Development of Three-Dimensional Feature-Based Vision Algorithm for Ship-Deck Landing

Victoria R. Britcher,* Inderjit Chopra,† and Anubhav Datta‡
University of Maryland, College Park, Maryland 20742

https://doi.org/10.2514/1.C038161

In this paper, we developed and validated a three-dimensional (3D) feature-based algorithm for tracking stochastic ship-deck motion at high sea states, specifically Sea-State 6 using data from the Navy SCONE data set. The new vision algorithm was developed from the structure-from-motion technique, which recovers the 3D structure of an object from a series of two-dimensional (2D) images, and was validated using a simulated 3D ship deck attached to a moving Stewart platform. Algorithm performance with different feature detectors and image resolutions was compared. In hand-held tests, the vision algorithm was demonstrated to accurately estimate the pose of a moving ship deck using a quadrotor. Visually degraded conditions were also evaluated; the algorithm was found robust to occlusion and low illumination, but performance reduced somewhat in severe glare. The vision algorithm was then validated in a simple free-flight test. All results were compared with Vicon ground-truth data. Additionally, as the 3D algorithm is computationally demanding, we developed and validated a method to improve the computational speed of the vision algorithm.

I. Introduction

LANDING a helicopter on a ship deck remains one of the most difficult and hazardous tasks for a human pilot. The goal of our research is to develop autonomous, vision-based ship-deck landing, which can significantly improve safety for pilots.

We previously created a two-dimensional (2D), feature-based vision algorithm to track ship-deck motion, which was able to function in visually degraded conditions [1]. However, laboratory testing demonstrated drawbacks to this approach. The algorithm requires a feature-rich, 2D image to be placed on the ship deck, which may not always be practical in some circumstances. From hand-held and free-flight data, we observed that, in visually degraded conditions, out-of-plane rotation of the landing pad image caused by pitch and roll motion can cause significant error in the estimated ship-deck pose [2].

In this paper, we have developed and investigated a three-dimensional (3D), feature-based vision algorithm to remedy these issues. This approach can work with arbitrary three-dimensional objects, eliminating the requirement for a special, detailed 2D landing pad image. The algorithm is also inherently able to detect the object from many different viewpoints, making it potentially robust to large rotations that could cause the 2D algorithm to fail [3].

Several prior studies have been conducted on using vision for rotorcraft landing on both stationary and moving platforms using simple visual tags or markings [4–8]. These approaches have also been explored for the problem of landing on ship decks, which present particular difficulty due to large, stochastic oscillating motions [9–11].

Our previous work explored the use of vision for ship-deck landing, given these challenges. We previously validated that it was feasible to track and autonomously land on a stochastically moving ship deck with vision alone. A fiducial-based algorithm was executed onboard a quadrotor to track and land on a stochastically moving platform [12]. Additionally, we developed a feature-based algorithm for tracking the ship deck, which works by matching features on a 2D reference image of the landing pad to camera images of the ship deck. This algorithm was able to work with generic landing pad patterns and could tolerate visually degraded conditions to an extent, such as occlusion or illumination differences, during bench-top tests [1]. However, its performance deteriorated in some other visually degraded conditions [13].

These prior studies have focused entirely on 2D computer vision methods. There now exists significant literature on 3D computer vision [14]. However, these concepts have not yet been applied to the problem of rotorcraft landing on a stochastically moving ship deck. This paper focuses on whether 3D computer vision approaches can mitigate the difficulties with the 2D feature-based algorithm. First, we describe our new 3D feature-based algorithm to estimate the pose of a moving ship deck in real time. Next, we discuss the results of controlled algorithm testing for ship-deck motion at high sea states and under visually degraded conditions; we additionally evaluate algorithm performance in free-flight hover. Finally, we evaluate a technique to improve the computational speed of the algorithm.

II. 3D Vision Algorithm

A new 3D feature-based vision algorithm has been developed to estimate the pose of a moving ship deck. This new approach can potentially address many of the deficiencies of 2D feature-based approaches. The algorithm incorporates multiple fundamental techniques used in 3D computer vision.

A. Feature Extraction and Matching

Features, for the purposes of this work, are small patches of interest, such as regions of high contrast or rich texture in an image, which are robust to transformations in scale, translation, illumination, and in- and out-of-plane rotation. Locations of good features are typically found by detecting strong corners or blobs (locations with significant change in pixel intensity in two directions). Image patches at these locations are extracted and converted to feature descriptors, which are a numerical vector representation of the image patch.

Features can be matched by comparing these extracted feature descriptors, which can be used to find corresponding points across different images. This technique was previously used in the 2D vision algorithm [1].

Received 30 July 2024; accepted for publication 24 March 2025; published online 6 June 2025.

*Graduate Research Assistant, Aerospace Engineering, Alfred Gessow Rotorcraft Center; vbritche@umd.edu. Student Member AIAA (Corresponding Author).

†Distinguished University Professor, Aerospace Engineering, Alfred Gessow Rotorcraft Center. Fellow AIAA.

‡Professor, Aerospace Engineering, Alfred Gessow Rotorcraft Center; datta@umd.edu. Associate Fellow AIAA.

B. Feature Detectors

Multiple algorithms have been developed to detect and extract image features. For this work, we used implementations of these algorithms provided in the OpenCV library.

The scale invariant feature transform (SIFT) algorithm, developed by Lowe [15], is a commonly used feature detector. It generates vectors of length 128 as feature descriptors. Features can be compared by calculating the L2-norm (Euclidean distance) between two descriptors. The algorithm is invariant to changes in scale and translation and is partially robust to changes in in- and out-of-plane rotation and illumination. While the algorithm provides generally good feature matching results, it is more computationally intensive than many other detectors [16].

The Oriented Features from accelerated segment test (FAST) and rotated Binary robust independent elementary features (BRIEF) (ORB) algorithm [17] is another common feature detector. The algorithm uses BRIEF binary descriptors; each of the 256 elements has the value 1 or 0. These features can be compared with the Hamming distance, which describes how many elements differ between two descriptors. Although it is significantly faster computationally than SIFT, it has lower robustness [16].

C. 3D Triangulation

Using two or more images, it is possible to determine the location of a point in 3D space. This process, triangulation, is illustrated in Fig. 1. A point in 3D space is projected onto two 2D camera images. Using the 2D image points corresponding to this point, it is possible to recover the point's location in three dimensions. Feature matching is typically used to find 2D image points corresponding to the same point in three dimensions [14]. The images can be obtained using a stereo (two-camera) setup or by using a monocular (single) camera to obtain two images at different locations.

The triangulation process requires the poses, or positions and orientations, of the camera(s) corresponding to each image. The camera poses can be measured directly or recovered using a visual reference of known size and pose, such as a fiducial marker. Alternatively, the relative camera poses can be estimated using multiple corresponding or matched 2D image points. The disadvantage of this approach is that it is ambiguous in scale; an additional scale reference must be used to recover this information.

Although only two images are necessary for triangulation, additional images can be used to recover additional 3D points for which correspondences could not be found in the previous images. If point correspondences are found across more than two images, the accuracy of the recovered 3D points can be improved further through numerical optimization techniques. This is described in more detail in the following section.

D. Ship-Deck Detection Algorithm

A new ship-deck detection algorithm was developed as part of this research program. The algorithm is separated into two stages: 1) extraction of features and their 3D positions from a set of images of a target object and 2) estimating the pose of the object using this extracted data.

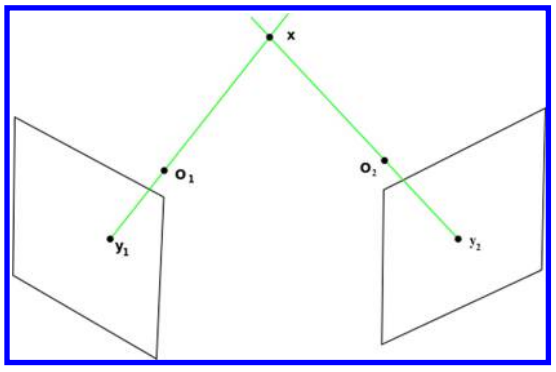


Fig. 1 Triangulation of 3D points using two images.

The feature extraction stage is developed from the structure-from-motion technique, which is used to recover the 3D structure of an object from a sequence of 2D images of it [14]. In this application, the 3D structure consists of a 3D point cloud (a collection of points in 3D space), and each point is associated with the 3D position of an extracted feature. An overview of this process is given as follows:

1) Obtain a sequence of images of the target object along with initial estimated camera poses. These images can be collected from one or more cameras and should include a sequence of varying object orientations relative to the camera(s), covering a range of object poses similar to those that may be observed during the pose estimation stage. The estimated camera poses can be obtained through physical measurement of the camera position or by use of a known visual reference, such as an attached fiducial marker.

2) Extract and match 2D features in the first two images to find corresponding points. Then, triangulate these points using the corresponding camera poses to estimate the 3D positions of these points. Store the matched features and their 3D positions. Implementations from OpenCV, an open-source computer vision library, are used for feature extraction and matching.

3) Extract features from each subsequent image. First, check for any matches with previously stored features. This is used to track features across multiple images, which are used later to refine the 3D point positions. Next, find any new feature matches between the current and previous image, and repeat the tasks in Step 2.

