Neural Computing and Applications
https://doi.org/10.1007/s00521-024-09589-y

ORIGINAL ARTICLE

Relative vectoring using dual object detection for autonomous aerial refueling

Derek Worth[1] · Jeffrey Choate[1] · James Lynch[1] · Scott Nykl[1] · Clark Taylor[1]

Received: 28 April 2023 / Accepted: 5 February 2024

Abstract
Once realized, autonomous aerial refueling will revolutionize unmanned aviation by removing current range and endurance limitations. Previous attempts at establishing vision-based solutions have come close but rely heavily on near perfect extrinsic camera calibrations that often change midflight. In this paper, we propose dual object detection, a technique that overcomes such requirement by transforming aerial refueling imagery directly into receiver aircraft reference frame probe-to-drogue vectors regardless of camera position and orientation. These vectors are precisely what autonomous agents need to successfully maneuver the tanker and receiver aircraft in synchronous flight during refueling operations. Our method follows a common 4-stage process of capturing an image, finding 2D points in the image, matching those points to 3D object features, and analytically solving for the object pose. However, we extend this pipeline by simultaneously performing these operations across two objects instead of one using machine learning and add a fifth stage that transforms the two pose estimates into a relative vector. Furthermore, we propose a novel supervised learning method using bounding box corrections such that our trained artificial neural networks can accurately predict 2D image points corresponding to known 3D object points. Simulation results show that this method is reliable, accurate (within 3 cm at contact), and fast (45.5 fps).

Keywords YOLO · Perspective-n-point · Relative navigation · Computer vision · Machine learning · Software simulations

1 Introduction

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 [15]. 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 Stingray successfully refueled a F/A-18F Super Hornet, a first-of-its-kind aerial refueling test involving an unmanned tanker [27]. 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 [23]. 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 [20].

✉ Derek Worth
derek.worth.2@au.af.edu

Jeffrey Choate
jeffrey.choate@au.af.edu

James Lynch
james.lynch.22@us.af.mil

Scott Nykl
scott.nykl@au.af.edu

Clark Taylor
clark.taylor.3@au.af.edu

[1] Department of Electrical and Computer Engineering, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433, USA

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 Fig. 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 using global navigation satellite systems such as 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 [2], IMUs are inherently noisy and drift over time [35], 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 paper 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 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

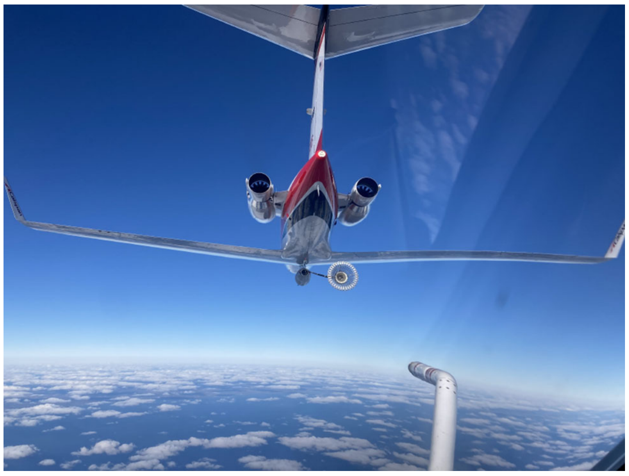


Fig. 1 The pilot must *see* both the probe and drogue during aerial refueling operations

work, but a scenario of a refueling receptacle and tanker boom could be adapted using the same method. Furthermore, only simulated refueling approaches are explored which opens the door to future work that could focus on bridging the gap between the virtual and real worlds through the use of machine transfer learning—an area of research that has already shown great promise [52].

To summarize, the research objective of this paper was to establish a computer vision technique for guiding one object to another, namely a probe tip to a drogue (or boom to receptacle), to help solve the autonomous aerial refueling problem. Contributions set forth by this paper include:

1. A novel machine learning technique for finding 2D image points that correspond to known 3D model points.
2. A novel vectoring technique using dual object detection for accurate, reliable, and real time relative navigation between two objects.
3. A simulation framework for developing and testing the relative vectoring technique discussed in this paper, which includes automation of labeled training data for machine learning.

The remainder of this paper is divided into four sections. Section 2 discusses related works in the AAR domain. Section 3 details the relative vectoring pipeline and how it is evaluated in a 3D virtual world. Pipeline performance results from different camera and feature selection configurations are summarized in Sect. 4. Finally, Sect. 5 discusses closing thoughts and future work.

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

2 Related works

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) [37, 53]. 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 [49]. 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, 2D feature detection, 2D to 3D feature matching, and pose estimation.

Analytical approaches Early attempts at feature detection involved processing pixels using methods such as infrared LED blob detection [5], Harris corner detection [3], VisNav [9], speeded up robust features (SURF), scale-invariant feature transform (SIFT), HSV color segmentation, and active contouring [10]. Each of these methods produced 2D image points that were subsequently matched to corresponding 3D model points using algorithms like mutual nearest point [3], classical assignment model, perspective transformation-based matching [57], maximum clique detection [34], and the Munkres algorithm [11]. Finally, the resulting 2D to 3D matches were transformed into 6° of freedom (6DoF) pose estimates using one of three algorithms, namely Gaussian least squares differential correction (GLSDC), the LHM algorithm (i.e., orthogonal iteration) [32], or perspective-n-point (PnP) [14].

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 [26, 47].

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 [12]. 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 [13]. Xufeng used rigid and flat uniformly colored rings that are easy to find in images, but do not exist in any real drogue configurations [51, 58]. 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 [63, 64]. Ellipse detection methods show potential, but currently remain untested in realistic scenarios.

Stereo vision approaches, like the one proposed by Parsons [39, 40], 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 [61]. 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 [1].

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 [4]. 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 [7]. Similarly, the FFB6D algorithm in [19] used depth from RGB-D images for accurate real time pose estimation, but the authors also reported certain textures resulted in decreased accuracy.

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 [17], segmentation [31], and tracking [21] 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 [18]. Dede implemented a convolutional neural network that could easily find semantic key-points [8]. 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 [45]. Some researchers, like Garcia [16], 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 [6], YOLO-6D+ [24], SSD-6D [25], SMOPE-net [29], BB8 [41], PoseCNN [56], and a few others [22, 46, 48]. 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.

The main contributions of this paper address these challenges by reframing the AAR problem of "drogue pose estimation relative to the vision sensor" to that of "position estimation relative to the probe." As explained in the next section, this subtle difference allows us to ignore extrinsic camera properties and mitigate any challenges related to automating detection and tracking of a symmetric object (i.e., the drogue).

3 Methods

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 Fig. 2. This produces a bonded pair of 6DoF pose estimates in the camera's local reference frame corresponding to the two objects. We use the resulting poses to compute a vector between the two objects, transformed into the local reference frame of the estimated receiver pose, see Fig. 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. 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.

The rest of this section describes our proposed implementation of the relative vectoring pipeline above (also see

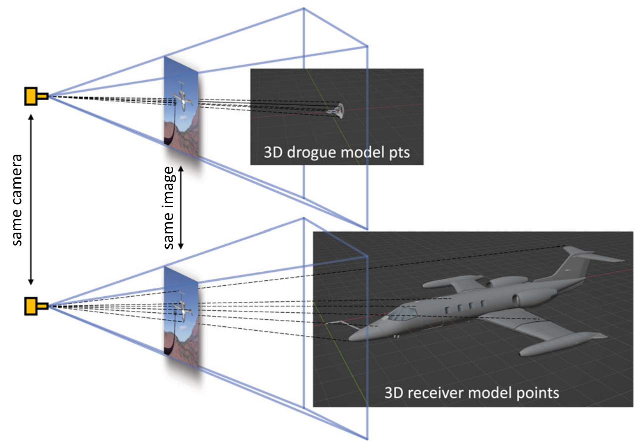


Fig. 2 Dual object detection uses machine learning to find 2D features of two separate objects in the same image and matches them to corresponding 3D model points before passing them to solve PnP for estimating relative poses of the objects

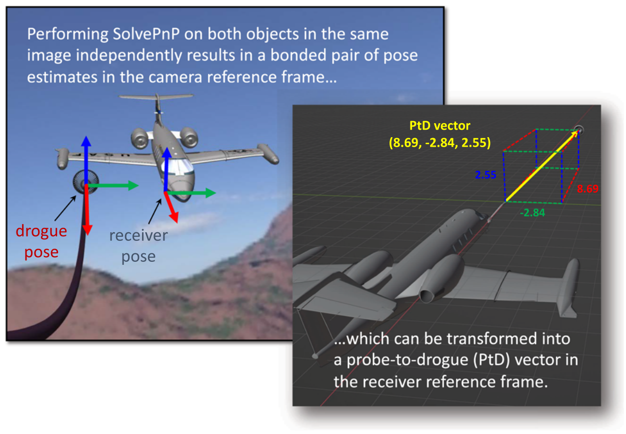

Performing SolvePnP on both objects in the same image independently results in a bonded pair of pose estimates in the camera reference frame...
drogue pose
receiver pose
PtD vector (8.69, -2.84, 2.55)
2.55
8.69
-2.84
...which can be transformed into a probe-to-drogue (PtD) vector in the receiver reference frame.

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

Fig. 4), experimentally justified design decisions, and methods for evaluating the pipeline.

