipeline stage, the intrinsic camera parameters listed in Table 1 were also supplied in addition to enabling extrinsic guess based on the previous estimate and setting iterations count, reprojection error threshold, and confidence threshold to 500, 4.0, and 0.9999 respectively. This method subsequently outputs each 6DoF pose estimate in the form of a Rodrigues rotation vector, *rvec*, and a z-forward translation vector, *tvec*.

3.1.5 Computing PtD Vectors

Each object's resulting vector pair, *rvec* and *tvec*, represents its estimated 6DoF pose in the camera's local reference frame. Using OpenCV's *cv::Rodrigues* method [65], the probe's *rvec* is converted into a direction cosine matrix, R_p^c, which transforms probe frame translation vectors, t^p, into camera frame vectors, t^c, such that:

$$t^c = R_p^c t^p \tag{2}$$

Maintaining consistency with the reference frames defined in Figure 12, the camera frame z-forward *tvec* for both the probe and drogue are also converted into camera frame x-forward translation vectors, t_p^c and t_d^c respectively. To obtain the receiver frame PtD vector, i.e., probe frame drogue vector, t_d^p, the camera frame PtD vector, $t_{p\to d}^c$, is first computed by subtracting the camera frame probe vector from the camera frame drogue vector:

$$t_{p\to d}^c = t_d^c - t_p^c \tag{3}$$

Then, the camera frame PtD vector is transformed into the receiver frame by multiplying it by the transpose of the probe's predicted direction cosine matrix:

$$t_d^p = t_{p\to d}^p = R_p^{c\top} t_{p\to d}^c \tag{4}$$

Theoretically, an autonomous receiver and tanker pair can use these PtD vector predictions to synchronize flight and perform autonomous aerial refueling—that is, as long as the predictions are accurate, reliable, and obtained in real time.

3.2 Evaluating Pipeline Performance

To show this pipeline can achieve the accuracy, reliability, and real time execution necessary for AAR operations, its performance is monitored while flying thousands of Monte Carlo approaches in simulation, capturing both a truth and predicted PtD vector every simulation frame.

3.2.1 Metrics

Accuracy is quantified as the magnitude difference, in meters, between the two vectors assessed as a function over distance between the probe and drogue. Based on the typical size of a drogue, acceptable error was also defined as accuracy within $\pm 7cm$ at contact. Execution *speed* is measured as average time in milliseconds per prediction and decomposed into individual pipeline operations. Finally, *reliability* is defined here as the percentage of successful predictions returned over the total number of predictions attempted. For example, PnP needs at least three matches to predict a pose. Thus, prediction attempts are considered unsuccessful, for example, when YOLO only finds two features in an image since PnP cannot make a prediction. Furthermore, some other relative navigation technique (e.g., GPS) is assumed to be used until the probe and drogue are within $120m$ of each other. Therefore, we automatically consider any predicted PtD vectors greater than $120m$ in magnitude as unsuccessful outliers.

3.2.2 Zoom Dilemma

Training a model to perform relative vectoring across the entire $120m$ approach range presents some inherent challenges. With the camera zoomed out, the drogue may occupy less than a few pixels at long range—making feature detection on it impossible. Similarly, zooming in to enlarge drogue features also limits visibility of the probe, a problem that hinders dual object detection and only gets worse at a closer range. To overcome this dilemma, *model switching* is implemented. That is, multiple models, each trained on unique features specific to a particular camera configuration (i.e., vantage point, orientation, and zoom), are deployed and switched between throughout different phases of the approach. This allowed the receiver to navigate toward the drogue based on tanker features until the drogue features were

large enough to detect with a different model.

3.2.3 Experiments

The following research questions motivated the experiments in this chapter:

1. Can relative vectoring using dual object detection with model switching effectively guide the probe to the drogue?
2. Does a YOLO model have enough capacity to learn the entire approach, or should it specialize within a specific range?
3. Are bounding box corrections really necessary?
4. What features are best suited for the relative vectoring pipeline—random versus specific?
5. Does feature size matter?
6. How does YOLO feature detection compare to true pixel projections with varied levels of random noise?
7. Can YOLO find occluded features based on surrounding spatial information?

Twelve trained YOLO models, as listed in Table 3 and depicted in Figure 13, helped answer these questions. Models A through C were trained to detect features on the drogue and facilitate the receiver navigating to it directly while on final approach, leading up to contact. In contrast, models D through L were trained to detect features on the tanker when further out, guiding the receiver to the general proximity where the drogue should be until within close range. Configuration parameters such as feature selection and training range were also varied according to their corresponding model training description.

Table 3. Experimental camera and feature configurations for model training.

Model ID	Camera view[a]	Training range	Testing domain[b]	Model training description
A	podcam1	0-20 m	1	Rear facing (zoomed in)
B	podcam2	0-120 m	1	Rear facing (zoomed out)
C	dashcam	0-20 m	1	Forward facing
D	wingcam	0-120 m	1-6	Full range (baseline wingcam config[c])
E	wingcam	0-25 m	2	Close range
F	wingcam	25-50 m	2	Mid-range
G	wingcam	50-120 m	2	Long range
H	wingcam	0-120 m	3	Uncorrected bounding boxes
I	wingcam	0-120 m	4	Random features
J	wingcam	0-120 m	5	Small features
K	wingcam	0-120 m	5	Medium features
L	wingcam	0-120 m	5	Large features

[a] *podcam1*, *podcam2*, and *dashcam* maintained a 15, 25, and 75 degree hFOV respectively and corresponding models trained on drogue features while *wingcam* maintained a 55 degree hFOV and trained on tanker features.

[b] Testing domains included: 1) model switching, 2) full vs limited range training, 3) bounding box corrections vs no corrections, 4) random vs specific features, 5) feature size, and 6) random pixel error.

[c] This configuration included forward facing tanker tracking, full range training, corrected labels, and strategically chosen small to medium sized features.

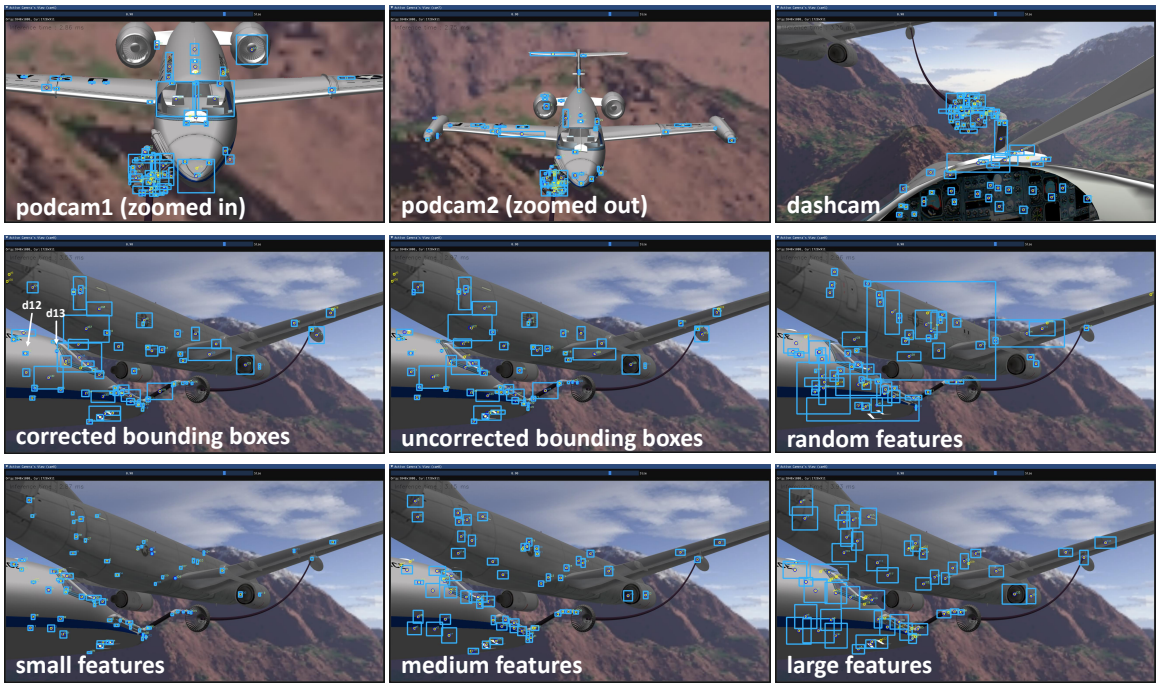


Figure 13. Experimental vantage points. Light blue rectangles and dark blue circles depict YOLO predicted bounding boxes and corresponding bounding box center points respectively, while the yellow circles depict true 2D feature points.

3.2.4 Feature Selection

Models D through H all trained on 80 common features, while the other models each trained on a unique set of features customized to test a particular aspect of general feature selection. For example, model J was trained on small features, model K trained on features co-located with and approximately quadruple the size of that of model J, and so on, see bottom row of Figure 13. Furthermore, the features used to train model D served as the baseline for the wingcam configurations and enabled us to compare pipeline performance with YOLO detections versus truth pixel feature projections perturbed with Gaussian noise.

Overall, over 250,000 predictions were collected across several hundred simulated Monte Carlo approaches for each of the experimental camera and feature configurations; one for each trained model and four additional collects for truth pixel projections (i.e., with YOLO disabled). For the truth projection collects, pixel noise was

applied such that the true feature 2D image points were perturbed from the truth in a random direction within a fixed threshold radius per data collect prior to the solve PnP stage. Threshold radii 0.0, 0.5, 1.0, and 2.5 pixels were used respectively. Finally, the mean, variance ($\pm 2\sigma$), and prediction error magnitudes[2] were plotted for each collect as a function of distance between the probe and drogue.

3.3 Results and Discussion

3.3.1 Qualitative analysis

Surprisingly, YOLO was highly effective at finding the 2D points corresponding to the 3D geometric feature centers once trained with bounding box corrections—even when trained to find symmetric and randomly selected features on the drogue. As depicted in Figure 13, most YOLO predictions appear closely aligned with the truth (i.e., yellow circles inside blue circles). There were instances where YOLO did not find a particular feature at all or was inaccurate by a considerable margin. However, using RANSAC during pose estimation helped filter out such outliers.

In simulation, as demonstrated in the video [8] from 7:14-24, PtD vector accuracy was not effected by significant camera orientation changes, showing pipeline resiliency to unknown and changing extrinsic parameters. The vectors also become noticeably more accurate the closer the receiver gets to the drogue. In simulation, the receiver always made successful contact with the moving drogue when "navigating" using predicted relative vectors. Also, several instances were observed in which YOLO accurately found occluded features, clearly indicating YOLO performs object detection using global spatial reasoning and not just pixels localized within close proximity to the predicted bounding box. This phenomenon is evident in the middle left image of

[2]Note that only a randomly sampled subset of error magnitudes are actually plotted to visualize the prediction spread.

Figure 13, where YOLO finds features d12 and d13 just behind the receiver cockpit windshield—these are actually 3D features on the tanker's left wing!

3.3.2 Quantitative Analysis

Figures 20 through 23 (see Appendix A) show how random pixel error effects the pipeline and validates its accuracy and reliability when precise 2D image points are found. Consequently, this also indicates the pipeline's machine learning 2D feature finder is the most critical part of the pipeline. When replacing perturbed truth points with actual YOLO predictions, Figures 24 through 35 indicate certain models performed better than others at different ranges. This validates the need for model switching. Although model C performed the best drogue tracking just prior to contact (within 2.5 meters), model A performed more consistently and with higher reliability leading up to contact from 20 meters out. Also, model D performed well across the entire range from 120 meters out, but navigated toward the drogue using tanker features and did not track the drogue directly. Selecting from only these three, any individual model would likely fail at navigating an entire AAR approach. However, the receiver could use model D to navigate within close range, then switch to model A from 20 to 2.5 meters out, switch again to model C to most accurately track the drogue during the final contact maneuver, and lastly switch back to model D to maintain synchronized flight with the tanker during refueling and perform a safe egress after disconnect.

In addition to showing pipeline viability using model switching, the other Monte Carlo results indicate individual YOLO models have sufficient capacity to learn the full range and need not specialize on a limited range. For example, model D outperformed models E through G, even within their corresponding specialized training ranges. Moreover, comparing results from model D to that of models H and I provide

evidence that bounding box corrections and intentionally chosen features improved pipeline performance. Furthermore, Figures 33 through 35 show medium size features produced the best results. The small features used to train model J were too small to detect past about 70 meters out, resulting in reduced accuracy and prediction reliability, while large features used to train model L produced less stable predictions at close range than that of model K.

3.3.3 Pipeline Speed

Table 4 shows the execution time for each pipeline operation. Experiments showed image processing (rendering, padding, scaling, etc.) and making YOLO predictions occupied the majority of the pipeline execution envelope. However, parallelizing the pipeline—that is, processing the current image while performing YOLO's forward propagation on the previous image—reduced execution time between predictions to approximately 22 ms, or 45.5 fps. This speed has high potential to meet the real time execution requirements of AAR.

Table 4. Pipeline execution time.

Pipeline operation	**Duration**
Virtual image capture	18 ms
YOLO prediction	22 ms
Pre-processing (making predictions)	19-21 ms
pad image (into square)	2 ms
blob from image	4-5 ms
network forward propagation	13-14 ms
Post-processing (interpreting predictions)	<1 ms
Remainder of pipeline	<1 ms
Total	**41 ms**

3.4 Summary

In this chapter, a 5-stage relative vectoring pipeline was presented that uses machine learning to find corrected 2D image points corresponding to 3D model points. The pipeline ultimately transforms the points into probe tip to drogue center vectors relative to the receiver aircraft's local reference frame. This provides the receiver enough information to navigate its probe into the drogue and maintain synchronized flight with the tanker throughout the entire refueling process. As part of this pipeline, an automated image labeling technique was proposed, which includes bounding box corrections for precise error free supervised machine learning. A technique call dual object detection was also proposed that transforms pose estimates of two objects observed within the same image into a relative vector between the two. Experimental results through Monte Carlo simulation showed that this pipeline effectively produces accurate, reliable, and real time predictions, is resilient to occlusions, and does not rely on extrinsic camera parameters.

Unfortunately, the pipelines has some prominent limitations. First, it relies on accurate rigid 3D object models. This pipeline would likely not perform well on objects that undergo significant change in shape or size midflight, e.g., features on long wings susceptible to large flex. Also, individual YOLO models must be uniquely trained based on camera and feature configurations. For example, a YOLO model trained from a forward-facing receiver vantage point cannot simply be moved to a rear-facing tanker vantage point and achieve the same performance without further training. The more obvious drawback to this pipeline currently is that we have only showed success in simulation.

To move this pipeline forward into the real world, future work could include applying more realistic 3D digital twins with accurate flight dynamics models. Computer generated scenes today are becoming more realistic and indistinguishable from real

Figure 14. Learjet nose cone and refueling drogue in a Vicon motion capture room.

world imagery. Training YOLO in such an environment will have a better chance at successful transfer learning to real AAR scenarios. Another way to bridge this pipeline into the real world is through the use of augmented reality. Using a motion capture system to project known 3D truth into images—a technique later outlined in Chapter IV—can enable precise and automated feature labeling, such as projecting corrected bounding boxes as described in Section 3.1.2, but on augmented images with real aircraft parts, see Figure 14.

Other future work could also include the application of Kalman filters to remove noise from predictions, pipeline optimizations to obtain better combinations of input image sizes, aspect ratios, augmentation methods, machine learning hyperparameters, feature types, etc., and automating feature selection through the use of other feature detection techniques (e.g., SURF and SIFT). YOLOv5 was chosen in this chapter for 2D feature point detection, but perhaps future research could also apply newer object detectors for increased accuracy, reliability, and speed—e.g., YOLOv8 was recently released.

Although this effort primarily focused on applications in autonomous aerial refueling, the proposed relative vectoring pipeline has several applications in other research fields. Examples include self-driving cars, spacecraft rendezvous in orbit, automating helicopter landings, micro aerial vehicle swarms, and just about any other problem involving relative navigation between multiple objects using imagery.

IV. Transitioning Relative Vectoring to the Real World

Chapter III outlined the relative vectoring pipeline and demonstrated how it can be effectively deployed in simulation. However, it required the automation of image labeling with scene augmentation to satisfies the supervised machine learning pipeline requirements, a capability not readily available outside the virtual world. This chapter proposes a machine transfer learning technique designed to address real world limitations on automated image labeling. This topic will be the focus of research for the coming year.

4.1 Transfer Learning

Machine transfer learning, in the context of training artificial neural networks, involves a source training task in which a prediction function is defined and parameters of the function are updated (typically through a process known as backward propagation) to minimize output error after feedforward of input features with known output labels from a source domain. The transfer occurs when the source prediction function is further trained on input features with corresponding output labels from a target domain [7]. In this effort, YOLO is the prediction function while synthetic and real images compose the source and target domains respectively. Furthermore, *negative transfer learning* occurs when a transfer method decreases model performance while a *positive transfer* increases performance [66]. Thus, the goal in this effort is to minimize negative transfer while maximizing positive transfer.

Figure 15. Image of a real airplane on top vs a synthetic image of an existing model on the bottom.

4.1.1 Digital Twins

But why use transfer learning at all? Why not only use real imagery? First, experimenting with real imagery is very expensive (think cost of pilot, flight crew, aircraft, jet fuel, etc.), especially when customized equipment and infrastructure installed on the aircraft is needed to validate results. Automating accurate labeling of such images also presents challenges—3D truth is not available as it is in simulation and real cameras suffer from image distortion which could negatively impact truth projections and corresponding labels into the image frame. Transfer learning, however, may be the key to transitioning relative vectoring to the real world. As described in Section 3.1.2, simulations can produce tens of thousands of accurately labeled synthetic images every hour. These are generated from digital twins of real objects. The sources of these digital twins may originate from existing 3D models as well as 3D scans of real objects. As shown in Figures 15 and 16, synthetic imagery closely resembles real imagery and, when used to train YOLO, has high potential for positive transfer learning.

Figure 16. Image of a real nose cone on the left vs a synthetic image of its 3D scan on the right.

4.1.2 Motion Capture Image Re-projection

Automation of real image labeling is also feasible in a motion capture (mocap) space using calibrated cameras and visual targets as demonstrated in the written portion of my specialty exam, see Appendix B. The study validated a camera localization procedure using the visual targets and mocap 3D truth. Once localized, other features with known 3D truth, including 3D feature points for automated image labeling, could be accurately projected into the camera's image plane. In the study, manual target labeling was accomplished, which this effort will automate through the use of *AprilTag*, a popular visual fiducial system [67]. The AprilTag 36h11 family contains 587 unique tags, of which at least 50 will be used in this effort. Fig. 2 of Appendix B depicts the full mocap image re-projection pipeline. Real images generated from this pipeline will contain real aircraft parts, including a full scale tanker drogue basket and receiver nose cone with attached probe as shown in Figure 14. With accurate world space camera localization and 3D mocap truth, we can suspend real features in simulated scenes through the use of augmented reality. By incorporating real textures, hybrid images have an even higher potential for positive transfer learning than synthetic imagery alone.

Figure 17. Green screen with umbrella and softbox lighting.

4.1.3 Scene Augmentation

In addition to projecting 3D truth into virtual, real, and hybrid imagery, scene augmentation is an important step in promoting YOLO's ability to generalize predictions to previously unseen data. This includes variations in backgrounds, lighting effects, camera perspective relative to the objects, and orientation of the objects relative to each other. Scene augmentation is easily implemented programmatically in simulation. To implement background and lighting augmentation in the real world while employing mocap image re-projection, green screen backdrops will be used with varied placements of umbrella and softbox lighting, as shown in Figure 17. Perspective and orientation augmentation will be performed manually by varying the position of the camera and objects within the mocap space.

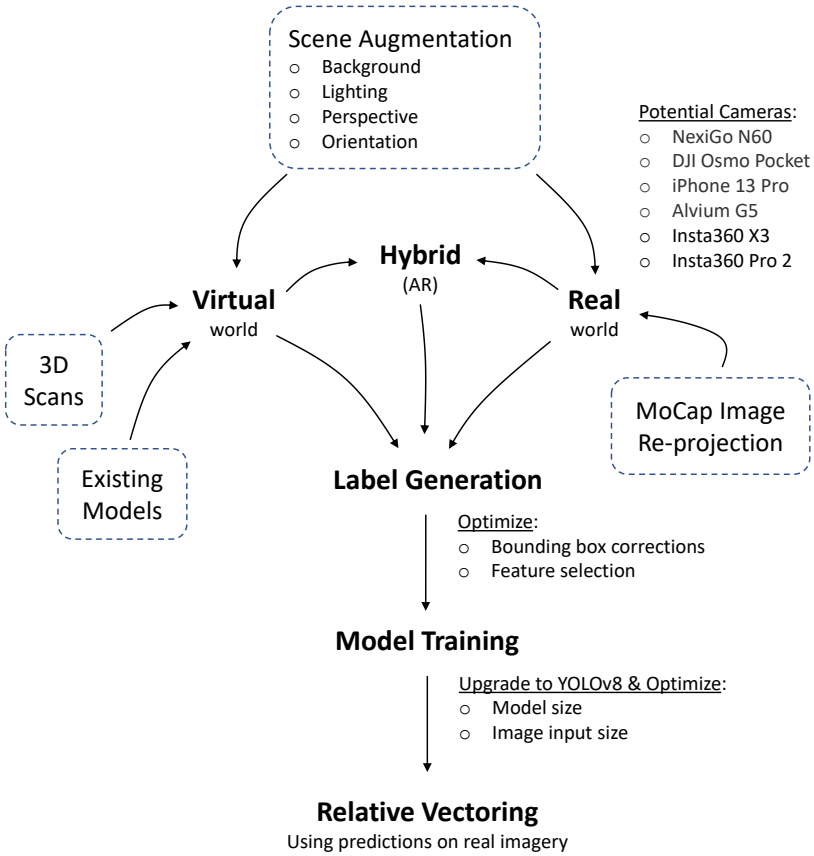


Figure 18. Transitioning relative vectoring to the real world using transfer learning.