4) Repeat this process for all images. The end result will be a list of extracted features, triangulated 3D point positions, and 2D positions of these features in each image.

5) Refine the estimated 3D point positions and camera poses with numerical optimization, also known as bundle adjustment. For each image, the estimated 3D points are projected back to 2D and are then compared to the actual 2D feature positions in the image. The numerical optimization minimizes the error between the projected and actual 2D points, resulting in more accurate 3D point positions.

6) Remove any 3D points, and their corresponding features, that are not located on the object itself. These points may be features located in the background or may be outliers resulting from incorrect matching. In this paper, these points are removed by generating a 3D bounding box (an imaginary 3D box which fully and tightly encloses the object geometry). Points within the box are assumed to fall on the object and are retained, while points that fall outside it are removed. This procedure removes background features and incorrect 3D points, preventing unwanted matches to these features during pose estimation.

7) Finally, save the refined 3D points and associated 2D feature descriptors.

The next stage is used for actual pose estimation. This is the portion that will be run onboard a rotorcraft during landing. This stage uses a monocular (single) camera, as 2D features from a single image are matched to stored 3D features. The process is described in detail as follows:

1) Load the saved features and their corresponding 3D positions.

2) Obtain a new 2D image of the target object (such as a frame from an onboard camera). Extract features from this image, and find matches with the features loaded in the previous step. This provides a list of 2D points in the image and their corresponding points in the stored 3D point cloud.

3) Solve the perspective-*n*-point (P*n*P) problem, which estimates the pose of the target object from corresponding 3D and 2D points. An OpenCV implementation of P*n*P, employing the random sample consensus (RANSAC) method, is used for this step. RANSAC iterates through many random samples of the corresponding points, finding pose estimates for each. The estimate with the largest number of inliers, or points consistent with the pose estimate, is selected as the final result. This makes the result more robust to outliers (poor matches).

4) The final output will be the pose (position and orientation) of the object in the 2D image. This process is repeated independently for any new images inputted.

5) If additional filtering of the raw data is necessary, the results are fed into a Kalman filter to mitigate any noise or small errors in the pose estimation results.

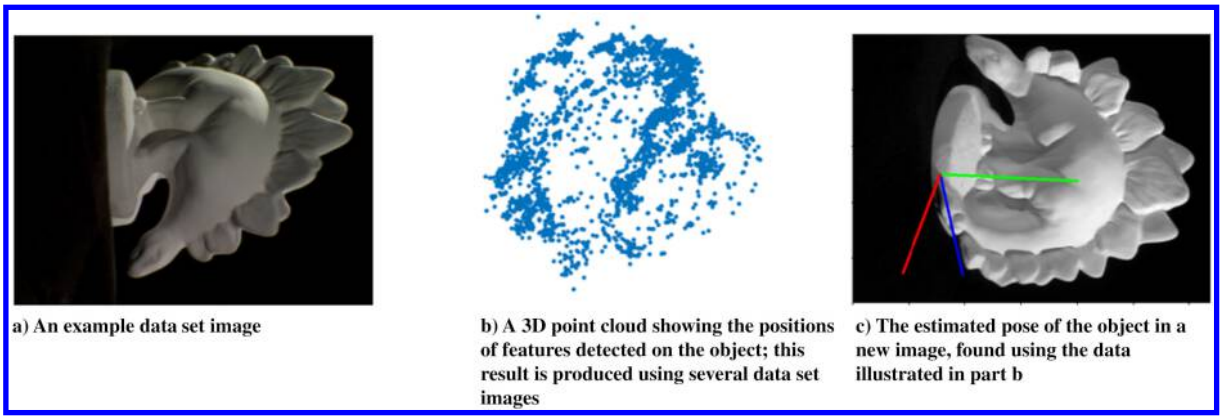


Fig. 2 Images illustrating the steps in the algorithm.

The features in a camera frame will be extracted and matched to the saved features from the first step. The matches between the 3D object points and the 2D points in the camera frame are used to recover the pose of the 3D object (the ship deck). Because this process extracts features from many viewpoints of the object, instead of a single 2D image, the pose estimation is expected to be far more robust to rotation of the ship deck.

E. Initial Algorithm Validation

The algorithm was initially validated using a publicly available data set, consisting of photos of an object taken from many viewpoints [18]. This data set is useful for basic validation because it contains both predetermined camera poses and accurate ground truth data. We inputted a subset of the images in the data set, consisting of one full revolution about the object by the camera, into the feature extraction stage. Figure 2a provides an example of a data set image used for feature extraction. A total of 962 features and their corresponding 3D positions were successfully extracted, as shown in Fig. 2b.

For pose estimation, another subset of images was used, consisting of viewpoints not used to extract features. These images form a sequence in which the camera again revolves around the object, though with the camera rotated 15 deg from the original images to ensure these viewpoints are not the same as those used earlier to extract features. The sequence covers a 315 deg view around the object. One of these estimated object poses is illustrated in Fig. 2c. The estimated poses for the sequence were then compared with ground truth, shown in Fig. 3. Note that images in the sequence are numerically labeled in order, starting from image 0 and ending with image 40. For each image, there is a corresponding object pose. The plot shows that these poses can be recovered precisely.

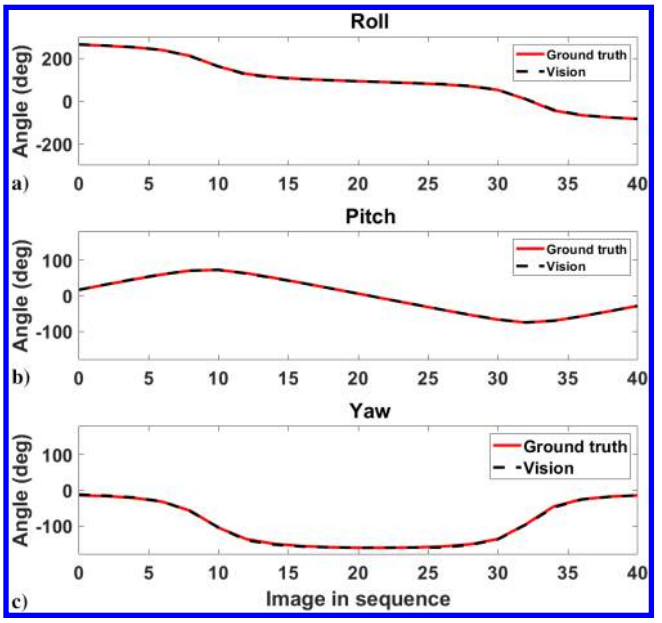


Fig. 3 Estimated object pose compared to ground truth across a sequence of new images.

III. Vision System Integration

The vision algorithm is integrated into a quadrotor unmanned aerial vehicle (UAV) to demonstrate the feasibility of vision-based ship-deck landing. Because external communication may be unreliable during real ship-deck landings, the vision algorithm must run using only onboard hardware. This hardware is described in the following for completeness; this is the same as was used for testing the 2D vision algorithm [2].

A. Camera for Vision

A See3CAM USB camera, manufactured by e-con Systems, was used to obtain images of the ship deck. It features a built-in autoexposure algorithm, which automatically varies the camera shutter speed for varying light intensity received by the camera. This allows the camera to compensate for a wide range of scene illumination, though at the cost of increased motion blur for low illumination levels. On the quadrotor, the camera is mounted on the underside, facing downward.

B. Flight Computer

An Up Xtreme single-board computer, featuring an Intel i5-8365UE processor, was used for the real-time pose estimation portion of the algorithm. This computer was previously used for our 2D feature-based algorithm [2]. This computer is mounted to the quadrotor to run the algorithm onboard.

As the feature extraction stage is computationally demanding, this portion was performed on an external desktop computer. The saved features and 3D points were then copied onto the flight computer for use in pose estimation. This also replicates the realistic operational scenario.

C. Code Integration

The pose estimation algorithm was implemented into the onboard flight computer. The code uses Robot Operating System 2 (ROS 2), a set of robotics libraries, for asynchronous communication between individual processes, or nodes. The pose estimation algorithm and logging runs on a single node. An additional node is dedicated to continuously retrieving and publishing frames from the camera, which are then received as needed by the vision algorithm.

D. Quadrotor

The camera and flight computer were integrated into a quadrotor UAV for testing, shown in Fig. 4. The quadrotor, which has a gross takeoff weight of 2.75 kg, was developed and fabricated in house with a combination of off-the-shelf and custom components. It has a frame size of 450 mm × 450 mm (measured from the centers of the

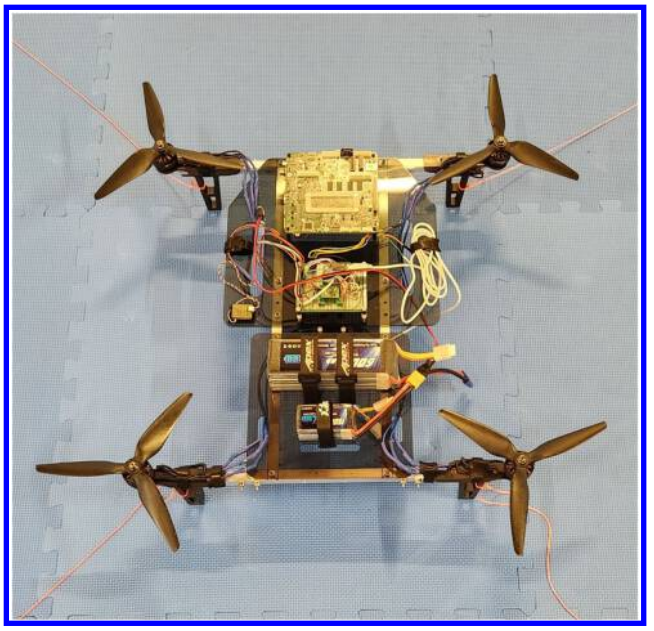


Fig. 4 Quadrotor for testing the vision algorithm.

motors) and a 9 in. (0.229 m) rotor diameter. This quadrotor was previously used to validate the 2D feature-based algorithm [2].