3.1 Capturing imagery

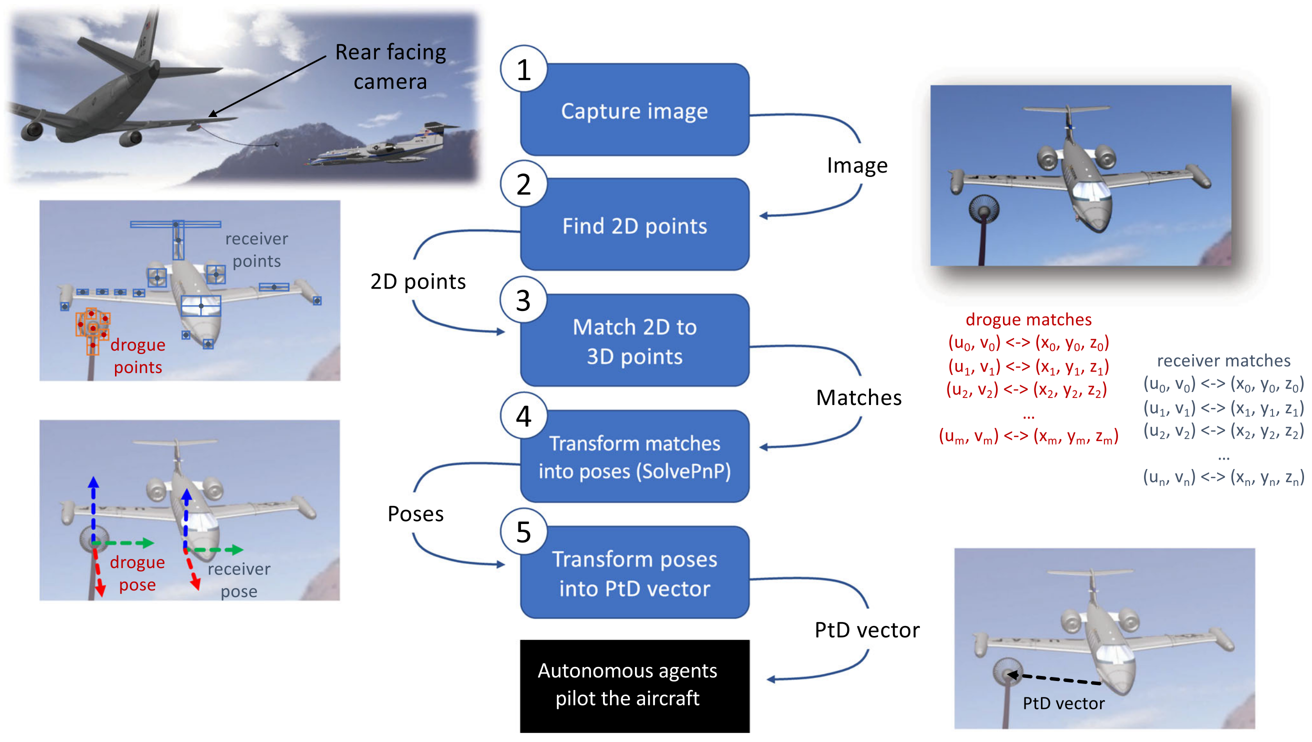
This may seem like the most simple and trivial stage—*just point the camera and snap a picture, right?* However, as any professional photographer or cinematographer will attest, images come in many different shapes and forms. The domain has several dimensions that effect computer vision, including pixel 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, we chose the camera properties in this effort with these trade spaces in mind.

Since the scope of this effort is limited to proving the viability of the proposed relative vectoring technique in simulation—leaving real world implementation for future work—we simplify the problem 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. We also vary the camera fields of view 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 [62], we restricted our simulated cameras 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, we chose a 2K resolution with a relatively wide aspect ratio of 1.90:1 for all cameras. Table 1 summarizes the camera parameters used in this effort.

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, our two primary camera locations on the receiver were forward facing on the probe side wing pod (see bottom of Fig. 6) and inside the cockpit. We also experimented with a rear facing camera mounted on the tanker buddy pod next to the drogue hose feed port.

Software simulation Implementing these cameras to model realistic aerial refueling scenarios 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. We chose the *AftrBurner* computer graphics engine [36] in this effort 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 [28] and depicted in Fig. 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 our experiments included the receiver aircraft (with attached refueling probe),



Rear facing camera
receiver points
drogue points
drogue pose
receiver pose
1 Capture image
2 Find 2D points
3 Match 2D to 3D points
4 Transform matches into poses (SolvePnP)
5 Transform poses into PtD vector
Autonomous agents pilot the aircraft
Image
2D points
Matches
Poses
PtD vector
drogue matches
$(u_0, v_0) <-> (x_0, y_0, z_0)$
$(u_1, v_1) <-> (x_1, y_1, z_1)$
$(u_2, v_2) <-> (x_2, y_2, z_2)$
...
$(u_m, v_m) <-> (x_m, y_m, z_m)$
receiver matches
$(u_0, v_0) <-> (x_0, y_0, z_0)$
$(u_1, v_1) <-> (x_1, y_1, z_1)$
$(u_2, v_2) <-> (x_2, y_2, z_2)$
...
$(u_n, v_n) <-> (x_n, y_n, z_n)$
PtD vector

Fig. 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

Table 1 Intrinsic camera parameters

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

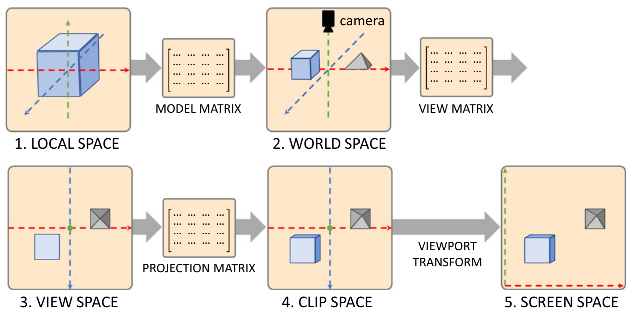


Fig. 5 Local to screen space transformation [28]

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, our 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, we modeled the receiver aircraft approaching the drogue basket extended behind a tanker flying straight and level (and while performing banking maneuvers), as shown in Fig. 6. This allowed us to validate the relative vectoring pipeline with 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

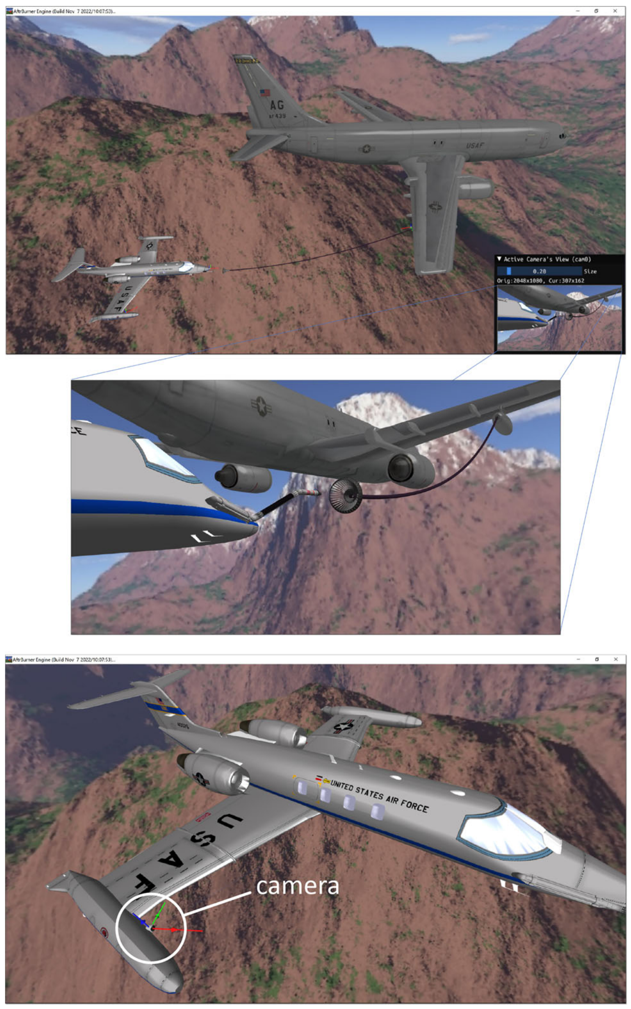


Fig. 6 The top image shows the AftrBurner simulation with the tanker and receiver flying in an aerial refueling echelon formation. A camera is mounted on the right wing of the receiver such that it can capture realistic imagery, as shown in the middle image, containing both receiver and tanker features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

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, we applied a simple relative motion model 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 Fig. 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

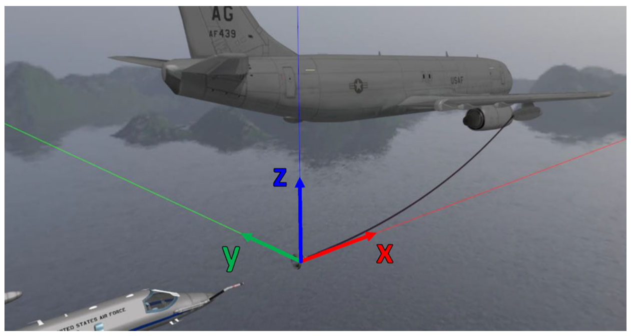


Fig. 7 Drogue-centered local reference frame

dynamic pose offset set to a random position within a specific range behind the drogue and an orientation set to match that of the tanker. Then, we randomly perturbed the pose between ± 10, ± 15, and ± 45° 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 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, we applied spherical linear interpolation over quaternions 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. We continued this process until contact was made, which we defined as less than 1 cm 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. We repeated this process continuously throughout all experiments and data collects.

3.2 Finding 2D image points

Using the camera pinhole model [49], 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, we turned to machine learning.