4.1.4 Summary

Automation of accurate image labeling is necessary for the relative vectoring pipeline proposed in this work. Unfortunately, automation from real aerial refueling sorties is infeasible due to several factors including high cost and lack of 3D truth. Machine transfer learning from virtual, real, and hybrid imagery offers a potential solution that, when combined with scene augmentation, has a high likelihood of positive transfer learning to real aerial refueling scenarios. The proposed transfer learning approach, as summarized in Figure 18, serves one of the two major research topics that will be investigated over the next year and explores label automation across the three image domains as follows:

- Virtual World - synthetic images captured by virtual cameras and labeled using 3D truth projections, as described in Chapter III.
- Real World - real images captured from a variety of different cameras listed at the top right of Figure 18 and labeled using mocap data, as proposed in Section 4.1.2.
- Hybrid World (Augmented Reality) - a blend of virtual and real world images alignment with mocap data and corresponding digital twins.

4.2 Evaluation Methods

4.2.1 Motion Capture Validation

Once transitioned to the real world using transfer learning, the relative vectoring pipeline will be evaluated using metrics similar to those described in Section 3.2.1 (i.e., accuracy, reliability, and speed), but truth vectors will come from the mocap system instead of simulation. Both qualitative and quantitative analysis will be conducted. The nose cone and drogue basket shown in Figure 14 will serve as the objects of interest and the experiment will involve a feedback loop that guides the probe to the drogue using relative vectors.

4.2.2 Boeing Test Flight

Sponsor organizations are conducting autonomous aerial refueling flight testing in October later this year. They have invited our team to test our vision algorithms, including the relative vectoring pipeline proposed in this work, onboard these flights. Leading up to the test flights, I indent to travel early to perform 3D scans of the actual aircraft flown and capture several labeled images using a mobile mocap image re-projection setup for adequate machine transfer learning. YOLO models, fully

trained in advance, will be used to predict relative vectors during the test flights. At a minimum, both qualitative and quantitative analysis will be performed using collected data such as 3D truth if available (e.g., timestamped differential GPS data) during and after the test flights. These test flights will likely uncover publishable relative vectoring pipeline results.

4.3 Summary

Within simulation, the relative vectoring pipeline proposed in this work successfully enabled a receiver aircraft to navigate aerial refueling approaches and dock into a refueling drogue. This chapter proposes research to be conducted over the next year that will facilitate the transition of this pipeline into real world aerial refueling scenarios. More specifically, this entails the use of machine transfer learning to overcome the lack of automated labeling.

V. Timeline

The remaining research prior to my dissertation defense includes the transition of relative vectoring to the real world. This includes, but is not limited to, machine transfer learning and applications of omnidirectional imagery, as outlined in Chapter IV. The proposed timeline for this work is shown in Figure 19. The work outlined in Chapters I through III is currently pending publication and was submitted to the journal of *Neural Computing and Applications* on 28 April 2023. I intend for the remaining work outlined in Chapter IV to result in at least one other joural publication. A draft dissertation will be submitted to the committee no later than April 2024 with a defense date no later than July 2024.

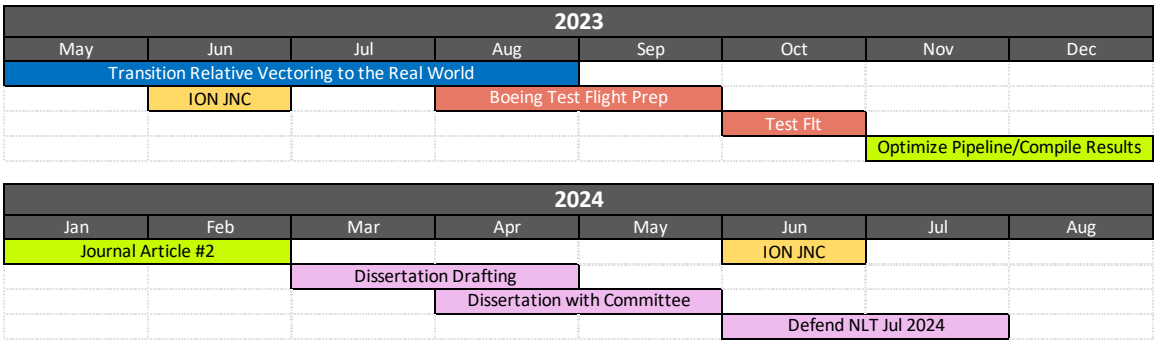


Figure 19. Proposed timeline.

Appendix A. Simulation Results

The plots in this appendix each consist of at least 300K relative vectoring observations, captured during several hundreds of random approaches in Monte Carlo simulation. Only a small subset of error magnitudes are plotted to visualize the spread of observations. The first 4 plots highlight pipeline performance when replacing YOLO predictions with true 2D image points perturbed with varied levels of random Gaussian noise. The other plots reflect pipeline performance when restricting model training to the camera and feature configurations listed in Table 3.

Error here is defined as the difference between true and predicted PtD vectors, as computed in simulation using floating point precision. Similarly, prediction reliability is defined as the percent of all predictions attempted that were valid. A prediction was considered valid if the pipeline found enough 2D image points to estimate a pose for both probe and drogue and the Euclidean distance between the two estimated poses was less than 120 meters. Anything beyond 120 meters was discarded as an outlier. Both error and prediction reliability was plotted as a function of true distance between the probe and drogue.

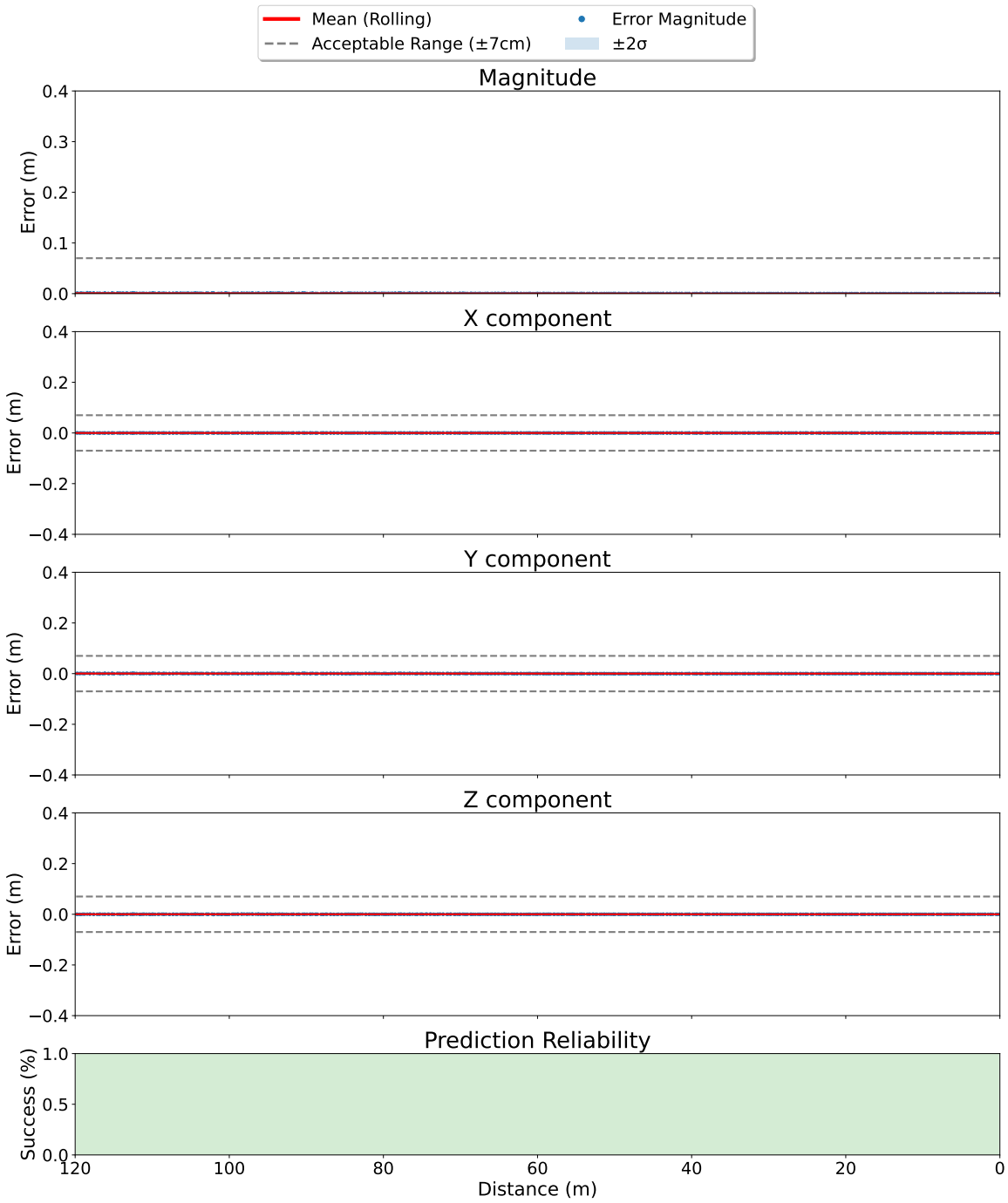


Figure 20. Pipeline results from truth projections (floating point precision) with zero random noise.

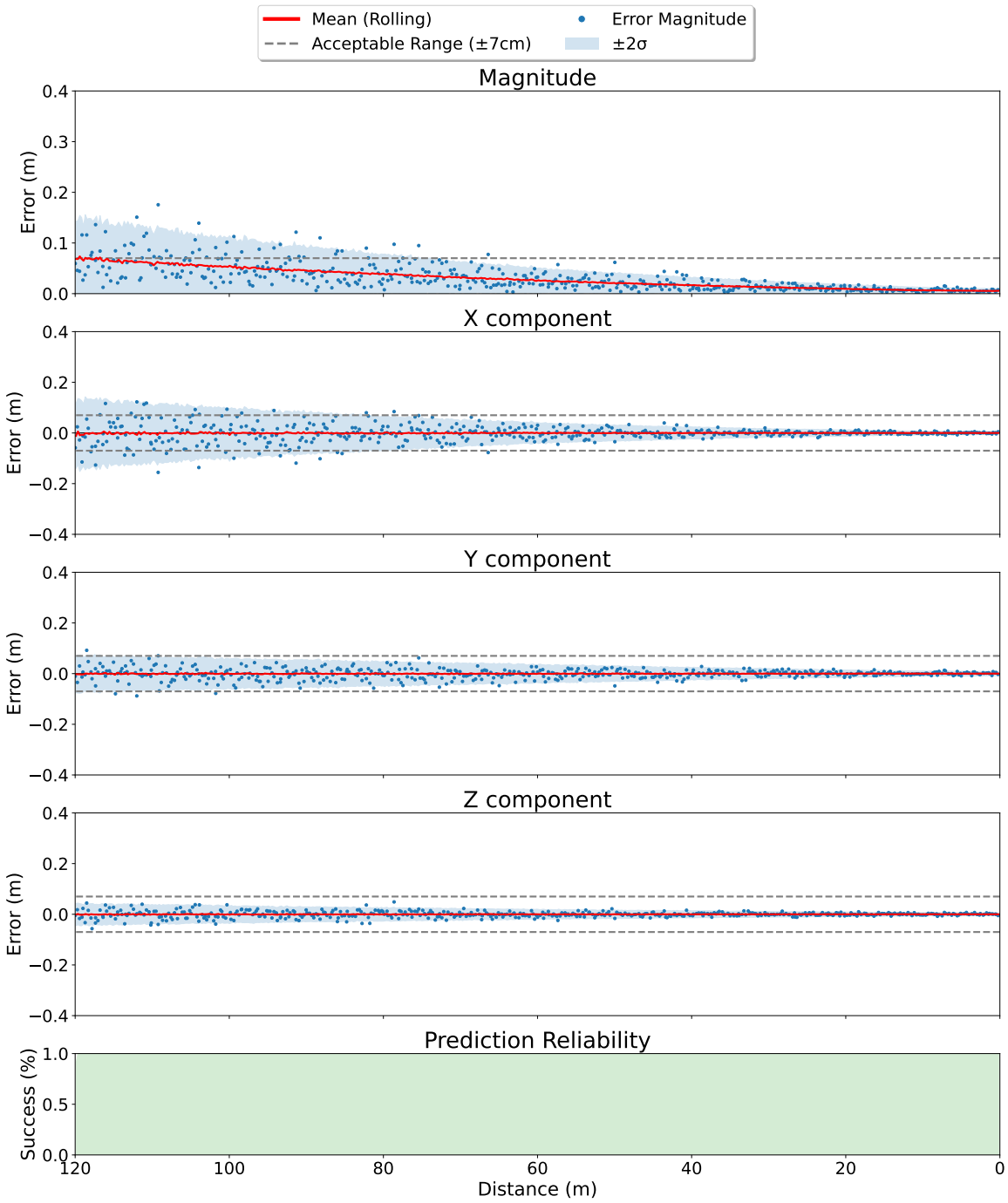


Figure 21. Pipeline results from truth projections perturbed with 0.5 pixels of random noise.

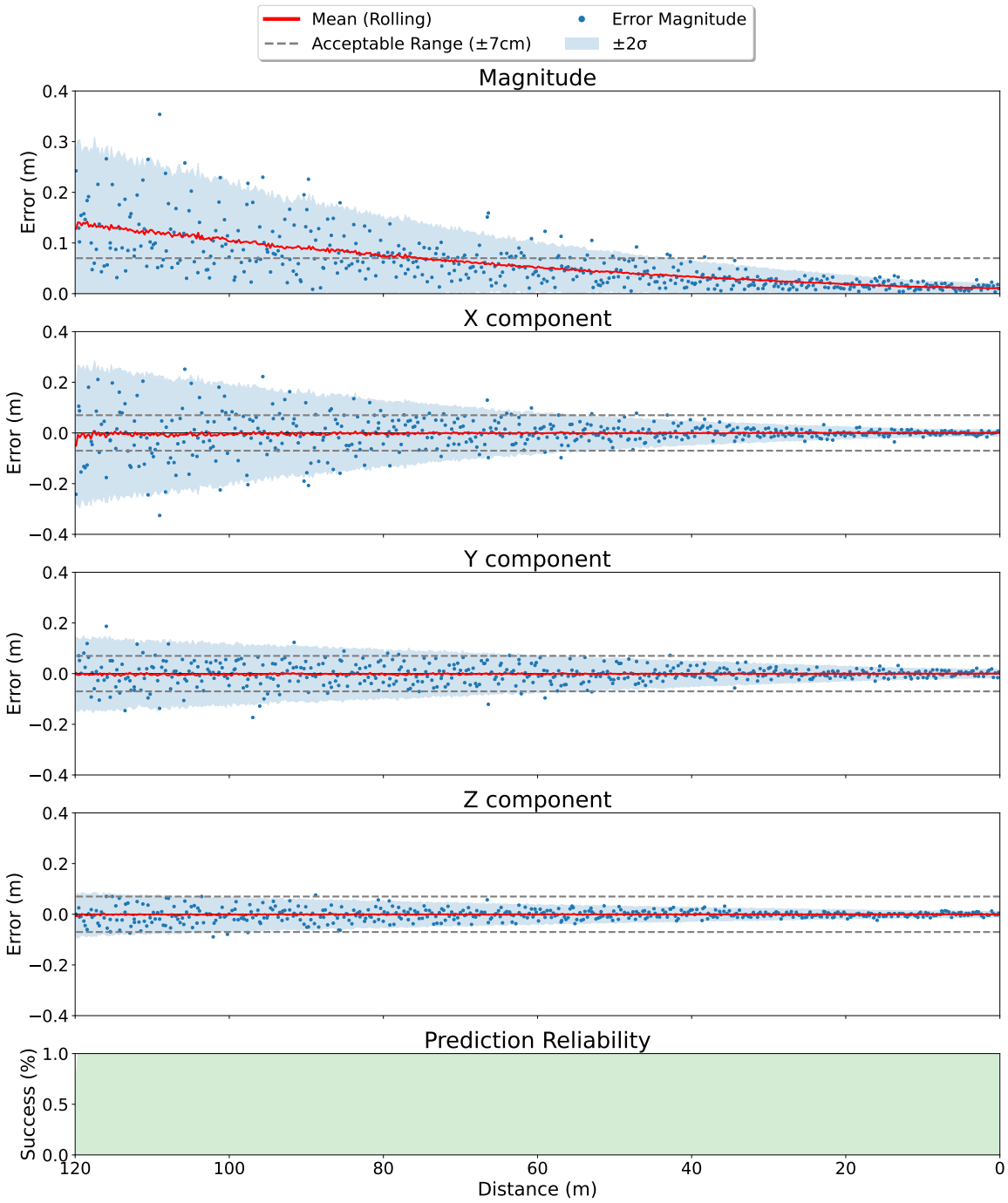


Figure 22. Pipeline results from truth projections perturbed with 1.0 pixel of random noise.

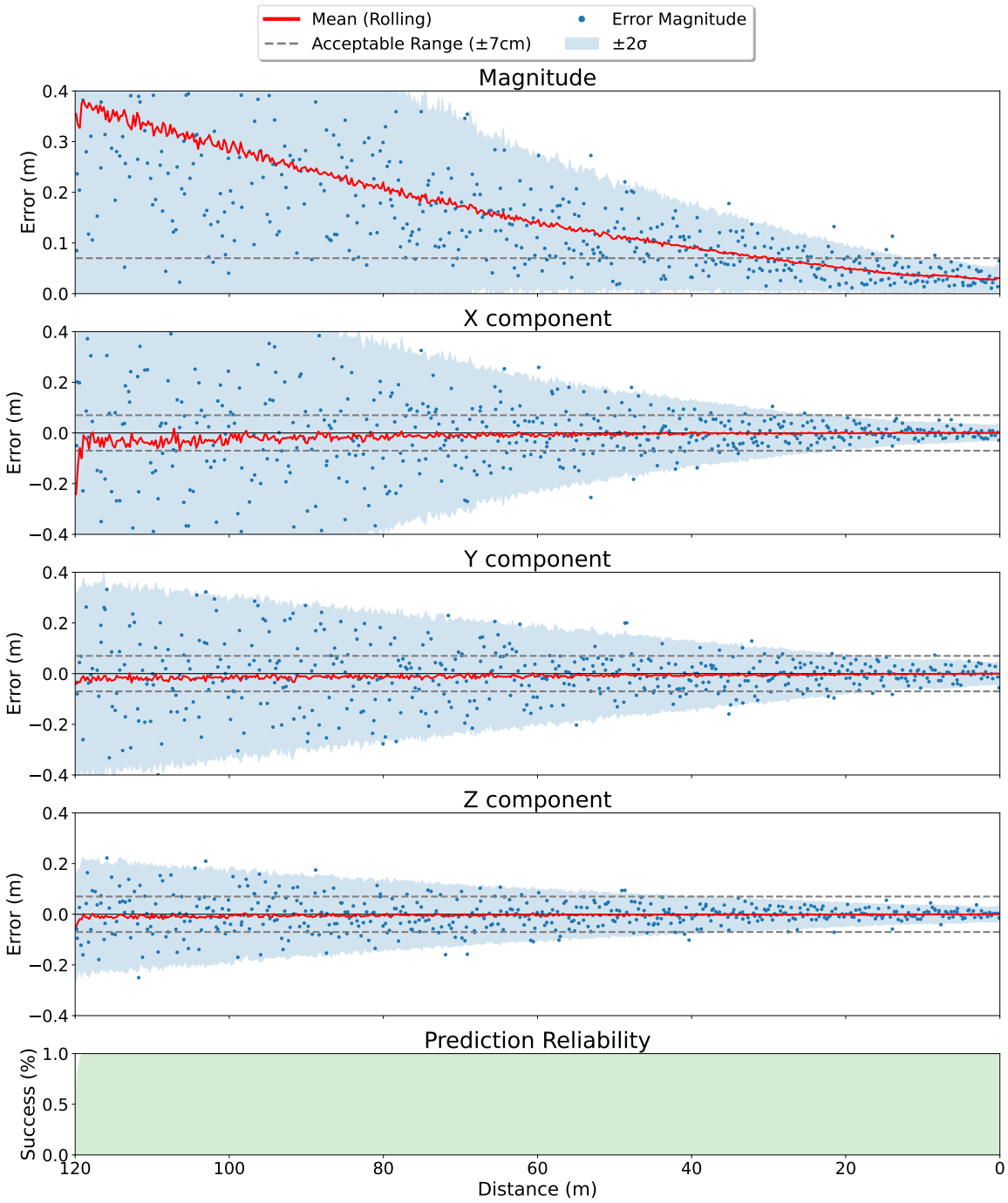


Figure 23. Pipeline results from truth projections perturbed with 2.5 pixels of random noise.