IV. Algorithm Validation with Simulated Ship Deck

The performance of the new algorithm must be evaluated with real-world objects before it can be incorporated into an aircraft for ship-deck landing.

A. Ship Deck

The vision algorithm was validated with a 3D object simulating a ship deck. The deck measures $266.7 \times 266.7 \times 104.8$ mm ($10.5 \times 10.5 \times 4.125$ in.) and incorporates several simple 3D shapes. ArUco markers of varying sizes have been affixed to the surfaces of the ship deck to provide many good features for detection. ArUco markers were specifically chosen to provide some consistency with the ArUco marker grid used to validate the 2D feature-based algorithm [2].

To extract features and 3D points from the object, we created a set of images of the object in a controlled manner. A camera was placed at three different fixed positions, and the object was rotated on a base in 10 deg increments to collect images around the entire object. With this process, a total of 111 photographs were collected from different camera poses. Example images from each camera position are provided in Fig. 5. For improved image quality, these photographs were taken using an illuminated photography tent with a monochrome background. This reduces illumination differences and shadows and minimizes unwanted detections of background features. One of the ArUco fiducial markers on the object, which was fully visible in all views, was used to obtain initial estimates of the camera poses relative to the ship deck during feature extraction. OpenCV functions were used to detect the fiducial marker in each image and compute the relative pose of the camera.

B. Algorithm Testing

The performance of the algorithm was evaluated using the simulated ship deck in various scenarios. For all tests, the algorithm was run in real time using the previously described flight computer and camera. Unless otherwise specified, the camera was mounted to the underside of the quadrotor UAV, with the vehicle held and moved by hand to allow large angles and for more controlled results.

An important consideration is that the vision algorithm outputs the pose of the ship deck relative to the camera. Therefore, all results presented here are relative poses. Note that these poses are affected by both the motion of the camera and the motion of the ship deck.

For all validation tests, ground truth data were provided through a Vicon motion-capture system with eight Vicon cameras. The setup was calibrated before each experimental session, with a final measurement error of less than 0.5 mm reported by Vicon software. The absolute poses of the object and camera recorded by the Vicon were converted to relative poses, in order to be consistent with the vision algorithm output. The raw algorithm output, rather than the Kalman filtered data, is presented to more accurately evaluate algorithm performance.

With the 3D vision algorithm, a time delay of approximately 500 ms was observed in the time stamps of logged pose estimates compared to ground truth. This delay is due to a combination of computational time and latency in receiving camera frames. To allow for better visual comparison in plots, the delay is corrected for in all real-time results presented in the paper. The root-mean-square error (RMSE) was used to quantitatively evaluate error in real-time test results after incorporating the time delay correction. Figure 6 provides an example of vision algorithm output with the time delay present.

V. Comparison of Feature Detectors

Different feature detectors present critical tradeoffs for algorithm performance. These include the number of extracted features, matching accuracy, robustness to scale and rotation, and computational cost [16]. Therefore, it is necessary to quantitatively compare the performance of the algorithm using different feature detectors in controlled conditions. For this test, the feature detectors SIFT and ORB were compared.

With the OpenCV implementation of ORB, it is also possible to specify the maximum number of features retained from each image (nFeatures). After features are extracted from an image, they are ranked according to their Harris score. This score is a measure of intensity change in all directions across the image patch. It will be

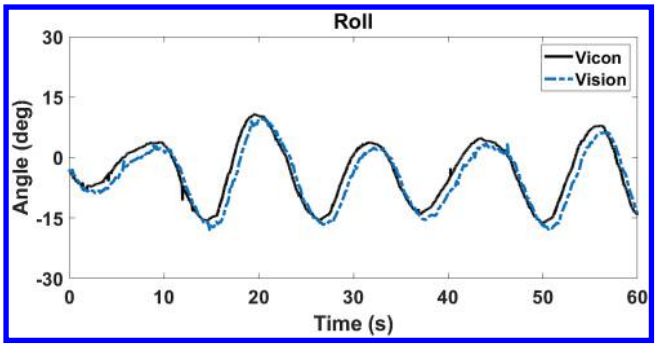


Fig. 6 Example of vision algorithm result for roll motion with uncorrected 500 ms delay.

Fig. 5 Example photographs of the ship deck used for feature and 3D point extraction, collected at each of the three camera positions.

higher for strong corners, which constitute good features, and lower for more edgelike patches or flat patches with little variation [19]. The highest-scoring features, up to a maximum of nFeatures, will be retained. A smaller value of nFeatures will result in fewer features extracted and therefore fewer possible feature matches but will increase computational speed.

A. Feature Extraction Results

Features and their corresponding 3D points were extracted from the ship-deck images using both SIFT and ORB. For ORB, this was performed with nFeatures = 1000 and nFeatures = 500. The resulting 3D point clouds are shown in Fig. 7. ORB with nFeatures = 500 and nFeatures = 1000 results in 5131 and 10,602 unique features, respectively, while SIFT extracted 11317 unique features.

B. Algorithm Performance Comparison

The object simulating the ship deck was placed horizontally on the floor in front of the camera, as seen in Fig. 8. For image collection, the camera was mounted to a miniature tripod, allowing the camera position and pitch to be adjusted manually. The camera was set to three different linear positions and pitch angles, ensuring that the deck is fully visible in the image for each position. Three photos were collected at each camera position to account for random image noise and error. For each test case, results were calculated for each image and averaged for each camera position.

The resolution of the camera image is also an important consideration for the vision algorithm. Extracting features from a lower-resolution image is less computationally expensive but can result in failure to detect smaller features in the image, especially at larger distances from the object [1]. The algorithm was therefore tested on images at the original camera resolution (1920×1080) as well as images scaled down by 50% (960×540).

The mean computational times required for the algorithm, averaged across all evaluated camera locations, are given in Table 1. Note that this is the time for an entire pose estimation update, which includes the total time to 1) extract features from the image of the ship deck, 2) match features between the camera image and stored data, and 3) estimate the pose of the deck.

For full-resolution images, SIFT is unacceptably slow, taking a mean of 0.569 s to process a single image. Reducing the resolution reduces this time dramatically to 0.164 s, though this is still longer than the ORB detector. In contrast, for ORB, the mean computational time is not meaningfully reduced by decreased image resolution. Varying nFeatures does show a significant impact, as the computational time needed for nFeatures = 1000 is almost double the time for nFeatures = 500.

The mean errors in pitch and position estimate, again averaged across all three camera locations, are given in Tables 2 and 3, respectively. For 100% resolution images, SIFT demonstrates the highest accuracy, with an error of 0.21 deg in pitch estimation. For ORB, an error of approximately 1 deg is observed for both values of nFeatures. For 50% resolution images, SIFT displays very similar results to the 100% resolution case. However, the performance of ORB degrades with reduced resolution, with errors for both estimated pitch and position increasing significantly. The error is especially severe for estimated pitch with nFeatures = 1000.

Figure 9 shows the variation of error with distance from the camera to the object for the 50% resolution images. With SIFT, the pose estimation error is similar for all distances tested. ORB shows similar error to SIFT when the camera is closer to the object. However, the error increases severely when the camera moves farther from the object, indicating ORB fails at longer distances. In particular, ORB with nFeatures = 1000 displays a dramatic increase in error. Inspection of the matched features on the image indicates that many of the

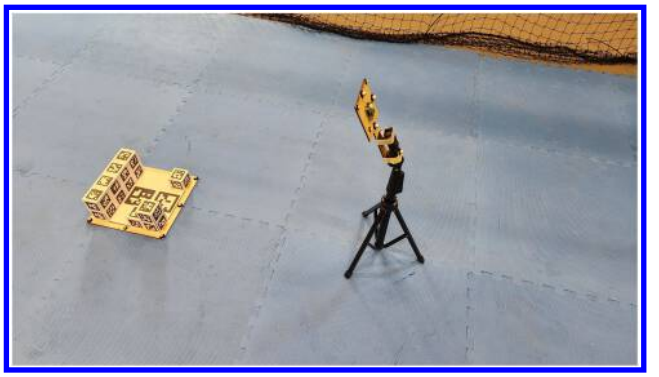


Fig. 8 Setup used for collecting images of the ship deck.