Object detection with YOLO Instead of creating or modifying an existing machine learning algorithm to perform pose estimation directly, we exploit what modern object detectors already can do with high precision, recall, and speed—namely, predicting 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 [42–44]. As originally proposed by Lynch [33] and Worth [54], 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+ [24] which both perform depth and pose estimation directly. However, the one used in this effort was the unaltered Ultralytics YOLOv5 PyTorch implementation found at [50]. 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, we train our models with labeled images generated with *bounding box corrections* specifically designed 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 [49]. In this effort, we refer to this phenomenon 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 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 Fig. 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 [30]. Even more concerning, manually labeled images are prone to human error and are often laced with missing annotations, misclassifications, and imprecise bounding boxes [59], all of which will most certainly decrease the accuracy of the proposed relative vectoring pipeline—thus, the need for automation.

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 Fig. 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. We do this 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 Fig. 10. An example of applying this process to synthetic imagery is depicted in Fig. 11.

Also, we excluded corrected bounding boxes from image labels that extended near (we chose a threshold of 10 pixels) or off the edge of the image frame. We did not want YOLO to 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, we chose to include all projected feature points 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

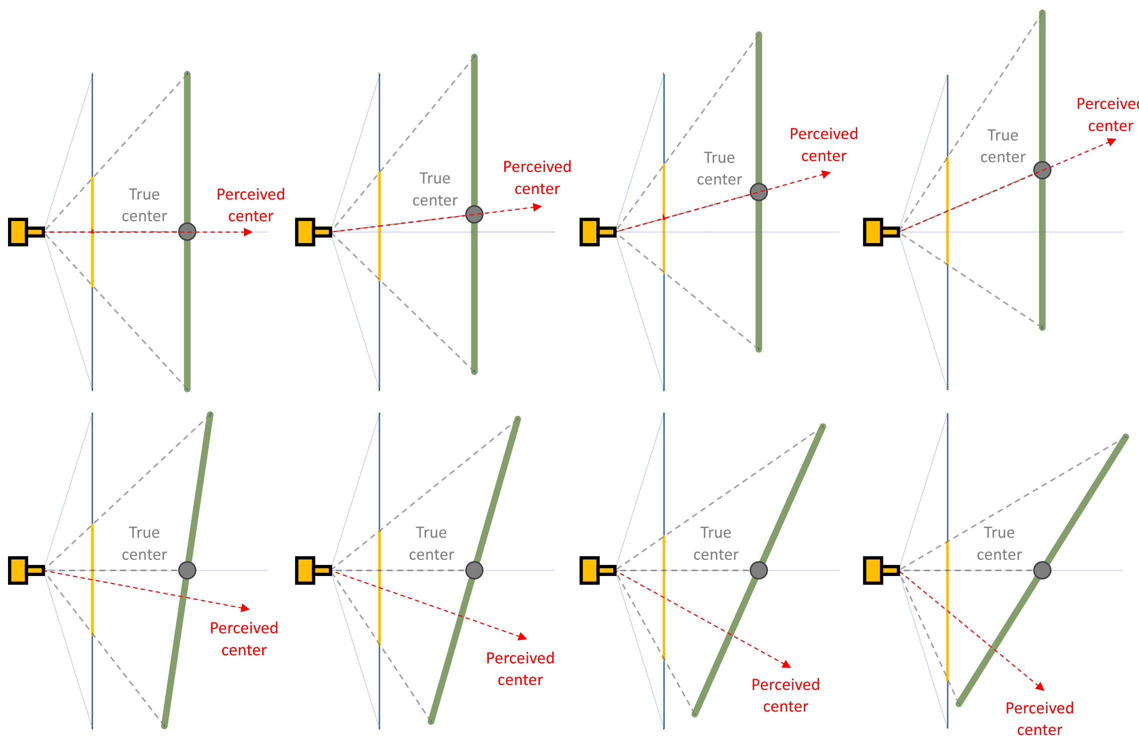


Fig. 8 The top row shows that translation does not effect the alignment of the true to perceived object center, as long as the object surface remains parallel to the camera's sensor plane. However, rotation (bottom row) has a parallax effect, forcing the perceived center to diverge from the truth

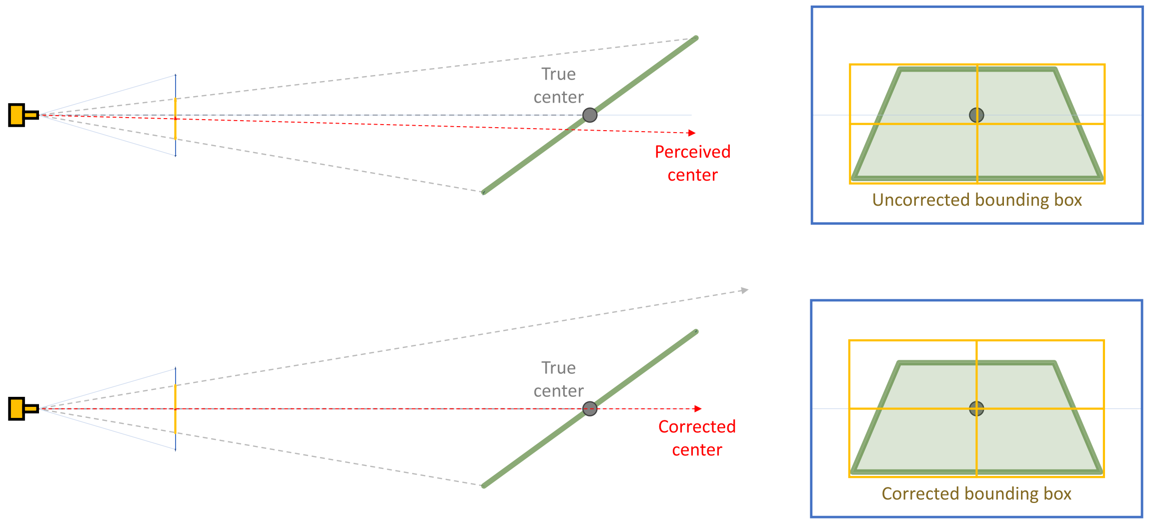


Fig. 9 Bounding box corrections align the perceived 2D center (i.e., original uncorrected bounding box center) with the true 3D center. Corrected image labels, as depicted on the bottom right, allow YOLO to learn these corrections and accurately predict 2D points corresponding to 3D feature centers

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 the results section, we did observe this in some of our experiments. However, this effect was not extensively tested in this effort and could serve as a good research area in future works.

Other machine learning decisions. With the above image labeling method and our Monte Carlo simulated approaches, we automated precise error-free labeling of thousands of high-resolution synthetic images in a relatively short

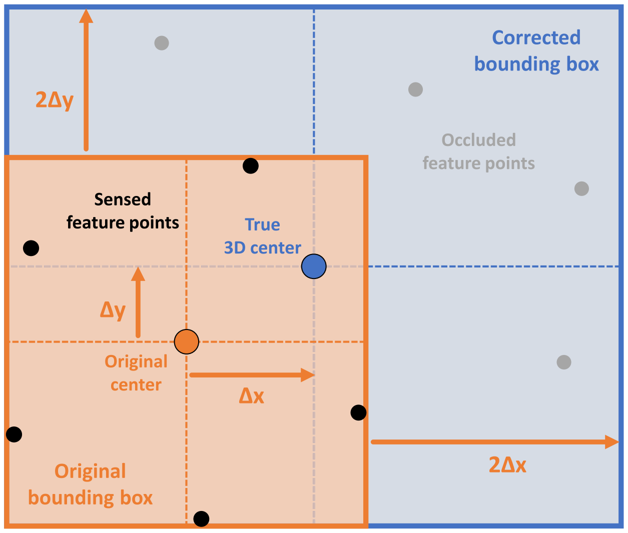


Fig. 10 Two sides grow outward until the bounding box center aligns with the true 3D feature center

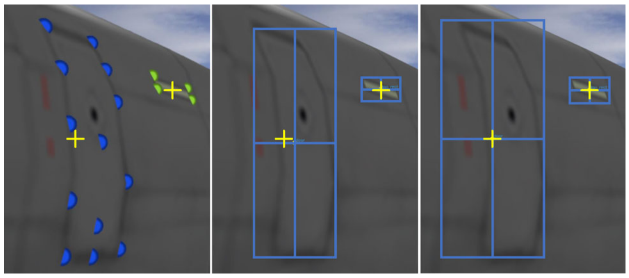


Fig. 11 The yellow crosshairs coincide with image projections of true 3D feature centers—these 2D center points are what we want YOLO to learn. On the left, we select 3D model points representing tanker features. The middle image shows the tightest fitting bounding boxes surrounding the 2D image projections 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 we train YOLO with

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 [65]. 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 our platform, listed in Table 2, limited the amount of 2K images we were able to cache and store in RAM. Thus, we limited each dataset to between 8000 and 10,000 images, each labeled with at most 40 receiver/probe and 40 tanker/drogue features, i.e., 80 features total per image. We chose an 80/20 training and validation data split and trained each model with 300 epochs and a batch size of 16.

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

For all experiments, we chose a *small* YOLOv5 model[2] with a relatively low resolution input size of 864×864, forcing us to resize and pad the 2K images generated by our 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 effort. All simulations, image labeling, training, and experimentation took place on a laptop with the configurations listed in Table 2.

3.3 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, we must pair them to correct corresponding 3D object model points. Fortunately, YOLO not only finds features, but also assigns each a class ID.