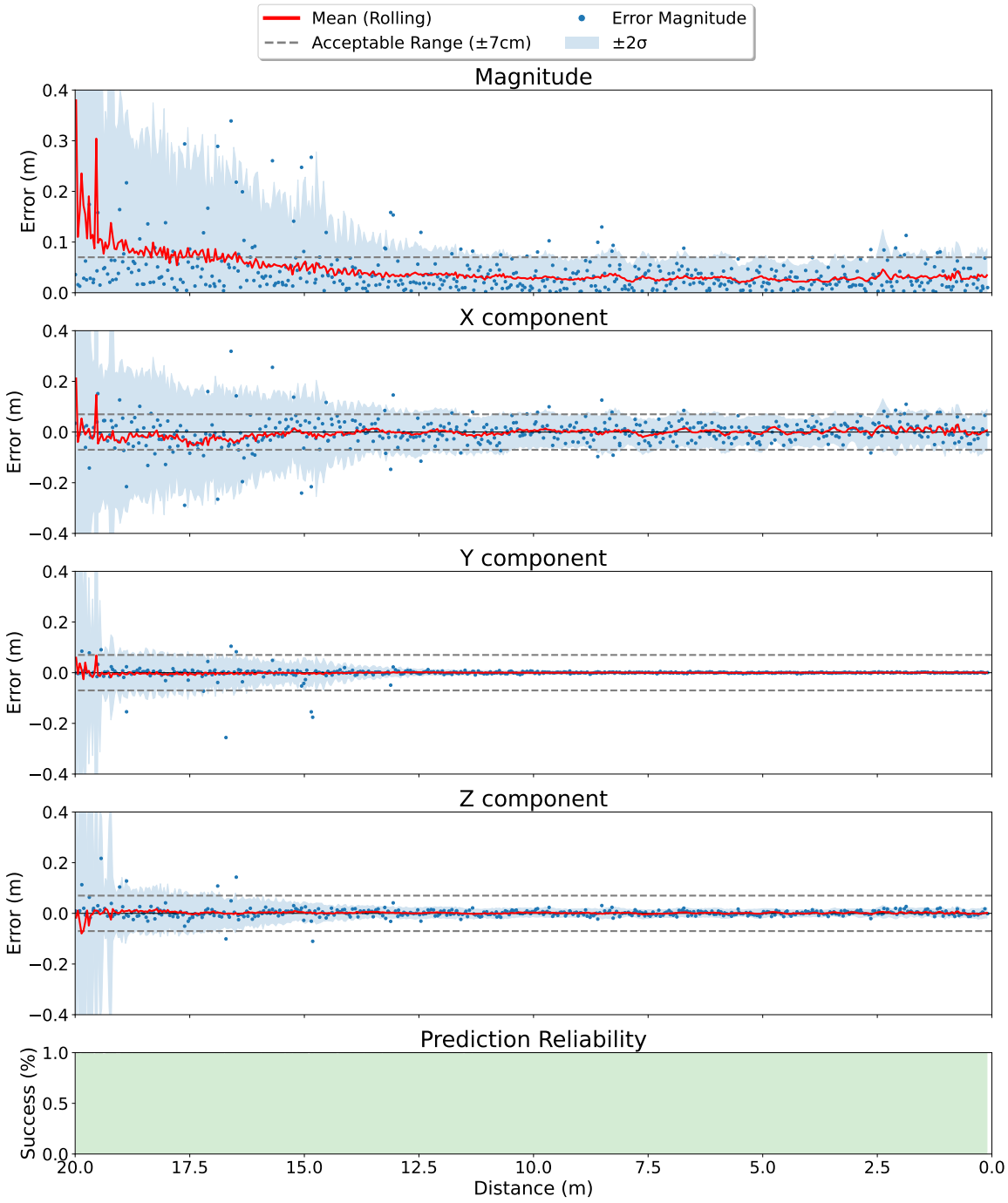


Figure 24. Pipeline results from tanker podcam trained while *zoomed in* (model A).

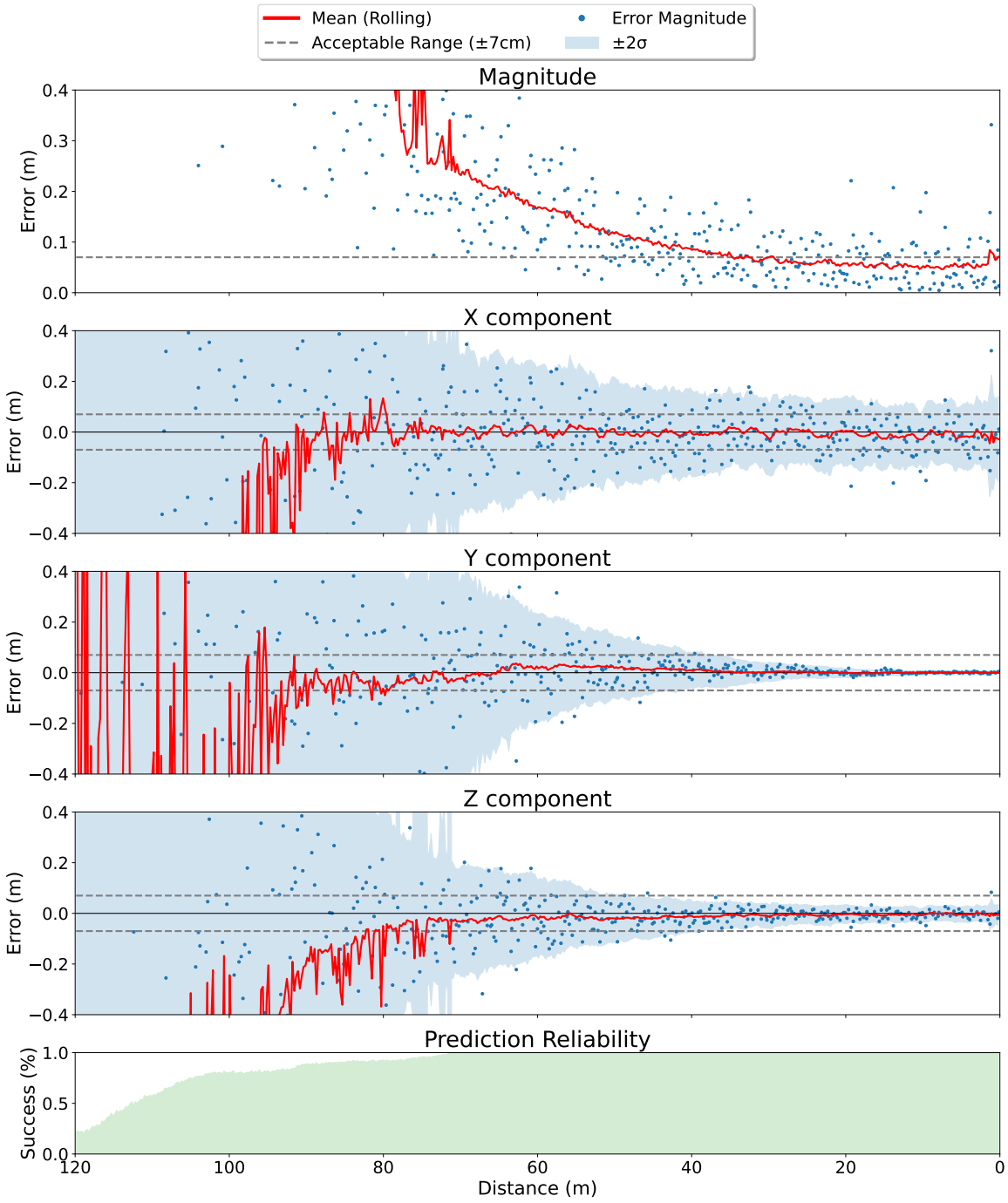


Figure 25. Pipeline results from tanker podcam trained while *zoomed out* (model B).

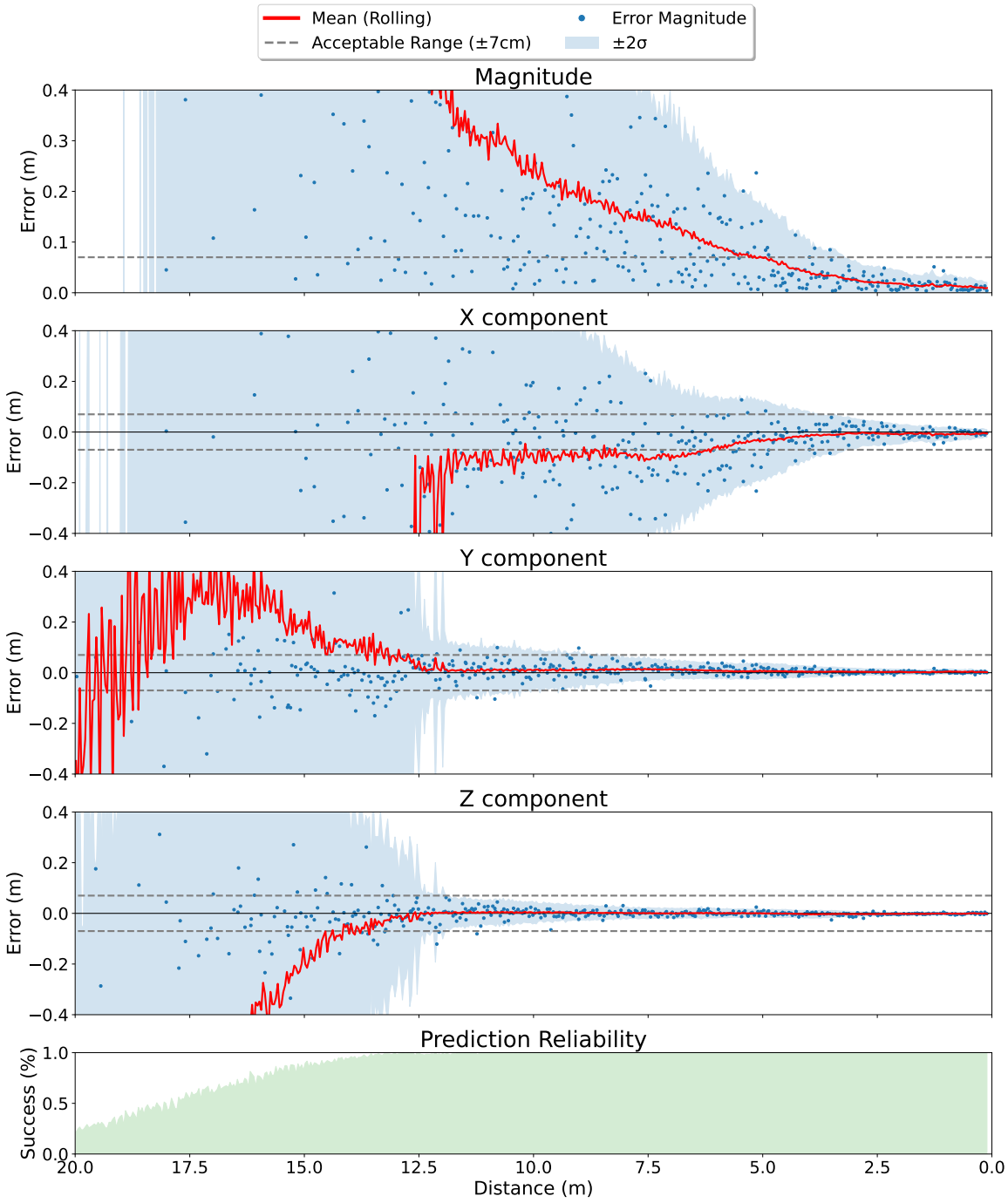


Figure 26. Pipeline results from receiver dashcam trained at *close range* (model C).

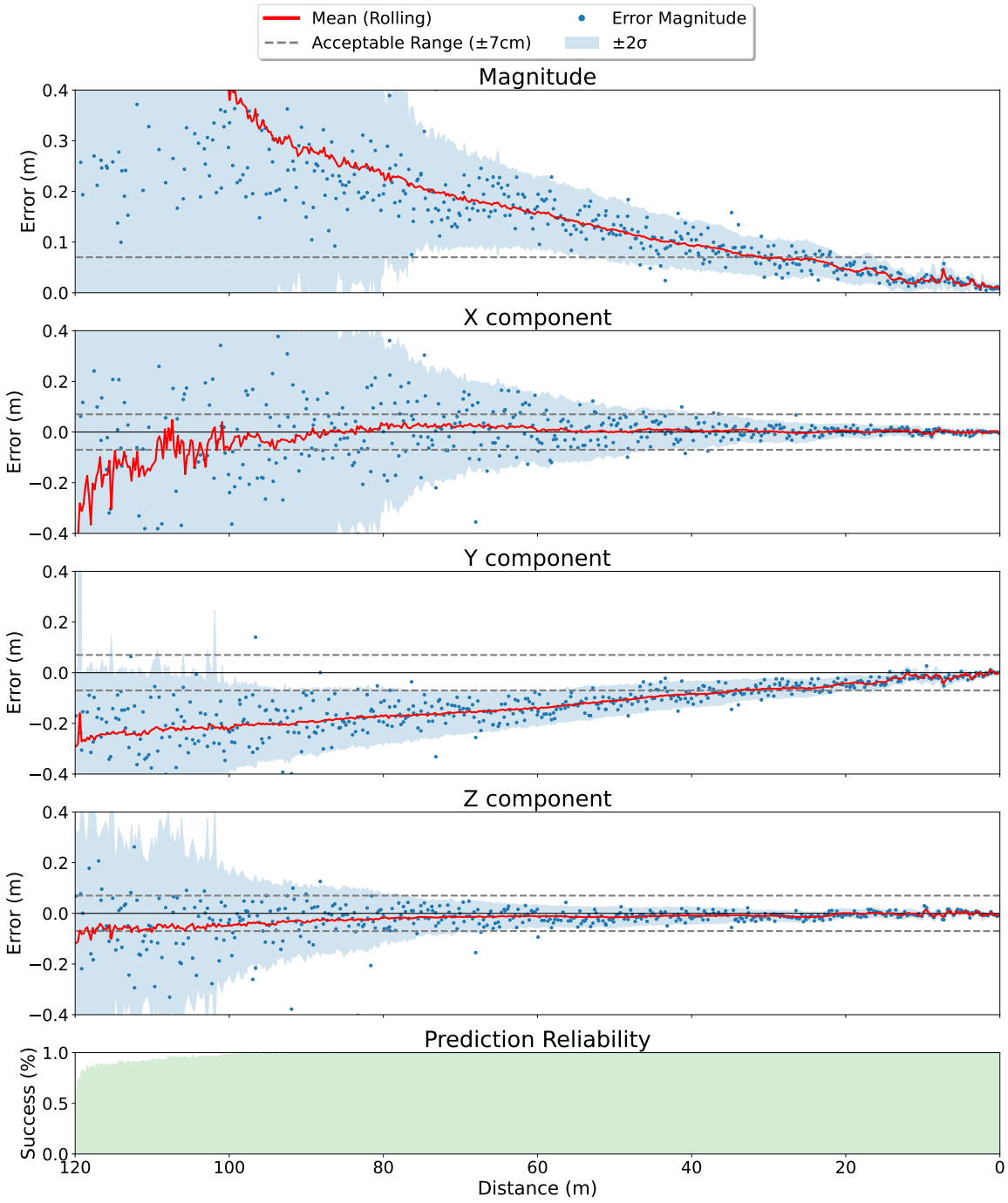


Figure 27. Pipeline results from receiver wingcam trained at *full range* (model D).

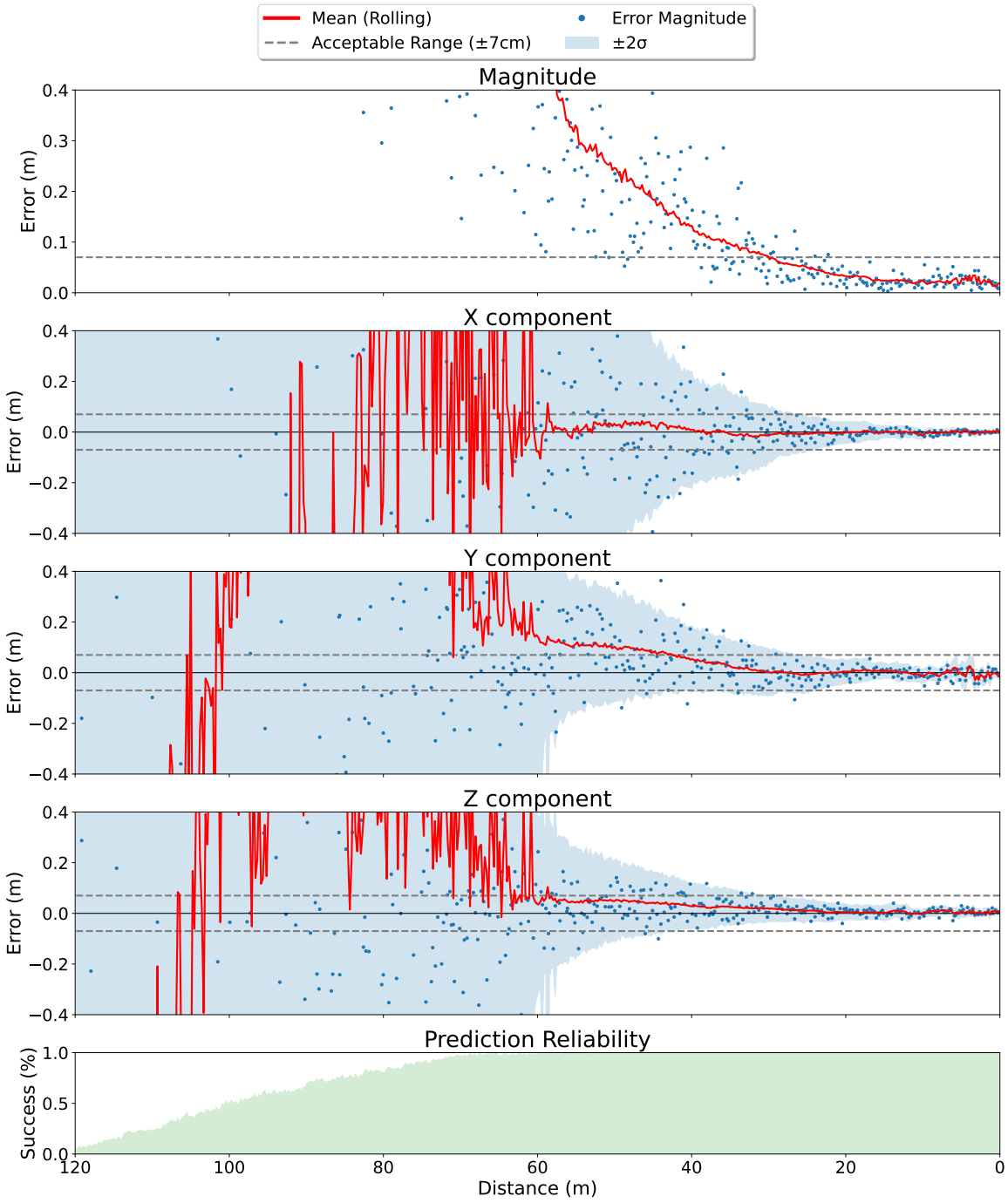


Figure 28. Pipeline results from receiver wingcam trained at *close range* (model E).

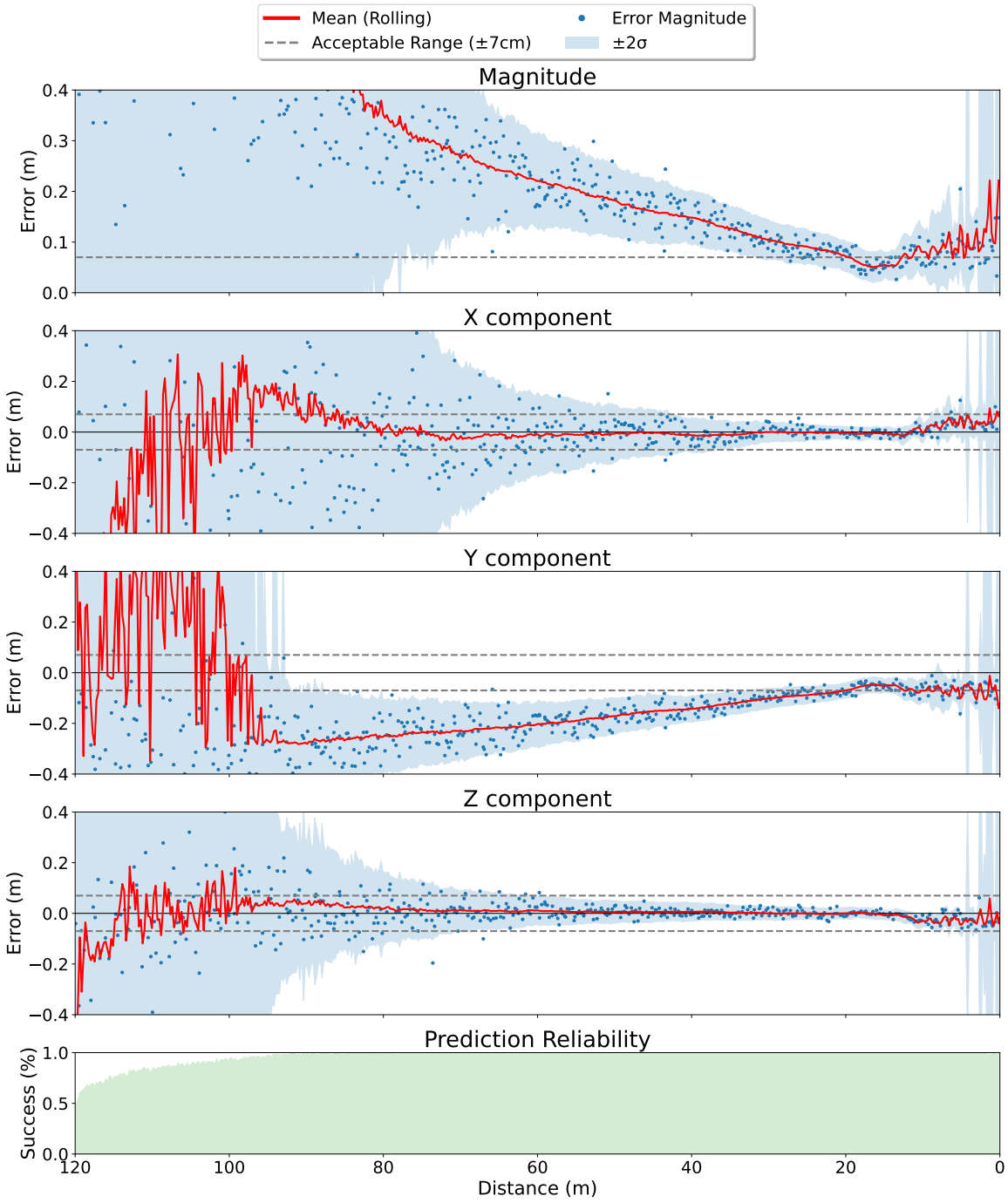


Figure 29. Pipeline results from receiver wingcam trained at *mid-range* (model F).