Table 1 Mean computational time for different feature detectors and image resolutions

Feature detector	100% resolution, s	50% resolution, s
SIFT	0.569	0.164
ORB (nFeatures = 1000)	0.156	0.156
ORB (nFeatures = 500)	0.080	0.078

Table 2 Mean pitch estimation error for different feature detectors and image resolutions

Feature detector	100% resolution, deg	50% resolution, deg
SIFT	0.21	0.23
ORB (nFeatures = 1000)	1.00	28.60
ORB (nFeatures = 500)	0.97	1.99

Table 3 Mean position estimation error for different feature detectors and image resolutions

Feature detector	100% resolution, cm	50% resolution, cm
SIFT	6.43	6.44
ORB (nFeatures = 1000)	5.44	14.94
ORB (nFeatures = 500)	5.41	8.85

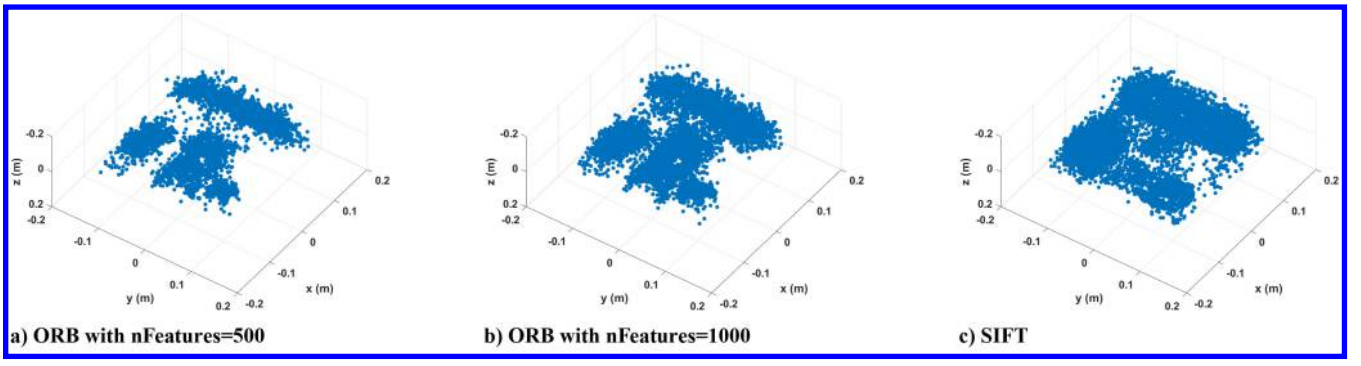


Fig. 7 Feature extraction results with different feature detectors.

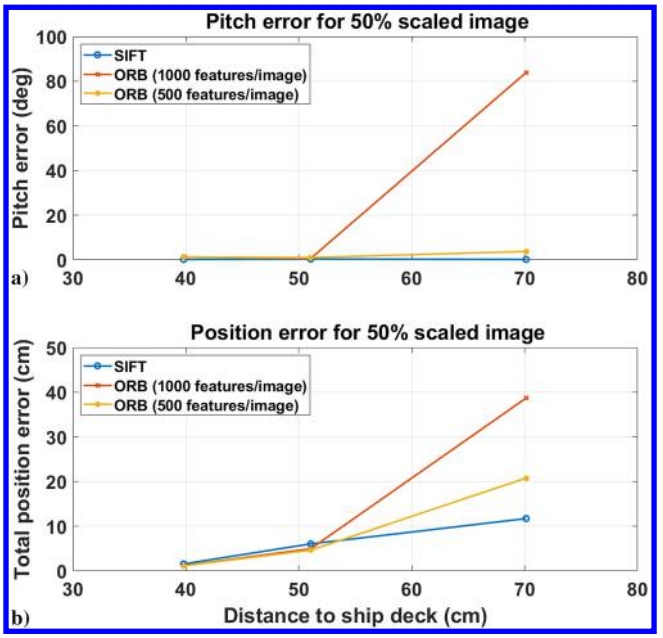


Fig. 9 Pose estimation error at varying distance from the object for 50% resolution images.

features on the 3D object were incorrectly matched to background details in the image. With the value of nFeatures = 1000, additional, lower-quality features are extracted compared to nFeatures = 500. This results in a higher number of erroneous matches. This failure is not acceptable for simulated landing because for the experimental algorithm tests the camera is located a distance of 50–90 cm from the ship-deck origin.

From these results, ORB with nFeatures = 500 and 100% resolution camera frames was selected for testing real-time pose estimation. This combination was chosen as it provides both relatively low computational time and acceptable performance.

An important note is that in this scenario pose estimation was completed on single, precollected images, rather than updating and logging continuously alongside other running scripts. Because of this increased computational load, the update frequencies are expected to be lower during real-time operation.

VI. Inlier Thresholding

Estimating a pose with P*n*P requires a minimum of four point correspondences. However, with a small number of corresponding points, errors in matching can result in an incorrect pose estimate. RANSAC iterates to find an estimated pose with the highest number of inliers, or point correspondences which are consistent with the estimated pose. The number of inliers can be used to evaluate the quality of a pose estimate; a small number of inliers may mean a poor result. The quality of pose estimation could potentially be improved by implementing a minimum threshold for the number of inliers, rejecting any estimates that fall below this threshold.

A vision algorithm test was completed to determine a good threshold for inliers. Algorithm output was collected with the quadrotor held by hand above the 3D ship deck. The quadrotor was moved in quick, random motions in attitude and position, with the ship deck either fully or partially in view of the camera, so as to generate pose estimates with varying quantity of inliers. For each pose estimate, the quantity of inliers was logged.

The attitude and position results are shown, respectively, in Figs. 10 and 11. The raw pose estimates (no thresholds applied) are plotted alongside data, which are filtered to remove pose estimates falling below the minimum inlier threshold. It was found that a threshold of 50 inliers yielded the best improvement in these results.

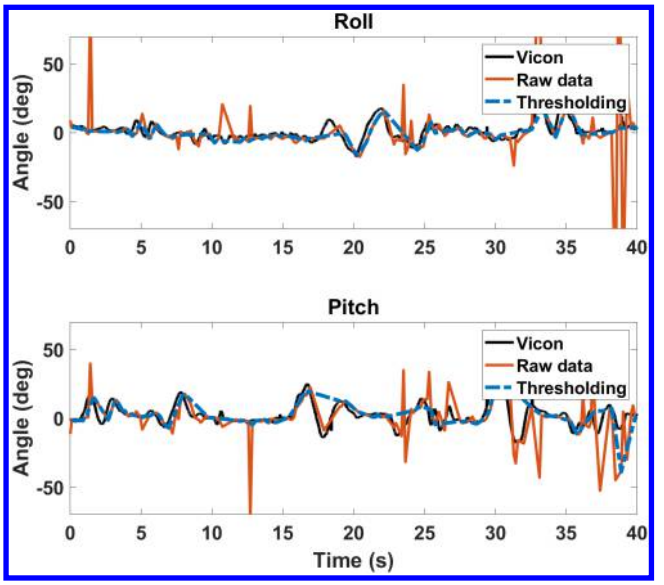


Fig. 10 Estimation of relative attitude angles. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.

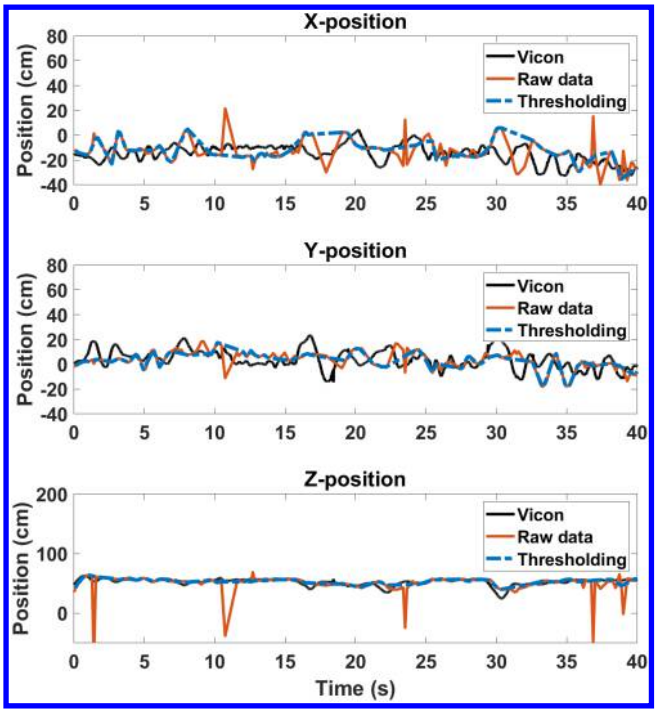


Fig. 11 Estimation of relative position. Raw algorithm output is plotted alongside results with a minimum inlier threshold applied.

VII. Stationary Camera and Landing Pad

A test was completed to show a baseline of performance. The quadrotor, with camera attached, was held stationary above the stationary 3D object, with the object fully in view of the camera.

The attitude and position results for this test are shown in Figs. 12 and 13, respectively. These plots show very accurate estimation of the ship-deck pose. The RMSE for pitch and roll are less than 2 deg, while the RMSE for linear position is less than 1.3 cm on any axis.