Interpreting predictions The internal complexity of YOLO and how it makes its predictions is beyond the scope of this paper. 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

[2] YOLOv5 model sizes include nano, small, medium, large, and xlarge with a performance (mean average precision) vs speed trade space outlined at [50].

P predictions on a square image with s pixel side lengths, where:

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

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 our 864×864 input images. To obtain the predictions corresponding to our 80 trained features, we apply objectness, class probability, and non-maximum suppression thresholds of 0.200, 0.250, and 0.200, respectively. This quickly narrows our search by filtering out any predictions with values below such thresholds. Of the remaining predictions, we perform feature assignment based on highest class probability and only retain the highest objectness scoring prediction per class. Finally, we store the bounding box (x, y) center coordinates as 2D feature image points and match them by class ID to their corresponding 3D model points. For any missing feature predictions, we simply omit them 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.

3.4 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, we consider the receiver and probe as a single object, i.e., any point on the receiver is considered a probe point, and define the object origins as the probe tip, drogue center, and average drogue center (relative to the tanker) for the receiver, drogue, and tanker, respectively. We also align the axes of the three reference frames similar to the previously described drogue-centered local reference frame (recall from Fig. 7) where the x, y, and z components of each reference frame point in the object forward, left, and up directions, respectively, see Fig. 12.

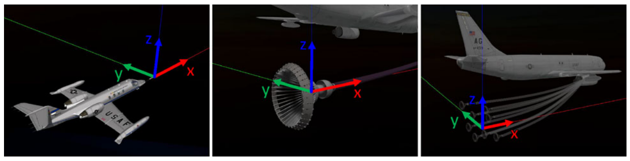


Fig. 12 Local reference frames. Note that the tanker reference frame origin is located at a fixed offset from the tanker at the average drogue center

With 3D points defined in their corresponding local reference frames, we can transform the two lists of 2D to 3D matches directly into probe tip and drogue centric pose estimates using one of the three analytical algorithms mentioned in Sect. 2. In this effort, we chose PnP—namely, OpenCV's RANSAC enabled *cv::solvePnPRansac* method [38]. In addition to the feature matches from the previous pipeline stage, we also supply the intrinsic camera parameters listed in Table 1, enable extrinsic guess based on the previous estimate, and set the 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*.

3.5 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 [38], we convert the probe's *rvec* into a direction cosine matrix, R_p^c, which transforms probe frame translation vectors, t^p, into camera frame vectors, t^c, such that:

$$t^c = R_p^c t^p$$

Maintaining consistency with our reference frames defined in Fig. 12, we also convert the camera frame z-forward *tvec* for both the probe and drogue 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, we first compute the camera frame PtD vector, $t_{p\to d}^c$, by subtracting the camera frame probe vector from the camera frame drogue vector:

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

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

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

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.

3.6 Evaluating pipeline performance

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

Metrics We quantify *accuracy* 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, we also define acceptable error 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, we consider prediction attempts unsuccessful, for example, when YOLO only finds two features in an image since PnP cannot make a prediction. Furthermore, we assume some other relative navigation technique (e.g., GPS) will be used until the probe and drogue are within 120 m of each other. Therefore, we automatically consider any predicted PtD vectors greater than 120 m in magnitude as unsuccessful outliers. Lastly, outliers greater than six standard deviations from a rolling average were also thrown out and considered unsuccessful.

Zoom dilemma Training a model to perform relative vectoring across the entire 120-meter 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 the probe's visibility, a problem that hinders dual object detection and only gets worse at a closer range. To overcome this dilemma, we implemented *model switching*. That is, we deployed multiple models, each trained on unique features specific to a particular camera configuration (i.e., vantage point, orientation, and zoom), and switched between them 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.

Experiments The following research questions motivated the experiments attempted in this effort:

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 Fig. 13, helped us answer these questions. We trained models A through C to detect features on the drogue and facilitate the receiver navigating to it directly while on final approach, leading up to contact. In contrast, we trained models D through L 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. We also varied configuration parameters such as feature selection and training range according to its corresponding model training description.

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 Fig. 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, we collected over 250,000 predictions 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, we applied pixel noise 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. We used threshold radii 0.0, 0.5, 1.0, and 2.5 pixels, respectively. Finally, we plotted the mean, variance ($\pm 2\sigma$), and prediction error magnitudes[3] of each collect as a function of distance between the probe and drogue.

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

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 drogue tracking (zoomed in)
B	podcam2	0–120 m	1	Rear facing drogue tracking (zoomed out)
C	dashcam	0–20 m	1	Forward facing drogue tracking
D	wingcam	0–120 m	1–6	Full range (baseline wingcam configuration[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° hFOV, respectively, with corresponding models trained on drogue features, while *wingcam* maintained a 55° hFOV and trained on tanker features

[b]Testing domains included: 1) model switching, 2) full versus 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

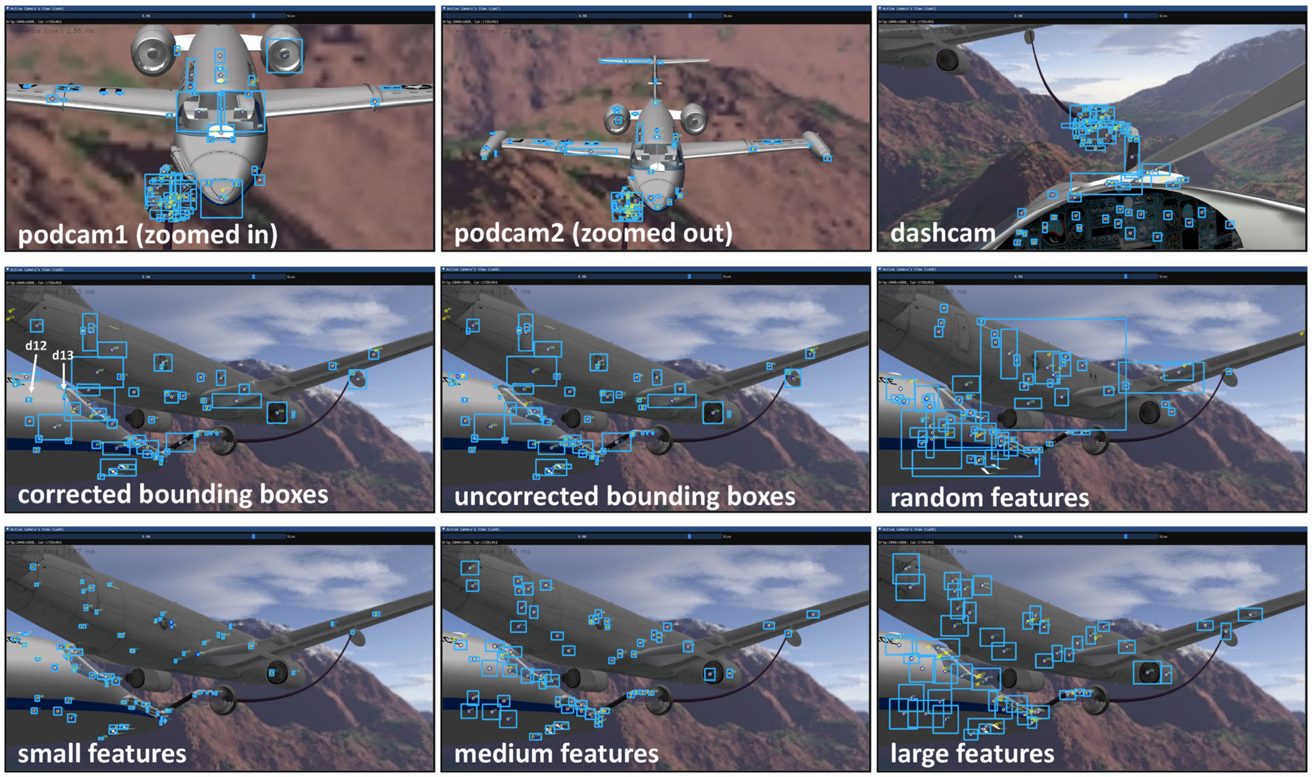


Fig. 13 Experimental vantage points. Light blue rectangles and dark blue circles depict YOLO predicted bounding boxes and corresponding bounding box center points, respectively, while the yellow circles depict true 2D feature points

4 Results and discussion

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 Fig. 13, most YOLO predictions appear closely aligned with the truth (yellow circle inside blue circle). 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 our video [55] 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. We also observed in simulation the receiver can always make a successful contact with the moving drogue when "navigating" using predicted relative vectors. Also, we observed several instances 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

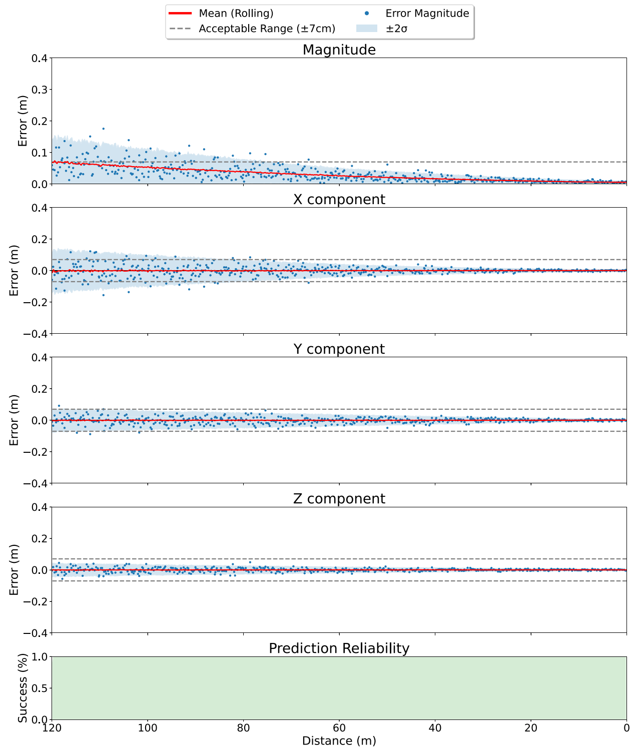


Fig. 15 Pipeline results from truth projections perturbed with 0.5 pixels of random noise