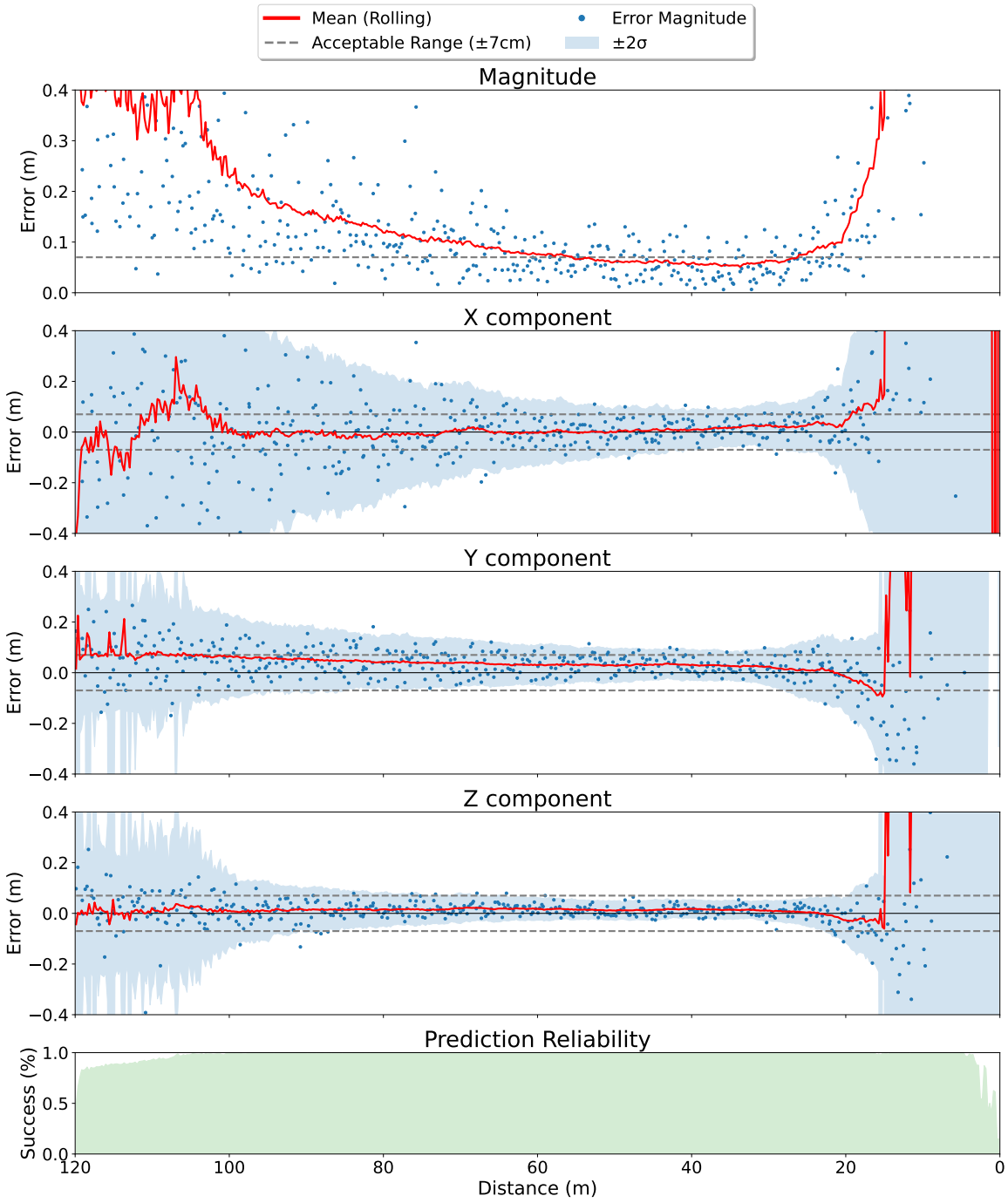


Figure 30. Pipeline results from receiver wingcam trained at *long range* (model G).

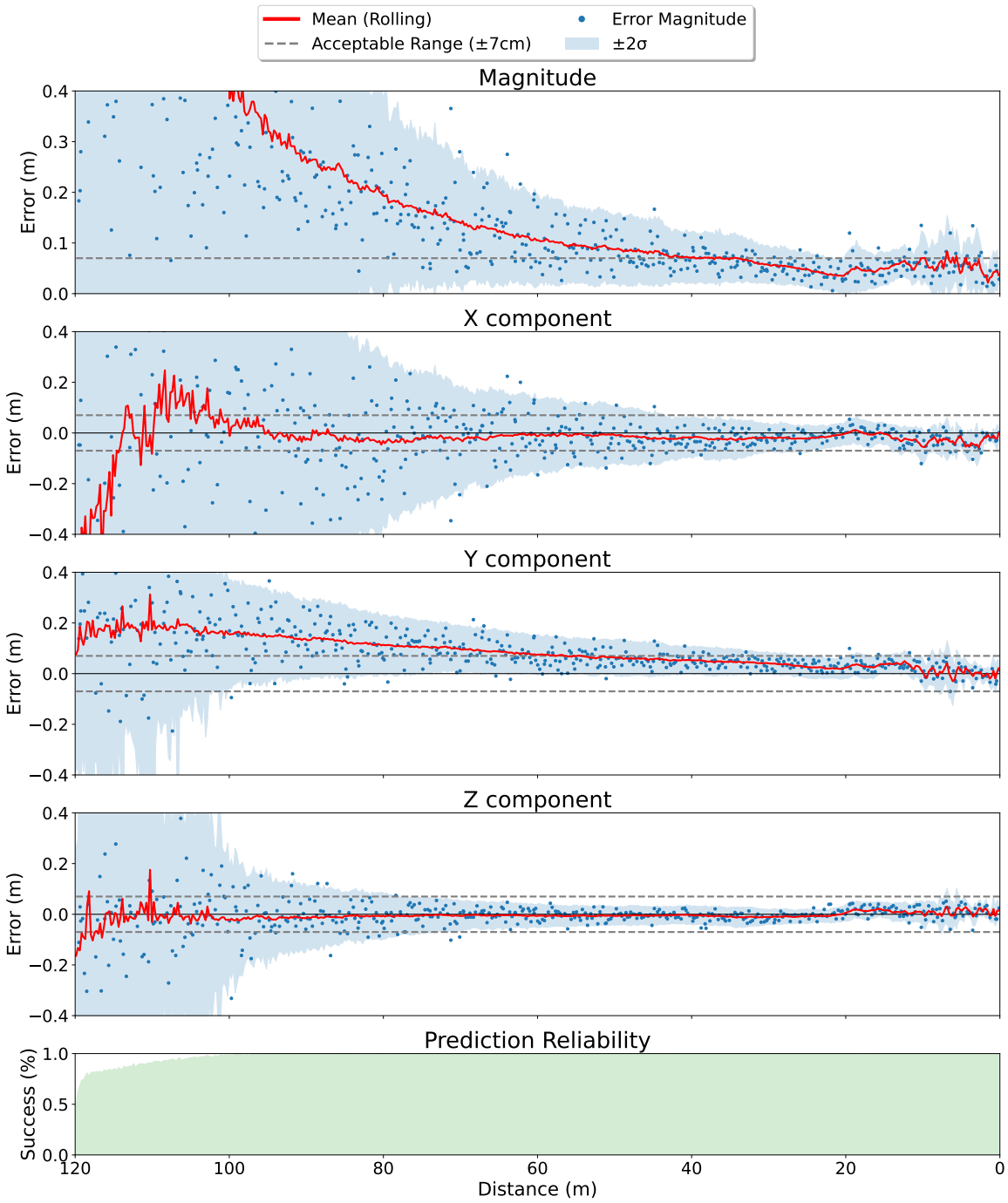


Figure 31. Pipeline results from receiver wingcam trained with *uncorrected bounding boxes* (model H).

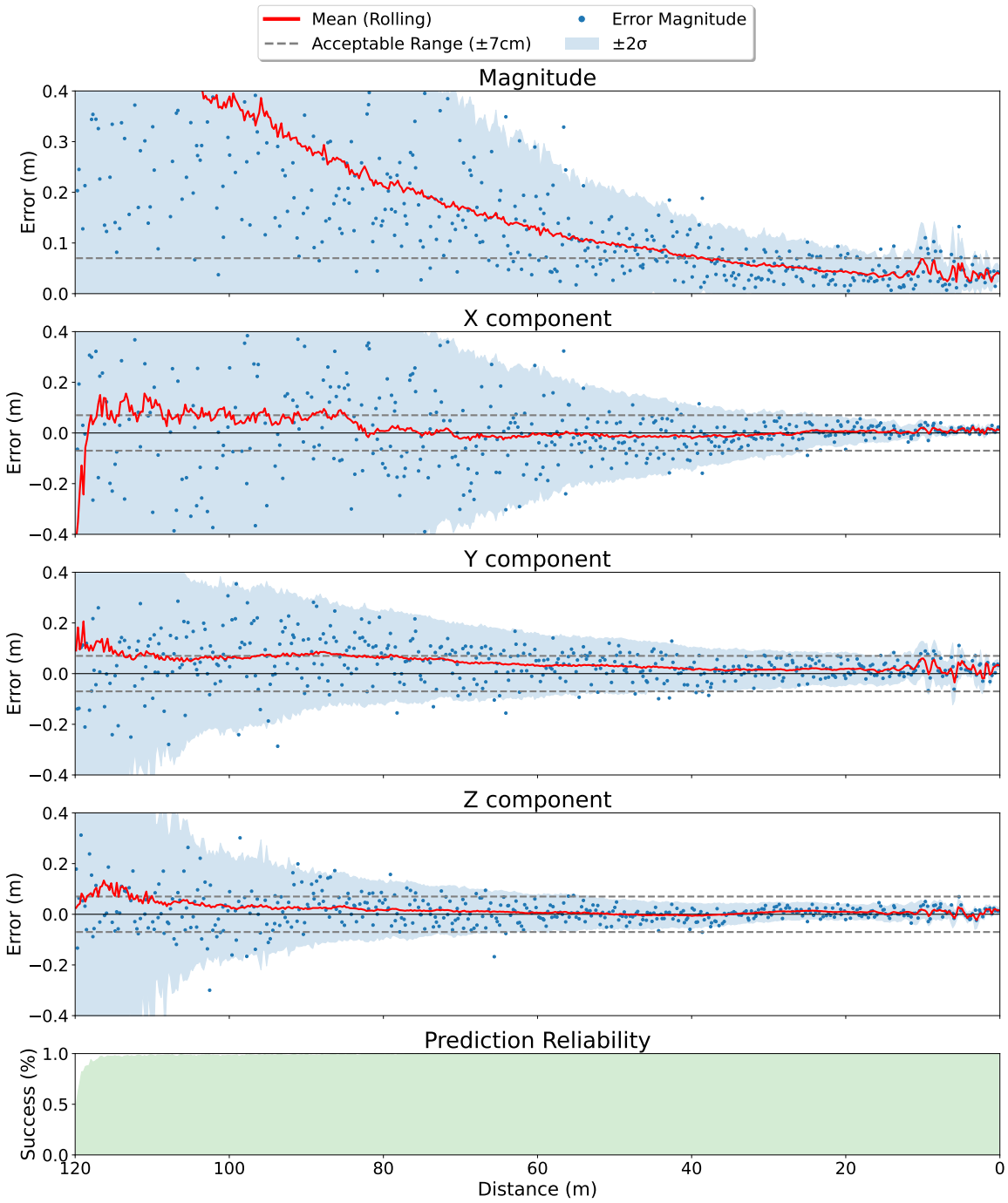


Figure 32. Pipeline results from receiver wingcam trained with *randomly selected features* (model I).

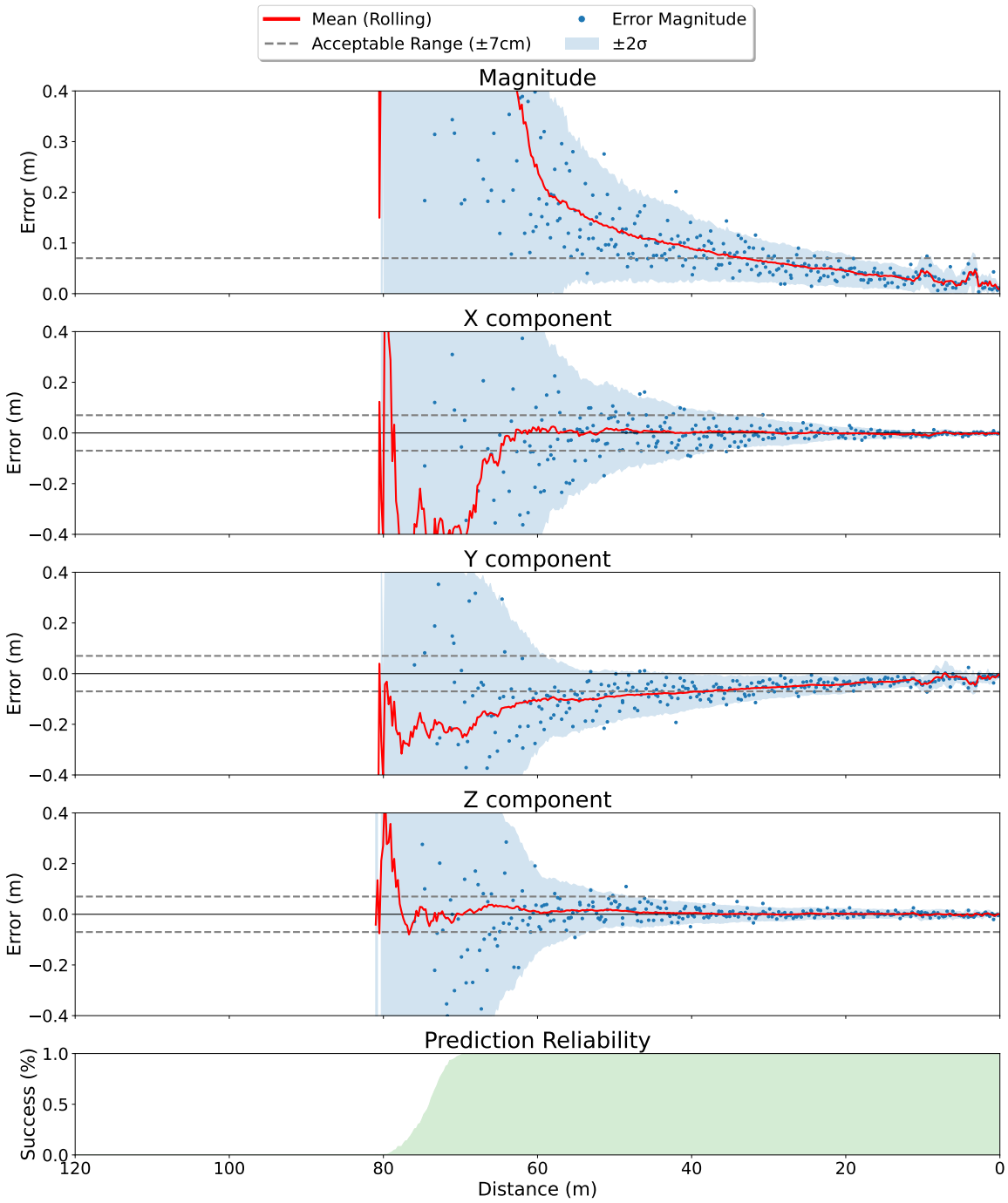


Figure 33. Pipeline results from receiver wingcam trained with *small features* (model J).

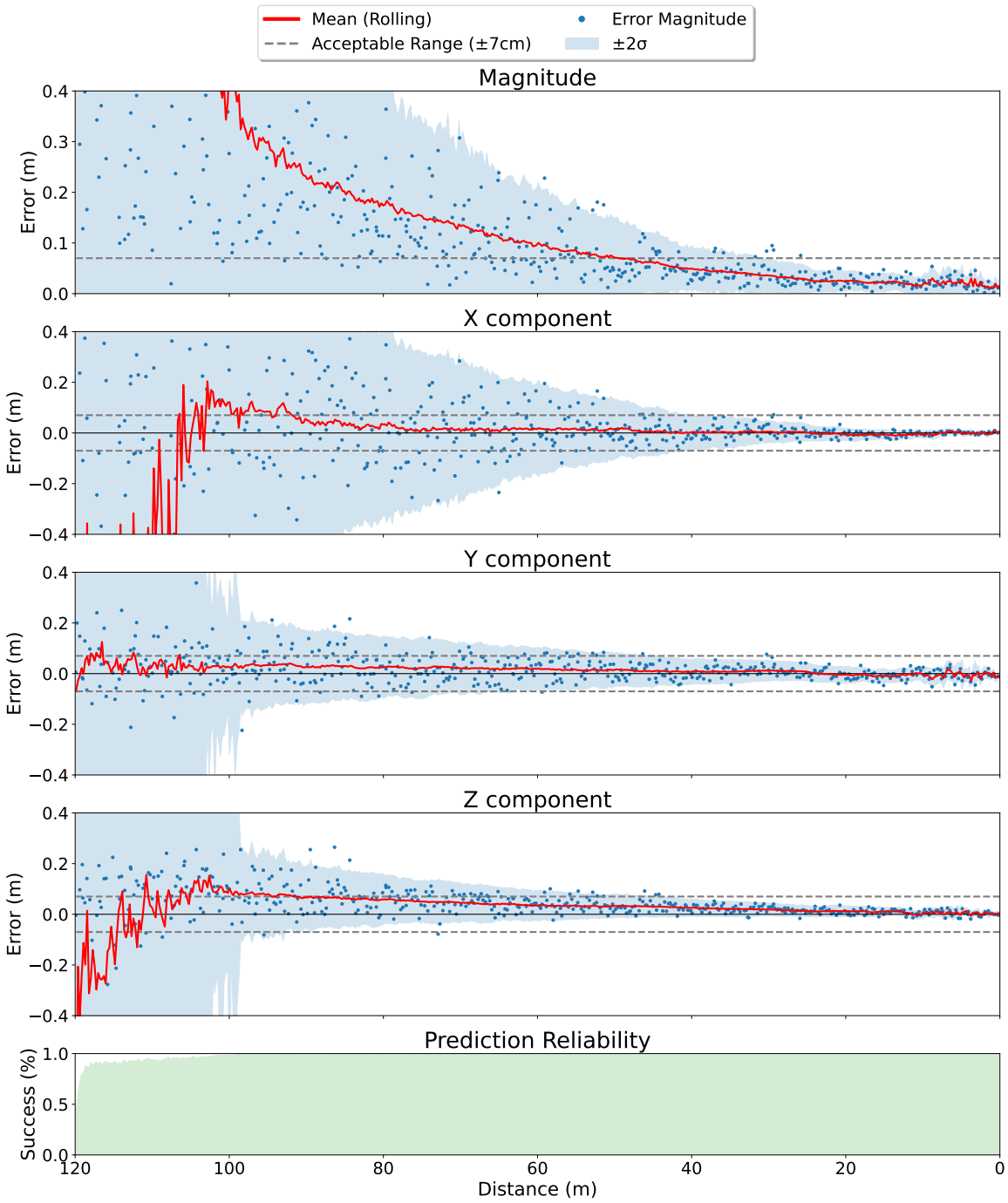


Figure 34. Pipeline results from receiver wingcam trained with *medium features* (model K).

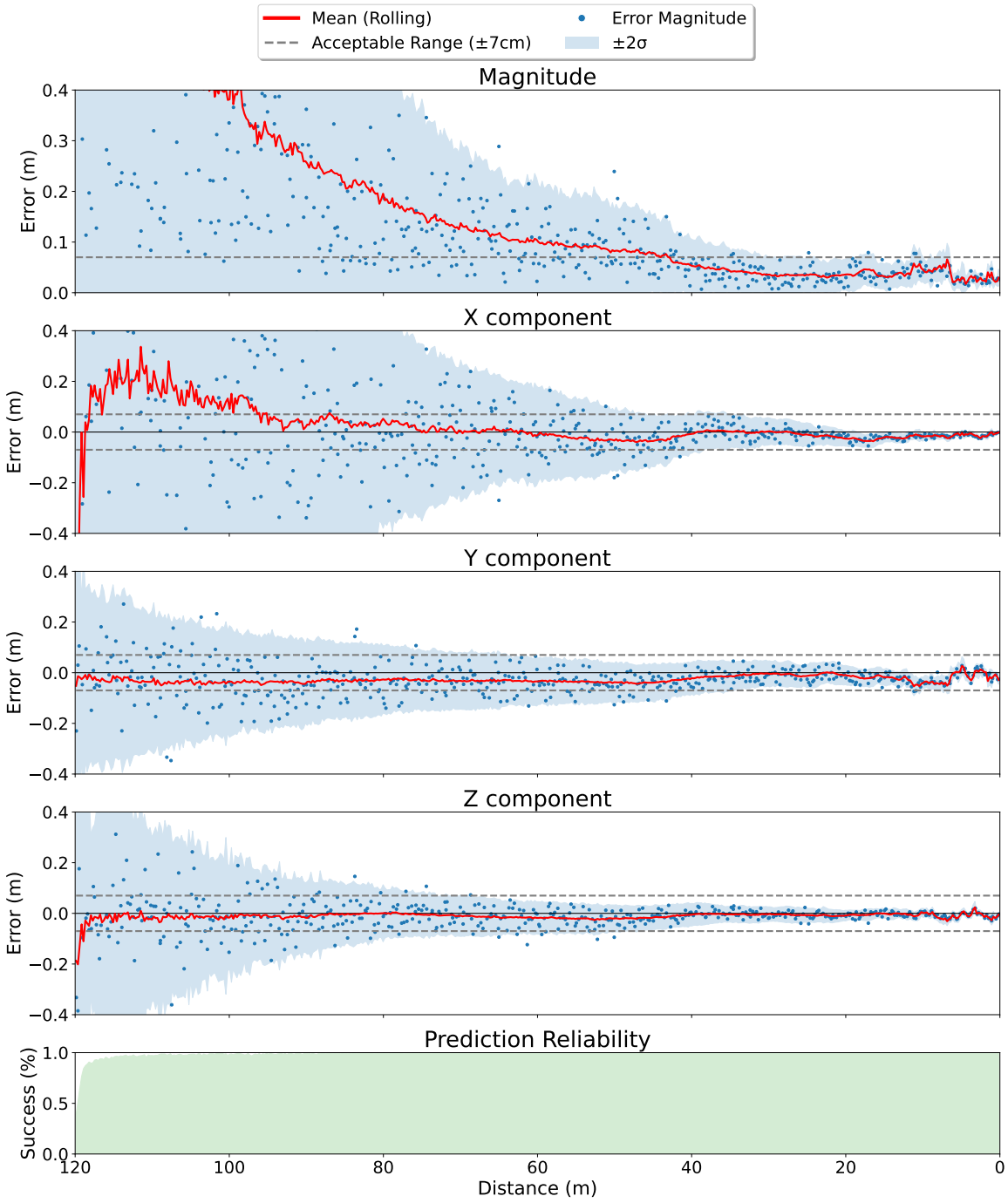


Figure 35. Pipeline results from receiver wingcam trained with *large features* (model L).