VIII. Ship-Deck Motion

The performance of the algorithm was then evaluated for a landing pad undergoing stochastic ship-deck motion. The ship-deck motion, representing Sea-state 6, was obtained from time histories provided in the Navy SCONE data set, representing the motions of a

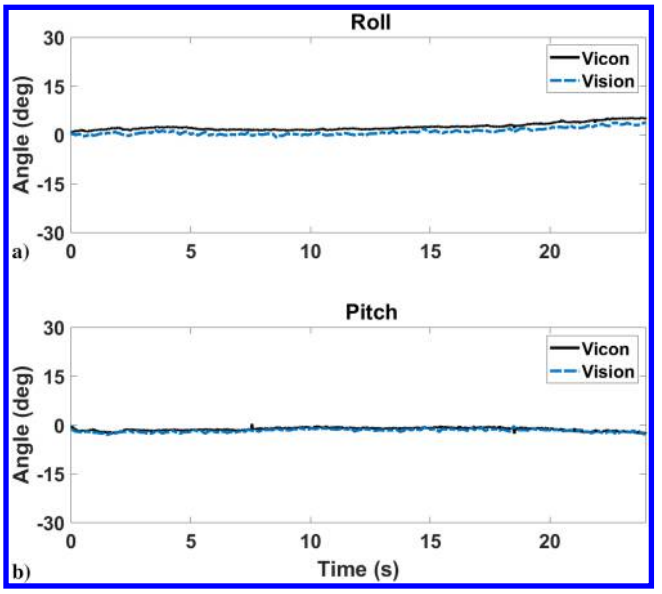


Fig. 12 Estimation of relative attitude angles, with the quadrotor held stationary above the ship deck.

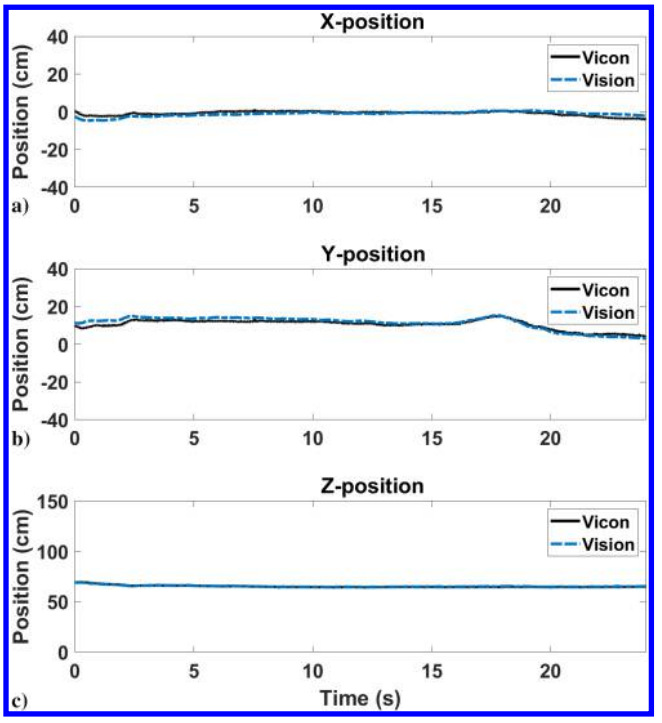


Fig. 13 Estimation of relative position, with the quadrotor held stationary above the ship deck.

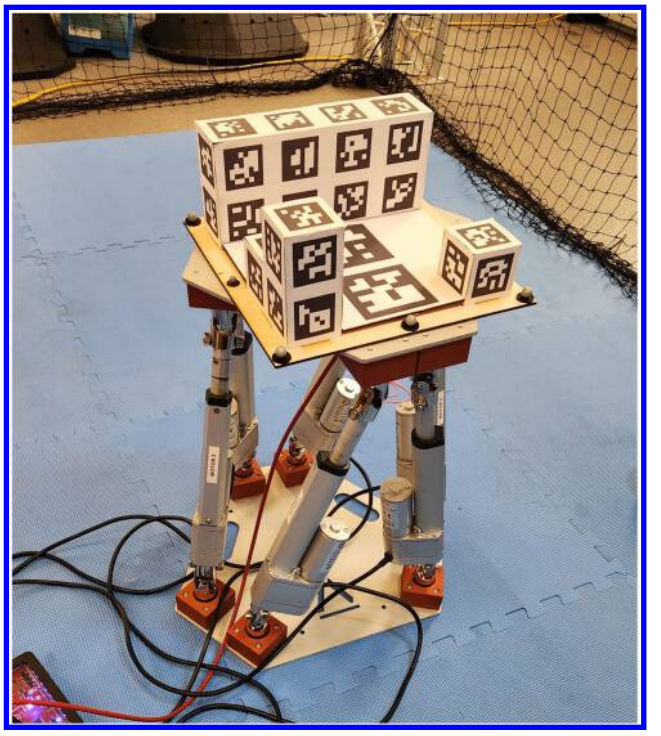


Fig. 14 Stewart platform with 3D object attached. The Stewart platform is used to simulate ship-deck motion.

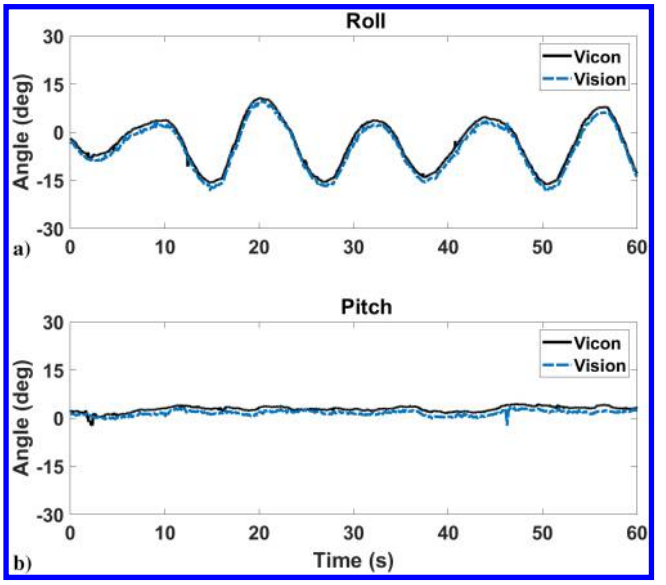


Fig. 15 Estimation of relative attitude angles, with the quadrotor held stationary above the ship deck attached to moving Stewart platform.

full-scale DDG-51 type ship [20]. The roll, pitch, and yaw motions are generated exactly from the time history without scaling, while the translational (surge, sway, and heave) motions are scaled to 0.5% of their original values due to the smaller scale of the experimental setup. A Stewart platform with the simulated ship deck attached, as shown in Fig. 14, was used to generate this ship-deck motion for vision algorithm testing. The vision algorithm is used to estimate the ship-deck pose in real time.

First, the quadrotor was held stationary above the moving ship deck, simulating a stable hover condition. The attitude and position results are provided in Figs. 15 and 16, respectively. The results indicate that the algorithm can estimate the deck pose accurately in this case, as the pose estimates closely follow the ground truth.

Next, the quadrotor was moved randomly with varying attitude and position. This creates large, complex relative motions between the camera and ship deck which must be captured by the vision algorithm. The results of the test are shown in Figs. 17 and 18. The attitude results closely follow the ground truth; the algorithm is able to accurately capture complex pitch and roll motions in excess of 20 deg. Additionally, the estimate of Z position closely matches ground truth. However, some errors are present in the estimates of X and Y position, as the vision algorithm estimates do not capture all of the rapid linear motions present in the ground truth.

The algorithm performance in both cases is quantitatively compared using the RMSE of position and attitude, provided in Table 4. With the quadrotor stationary, the attitude results show similar error to the baseline case, though the X- and Y-position errors are increased. With significant relative motion of the quadrotor and platform, the X- and Y-position errors are significantly increased, while the attitude and Z-position errors are modestly increased. The

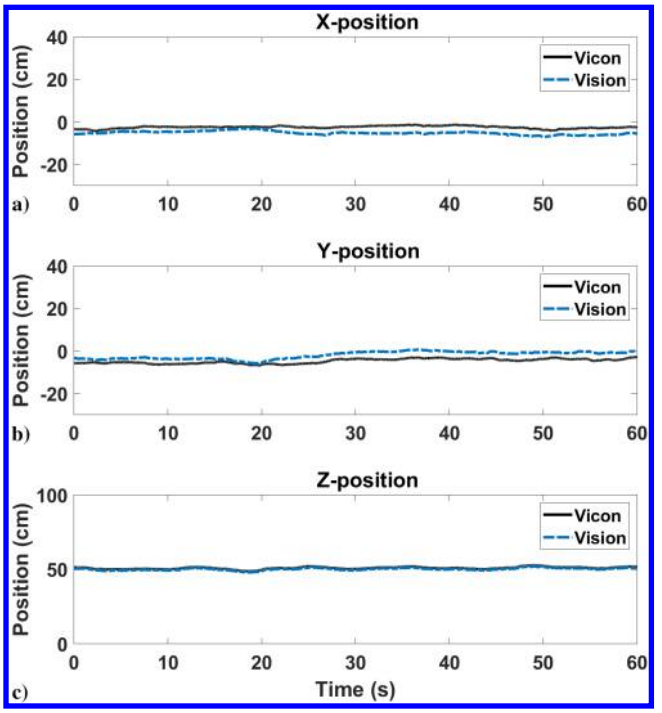


Fig. 16 Estimation of relative position, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