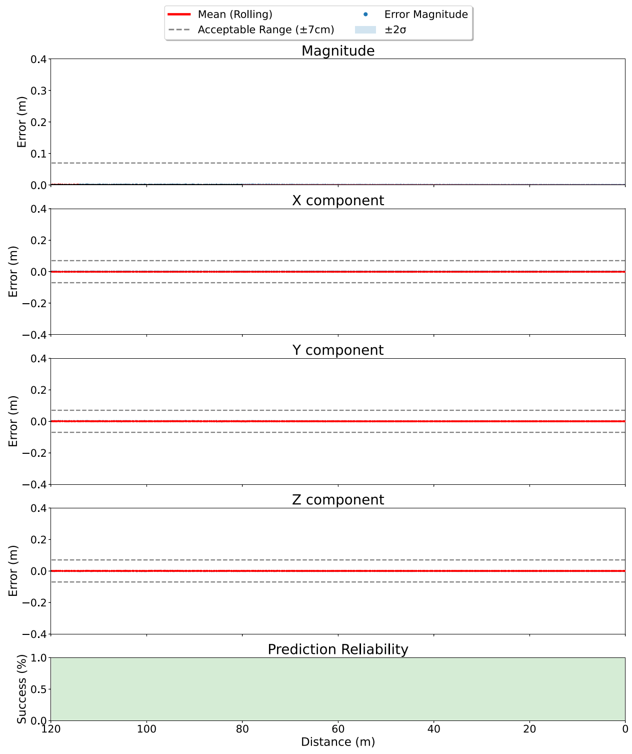


Fig. 14 Pipeline results from truth projections (floating point precision) with zero random noise

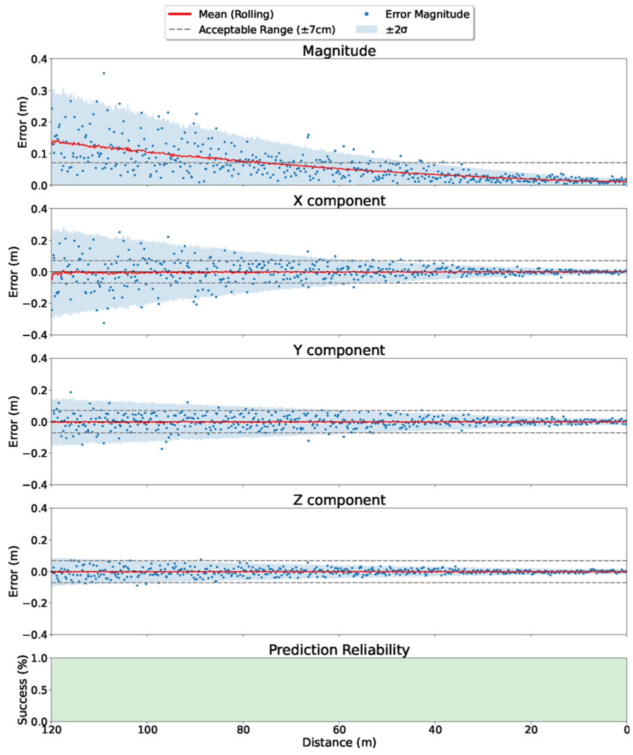


Fig. 16 Pipeline results from truth projections perturbed with 1.0 pixel of random noise

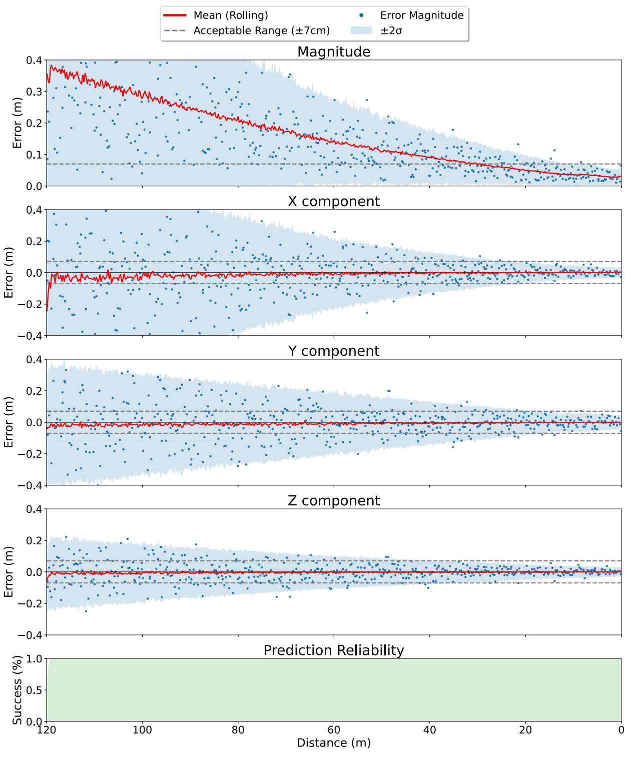


Fig. 17 Pipeline results from truth projections perturbed with 2.5 pixels of random noise

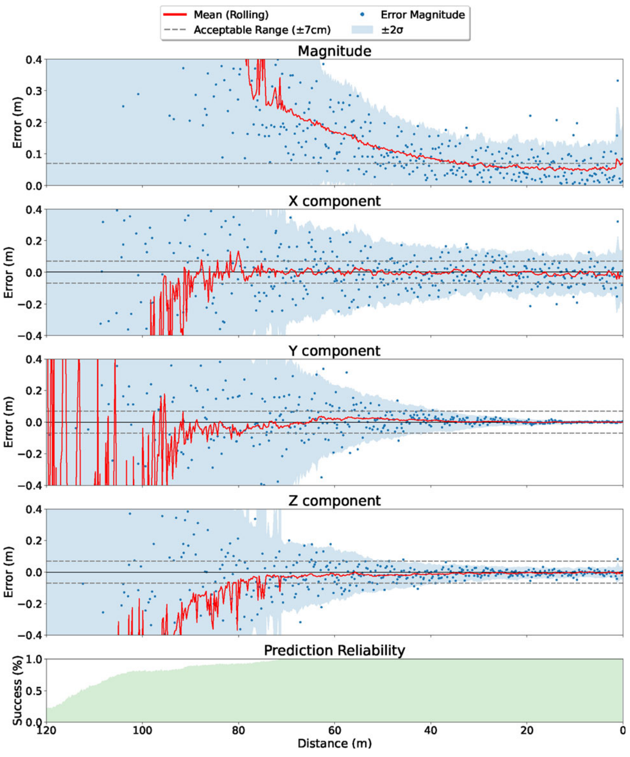


Fig. 19 Pipeline results from tanker podcam trained while *zoomed out* (model B)

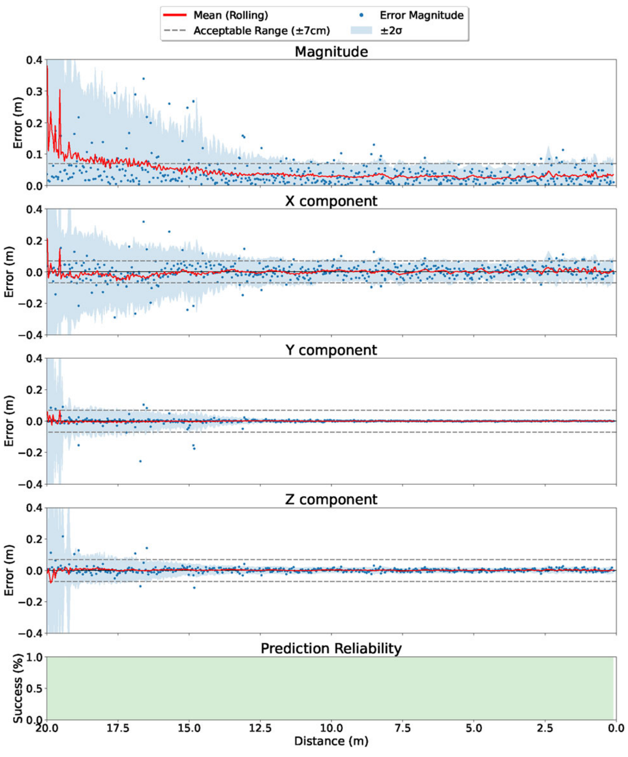


Fig. 18 Pipeline results from tanker podcam trained while *zoomed in* (model A)