Appendix B. Specialty Exam

The following specialty exam manuscript was compiled and submitted to the dissertation advisory committee on 24 January 2023.

The Next Step for Dual Object Detection: Bridging the Gap Between the Virtual and Real Worlds
PhD Specialty Exam

Derek Worth, Maj, USSF
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, USA
https://orcid.org/0000-0002-9635-6022

Fig. 1. Relative vectoring using dual object detection (simulation only).

I. Introduction

My previous research has uncovered a technique, called *relative vectoring using dual object detection (DOD)*, which has shown great promise. When given image data, it reliably provides accurate relative navigation (less than 3 cm of error at contact) between two objects in real time (+40 Hz on a Nvidia RTX A5000) without the need for any extrinsic calibrations—exactly what is needed in autonomous aerial refueling! Unfortunately, we have only shown success thus far in simulation, see Fig. 1. Now, it's time to move this technique to the real world...

The DOD pipeline works first by employing machine learning to find 2D image coordinates of features across two objects within the image with known 3D models. Exploiting geometry of the camera pinhole model, it subsequently applies a 2D to 3D point correspondence to solve the perspective-n-point (PnP) problem and find the pose of both objects in the camera's local reference frame. Finally, a vector between the two poses is transformed in the local reference frame of one of the two objects, effectively telling that object where the other object is in relation to itself. The pipeline is robust to occlusions and, as previously mentioned, accurate, reliable, fast, and unreliant on extrinsic calibrations. Unfortunately, the first and most critical stage of the pipeline relies heavily on machine learning, which subsequently relies on a large training dataset to learn how to accurately find specific features. In other words, DOD only works if we have a lot of accurately labeled images containing the objects of interest.

Generating such data in simulation is relatively easy. Simply project known 3D model points into the perfectly calibrated simulated camera using the camera pinhole model, then surround the corresponding pixel coordinates with corrected bounding boxes, a process describe in [1]. With this approach, we can automate precise error-free labeling of thousands of high-resolution synthetic images in a relatively short amount of time (approximately 20K images, each with over 50 features, per hour). This includes data augmentation, where an assortment of different lighting effects, backgrounds, orientations/vantage points, and occlusions are applied—an important process the Google Research Brain Team highlights as a critical component of training deep learning models [2]. Unfortunately, precise labeling of *real* imagery is orders of magnitude more difficult and time consuming to produce. For example, the 328K images from the popular MS COCO dataset required over 30K worker hours to produce [3]. Even more concerning, manually labeled images are prone to human error and are often laced with missing annotations, misclassifications, and imprecise bounding boxes [4]—all of which will most certainly decrease the accuracy of the DOD pipeline. Thus, automating the labeling process is a must!

So, how do we automate labeling of real images necessary for DOD? In this effort, I attempt to find answers to this question using re-projections of digital twin data into real images by aligning 2D image points with 3D model points within a motion capture system (MCS). Projecting 3D truth from an image aligned MCS means we can localize 2D truth in images and use that information to automate accurate image labeling, unblemished from human error. Furthermore, if successful, this could enable our longer term vision of eventually deploying augmented reality to accurately overlay real images of aircraft parts (e.g., Calsan Learjet nose cone) with simulated parts (e.g., the rest of the aircraft) in a variety of environmental conditions. In theory, this pseudo-realism image manufacturing process will result in photorealistic images containing both true features that machine learning models

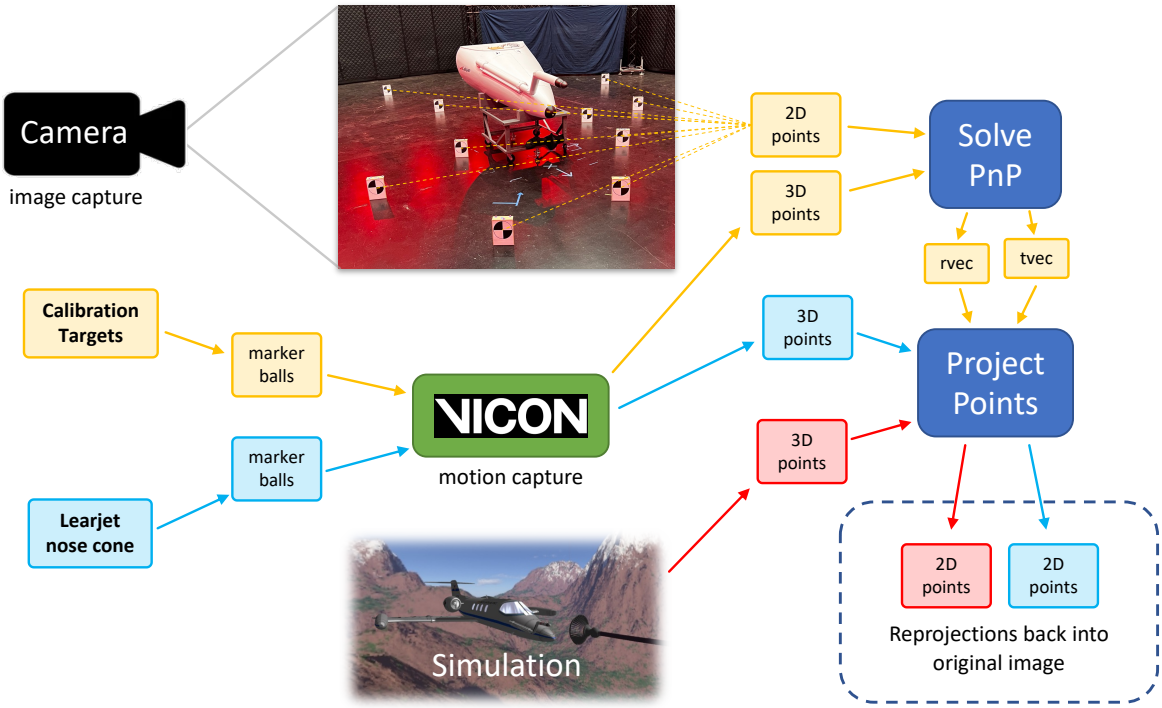


Fig. 2. Proposed motion capture aligned re-projection pipeline.

will encounter in the real world and augmented image space surrounding those features with more general, but equally important spacial information the models need to generalize well in new and previously unseen scenarios.

Because capturing training data from actual aerial refueling operations is cost prohibitive and untenable to validate and accurately label, the ultimate goal of this research is to create high-performing machine learning models from more easily obtained data. Transfer learning, which research has already shown can be used to create such models [5], is the key this research intends to exploit. Transfer learning from scenarios in simulation to real autonomous aerial refueling operations will have a better chance of success with more realistic images, and automating the labeling and augmentation of real imagery in a camera aligned MCS has tremendous potential to provide this exact capability.

My proposed pipeline, as summarized in Fig. 2, captures real images of static calibration targets surrounding objects of interest, all tracked by a MCS. Using the targets, which are clearly visible in the camera's image frame with known 3D positions reported by the MCS, it computes the camera's orientation from the 2D to 3D calibration target point correspondences using a solution to the perspective-n-point (PnP) problem. The resulting orientation, composed of rotation and translation vectors, can subsequently be used to compute the re-projections of known 3D points of non-target objects into the original image frame. Results show that this method can project real 3D objects into an image with high accuracy and consistency. This effort also shows that various calibration target configurations produce different amounts of re-projection error, with the most optimal configuration tested resulting in average error less than 0.7 pixels with a standard deviation of 0.31 pixels across all tested image points. Furthermore, this method can also inherently overcome time alignment issues that plague similar motion capture setups.

The remainder of this paper is divided into five sections. Section II uncovers the system performance track record of Vicon, the MCS used in this effort, and discusses related works in deploying camera calibrations in a MCS for aligning image data with re-projections from the real world. Section III details how the proposed motion capture alignment re-projection pipeline was implemented as well as how it was both quantitatively and qualitatively evaluated. Experimental results of the pipeline's performance are summarized in Section IV. Finally, Section V discusses closing thoughts and future work.

II. Background & Related Work

A. Use of the Vicon Motion Capture System

Vicon is considered one of the most well known, respected, and accurate systems in the motion capture (mocap) industry today [6]. Rokoko, a competing mocap provider, names Vicon as a top-tier mocap system, stating "they can safely be called the Rolls-Royce of mocap setups" [7]. In a side-by-side comparison, Serpiello [8] highlights Vicon as having the highest 3D position accuracy (sub-millimeter) when compared to 10 other high-end mocap cameras produced by OptiTrack, Qualisys, Simi Reality Motion Systems, and STT Systems.

Furthermore, a 2016 Vicon performance study [10] formally quantified the system's trueness (mean marker distance error),

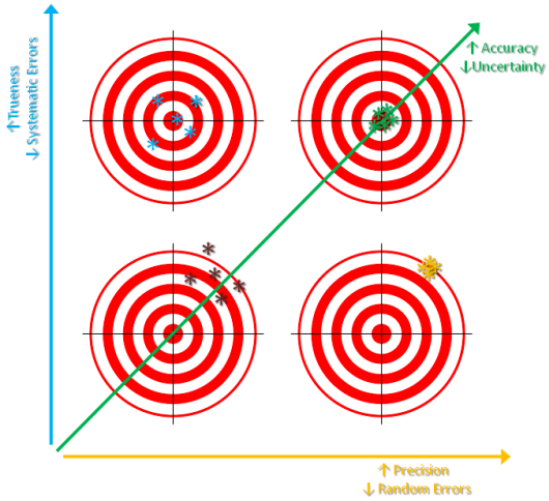


Fig. 3. Metrics used throughout this effort include trueness, accuracy, and precision as defined by ISO 5725. Image courtesy of [9].

precision (standard deviation of marker distances), and uncertainty (confidence interval)—measurements as defined in ISO 5725 by the International Organization for Standardization [11] and summarized in Fig. 3. The study used Vicon Bonita series cameras (now considered legacy and a slightly lower end version of the T-series cameras used here at AFIT and in this effort), and primarily focused on system performance in human performance scenarios, i.e., markers placed on body parts during 3D human movement. However, it uncovered evidence of optimal mocap setup parameters for more generalized scenarios. Overall, the study supported intuition where fewer cameras, reduced marker ball visibilty, and dynamic movement contributed to lower trueness, precision, and accuracy. However, the decrease in performance reported by this study remained in the sub-millimeter error range, further promoting confidence in the Vicon system. Bottom line, the literature shows Vicon can be trusted, especially when maximizing camera count and reflective marker visibility in static scenarios.

B. Camera calibrations

When using imagery to correlate or translate between 2D image points and 3D world points, camera calibrations are essential. One must know where the camera is pointed (extrinsic parameters) and how the pixel sensors across the camera's image plane map to projections out the camera lens and into the world around it (intrinsic parameters). Fortunately, intrinsic calibrations are fairly straight forward ever since Zhang published his method in 2000 [12]. It is commonly used and accepted across the literature landscape and has many open source implementations, including OpenCV's *calibrateCamera* method. Additionally, many tutorials are available, including [13], which served as the baseline intrinsic calibration code for this effort.

On the extrinsic side, the literature is rife with numerous intricate and elaborate approaches to automating accurate targetless extrinsic calibrations. Lee performed extrinsic calibrations using synchronous labeled imagery, but requires multiple synchronized cameras with manually labeled 2D estimates for each camera view [14]. The manual labeling process alone makes this approach cumbersome and unappealing for the automation needed in this effort. Others attempted to automate the extrinsic calibration process through the use of keyframes, simultaneous localization and mapping (SLAM), and bundle adjustments without requiring supporting infrastructure, calibration patterns, or even overlapping camera views [15], [16]. Though reporting a final mean squared re-projection error of less than 1 pixel, these methods require multiple statically mounted cameras on rotating objects and many image captures before applying lengthy bundle adjustments for post processing re-projection optimization, making this pipeline untenable for this effort as well.

Another popular approach to extrinsic calibrations applies sensor fusion by combining the edge detection of an optical camera with depth mapping of tightly coupled laser scanners or LiDAR sensors [17]–[19]. Like the multi-camera bundle adjustment method, sensor fusion has the benefits of achieving extrinsic calibrations automatically and without the use of targets. Unfortunately, the tradeoffs for these benefits include high implementation complexity, long scan times, and the requirement of good initial calibration seeds. Authors admit that the sensor fusion approach "is not trivial due to the vastly differing nature of the data recorded from the two sensors," highlighting the difficult of coalescing 2D dense color image data with 3D sparse point clouds. Although these automated extrinsic calibration approaches have specific situations when they would prove most useful, their complexity and slow runtimes make them unusable in solving the DOD labeling problem in this effort.

Optimizing extrinsic calibrations within a MCS is not a novel concept. Lee et al. used it to align the reference frame of their cameras and micro aerial vehicles to validate IMU navigation data [20]. They also ran into time alignment dependencies where their setup required synchronization between camera image, Vicon, and IMU data, a problem they eventually overcame by taking advantage of the precise sampling rate of their Vicon system. In a RGB-D SLAM benchmark [21], the authors placed reflective markers on the camera and a small checkerboard, then computed the camera pose based on image projections into the checkerboard corners, which where previously measured and tracked by the MCS. After performing their extrinsic calibrations to align the camera markers with the camera's optical image frame, they realized that the image data not only uncovered the location of the checkerboard, but also allowed them to rectify latency between camera and MCS data collects by comparing timestamps of checkerboard locations calculated from imagery with timestamps of the same checkerboard locations reported by the MCS—a rather unintended, yet pleasant discovery.

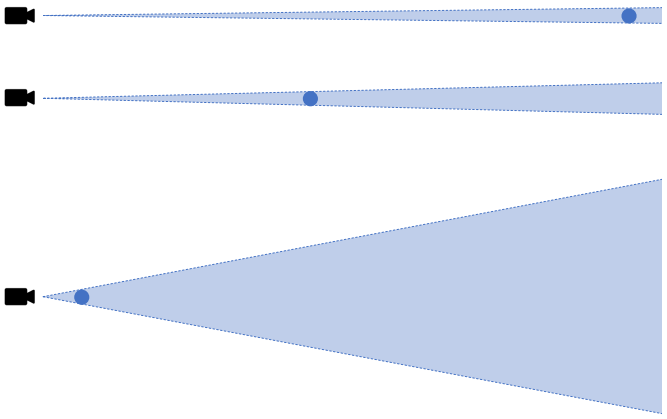


Fig. 4. The dark blue circle represent the precision of the MCS when reporting on the position of a point at the center of the circle. The span of possible camera projections through such precision is depicted in light blue. This simple illustration shows the effects on which the distance between the camera and sensed object have on re-projection error, where re-projections of the point into the top camera will have significantly less pixel error, on average, than that of the bottom.

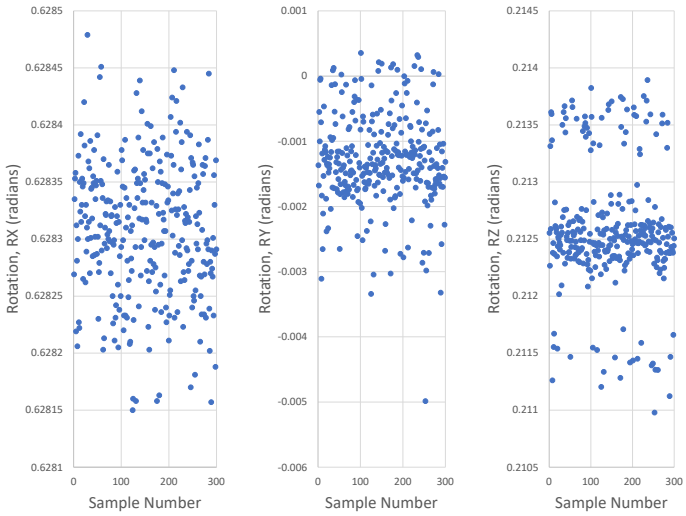


Fig. 5. These plots show 300 rotation measurements of a stationary object as reported by Vicon. The standard deviation of these samples for RX, RY, and RZ are 0.0034, 0.0412, 0.0307 degrees respectively.

Chiodini et al. used a similar method to align the pose of their camera to the reflective markers attached to the camera in a robotic exploration mission scenario [22]. However, instead of using a checkerboard with markers on it, they simply used static markers scattered on the ground, moving the camera into different positions while minimizing camera re-projection error of the markers. There are several other examples throughout the literature detailing various methods and applications of extrinsic camera calibrations inside a MCS (including [23], [24]) and they all are aimed at solving the same problem: estimating the transformation between the camera reference frame tracked by a MCS and the true optical camera reference frame. Unfortunately, solving for this transformation alone maintains tight coupling to the MCS error thresholds, where even miniscule error in rotation (e.g., small fraction of a degree) and translation (e.g., sub-millimeter) equate to significant re-projection error when extended over large distances, see Fig. 4.

One of the major contributions of this paper is ignoring the MCS estimates of the camera entirely in order to decouple it from the MCS. In this effort, the MCS has no awareness of the camera and only MCS-reported 3D points extended over a variety of distances from the camera are used to orient the camera in relation to other objects tracked by the MCS. In the next section, we will explore the importance of this subtle difference.

III. Approach

A. Proposed pipeline

In this effort, I propose a variation to the MCS aligned extrinsic calibrations describe in the previous section. Instead of relying on the MCS to track the pose of a single calibration target (e.g., chessboard with reflective markers) and the camera, I replace the camera tracking with more targets. More specifically, I explore the use of multiple MCS-tracked calibration targets distributed throughout the motion capture space such that the true 3D position of each target center is known and its corresponding true 2D projection into the camera's image frame can easily be found.

To justify why this approach should be more accurate than the more popular one outlined at the end of Section II, consider the tight coupling between the camera's true optical reference frame and its corresponding MCS-reported reference frame. The problem with computing, then using the transformation offset between these two is that this offset is static along with the true optical reference frame, while the MCS-reported reference frame fluctuates within the system's precision thresholds. Fig. 5 shows 300 samples of a stationary target tracked by Vicon. The variance of these measurements, although impressively small, fluctuates. This fluctuation, as evident in Fig. 4, introduces significant error at long distances. Therefore, using the MCS-reported 6DoF pose of the camera in conjunction with a highly optimized extrinsic calibration still results in fluctuations from the truth. In other words, a single perfect extrinsic calibration between the camera's true optical reference frame and the MCS-reported camera reference frame does not exist—because it is always moving.

The decoupled approach used in the pipeline for this effort has to perform an extrinsic calibration using solve PnP for every image captured, but the call to OpenCV's *solvePnPRansac* method takes less than 2 ms to execute and produces consistently accurate pose predictions as reported later in this paper. Furthermore, using targets spread throughout the motion capture space takes advantage of longer projection "control arms" less susceptible to error from MCS measurement fluctuations.

One main drawback to this approach is the need for manual labeling of image data. However, as later discussed in Section V, this process could potentially be automated using

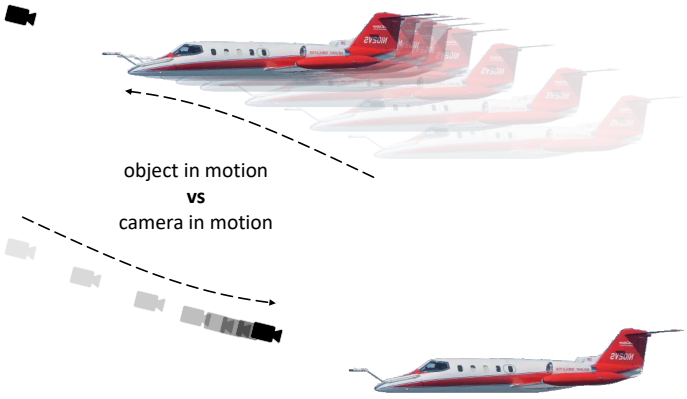


Fig. 6. Relative object movement in the image frame appears the same whether the object moves in front of a stationary camera, or the camera moves while pointing at a stationary object, as long as the two maneuvers are performed in reverse.

machine learning with very high accuracy. Additionally, this pipeline can also be deployed such that the camera remains stationary throughout the data collect forcing the targets to also remain stationary for all images, thus only requiring the manual labeling of a single image. This is a viable solution to the automated image labeling problem required in DOD since the motion of the objects of interest within the camera's image frame can appear to move regardless of whether the camera remains stationary and the objects move, or the reverse, as depicted in Fig. 6.

B. Time alignment tradeoff

As long as the objects of interest remain stationary, time alignment between the MCS and camera is unnecessary. However, this means the camera is the one moving and thus requires new labels per image. Conversely, a stationary camera configuration means the objects of interest are moving, and thus time alignment is necessary.

C. Digital twin

As stated earlier, the Vicon MCS used in this effort is reliable with high accuracy, which means it has high precision and trueness. The YouTube video [25] demonstrates this accuracy with a rigid object (i.e., colorful wooden airplane) moving in both the real and simulation worlds simultaneously in which the AftrBurner simulation is driven by Vicon data via a UDP socket to the Vicon Tracker software through the use of the Vicon DataStream SDK [26]. In this brief demonstration, I precisely set the real object at the origin inside the motion capture room, as marked by the white tape on the floor. As the object moves away from the origin, the sychronized simulated A-10, which also started at the virtual world origin, has precisely the same changes in rotation and translation since the simulated aircraft uses Vicon data to set its pose in simulation. The remarkable climax of the demonstration culminates when the aircraft returns to the white taped origin in the real world, which reflects an align in simulation with sub-millimeter positional accuracy (Global XYZ measured in meters) and less than 0.09 degrees of RPY rotational error. This demonstrates that real world data can easily be modeled in simulation. However, the reverse (placing modeled data from simulation into the real world) is more difficult and the primary research focus for this paper.

D. Preparing the Vicon motion capture system

The MCS was an essential component of this research effort and thus a significant amount of time was spent up front learning how to properly calibrate and use it. Below are the steps I took in order to prepare the system for optimal results.