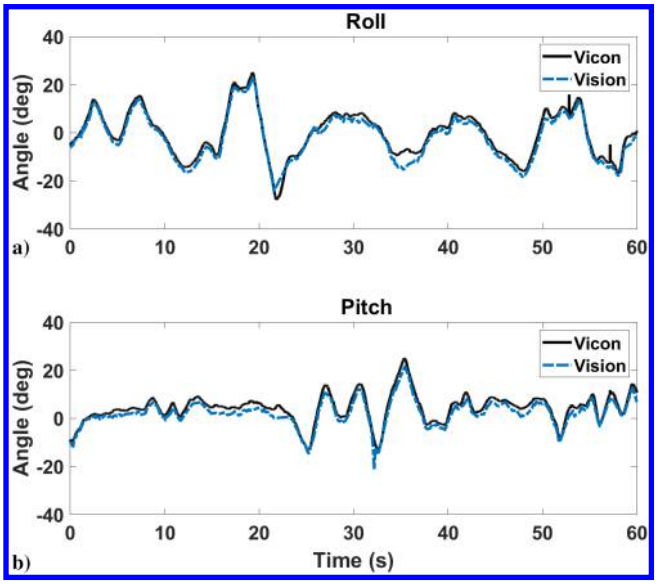


Fig. 17 Estimation of relative attitude angles, with the quadrotor moved in attitude and position above the ship deck attached to the moving Stewart platform.

movement of the quadrotor introduces motion blur, which contributes to these errors. Additionally, the 3D object is only partially visible in frame during portions of the test, resulting in fewer feature matches and therefore increased error.

IX. Visually Degraded Conditions

The algorithm's performance was then evaluated in visually degraded conditions. Three conditions were specifically evaluated: 1) occlusion, 2) low illumination, and 3) glare. For all tests, the ship deck was mounted to the Stewart platform simulating Sea-state 6 motions, with the quadrotor held stationary above the moving platform.

The vision algorithm was first evaluated under occlusion. The visible portions of the 3D object were partially occluded using paper

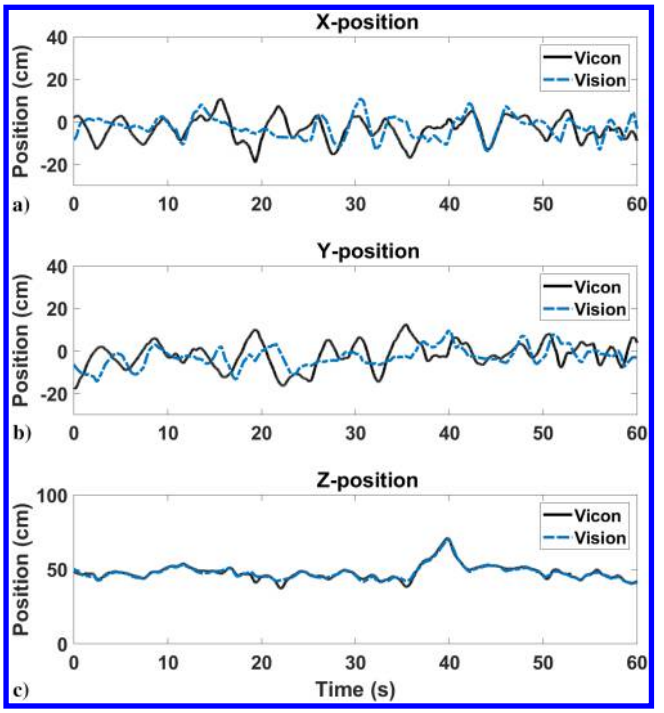


Fig. 18 Estimation of relative position, with the quadrotor moved in attitude and position above the ship deck attached to the moving Stewart platform.

Table 4 RMSE for ship-deck motion cases

Test case	X, cm	Y, cm	Z, cm	Roll, deg	Pitch, deg
Quadrotor stationary	2.9	3.0	0.8	1.5	1.4
Quadrotor moving	5.7	6.3	1.1	2.2	2.4

coverings, as shown in Fig. 19. The results for attitude and position are shown in Figs. 20 and 21, respectively. The pose estimates in this case are accurate overall, closely matching the Vicon ground truth, with only minor noise in the pitch and roll estimates.

The algorithm performance was then evaluated for low illumination conditions. Low indoor lighting was used to create an illumination of approximately 10 lx. The results of the test are shown in

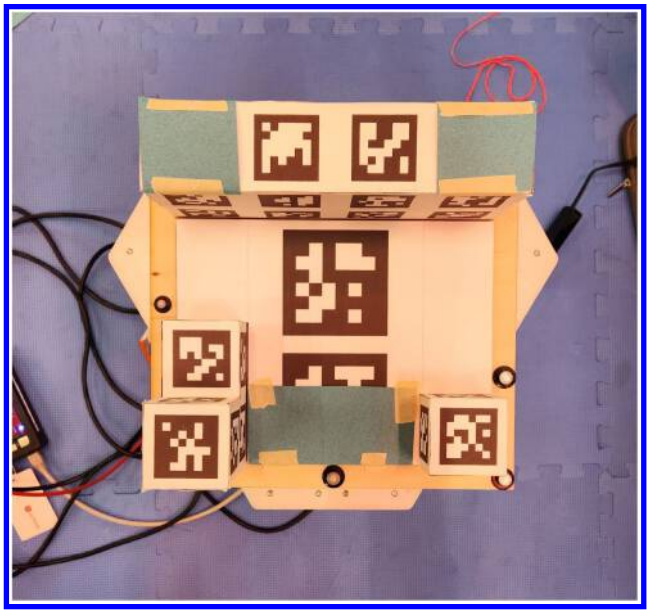


Fig. 19 The 3D object simulating the ship deck with added occlusions.

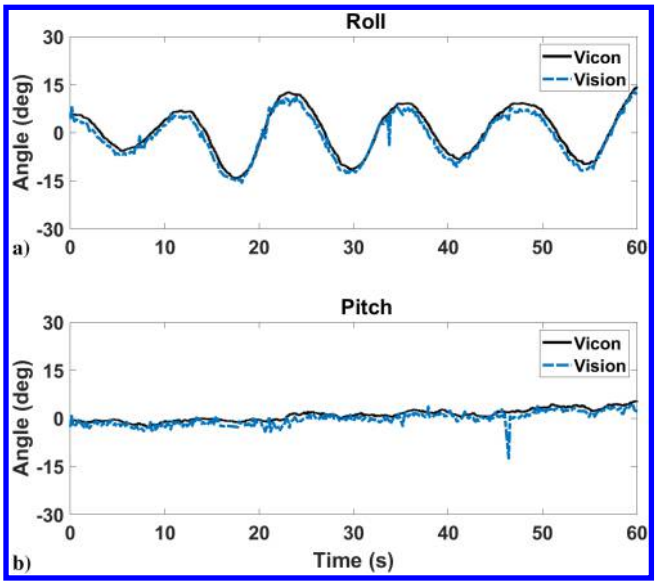


Fig. 20 Estimation of relative attitude angles of a partially occluded ship deck, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

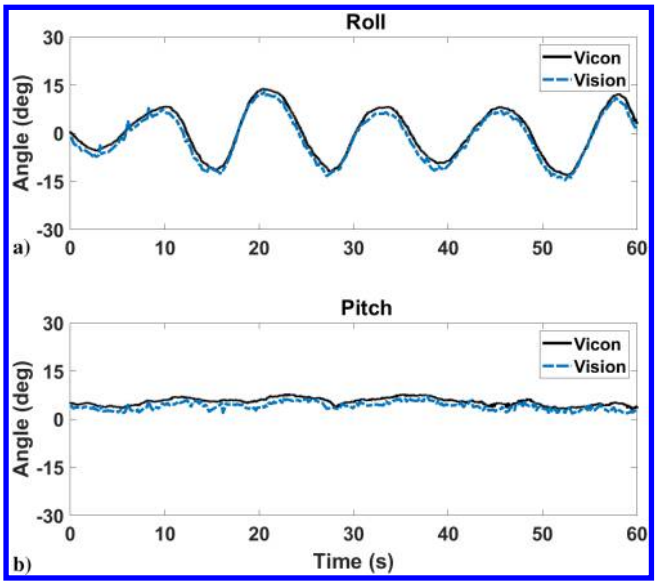


Fig. 22 Estimation of relative attitude angles of a ship deck in low illumination, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