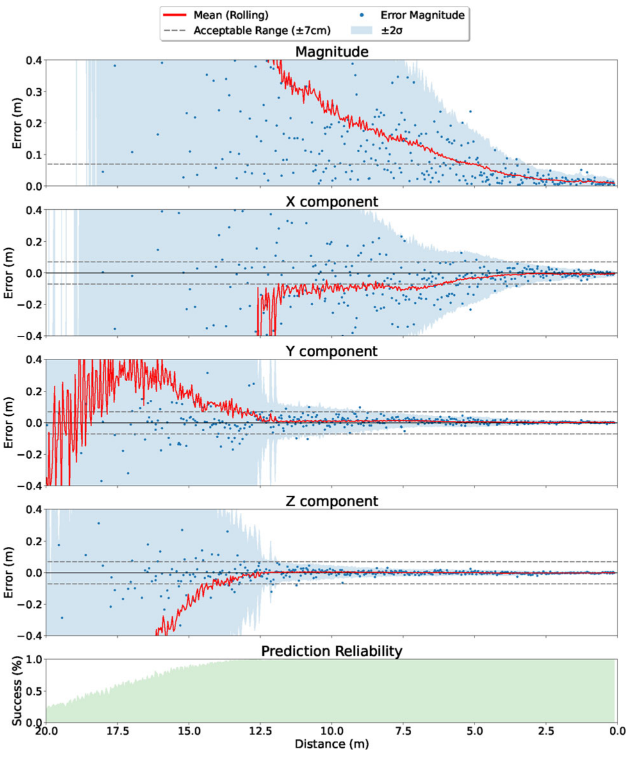


Fig. 20 Pipeline results from receiver dashcam trained at *close range* (model C)

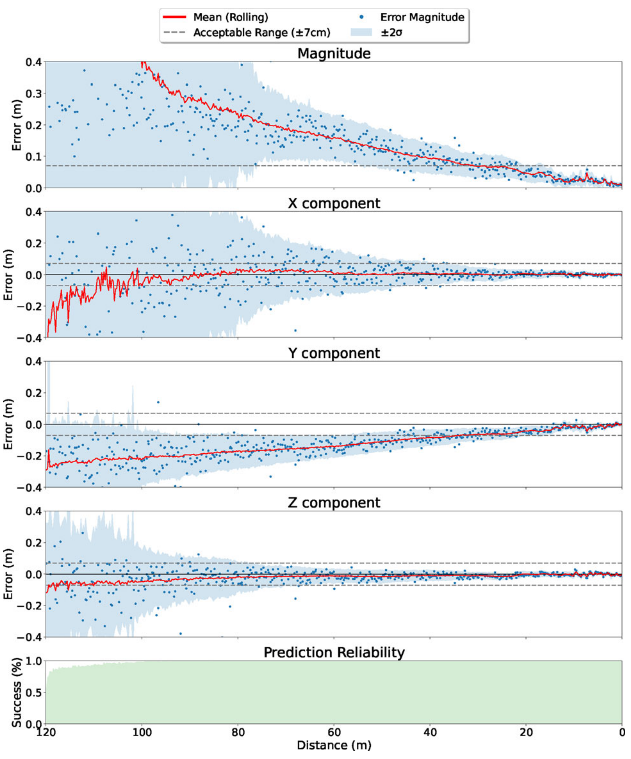


Fig. 21 Pipeline results from receiver wingcam trained at *full range* (model D)

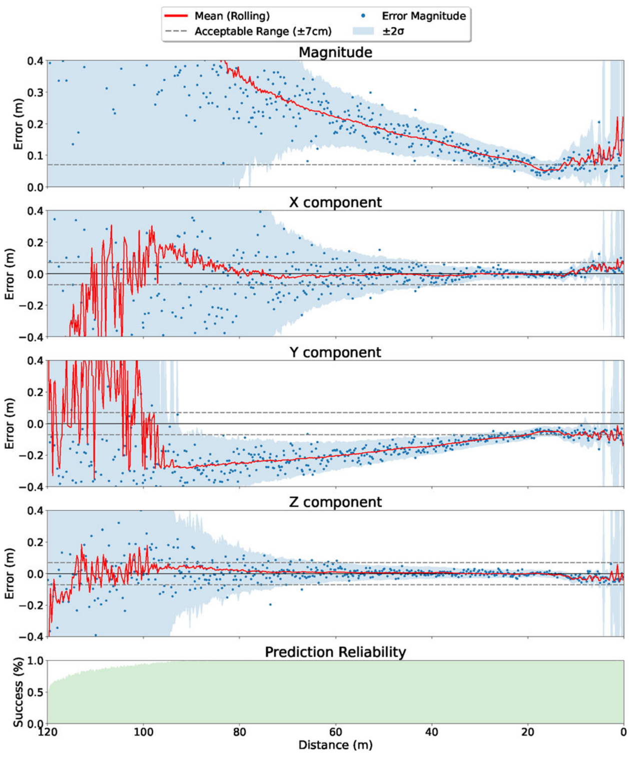


Fig. 23 Pipeline results from receiver wingcam trained at *mid-range* (model F)

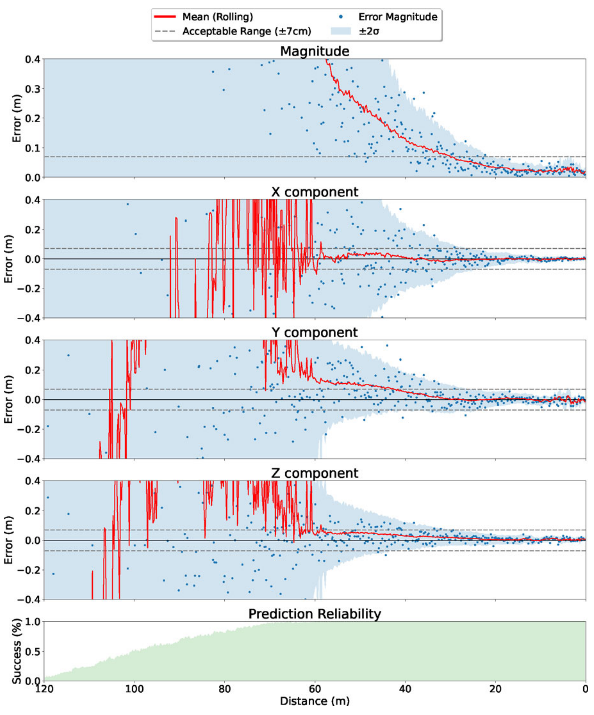


Fig. 22 Pipeline results from receiver wingcam trained at *close range* (model E)

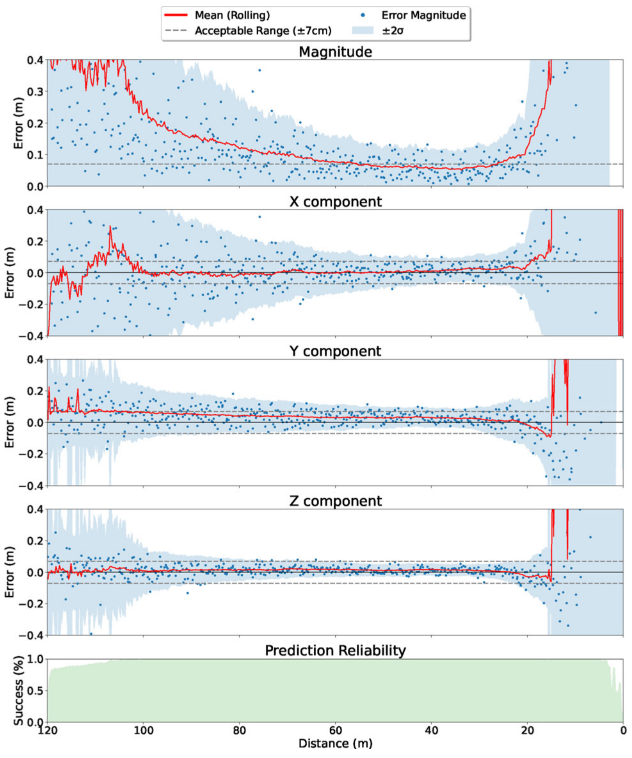


Fig. 24 Pipeline results from receiver wingcam trained at *long range* (model G)

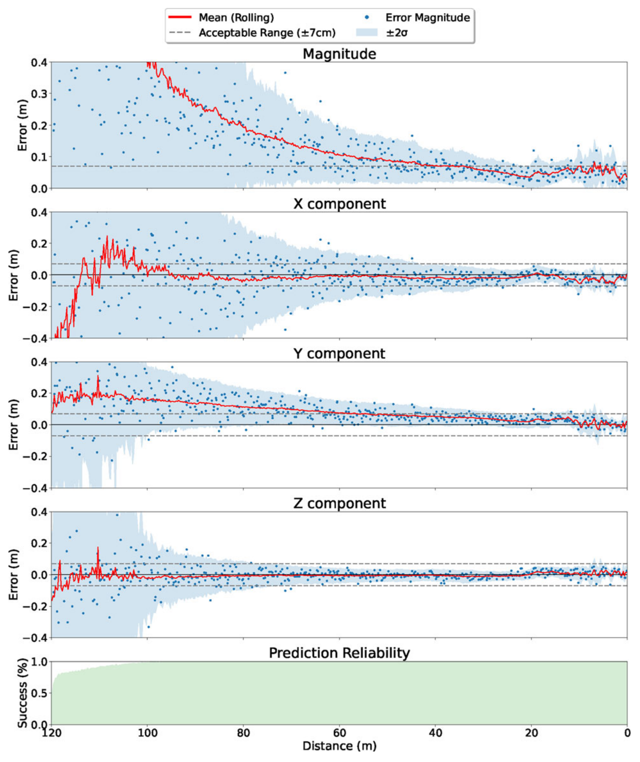


Fig. 25 Pipeline results from receiver wingcam trained with *uncorrected bounding boxes* (model H)