Re-orient cameras - The motion capture room used in this experiment is approximately 20 meters wide, 16 meters long, and 10 meters high. The first tier of 30 Vicon T-series cameras are mounted roughly evenly spaced around the room about 4.25 meters off the ground. The second tier of 30 additional cameras are also evently space around the room about 7 meters off the ground. The original configuration had all 60 cameras pointed downward toward the floor in various directions to capture motion of the entire room, including the corners. For the experiments in this effort, all motion was captured in the center of the room. To maximize Vicon accuracy and make the most of all the cameras available, I re-oriented all cameras (only adjusting those necessary) to point toward where my experimentation was taking place.

Removed reflective material - After my initial full calibration after reorienting the cameras, I noticed some "dead spots" in the center of the room near the origin where my objects would simply disappear from Vicon's view. I quickly realized the standard calibration procedures begin with masking current reflections within each Vicon camera's field of view. This included the paper targets taped to the floor as well as the white masking tape used to denote the room's origin and XY axes. Furthermore, all other reflective objects were removed or covered with dark sheets, including the F-15, pile of pallet wood, Quang's satellite, and Adam's 3D cargo loading scanner.

Firmware update - After troubleshooting various issues (e.g. Giganet 2 failing) with the Vicon engineering support team, we realized all Vicon cameras were running outdated firmware, and not all on the same version. The update took maybe 15 minutes and was a quick assurance the any further issues or communication warnings reported in Vicon Tracker were not due to the camera firmware.

Box 2 is dead! - Not really, but it is highly unrealiable at best. I got it working twice, but both times, its cameras randomly flickered off and the box disconnected from the Vicon Tracker software, leaving me with red X's of doom, making me wonder if the absence of those cameras induced any strange issues or errors I would later run into. Engineers at Vicon support noted our entire MCS is beyond service life and unrepairable. For system accuracy, I preferred using all 60 cameras, but Giganet 2's reliability issues forced me to disconnect it from the system entirely for all my experiments and work with 10 fewer cameras. I also intermittently experienced issued with Tracker connecting to Giganet 3 at startup, but

when it did connect, I did not experience any issues for the remaining duration of any of my captures, so I retained it as part of my experiments.

Vicon calibration - After all the cleanup stated above, I performed a full calibration of the system and used that calibration for all the data collects in this paper. Note that I used the default 0.75 Vicon LED strobe intensity for all cameras.

Post calibration assessment - To validate Vicon precision (i.e., standard deviation of measurements) and trueness (i.e., average measurements centered on truth) post calibration, I ran a few simple tests. Since the camera was the only moving object in my planned experiments and it was not tracked by Vicon, I only tested stationary measurements. I placed objects with markers in arbitrary, yet widespread locations around the room and observed the variance of Vicon's reported measurements over the course of over 1,000 sample. Overall, the standard deviation of all position measurements were less than 1 millimeter. I did not pay particularly close attention to any rotation measurements, since I did not intend to use Vicon rotations in any of my experiments. I tested trueness by placing two marker balls along the X axis as reported by Vicon (using maximum zoom in Vicon Tracker to be as precise as possible). I then set up a laser level to cast a straight line between the two markers, as shown in Fig 8. Placing a third reflective marker at several locations along the laser line and observing Vicon reported measurements of sub-millimeter distances from the axis further confirmed precision and trueness. I repeated this laser procedure along the Y axis with the same results.

MoCap reflective marker balls - Marker balls available during this experiment varied in size and quality. Though not fully tested and evaluated in this effort, literature indicates larger markers produce higher MCS accuracy, likely due increased camera visibility [10]—i.e., larger markers reflect more light for the cameras to see. A simple experiment confirmed this finding, where different marker balls of various sizes were placed on a flat surface at the same position in the motion capture room, one at a time and free of any surrounding occlusions that could obstruct the line of sight between each ball and all the Vicon cameras. Then, I recorded how many cameras the Vicon Tracker software reported could see each marker, see results in Table I. Thus, I used up all the larger balls in our supply first, favoring larger balls with lower quality (e.g. used balls with blemishes or cheaper non-reflective balls with reflective tape stuck to it) over the smaller high quality balls (i.e. pristine OptiTrack markers).

E. Custom MCS tools

Centering wand - In order to establish objects and features with precise Vicon reference frame 3D positions, I had to fabricate a few custom tools. The first was what I call a "centering wand." The Vicon Tracker software does not allow you to create a measurable object with less than 3 reflective markers. However, it does allow you to establish an object with more than 3 markers, then later detach some, as long as the object always has at least 3 markers. The system also allows

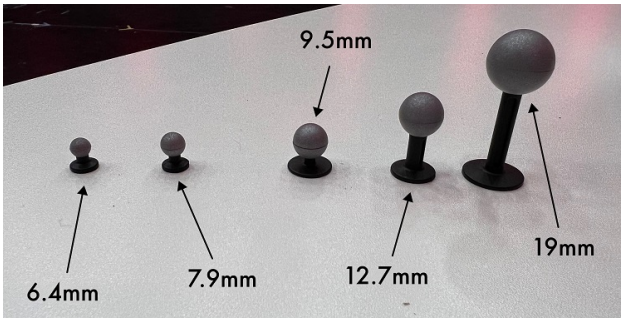


Fig. 7. MoCap marker sizes.

TABLE I
VICON MARKER VISIBILITY

Marker size	*Cameras with solid visibility*	**Total cameras with any visibility*
6.4 mm	6	8
7.9 mm	8	9
9.5 mm	11	12
12.7 mm	10	13
19 mm	15	15

*Includes cameras with sporatic/unstable marker visibility.

you to localize the object origin with any user specified offset, both in position and rotation. Furthermore, Tracker allows for precise alignment of the origin with individual marker balls or the geometric center of multiple marker balls, see Fig. 9. With these constraints, I ran a taut string between the base of two markers held in place by a custom wooden structure (I chose oak, a hardwood that would not deform during experiments), with the precise center of the string marked and visible through a viewing port as depicted in Fig 10. After fabricating this centering wand, I could subsequently transfer its precise 3D center point along the string to any object, then detach it after

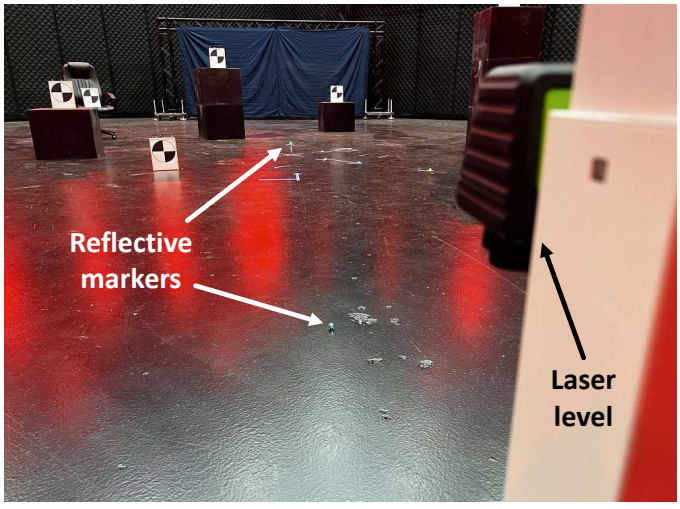


Fig. 8. In this image, the laser level casts a straight line across the center of two marker balls positioned along the X axis as reported by Vicon, providing a precise visual along the otherwise invisible axis.

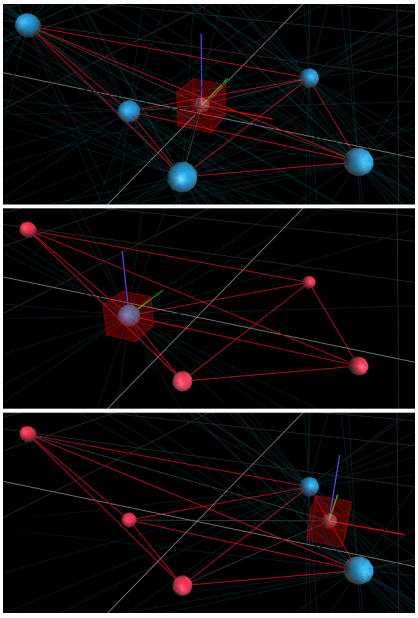


Fig. 9. The Vicon Tracker software allows you to easily align or "snap" an object's local reference frame origin to a single marker (center) or the geometric mean of multiple markers (top), which includes the exact midpoint between two markers (bottom).

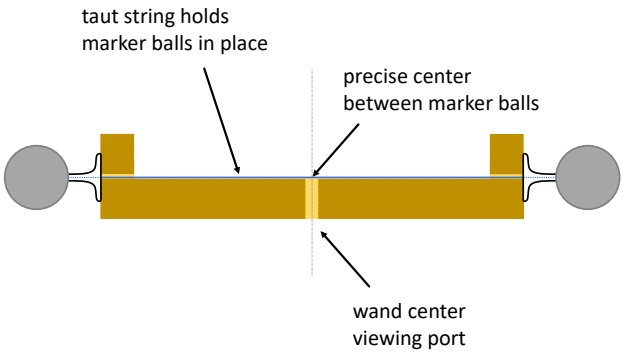


Fig. 10. This centering wand was used to precisely localize 3D Vicon object origins, such as calibration target centers.

the Vicon object's origin coincident to this center point was set, and reuse it for other objects.

Calibration targets - Next, I needed a truth set of targets for orienting the camera using solve PnP and a validation set for computing re-projection errors. I decided on 10 of each because that's how many 8.5"×11" target printouts I could fit on the wood that I purchased, and labeled them 1 through 10 and A through J respectively. I also needed these targets to be highly visible such that they could easily be localized to precise pixel coordinates within images of the targets. For this, I enlarged a chessboard corner as commonly used in intrinsic camera calibrations, but only retained a single corner and confined it to a circle as shown in Fig 11. Labels were necessary in order to differentiate the targets when performing the 2D to 3D correspondences for solve PnP. Also, I made sure to place a piece of double sided tape directly behind the the

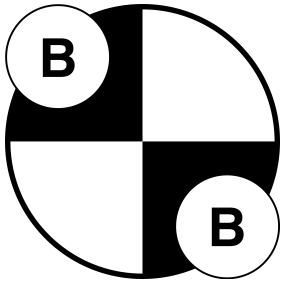


Fig. 11. Example of final calibration target design.

center of each paper target before attaching it to the face of the wooden placards. This was important for ensuring the paper did not flare up away from its original position during the experiments, potentially causing inconsistent measurements. This also ensured the target could be viewed from multiple angles and the target center would still project through the same 3D point.

Next, to localize the targets in the 3D Vicon reference frame, each target needed at least 3 reflective marker balls. Vicon detects objects by the geometric shape the balls make in 3D space, so each marker ball pattern, when placed on each target, had to be unique. For maximum Vicon visibility, I constructed each target with a rigid upper platform that the marker balls were mounted to. Otherwise, half the downward facing Vicon cameras at any given time would be blind to the marker balls mounted to one side. To localize the center of each target in its own 3D local reference frame, I clamped the centering wand to the target, ensuring the center of the string coincided with the center of the target. In Vicon Tracker, I could next select the target's markers as well as those of the centering wand, snap the target origin to the center of the centering wand markers, then detach the centering wand markers and unclamp the wand. I repeated this process for all 20 targets as well as the probe tip of the Learjet nose cone[1], see Fig 12.

Points finder block (PFB) - In order to localize a generic point in 3D space and/or validate already localized points, I created what I call the 3D points finder block, or PFB for short. Depicted in the upper left image of Fig. 12 with the centering wand, I created this block by clamping it to the centering wand using the same methods for establishing the target and nose cone centers. I used this primarily to find features across the skin of the Learjet nose cone, but also periodically used it to randomly test target centers for accuracy (ensuring target centers did not drift or get misaligned somehow).

Nose cone - I included 22 features across the skin of the Learjet nose cone in my experiments as part of a qualitative analysis. Although I did not measure re-projection error directly for the 3D points corresponding to each feature, I instead made a visual assessment, specifically looking for whether the 2D re-projections appeared to align with where I believe

[1]As a slightly unrelated side note, setting the Learjet probe tip as the receiver origin makes probe to drogue vectoring calculations much easier than dealing with offsets and reference frame transformations required in a receiver aircraft geometric center origin reference frame.

TABLE II
LEARJET NOSE CONE FEATURES

Alpha ID	**Feature description*	*X (mm)*	*Y (mm)*	*Z (mm)*
a	outer elbow	-768.2973906	-27.89274382	81.1959054
b	inner elbow	-743.0291471	27.76617757	-3.822033981
c	side of probe tip	-205.9776671	52.61945257	15.77493448
d	nose tip	-1290.085389	505.0422228	-685.3452475
e	top nose rivet line intersection	-1804.751425	504.5286008	-309.4463176
f	NPS nose rivet line intersection	-1820.795047	792.1364097	-512.781415
g	NPS square	-3129.592745	964.8074164	59.92651885
h	double bolt strap	-1848.07344	189.7426603	-258.5782814
i	PS tiny holes	-1972.575248	158.8567759	-519.4522253
j	NPS pie slice panel	-1954.786548	832.5464897	-562.7941358
k	NPS fin	-2574.021563	934.3604807	-647.0926169
l	NPS 3 ovals and circle	-2987.8444	692.9725408	102.7852802
m	PS 3 ovals no circle	-2985.682335	312.3753551	103.2349608
n	PS fin	-2563.469373	71.26852201	-639.1763474
o	NPS Calspan C	-3398.188806	1200.988497	-179.3306417
p	inner shoulder	-1607.58596	268.8039015	-324.3665598
q	PS nose rivet line intersection	-1817.059136	214.1545165	-515.0584856
r	PS Calspan N	-3241.489702	-168.5302529	-192.3698598
s	NPS rear rivet line corner	-2917.651397	1110.526606	-239.0707842
t	PS rear rivet line corner	-2916.067888	-101.3599516	-232.8971979
u	no bolt strap	-2749.129751	-85.41551785	-20.41212917
v	probe tip	1.560546184	0.603303156	1.95784376

Note that the 3D points are measured in the Vicon reference frame.
*NPS = non-probe side, PS = probe side.

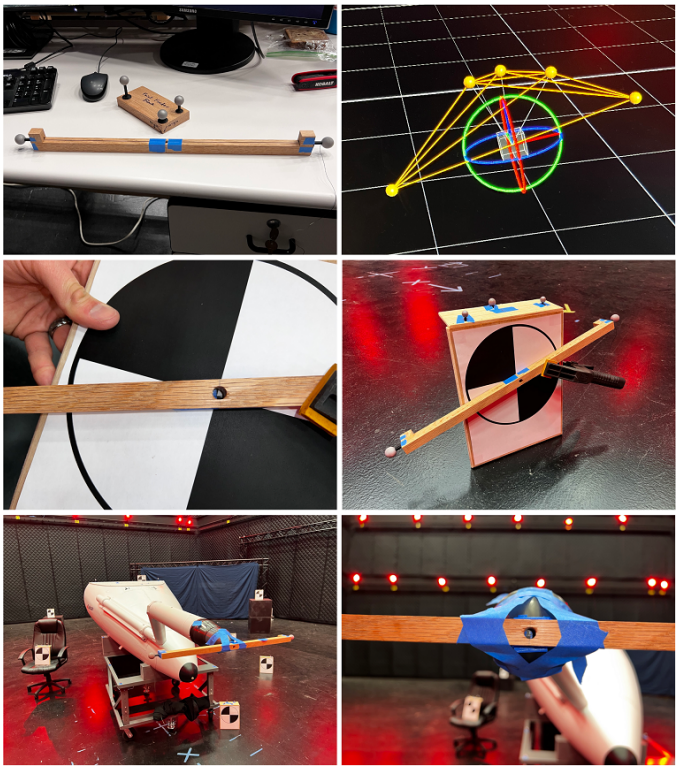


Fig. 12. The top left image depicts the centering wand (next to the 3D points finder block). The top right depicts the object established in Vicon Tracker from the centering wand clamped to the calibration target (center). After establishing the object in Tracker, the two centering wand markers were detached so the wand could be removed and resused on other objects. The bottom shows how the same process was applied to the Learjet probe tip.

they should be based on spatial information surrounding each feature. For example, a point on the probe tip should be re-projected in the image on to the end of the probe. The features I chose for this effort are listed in Table II. Note that I established the 3D Vicon reference frame coordinate for each feature using the PFB.

F. Gotchas and lessons learned

Throughout this effort, had a few stumbling blocks. Here a few I documented.

Target labels - Calibration targets need visible labels. Initially, I left the targets bare without any distinguishing markings, relying soley on the replay of Vicon log data. However, that approach was time consuming and I ultimately replaced the bare targets with labeled targets. Additionlly, I started with 10 targets labeled A through J. However, I quickly discovered 10 targets were insufficient, especially when assessing re-projection error. I suspect more targets even are better (i.e., perhaps re-projection error is inversely proportional to the number of orientation calibration targets used to orient the camera, this could be tested in a future work), but come at the cost of increased image labeling time.

Symmetry is bad in a MCS - Vicon quickly gets confused when you establish objects with symmetry. For example, Targets C, which you will later see in the results (captures 63 and 64), has missing data because Vicon got confused and reported the target was inverted upside down. Rather than reporting re-projection error for the invalid 3D coordinate, I omitted it from the dataset completely.

Target placement - Getting targets off the ground (above 5 feet) was difficult. The wheeled structures in the room where helpful and the scissor lift was a life saver! Next time, I would

Fig. 13. The iPhone 13 Pro and DJI OSMO Pocket were the two cameras used in this effort.

perhaps make some telescoping tripod mountable targets for each placement in 3D space. Also, a bit obvious, but you can't place targets under the nose cone—blocks line of sight to the Vicon cameras.

Marker collisions - Marker pattern collisions are easy to make, especially with lots of (similar) objects. I had to reposition marker balls multiple times after Vicon would get confused between two objects with similar marker configurations. If they are a little bit off, Vicon tries to match the nearest shaped object without warning (which is a bit concerning).

G. Cameras used

Initially, the obvious camera choice was the monochromatic Prosilica 2050s that Nate was using. However, learning how to use the eBUS driver, Vimba SDK, *aarimg* AftrBurner module, and the various batch scripts on Bertha proved highly time consuming. After this exam, I may revisit those technologies, but for the sake of time, I went with a DJI OSMO Pocket (model OT110) with a 4000×2668 resolution and my trusty iPhone 13 Pro with a 4032×3024 resolution, see Fig 13. The obvious concern here was the effects of autofocus negating camera calibrations. However, both devices feature a focus lock that disabled any focus changes during image capture, see Fig. 14. Per the iPhone User Guide [27], users can "lock manual focus" and "precisely set and lock the exposure for upcoming shots." This proved extremely useful in overcoming the poor lighting conditions in the motion capture room. Both OSMO Pocket and iPhone 13 Pro produced vivid, sharp, high resolution images that clearly showed the calibration targets from across the room. In contrast, I could only get the Prosilica 2050 to produce dark, blurry, and/or underexposed images that were difficult to analyze, see Fig 15. The only downside of using cameras with variable focus is that I had to perform an intrinsic calibration (i.e. take a lot of chessboard pictures) each time I changed settings, which include powering off the devices or locking the screens. There were a few times I had to completely redo a data collect because I accidentally tapped the wrong place on the screen of my iPhone.

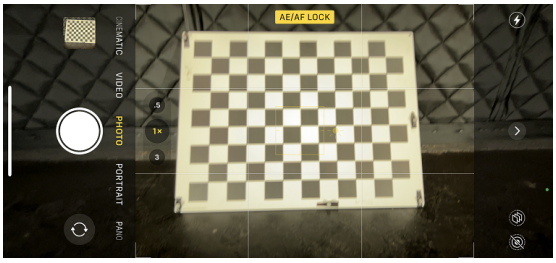


Fig. 14. iPhone screenshot depicting AE/AF LOCK enabled, preventing focus changes during image captures.

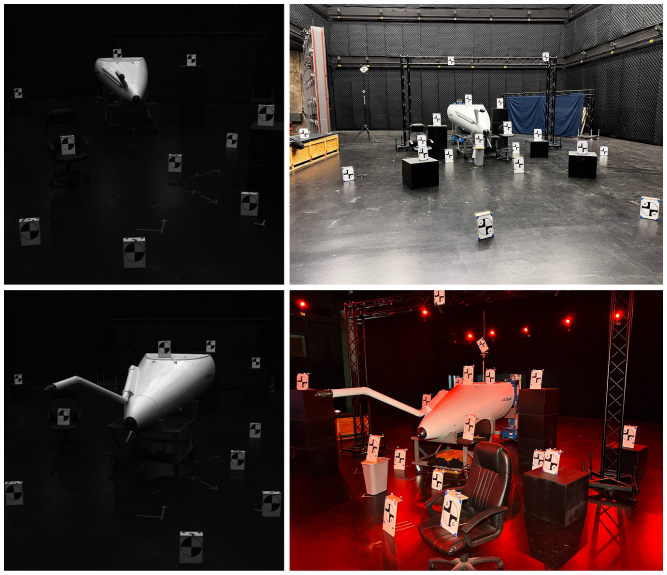


Fig. 15. The images on the left were taken under maximum lighting conditions with the Prosilica 2050, whereas the iPhone 13 Pro captured the images on the right. Note that the iPhone continued to capture highly vivid target information, even after turning off the main lights and the majority of lighting coming from the Vicon strobe LEDs (bottom right).

H. Manual labeling

To define the *truth* labels for each image, I used the line intersection technique: rather than selecting the whole pixel nearest to the target center, I used image processing software, e.g., Gimp [28], to select pixel pairs in which projected lines through them intersected with the target center.[2] The intersection of two lines using this method more precisely estimated target centers with subpixel accuracy, see Fig. 16. Unfortunately, this process is time consuming. Even with a spreadsheet that automatically computes the line intersections, each image with 20 targets took me about 25 minutes to label. In future work, machine learning with object detection could automate this process.

I. Scenarios

The proposed pipeline takes advantage of extrinsically calibrating the camera using solve PnP for each image, specifically

[2]Helpful hint: when selecting two points for each line, select the first point as one near the target center so you can select the second point using a longer control arm for smaller changes in the line's interection with the target center.