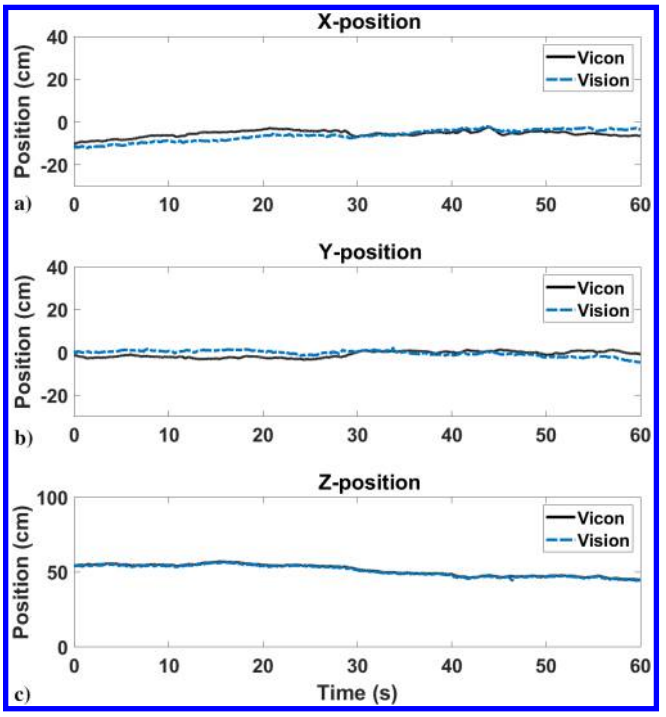


Fig. 21 Estimation of relative position of a partially occluded ship deck, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

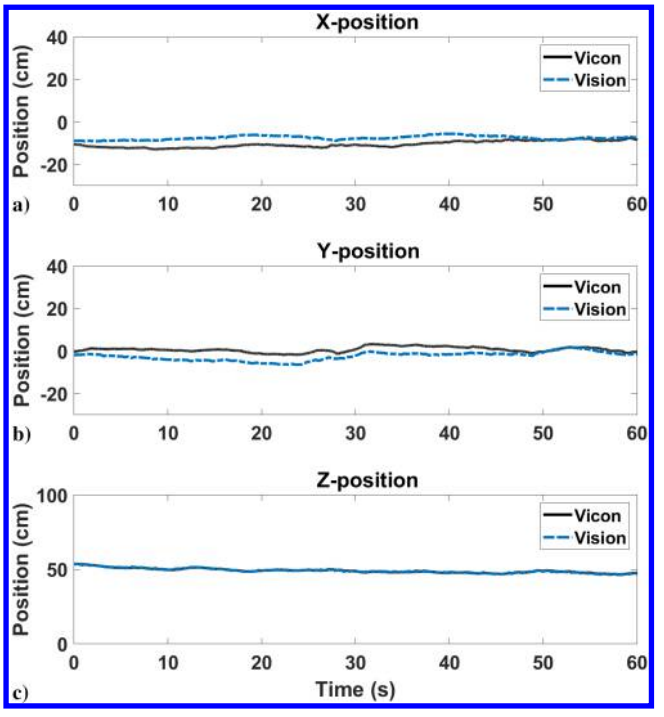


Fig. 23 Estimation of relative position of a ship deck in low illumination, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

Figs. 22 and 23. The attitude and position estimates for this case are accurate, closely matching the ground truth.

Finally, the algorithm was tested in glare conditions. A 900 lm flashlight was used to create significant glare and shadows on the ship deck, as shown in Fig. 24. The attitude and position estimates are provided in Figs. 25 and 26, respectively. The roll and pitch results acceptably follow ground truth, and the position estimates very accurately match ground truth.

Though the overall results appear accurate, there were some instances during the test in which the vision algorithm failed to detect the ship deck at all due to the glare obscuring a large portion of the ship deck. In particular, the algorithm did not successfully log any pose estimates between the times of 12–16, 24–29, and 40–43 s during the test. Such a failure could be problematic during an actual ship-deck landing, as the controller will not receive any updated ship-deck pose estimates for several seconds.

The RMSE of linear position and attitude, summarized in Table 5, can be used to further compare each test case. The low illumination and occlusion cases show comparable error to the ideal ship-deck motion test, indicating the algorithm is robust to these conditions. In the glare case, the errors are lower than in the occlusion or low illumination conditions. However, as the algorithm failed to detect the ship deck several times during the test, robustness in glare needs to be further improved.

A. Feature Match and Inlier Counts

As discussed earlier, the number of feature matches and inliers obtained during pose estimation can significantly affect the quality

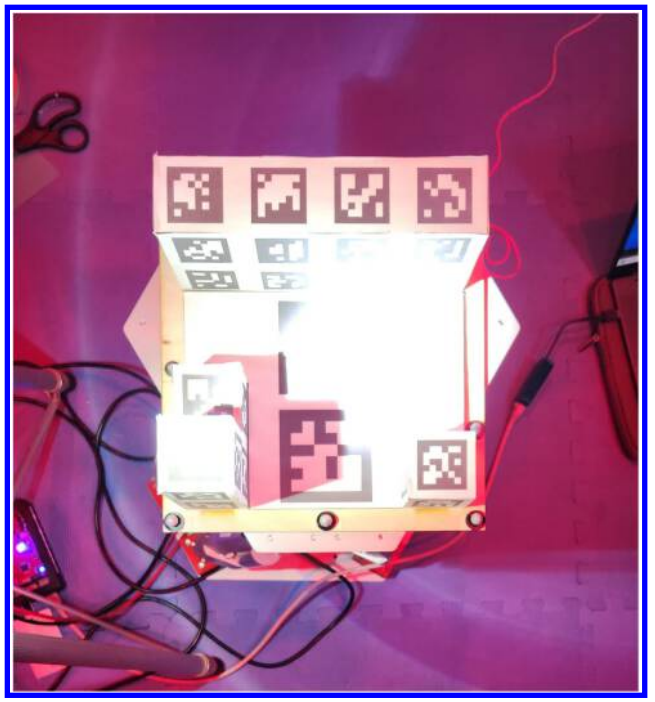


Fig. 24 The 3D ship deck with glare and shadows produced by a 900 lm flashlight.

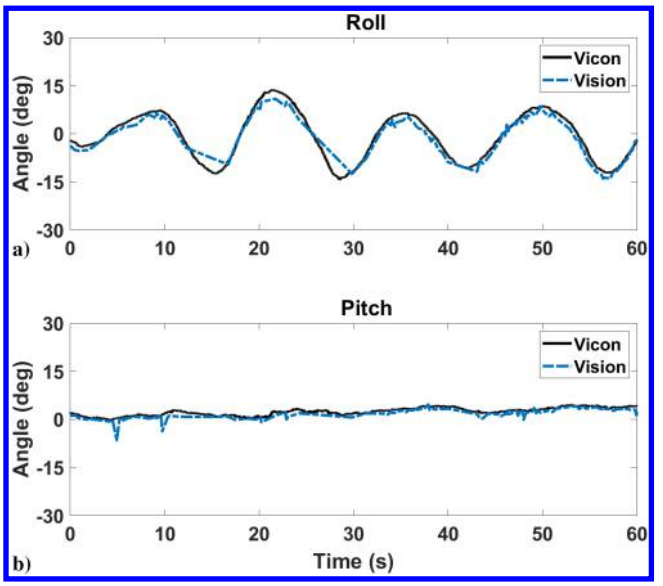


Fig. 25 Estimation of relative attitude angles of a ship deck with glare, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

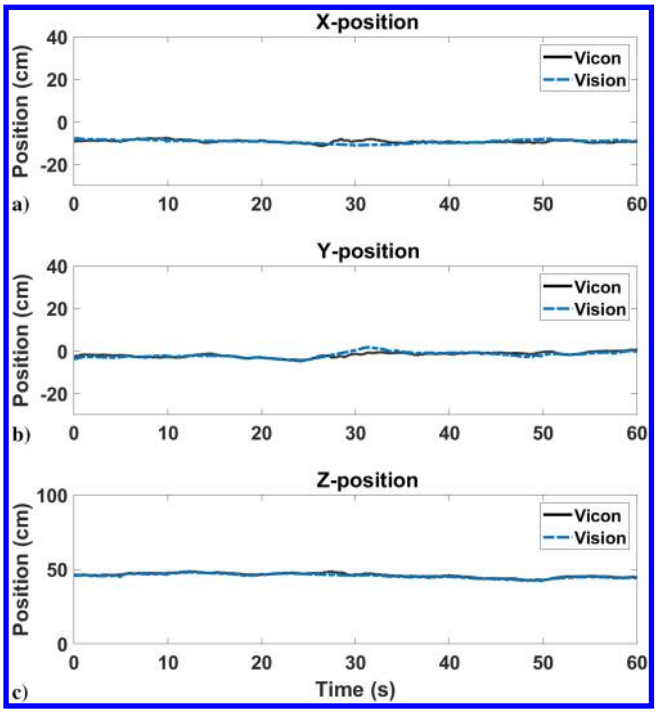


Fig. 26 Estimation of relative position of a ship deck with glare, with the quadrotor held stationary above the ship deck attached to the moving Stewart platform.

of results. Therefore, analyzing these values for various test cases can provide additional insight into algorithm performance in different scenarios. The mean number of feature match and inlier counts for each test case are summarized in Table 6.

The mean numbers of feature matches and inliers decrease somewhat for visually degraded conditions compared to the test cases without added visual degradations. However, a corresponding decrease in accuracy is not observed in pose estimation error. This suggests that, if a sufficient number of accurate feature correspondences can be obtained, additional matches may not significantly improve accuracy. On the other hand, additional inlier matches improve robustness to detection failure, compared to having a number of inliers very close to the minimum threshold. This is notable in the glare case, where pose estimates were not generated during multiple portions of the test due to an insufficient inlier count.