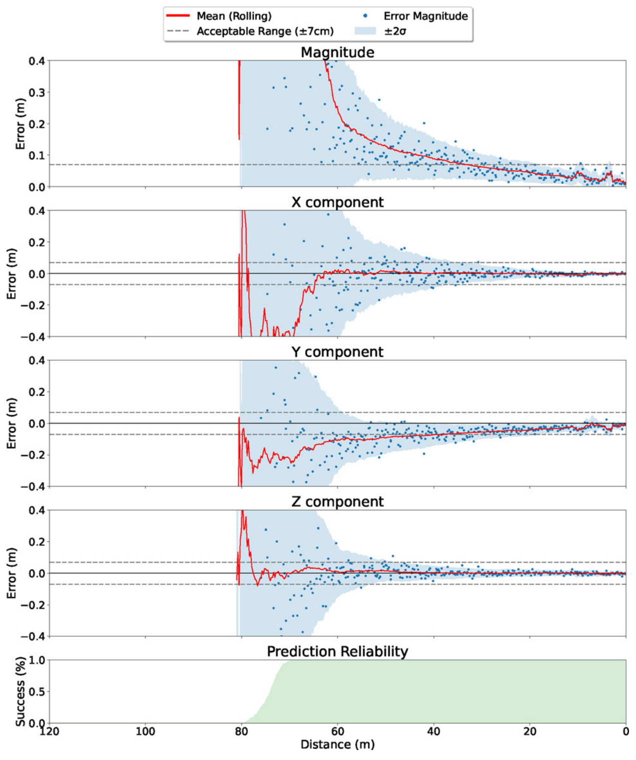


Fig. 27 Pipeline results from receiver wingcam trained with *small features* (model J)

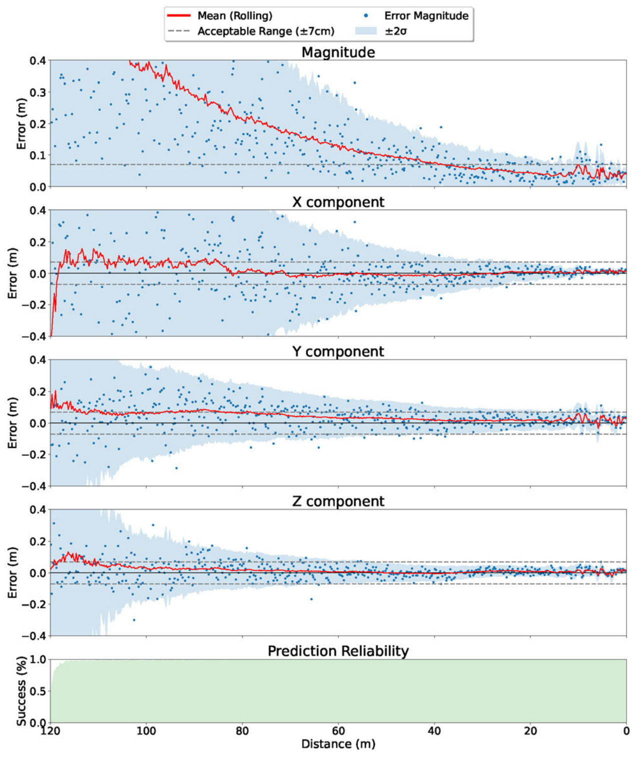


Fig. 26 Pipeline results from receiver wingcam trained with *randomly selected features* (model I)

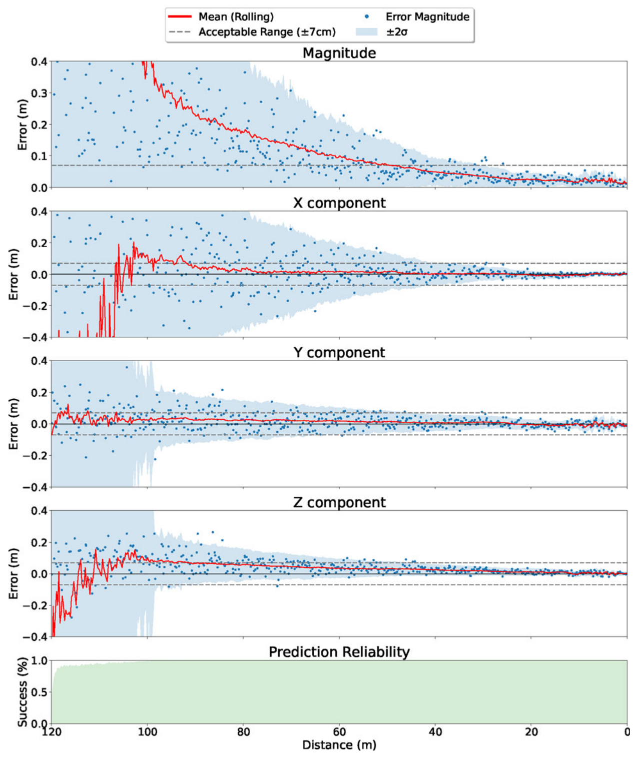


Fig. 28 Pipeline results from receiver wingcam trained with *medium features* (model K)

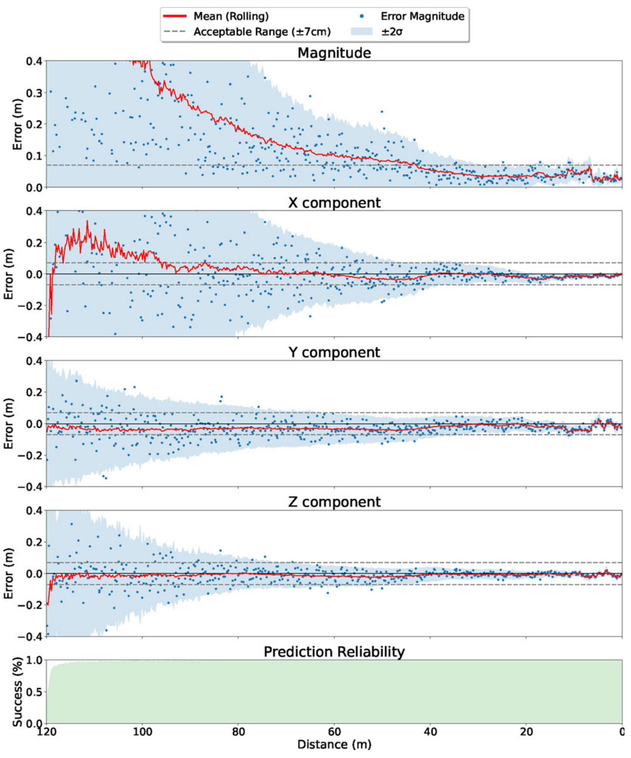


Fig. 29 Pipeline results from receiver wingcam trained with *large features* (model L)

Table 4 Pipeline execution time

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

phenomenon is evident in the middle left image of Fig. 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!

Quantitative analysis Figures 14 through 17 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

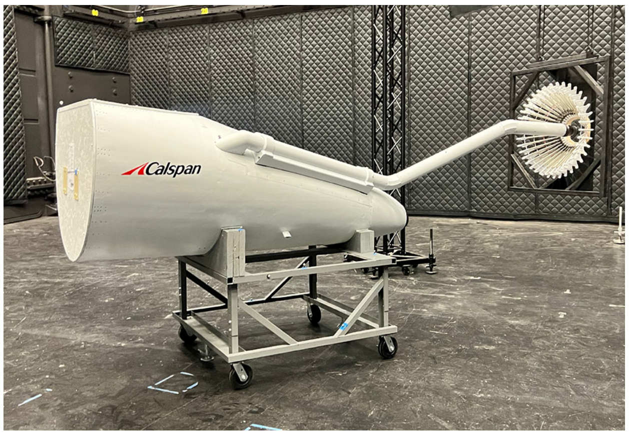


Fig. 30 Learjet nose cone and refueling drogue in a Vicon motion capture room

of the pipeline. When replacing perturbed truth points with actual YOLO predictions, Figs. 18 through 29 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 m), model A performed more consistently and with higher reliability leading up to contact from 20 m out. Also, model D performed well across the entire range from 120 m 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 m 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, Figs. 27 through 29 show medium size features produced the best results. The small features used to train model J were too small to detect past about 70 ms 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.

Pipeline speed Table 4 shows the execution time for each pipeline operation. In our experiments, image processing (rendering, padding, scaling, etc.) and making YOLO predictions occupied the majority of the pipeline execution envelope. However, processing the current image and performing YOLO's forward propagation on the previous image in parallel 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.

5 Conclusions

In this paper, we presented a 5-stage relative vectoring pipeline 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, we proposed an automated image labeling technique, which includes bounding box corrections for precise error free supervised machine learning. We also proposed a technique call dual object detection that transforms pose estimates of two objects observed within the same image into a relative vector between the two. We showed through Monte Carlo simulation 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 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 can enable precise and automated feature labeling, as described in this paper, but on augmented images with real aircraft parts—a technique we are already exploring, see Fig. 30.

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). We chose YOLOv5 in this effort 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.

6 Disclaimer

The views expressed are those of the author and do not reflect the official policy or position of the US Air Force, Department of Defense, or the US Government.

7 Supplementary information

This paper has an accompanied video which can be found at [55].

Acknowledgements We would like to thank our sponsors at the Naval Air Systems Command (USN/NAVAIR) and Aerospace Systems Directorate (USAF/AFRL/RQ). Your partnership and financial support made this research possible.

Funding This study was funded by the US Naval Air Systems Command and the Air Force Research Laboratory Aerospace Systems Directorate.