TABLE III
CAPTURE SCENARIOS

Capture	Camera used	Camera settings	chessboards	Scenario description
1	Osmo Pocket	ISO 800, 1/120s, fluorescent	249	Non-coplanar, many chessboards
2	Osmo Pocket	auto exposure/whitebalance	169	Non-coplanar, auto exposure
3	Osmo Pocket	auto exposure/whitebalance	23	Non-coplanar, auto exposure, few chessboards
4	iPhone 13 Pro	ISO 500, f1.5, 1/60s, RICH WARM	173	Non-coplanar, wide spread
5	iPhone 13 Pro	ISO 125, f1.5, 1/75s, VIBRANT	589	Coplanar (near, mid, far)
6	iPhone 13 Pro	ISO 125, f1.5, 1/75s, VIBRANT	345	Rings (XS, S, M , L)

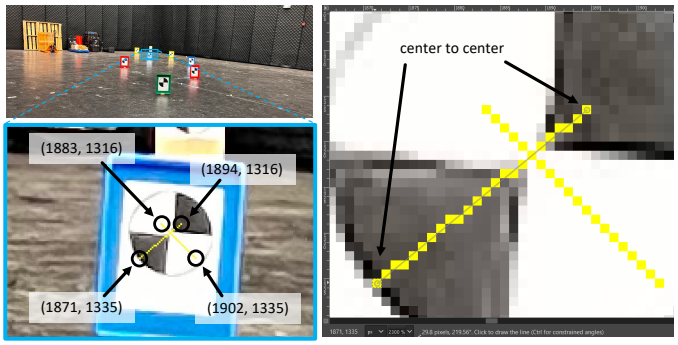


Fig. 16. In this example, solving for the interection of two lines that pass through the 4 selected pixel coordinates resulted in a target center at (1887.97619, 1320.97619), rather than simply selecting the estimated nearest whole pixel at (1888, 1321). The right zoomed in image shows how typical image processing software easies pixel pair selection in this task by projecting a higher resolution line (overlaid on top of the pixelated yellow line) between the centers of the two pixels.

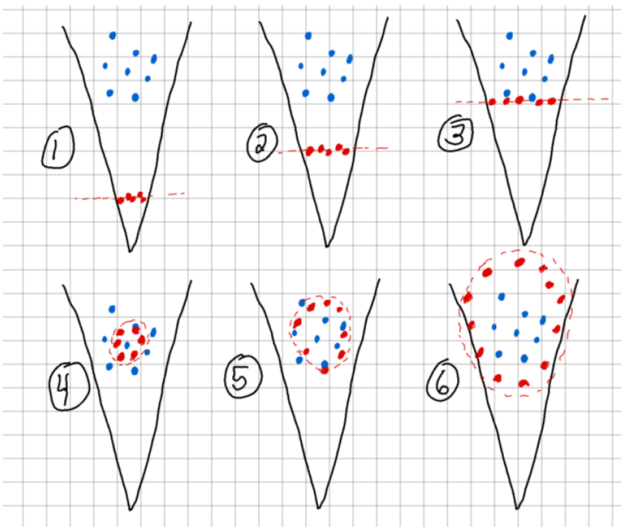


Fig. 17. This is a rough sketch of my initial experimental design. These diagrams are a top down view of 6 different configurations planned for this effort. The blue points represent features on the object of interest that we want to re-project back into the image, while the red points are the calibration targets used to orient the camera within the motion capture room. 1-3 correspond to coplanar near, middle, and far target configurations, while 4-6 correspond to non-coplanar small, medium, and large target spread ring configurations.

by orienting the camera using distant and widespread 3D points rather than the more traditional small group of coplanar checkerboard points. This has the added benefits depicted in Fig. 4. Thus, my intuition tells me more targets at further distances from the camera are better. To test this hypothesis, I have designed an experiment with a variety of capture configurations, as summarized in Table III. Some of my initial questions that helped motivate my experimental design included:

- How many chessboard images are necessary for a sufficient intrinsic calibration?
- What is the most optimal target configuration? Near the camera? Far away? Tightly clustered around the object of interest? Widespread?
- Does orienting the camera from a specific configuration while projecting to another matter? What happens when you reverse the points?
- Does the camera used matter? Effects from ISO, shutter speed, aperture? Resolution? Field of view?
- Does capture configuration matter? Should targets be closer to the camera? Midfield? Far away?

From these, I came up with a few target configurations, as shown in Fig. 17. The metric I used to assess configuration performance was image re-projection error, defined as the Euclidean distance measured in pixels between the re-projection and truth labels. In order to further and more objectively quantify this error for a given image collect, I assigned targets to 3 different categories:

- *Orient* - targets used to orient the camera during the extrinsic calibration process (i.e. passed into solve PnP).
- *re-project* - targets with 3D points re-projected into the image for re-projection error calculations.
- *Both* - targets that were used for both orient and re-project.

An example of a specific category combination is shown in Fig. 18, where targets 4-6 are only used to *orient*, targets 1-3 (2 resides outside the image frame) are only used to *re-project*, and targets 7-10 (9 resides outside the image frame) are used for *both*. For the final results of this effort, I performed the re-projection error for all targets for all capture configurations and used 3 variations of orientation configurations: 1-10 only, A-J only, and 1-10 and A-J combined. In other words, of the 3 categories above, final results were only reported on *orient* and *both*, since *re-project* of an single target had no influence on the other remaining targets. As shown in Table III, I performed 6 data captures total, each with their own combination of camera and target configurations. Each capture consists of

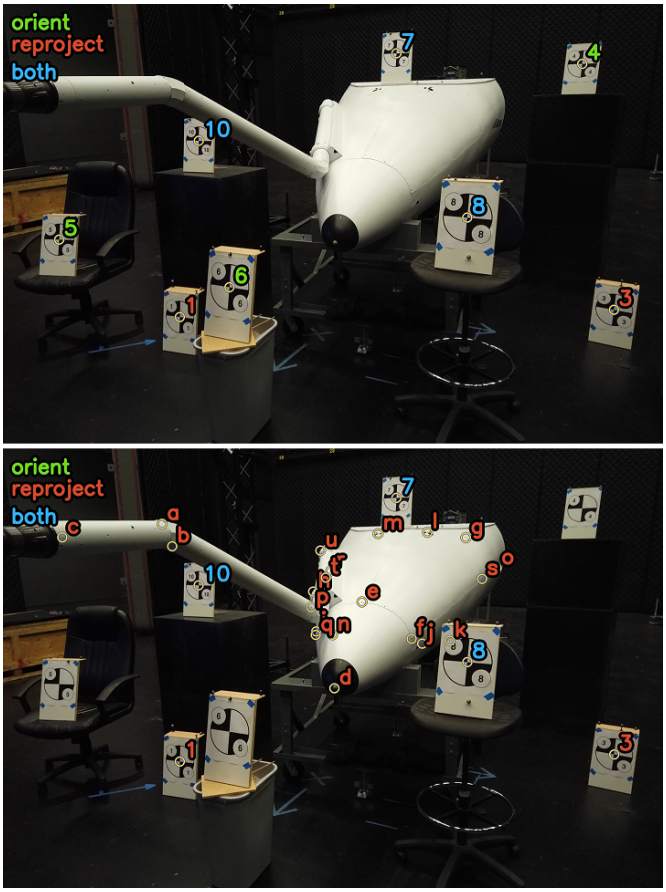


Fig. 18. Example image loaded with *orient*, *re-project*, and *both* targets. The top image depicts the truth points with their assigned categories, while the bottom image shows the re-projected points, which includes the nose cone features listed in Table II.

several images of targets like the one shown in Fig 18, as well as chessboard images for intrinsic calibrations, as noted in the chessboards column of Table III. Unfortunately, due to the lengthy manual labeling process, I was only able to annotate a limited subset of each image capture. Such images are listed in Fig. 19.

When solving the perspective-n-point problem, coplanar 3D points are well known to produce bad results such as reflections across the image plane or have multiple pose solutions [29]. Regardless, I wanted to test the resiliency of this pipeline, so I added the near, midfield, and far coplanar configurations as part of my experiments. In addition to the coplanar configurations, I also implemented non-coplanar ring configurations in which the targets evenly spaced across the center of the room represented an object of interest, while a large ring of surrounding targets were used to *orient* the camera. I then migrated the targets comprising the ring inward to form medium, small, and extra small ring configurations as depicted at the bottom of Fig. 19.

J. Software implementation

I implemented the pipeline proposed in this effort using the AftrBurner graphics engine [30]. As previously mentioned I used OpenCV's API for intrinsic camera calibrations, performing solve PnP, and re-projecting points back into the image plane. All code for this effort has been committed and pushed to the *aarviz2020* repo under the *derek-worth* branch. The code implementations specific to this effort are contained in the *ViconCamera.h* and *ViconCamera.cpp* files.

IV. Results, Analysis, and Discussion

The re-projection results for all configurations listed in Section III are reported in Fig 20.[3] The same results are reported again in Figures 21 and 22, but sorted differently for readability.

Overall, targets H, I, and J reported significant reprojection error across captures 1-5 and capture configuration ID 64 (large ring). It appears, in these configurations, the pipeline struggles with re-projecting accurately in the far and near fields when the majority of the orientation points are in the midfield. Additionally, all targets produced significant re-projection errors in capture 5 (coplanar) across the board for near, midfield, and far as expected. These results align with what the literature says about perspective-n-point and coplanar points. The remaining outliers that spiked high re-projection error appear to be outliers in the motion capture space of targets. In other words, the data shows that this pipeline works well when orienting off of points intermingled in the space surrounding the objects of interest that are re-projected.

The top 6 configurations with the lowest standard deviation of re-projection errors were from capture 6, images 2 and 3. These were the medium and small rings, which just happened also intermingle with the objects of interest.

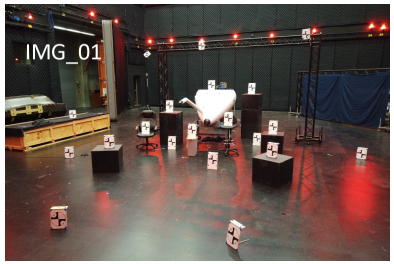
For these results, I am assuming the Vicon 3D truth data is accurate and consistent across all targets at all locations throughout the motion capture space, may very well not be the case. I am also assuming my manual hand labeling is highly accurate as well. Either of these assumptions being incorrect could negatively bias the results.

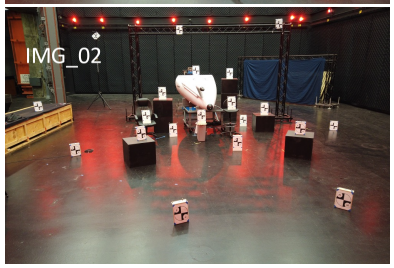
When assessing re-projections of the Learjet nose cone (sample shown in Fig 18), all points seemed to be highly accurate. The points re-projected on the side flaps, nose tip, probe tip, and rivet lines seemed to all line up. If this was an augmented reality setup with simulation data projected over the image instead of simple point clouds, I believe the blending of the nose cone with the rest of the aircraft would match pretty well.

V. Conclusions & Future Work

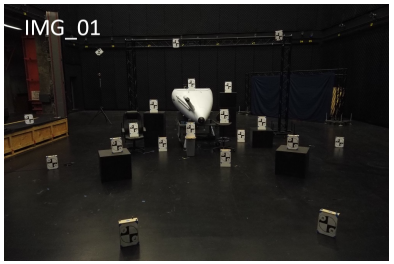
Overall, the pipeline presented in this effort shows great promise for solving the automated labeling problem needed for DOD to translate through machine transfer learning into the real world. It circumvents the re-projection error due to tight coupling of a MCS tracked camera and narrows the source of error the the PnP algorithm and accuracy of labeling 2D truth image points. The top performing capture configurations

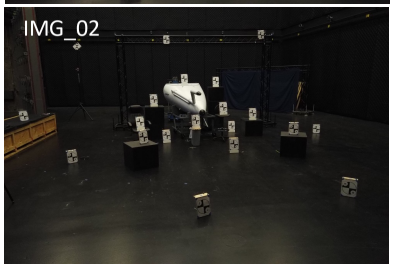
[3]Please excuse the horrendous faux pas of pasting Excel spreadsheets into this document! I had troubles getting Matplotlib to graph the data in a meaningful and legible may. I am definitely open to suggestions.



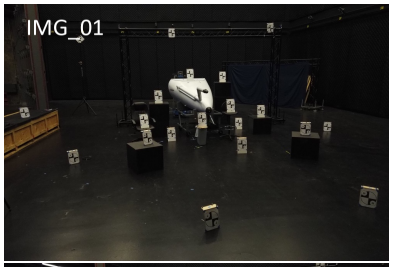


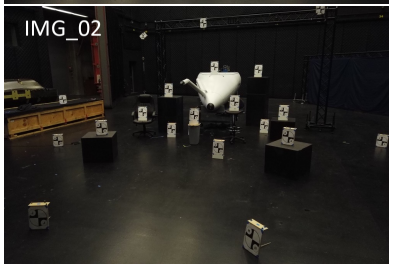
capture 1



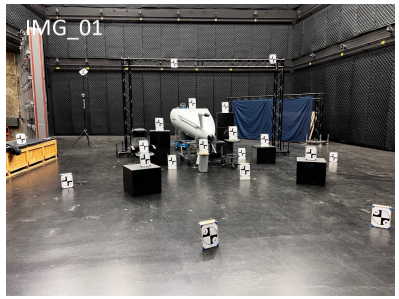


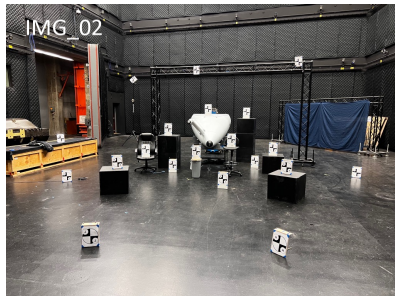
capture 2



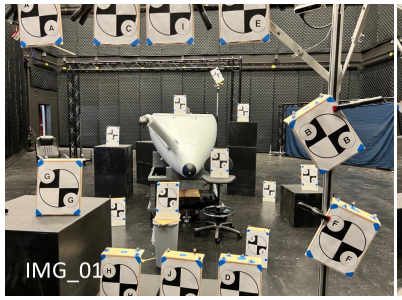


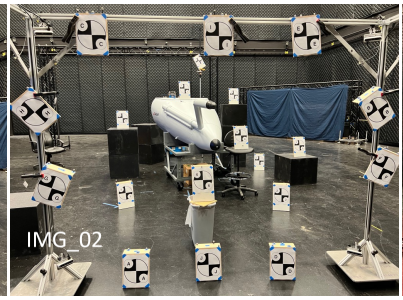
capture 3



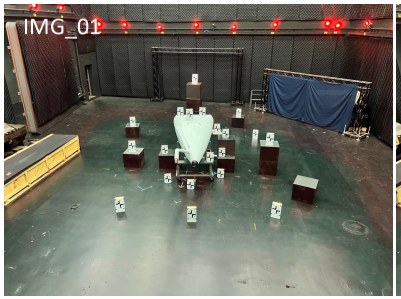


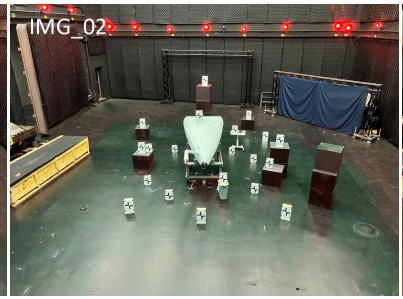
capture 4

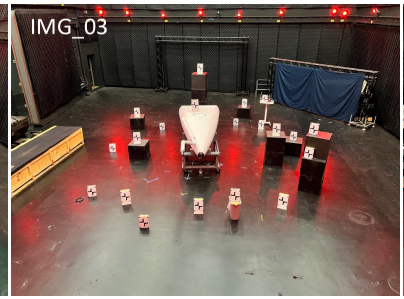


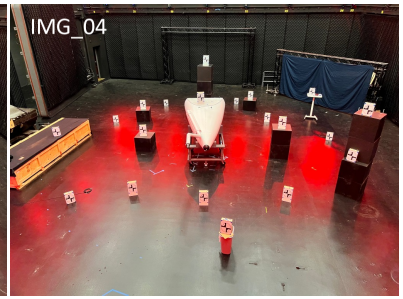


capture 5









capture 6

Fig. 19. Each individual capture comprises a collection of both calibration target images and a full set of chessboard images taken from a common camera with locked focus. Disengaging focus lock (e.g., powering off device, switching apps, etc.) initiated a new capture with its own corresponding intrinsic camera calibration. A DJI OSMO Pocket captured 1-3, while an iPhone 13 Pro captured 4-6.

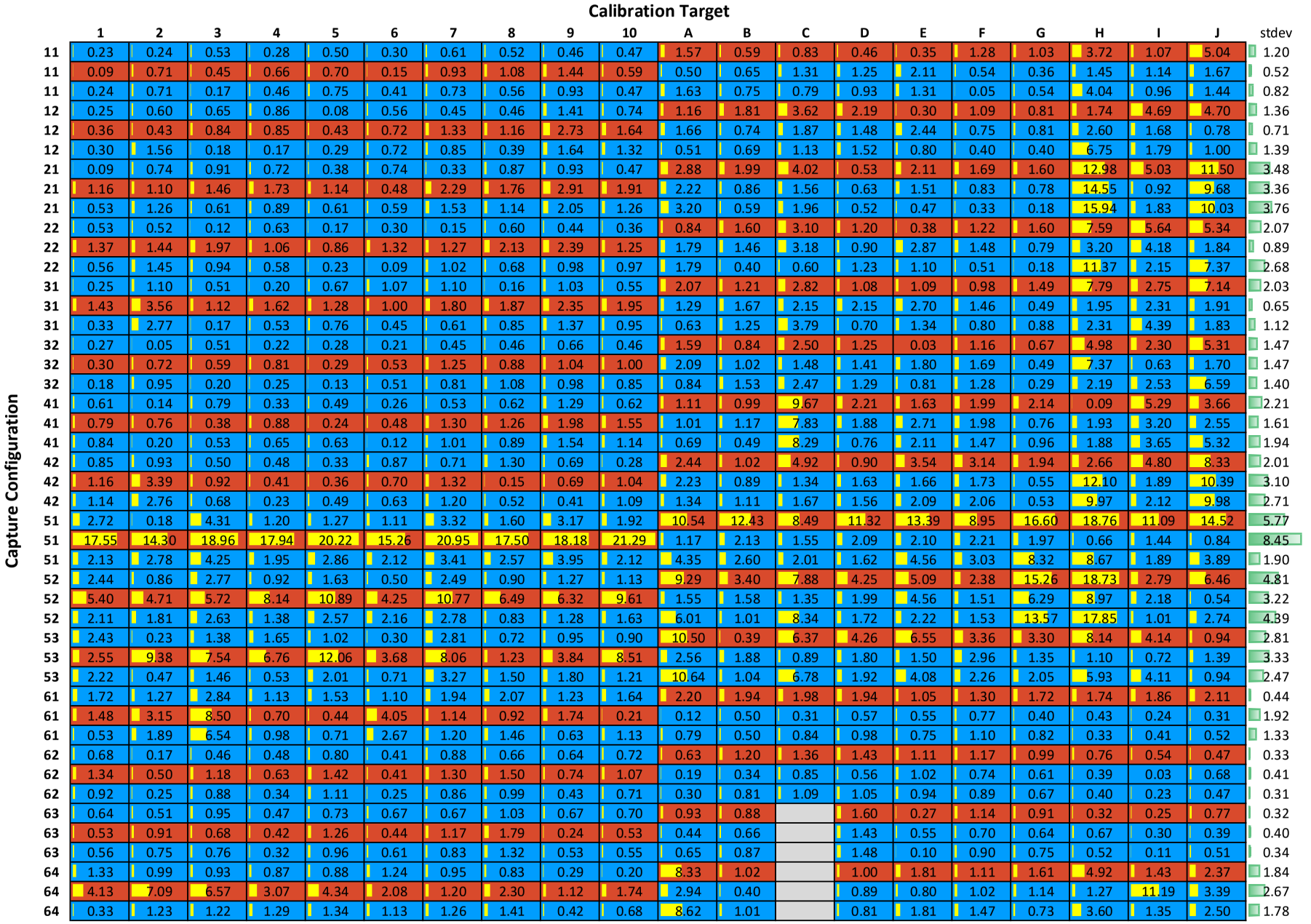


Fig. 20. Camera re-projection error, sorted by camera orientation source, then capture configuration. The 2-digit capture configuration identifiers along the left side of the table signify the capture configuration, where the first and second digits correspond to the capture and image numbers respectively, see Fig. 19. Omitted re-projection errors for Target C are greyed out, indicating no re-projection error reported due to invalid Vicon data. Also, blue indicates targets used as orientation sources for the row's re-projection calculations, while red indicates re-projections reported for the corresponding target, but the target was not used in the orientation of the camera. The yellow data bars graphically represent the numerical re-projection error values with respect to the span of all values reported.

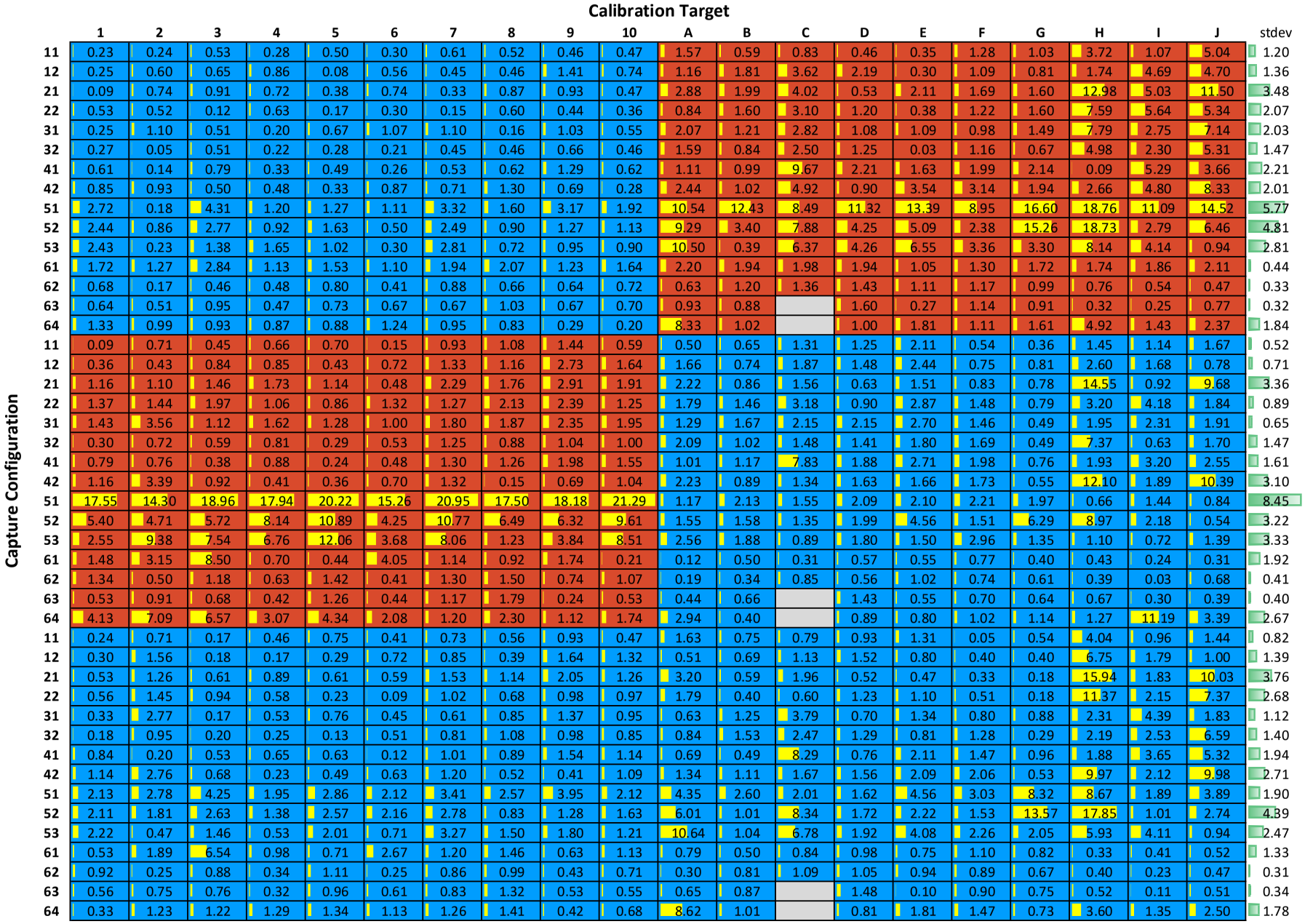


Fig. 21. Camera re-projection error, sorted by capture configuration, then camera orientation source. See Fig. 20 for an explanation of the table colors and axes.

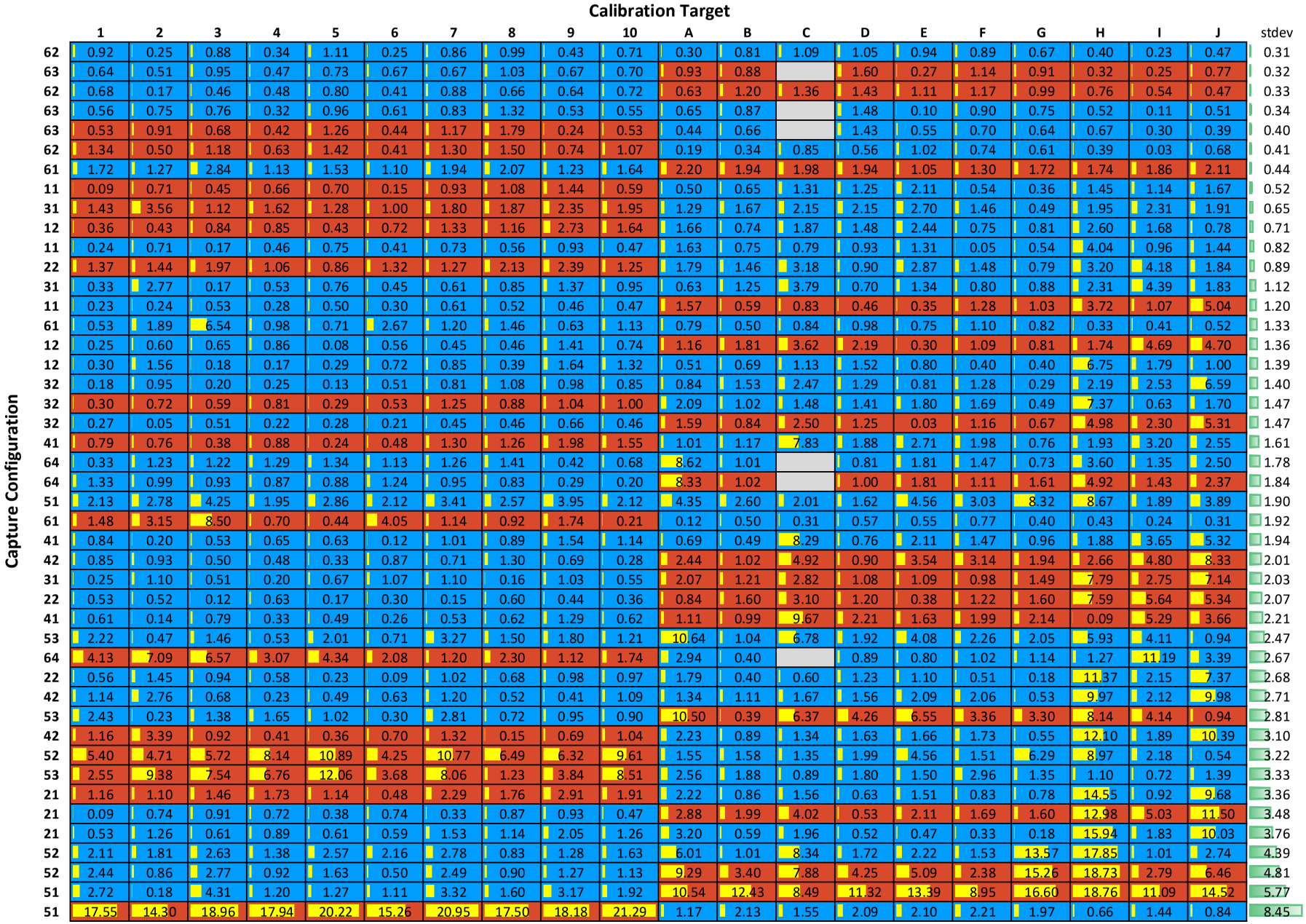


Fig. 22. Camera re-projection error, sorted by the standard deviation of capture re-projection error for each row. See Fig. 20 for an explanation of the table colors and axes.

indicate that this method works best when orientation targets are near or intermingled with the objects of interest that we want to re-project.

In future work, we could explore the following:

- Up close configurations weren't explored, all captures were take at a relatively far distance away from the targets. Future captures could test how well this pipeline performs at close distances that are relatively common in AAR operations.
- We should probably run this pipeline on the different image quadrants (e.g., upper left quadrant, corners, edges, etc.), instead of just focusing on centered objects of interest.
- We could explore varying Solve PnP parameters such as iteration count, etc.
- What are the effect of random noise to the truth pixels?
- Automate target localization in imagery using YOLO and targets parallel with camera image plane to overcome the parallax effect.

REFERENCES

[1] D. Worth, J. Lynch, and S. Nykl, "Single-shot pose estimation for aar with yolo and perspective-n-point," in *Proceedings of the ION Joint Navigation Conference*, 6 2022. [Online]. Available: https://www.ion.org/jnc/abstracts.cfm?paperID=11039

[2] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European conference on computer vision*. Springer, 2020, pp. 566–583.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[4] Y. Yang, K. J. Liang, and L. Carin, "Object detection as a positive-unlabeled problem," *arXiv preprint arXiv:2002.04672*, 2020.

[5] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[6] C. M. Insights, "Motion capture technology market 2022 – 2028 increasing demand, key players to taking advantage of this opportunity: Vicon, xsens, mo-sys, optitrack," 2022. [Online]. Available: https://www.digitaljournal.com/pr/motion-capture-technology-market-2022-2028-increasing-demand-key-players-to-taking-advantage-of-this-opportunity-vicon-xsens-mo-sys-optitrack

[7] Rokokol, "The complete guide to professional motion capture," 2022. [Online]. Available: https://www.rokoko.com/insights/the-complete-guide-to-professional-motion-capture

[8] F. Serpiello, "Compare motion capture systems," 2021. [Online]. Available: https://www.comparesportstech.com/compare-motion-capture-systems

[9] C. Biotech, "Accuracy and precision of measurements," 2022, [Online; accessed January 8, 2023]. [Online]. Available: https://www.cherrybiotech.com/wp-content/uploads/2018/06/Figure-1.png

[10] P. Eichelberger, M. Ferraro, U. Minder, T. Denton, A. Blasimann, F. Krause, and H. Baur, "Analysis of accuracy in optical motion capture–a protocol for laboratory setup evaluation," *Journal of biomechanics*, vol. 49, no. 10, pp. 2085–2088, 2016.

[11] International Organization for Standardization, *ISO 5725-1 Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions*. International Organization for Standardization, 1994.

[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[13] K. Sadekar and S. Mallick, "Camera calibration using opencv," 2020. [Online]. Available: https://learnopencv.com/camera-calibration-using-opencv/

[14] S.-E. Lee, K. Shibata, S. Nonaka, S. Nobuhara, and K. Nishino, "Extrinsic camera calibration from a moving person," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 344–10 351, 2022.

[15] Y. Chen, Y. Chen, and G. Wang, "Bundle adjustment revisited," *arXiv preprint arXiv:1912.03858*, 2019.

[16] G. Carrera, A. Angeli, and A. J. Davison, "Slam-based automatic extrinsic calibration of a multi-camera rig," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2652–2659.

[17] A.-S. Vaida and S. Nedevschi, "Automatic extrinsic calibration of lidar and monocular camera images," in *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2019, pp. 117–124.

[18] Y. Zhuang, F. Yan, and H. Hu, "Automatic extrinsic self-calibration for fusing data from monocular vision and 3-d laser scanner," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 7, pp. 1874–1876, 2014.

[19] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[20] G. H. Lee, M. Achtelik, F. Fraundorfer, M. Pollefeys, and R. Siegwart, "A benchmarking tool for mav visual pose estimation," in *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, 2010, pp. 1541–1546.

[21] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[22] S. Chiodini, M. Pertile, R. Giubilato, F. Salvioli, M. Barrera, P. Franceschetti, and S. Debei, "Camera rig extrinsic calibration using a motion capture system," in *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE, 2018, pp. 590–595.

[23] E. Rodenburgh and C. Taylor, "A system for evaluating vision-aided navigation uncertainty," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, 2020, pp. 2272–2280.

[24] P. Ganesh, K. Volle, P. Buzaud, K. Brink, and A. Willis, "Extrinsic calibration of camera and motion capture systems," in *SoutheastCon 2021*. IEEE, 2021, pp. 01–08.

[25] D. Worth, "Vicon aftrburner sync," 2022. [Online]. Available: https://www.youtube.com/watch?v=ZDAO1cpLSfY

[26] V. Inc., "Datastream sdk documentation," 2022. [Online]. Available: https://docs.vicon.com/display/DSSDK111

[27] A. Inc., "iphone user guide," 2023, [Online; accessed January 15, 2023]. [Online]. Available: https://support.apple.com/guide/iphone/set-up-your-shot-iph3dc593597/ios

[28] T. G. Team, "Gnu image manipulation program," 2023, [Online; accessed January 20, 2023]. [Online]. Available: https://www.gimp.org/

[29] X. X. Lu, "A review of solutions for perspective-n-point problem in camera pose estimation," in *Journal of Physics: Conference Series*, vol. 1087, no. 5. IOP Publishing, 2018, p. 052009.

[30] S. Nykl, "Aftrburner 3d visualization engine," May 2022. [Online]. Available: http://www.nykl.net/aburn

Bibliography

1. J. C. Fredriksen, *The United States Air Force: A Chronology*. ABC-CLIO, 2011.

2. S. LaGrone, "Mq-25a unmanned aerial tanker refuels super hornet in successful first test," *USNI News*, 6 2021. [Online]. Available: https://news.usni.org/2021/06/07/mq-25a-unmanned-aerial-tanker-refuels-f-a-18-hornet-in-successful-first-test [Accessed: April 12, 2023]

3. K. Johnson, "Airbus a330 mrtt certificated for automatic aerial refueling," *Flying*, 7 2022. [Online]. Available: https://www.flyingmag.com/airbus-a330-mrtt-certificated-for-automatic-aerial-refueling [Accessed: April 12, 2023]

4. J. Hinchman and D. Schreiter, "Automated aerial refueling presentation to 2007 arsag conference (preprint)," Air Force Research Lab Wright-Patterson AFB OH Air Vehicle Directorate, Tech. Rep., 2007.

5. G. Balamurugan, J. Valarmathi, and V. Naidu, "Survey on uav navigation in gps denied environments," in *2016 International conference on signal processing, communication, power and embedded system (SCOPES)*. IEEE, 2016, pp. 198–204.

6. M. Narasimhappa, A. D. Mahindrakar, V. C. Guizilini, M. H. Terra, and S. L. Sabat, "Mems-based imu drift minimization: Sage husa adaptive robust kalman filtering," *IEEE Sensors Journal*, vol. 20, no. 1, pp. 250–260, 2019.

7. K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

8. D. Worth, J. Choate, J. Lynch, S. Nykl, and C. Taylor, "Relative vectoring using dual object detection for autonomous aerial refueling," 2023. [Online]. Available: https://youtu.be/RXbrBl8Re7M [Accessed: April 29, 2023]

9. W. Williamson, J. Min, J. Speyer, and J. Farrell, "A comparison of state space, range space, and carrier phase differential gps/ins relative navigation," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 4. IEEE, 2000, pp. 2932–2938.

10. E. A. Olsen, C.-W. PARK, and J. P. How, "3d formation flight using differential carrier-phase gps sensors," *Navigation*, vol. 46, no. 1, pp. 35–48, 1999.

11. C. Tomasi, "A simple camera model," in *Notes from computer science 527*, 2015. [Online]. Available: https://courses.cs.duke.edu//fall16/compsci527/notes/camera-model.pdf [Accessed: April 12, 2023]

12. S. Chen, H. Duan, Y. Deng, C. Li, G. Zhao, and Y. Xu, "Drogue pose estimation for unmanned aerial vehicle autonomous aerial refueling system based on infrared vision sensor," *Optical Engineering*, vol. 56, no. 12, p. 124105, 2017.

13. G. Campa, M. R. Napolitano, and M. L. Fravolini, "Simulation environment for machine vision based aerial refueling for uavs," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 1, pp. 138–151, 2009.

14. J.-Y. Du, *Vision based navigation system for autonomous proximity operations: an experimental and analytical study*. Texas A&M University, 2004.

15. H. Duan and Q. Zhang, "Visual measurement in simulation environment for vision-based uav autonomous aerial refueling," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 9, pp. 2468–2480, 2015.

16. L. Xin, D. Luo, and H. Li, "A monocular visual measurement system for uav probe-and-drogue autonomous aerial refueling," *International Journal of Intelligent Computing and Cybernetics*, 2018.

17. M. Mammarella, G. Campa, M. R. Napolitano, and M. L. Fravolini, "Comparison of point matching algorithms for the uav aerial refueling problem," *Machine Vision and Applications*, vol. 21, no. 3, pp. 241–251, 2010.

18. H. Duan, L. Xin, and S. Chen, "Robust cooperative target detection for a vision-based uavs autonomous aerial refueling platform via the contrast sensitivity mechanism of eagle's eye," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 3, pp. 18–30, 2019.

19. C.-P. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 6, pp. 610–622, 2000.

20. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

21. J. Kimmett, J. Valasek, and J. Junkins, "Autonomous aerial refueling utilizing a vision based navigation system," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4469.

22. M. D. Tandale, R. Bowers, and J. Valasek, "Trajectory tracking controller for vision-based probe and drogue autonomous aerial refueling," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 4, pp. 846–857, 2006.

23. T. Erkin, O. Abdo, Y. Sanli, H. Celik, and H. Isci, "Vision-based autonomous aerial refueling," in *AIAA SCITECH 2022 Forum*, 2022, p. 1384.

24. Y. Fan, J. Huang, T. Jia, and C. Bai, "A visual marker detection and position method for autonomous aerial refueling of uavs," in *2021 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2021, pp. 1006–1011.

25. W. Xufeng, D. Xinmin, and K. Xingwei, "Feature recognition and tracking of aircraft tanker and refueling drogue for uav aerial refueling," in *2013 25th Chinese Control and Decision Conference (CCDC)*. IEEE, 2013, pp. 2057–2062.

26. X. F. Wang, X. M. Dong, X. W. Kong, and J. H. Zhi, "Vision based measurement of refueling drogue for autonomous aerial refueling," in *Applied Mechanics and Materials*, vol. 590. Trans Tech Publ, 2014, pp. 618–622.

27. K. Zhao, Y. Sun, H. Li, Y. Fu, and Q. Zeng, "A novel drogue pose estimation method for autonomous aerial refueling based on monocular vision sensor," *IEEE Sensors Journal*, 2022.

28. K. Zhao, Y. Sun, H. Li, L. Wu, and Y. Fu, "Monocular visual pose estimation for flexible drogue by decoupling the deformation," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

29. C. Parsons and S. Nykl, "Real-time automated aerial refueling using stereo vision," in *International Symposium on Visual Computing*. Springer, 2016, pp. 605–615.

30. C. Parsons, Z. Paulson, S. Nykl, W. Dallman, B. G. Woolley, and J. Pecarina, "Analysis of simulated imagery for real-time vision-based automated aerial refueling," *Journal of Aerospace Information Systems*, vol. 16, no. 3, pp. 77–93, 2019.

31. J. Zhang, Z. Liu, Y. Gao, and G. Zhang, "Robust method for measuring the position and orientation of drogue based on stereo vision," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4298–4308, 2020.

32. J. Anderson, J. Miller, X. Wu, S. Nykl, C. Taylor, and W. Watkinson, "Real-time automated aerial refueling with stereo vision: Overcoming gnss-denied environments in or near combat areas," *Inside GNSS*, pp. 32–41, 8 2021. [Online]. Available: https://insidegnss.com/real-time-automated-aerial-refueling-with-stereo-vision-overcoming-gnss-denied-environments-in-c [Accessed: April 12, 2023]

33. C.-I. Chen, R. Koseluk, C. Buchanan, A. Duerner, B. Jeppesen, and H. Laux, "Autonomous aerial refueling ground test demonstration—a sensor-in-the-loop, non-tracking method," *Sensors*, vol. 15, no. 5, pp. 10 948–10 972, 2015.

34. J. Curro, J. Raquet, T. Pestak, J. Kresge, and M. Smearcheck, "Automated aerial refueling position estimation using a scanning lidar," in *Proceedings of the 25th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2012)*, 2012, pp. 774–782.

35. Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3003–3013.

36. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

37. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

38. S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International conference on machine learning*. PMLR, 2015, pp. 597–606.

39. Q. Guan, W. Li, S. Xue, and D. Li, "High-resolution representation object pose estimation from monocular images," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 980–984.

40. M. A. Dede and Y. Genc, "Object aspect classification and 6dof pose estimation," *Image and Vision Computing*, p. 104495, 2022.

41. S. Sun, Y. Yin, X. Wang, and D. Xu, "Robust landmark detection and position measurement based on monocular vision for autonomous aerial refueling of uavs," *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4167–4179, 2018.

42. J. A. B. Garcia and A. B. Younes, "Real-time navigation for drogue-type autonomous aerial refueling using vision-based deep learning detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 4, pp. 2225–2246, 2021.

43. J. Cheng, P. Liu, Q. Zhang, H. Ma, F. Wang, and J. Zhang, "Real-time and efficient 6-d pose estimation from a single rgb image," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–14, 2021.

44. J. Kang, W. Liu, W. Tu, and L. Yang, "Yolo-6d+: single shot 6d pose estimation using privileged silhouette information," in *2020 International Conference on Image Processing and Robotics (ICIP)*. IEEE, 2020, pp. 1–6.

45. W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

46. C. Li, S. Sun, X. Song, H. Song, N. Akhtar, and A. S. Mian, "Simultaneous multiple object detection and pose estimation using 3d model infusion with monocular vision," *arXiv preprint arXiv:2211.11188*, 2022.

47. M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3828–3836.

48. Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

49. B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 292–301.

50. M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 699–715.

51. W.-L. Huang, C.-Y. Hung, and I.-C. Lin, "Confidence-based 6d object pose estimation," *IEEE Transactions on Multimedia*, 2021.

52. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

53. Learn OpenGL, "Coordinate systems," 2015. [Online]. Available: https://learnopengl.com/Getting-started/Coordinate-Systems [Accessed: April 12, 2023]

54. S. Nykl, "Aftrburner 3d visualization engine," May 2022. [Online]. Available: http://www.nykl.net/aburn [Accessed: April 12, 2023]

55. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

56. J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

57. ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

58. J. Lynch, "Monocular pose estimation for automated aerial refueling via perspective-n-point," Master's thesis, Air Force Institute of Technology, 2022.

59. D. Worth, J. Lynch, and S. Nykl, "Single-shot pose estimation for aar with yolo and perspective-n-point," in *Proceedings of the ION Joint Navigation Conference*, 6 2022.

60. J. Yu and H. Choi, "Yolo mde: Object detection with monocular depth estimation," *Electronics*, vol. 11, no. 1, p. 76, 2021.

61. Ultralytics, "Yolov5 in pytorch," 2022. [Online]. Available: https://github.com/ultralytics/yolov5 [Accessed: April 12, 2023]

62. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

63. Y. Yang, K. J. Liang, and L. Carin, "Object detection as a positive-unlabeled problem," *arXiv preprint arXiv:2002.04672*, 2020.

64. B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European conference on computer vision*. Springer, 2020, pp. 566–583.

65. OpenCV Team, "Open source computer vision library v4.5.5," Dec. 2021. [Online]. Available: https://opencv.org/opencv-4-5-5/ [Accessed: April 12, 2023]

66. L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

67. AprilRobotics, "Apriltag-imgs," 2018. [Online]. Available: https://github.com/AprilRobotics/apriltag-imgs [Accessed: April 25, 2023]