Data availability The datasets generated during and/or analyzed in this work are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anderson J, Miller J, Wu X et al (2021) Real-time automated aerial refueling with stereo vision: Overcoming gnss-denied environments in or near combat areas. Inside GNSS pp 32–41. https://insidegnss.com/real-time-automated-aerial-refueling-with-stereo-vision-overcoming-gnss-denied-environments-in-or-near-combat-areas/
2. Balamurugan G, Valarmathi J, Naidu V (2016) Survey on uav navigation in gps denied environments. In: 2016 International conference on signal processing, communication, power and embedded system (SCOPES), IEEE, pp 198–204
3. Campa G, Napolitano MR, Fravolini ML (2009) Simulation environment for machine vision based aerial refueling for uavs. IEEE Trans Aerosp Electron Syst 45(1):138–151
4. Chen CI, Koseluk R, Buchanan C et al (2015) Autonomous aerial refueling ground test demonstration-a sensor-in-the-loop, non-tracking method. Sensors 15(5):10948–10972
5. Chen S, Duan H, Deng Y et al (2017) Drogue pose estimation for unmanned aerial vehicle autonomous aerial refueling system based on infrared vision sensor. Opt Eng 56(12):124105
6. Cheng J, Liu P, Zhang Q et al (2021) Real-time and efficient 6-d pose estimation from a single rgb image. IEEE Trans Instrum Meas 70:1–14
7. Curro J, Raquet J, Pestak T et al (2012) 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), pp 774–782
8. Dede MA, Genc Y (2022) Object aspect classification and 6dof pose estimation. Image Vis Comput 124:104495
9. Du JY (2004) Vision based navigation system for autonomous proximity operations: an experimental and analytical study. Texas A &M University, TX
10. Duan H, Zhang Q (2015) Visual measurement in simulation environment for vision-based uav autonomous aerial refueling. IEEE Trans Instrum Meas 64(9):2468–2480
11. Duan H, Xin L, Chen S (2019) Robust cooperative target detection for a vision-based uavs autonomous aerial refueling platform via the contrast sensitivity mechanism of eagle's eye. IEEE Aerosp Electron Syst Mag 34(3):18–30
12. Erkin T, Abdo O, Sanli Y et al (2022) Vision-based autonomous aerial refueling. In: AIAA SCITECH 2022 Forum, p 1384
13. Fan Y, Huang J, Jia T et al (2021) A visual marker detection and position method for autonomous aerial refueling of uavs. In: 2021 IEEE international conference on unmanned systems (ICUS), IEEE, pp 1006–1011
14. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395
15. Fredriksen JC (2011) The United States air force: a chronology. ABC-CLIO
16. Garcia JAB, Younes AB (2021) Real-time navigation for drogue-type autonomous aerial refueling using vision-based deep learning detection. IEEE Trans Aerosp Electron Syst 57(4):2225–2246
17. Girshick R, Donahue J, Darrell T et al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
18. Guan Q, Li W, Xue S et al (2021) High-resolution representation object pose estimation from monocular images. In: 2021 China automation congress (CAC), IEEE, pp 980–984
19. He Y, Huang H, Fan H et al (2021) Ffb6d: a full flow bidirectional fusion network for 6d pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3003–3013
20. Hinchman J, Schreiter D (2007) Automated aerial refueling presentation to 2007 arsag conference (preprint). Tech. rep, Air Force Research Lab Wright-Patterson AFB OH Air Vehicle Directorate
21. Hong S, You T, Kwak S et al (2015) Online tracking by learning discriminative saliency map with convolutional neural network. In: International conference on machine learning, PMLR, pp 597–606
22. Huang WL, Hung CY, Lin IC (2021) Confidence-based 6d object pose estimation. IEEE Trans Multimed 24:3025–3035
23. Johnson K (2022) Airbus a330 mrtt certificated for automatic aerial refueling. Flying https://www.flyingmag.com/airbus-a330-mrtt-certificated-for-automatic-aerial-refueling
24. Kang J, Liu W, Tu W et al (2020) Yolo-6d+: single shot 6d pose estimation using privileged silhouette information. In: 2020 International conference on image processing and robotics (ICIP), IEEE, pp 1–6
25. Kehl W, Manhardt F, Tombari F et al (2017) Ssd-6d: making rgb-based 3d detection and 6d pose estimation great again. In: Proceedings of the IEEE international conference on computer vision, pp 1521–1529
26. Kimmett J, Valasek J, Junkins J (2002) Autonomous aerial refueling utilizing a vision based navigation system. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, p 4469
27. LaGrone S (2021) Mq-25a unmanned aerial tanker refuels super hornet in successful first test. USNI News https://news.usni.org/2021/06/07/mq-25a-unmanned-aerial-tanker-refuels-f-a-18-hornet-in-successful-first-test
28. Learn OpenGL (2015) Coordinate systems. https://learnopengl.com/Getting-started/Coordinate-Systems
29. Li C, Sun S, Song X et al (2022) Simultaneous multiple object detection and pose estimation using 3d model infusion with monocular vision. arXiv preprint arXiv:2211.11188
30. Lin TY, Maire M, Belongie S et al (2014) Microsoft coco: common objects in context. In: European conference on computer vision, Springer, pp 740–755
31. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
32. Lu CP, Hager GD, Mjolsness E (2000) Fast and globally convergent pose estimation from video images. IEEE Trans Pattern Anal Mach Intell 22(6):610–622
33. Lynch J (2022) Monocular pose estimation for automated aerial refueling via perspective-n-point. Master's thesis, Air Force Institute of Technology
34. Mammarella M, Campa G, Napolitano MR et al (2010) Comparison of point matching algorithms for the uav aerial refueling problem. Mach Vis Appl 21(3):241–251

35. Narasimhappa M, Mahindrakar AD, Guizilini VC et al (2019) Mems-based imu drift minimization: Sage husa adaptive robust kalman filtering. IEEE Sens J 20(1):250–260
36. Nykl S (2022) Aftrburner 3d visualization engine. http://www.nykl.net/aburn
37. Olsen EA, Park CW, How JP (1999) 3d formation flight using differential carrier-phase gps sensors. Navigation 46(1):35–48
38. OpenCV Team (2021) Open source computer vision library v4.5.5. https://opencv.org/opencv-4-5-5/
39. Parsons C, Nykl S (2016) Real-time automated aerial refueling using stereo vision. In: International symposium on visual computing, Springer, pp 605–615
40. Parsons C, Paulson Z, Nykl S et al (2019) Analysis of simulated imagery for real-time vision-based automated aerial refueling. J Aerosp Inf Syst 16(3):77–93
41. Rad M, Lepetit V (2017) 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, pp 3828–3836
42. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
43. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767
44. Redmon J, Divvala S, Girshick R et al (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
45. Sun S, Yin Y, Wang X et al (2018) Robust landmark detection and position measurement based on monocular vision for autonomous aerial refueling of uavs. IEEE Trans Cybern 49(12):4167–4179
46. Sundermeyer M, Marton ZC, Durner M et al (2018) Implicit 3d orientation learning for 6d object detection from rgb images. In: Proceedings of the European conference on computer vision (ECCV), pp 699–715
47. Tandale MD, Bowers R, Valasek J (2006) Trajectory tracking controller for vision-based probe and drogue autonomous aerial refueling. J Guid Control Dyn 29(4):846–857
48. Tekin B, Sinha SN, Fua P (2018) Real-time seamless single shot 6d object pose prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 292–301
49. Tomasi C (2015) A simple camera model. In: Notes from computer science 527, https://courses.cs.duke.edu//fall16/compsci527/notes/camera-model.pdf
50. Ultralytics (2022) Yolov5 in pytorch. https://github.com/ultralytics/yolov5
51. Wang XF, Dong XM, Kong XW et al (2014) Vision based measurement of refueling drogue for autonomous aerial refueling. Appl Mech Mater 590:618–622
52. Weiss K, Khoshgoftaar TM, Wang D (2016) A survey of transfer learning. J Big Data 3(1):1–40
53. Williamson W, Min J, Speyer J et al (2000) 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), IEEE, pp 2932–2938
54. Worth D, Lynch J, Nykl S (2022) Single-shot pose estimation for aar with yolo and perspective-n-point. In: Proceedings of the ION joint navigation conference
55. Worth D, Choate J, Lynch J et al (2023) Relative vectoring using dual object detection for autonomous aerial refueling. https://youtu.be/RXbrBl8Re7M
56. Xiang Y, Schmidt T, Narayanan V et al (2017) Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199
57. Xin L, Luo D, Li H (2018) A monocular visual measurement system for uav probe-and-drogue autonomous aerial refueling. Int J Intell Comput Cybern 11(2):166–180
58. Xufeng W, Xinmin D, Xingwei K (2013) Feature recognition and tracking of aircraft tanker and refueling drogue for uav aerial refueling. In: 2013 25th Chinese control and decision conference (CCDC), IEEE, pp 2057–2062
59. Yang Y, Liang KJ, Carin L (2020) Object detection as a positive-unlabeled problem. arXiv preprint arXiv:2002.04672
60. Yu J, Choi H (2021) Yolo mde: object detection with monocular depth estimation. Electronics 11(1):76
61. Zhang J, Liu Z, Gao Y et al (2020) Robust method for measuring the position and orientation of drogue based on stereo vision. IEEE Trans Industr Electron 68(5):4298–4308
62. Zhang Z (2000) A flexible new technique for camera calibration. IEEE Trans Pattern Anal Mach Intell 22(11):1330–1334
63. Zhao K, Sun Y, Li H et al (2022) A novel drogue pose estimation method for autonomous aerial refueling based on monocular vision sensor. IEEE Sens J 22(23):23064–23076
64. Zhao K, Sun Y, Li H et al (2022) Monocular visual pose estimation for flexible drogue by decoupling the deformation. IEEE Trans Instrum Meas 71:1–11
65. Zoph B, Cubuk ED, Ghiasi G et al (2020) Learning data augmentation strategies for object detection. In: European conference on computer vision, Springer, pp 566–583