Table 5 RMSE for visually degraded conditions

Test case	X	Y	Z	Roll	Pitch
Occlusion	2.2 cm	2.3 cm	0.6 cm	2.0	1.6
Low illumination	3.3 cm	3.4 cm	0.3 cm	1.5	2.4
Glare	0.9 cm	0.9 cm	0.5 cm	1.7	1.0

Table 6 Summary of mean feature matches and inliers for each test case

Test case	Mean feature matches	Mean inliers
Quadrotor stationary	391	102
Quadrotor moving	383	87
Occlusion	370	75
Low illumination	342	61
Glare	370	69

X. Free-Flight Test

A simple free-flight test was performed to validate the algorithm's performance in hover. In hover, the vision algorithm must tolerate vibration and small motions that are not present with a hand-held camera.

The ship deck was secured to the floor for safety. The quadrotor was manually piloted and held in hover above the ship deck for several seconds. The results of the test are given in Figs. 27 and 28.

In hover, the algorithm is able to accurately estimate position and attitude, indicating it is viable for use in actual flight. The RMSE for pitch and roll are only 1.2 and 1.4 deg, respectively, while an RMSE of 4 cm is present for the X-, Y-, and Z-position estimates. This indicates that the vibrations and small motions that occur in free hover do not significantly degrade algorithm performance.

XI. Performance Comparison with 2D Vision Methods

The performance of the 3D feature-based vision algorithm can be further evaluated by comparing its performance to that of more typical 2D vision methods for ship-deck landing. The 3D feature-based algorithm offers significant advantages over 2D approaches,

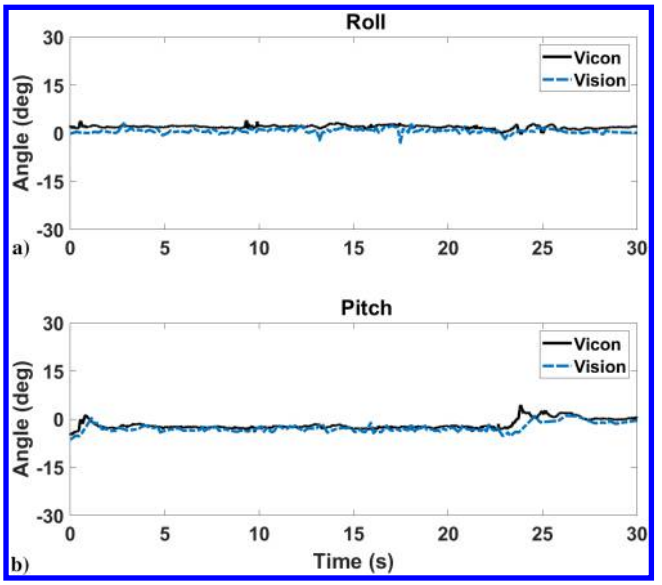


Fig. 27 Estimation of relative attitude angles, with the quadrotor hovering above the ship deck.

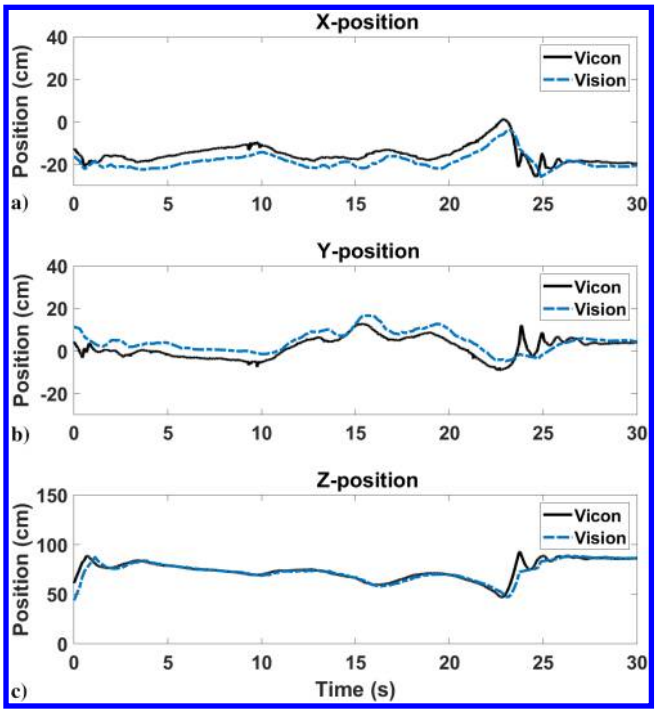


Fig. 28 Estimation of relative position, with the quadrotor hovering above the ship deck.

including its ability to work with 3D objects and its robustness to large rotations; it is therefore important to investigate whether it provides a similar level of pose estimation accuracy. For consistency, the comparison will be limited to work evaluating small-scale UAVs.

When evaluating our previously developed 2D feature-based algorithm under similar test scenarios as those presented in this paper, the pose estimation accuracy was found to be less than 4 cm along each linear axis and less than 2.5 deg for pitch and roll [13]. Using a recursive AprilTag fiducial marker, Nicholson et al. reported error in vision-based estimation of 8–23 cm in position and 1.8–3.1 deg in rotation [11]. Using a typical H-shaped landing target, Lin et al. observed position estimation error of 3-5 cm along each linear axis [21].

The test results for the 3D feature-based algorithm showed 0.5–6.3 cm of error along each linear axis and 1.2–2.4 deg of error in pitch and roll. Its performance is therefore comparable to that of 2D vision approaches.

XII. Increasing Algorithm Speed

The 3D feature-based algorithm is naturally far more computationally intensive than the 2D algorithm. A much larger number of features are extracted from the 3D object across many different camera positions, compared to the single image used for the 2D algorithm. Approximately 1200 features were extracted from the 2D ship-deck image, while the 3D ship-deck image set provided 5000–13,000 features, depending on feature detector and settings. This means that a much larger number of features must be matched in each update, increasing computational load dramatically.

For the ship-deck tests, the algorithm was able to update at a speed of 5–6 Hz. To be feasible for ship-deck pose estimation, at minimum, the algorithm must update at a sufficient frequency to adequately capture the stochastic ship-deck motion. From analyzing Fourier transforms of the SCONE ship-deck motion histories, the maximum frequency of ship-deck motion is approximately 0.34 Hz. Therefore, this update frequency is adequately fast to track the motion in real time. However, it is desirable to improve the update speed to improve the quality of pose estimation and to minimize delay in providing updated pose estimates to the controller. Additionally, more complex objects, such as real ships, may yield more features and corresponding points during feature extraction than the simplified object used here, increasing computational load. It is therefore important to explore techniques to potentially decrease computational time and increase update speed.

A. Filtering Visible Features

When updating the algorithm using a new frame, features detected in this frame are matched to features previously extracted from the 3D ship deck. However, only a subset of the features on the 3D ship deck will actually be visible in any single camera image. As feature matching is one of the most computationally demanding steps in the algorithm, restricting feature matching only to these visible features could potentially improve the speed of the algorithm.

The feature extraction process tracks which features are visible in each camera pose. From this, we can determine the ship-deck rotations where each feature should be visible. During real-time pose estimation, an initial pose estimate can be used to restrict feature matching in the next update to points visible in nearby rotations. The details of this process are described in the following subsections.

For this process, rotations were stored and processed as quaternions. This improves computational efficiency and allows necessary mathematical operations to be performed; in particular, using quaternions, it is possible to calculate distances between rotations as a single angle. Additionally, methods have been developed to efficiently compute the average of multiple quaternions [22].

B. Process for Filtering Visible Features

For each image of the ship deck used in feature extraction, there is an associated camera pose. Its orientation is stored as a rotation vector, which is converted to a quaternion representation for the filtering process. The visible rotations of each feature were obtained using the following procedure:

1) From the feature extraction results, we can recover the camera poses where a feature on the ship deck is visible. Each camera pose corresponds to a rotation.

2) Features may be visible in multiple camera poses. However, comparing multiple rotations for each feature is less computationally efficient than comparing a single rotation. Therefore, an average rotation for each feature is found by calculating the average of quaternions for all rotations where the feature was detected.

3) The average rotation for each feature is then stored in an array. Once this has been repeated for all of the features, save this array with the features and 3D points for later use.

The pose estimation process is modified to include this filtering. This is described in the following:

1) Choose a rotation threshold for a feature to be considered visible. When calculating the distance between rotations, if a rotation is within this threshold, it will be considered visible. The threshold should be large enough to account for reasonable differences between the camera pose and feature rotations, as well as movement of the camera and platform between algorithm updates.

2) Obtain an initial ship-deck rotation. This is done by matching image features to all of the saved object features, the same as in the original algorithm.

3) For each stored feature, calculate the distance between its rotation and the initial rotation. Create a modified list of features which meet the rotation threshold.

4) Complete the feature matching step using the modified list of features. Use this to find an updated pose estimate.

5) Repeat this process while the ship deck is in view of the camera. Each time a new pose estimate is obtained, the pose is used for the initial rotation in the next step.

Figure 29 illustrates the steps in this procedure.

C. Performance Comparison

Basic validation of the computational speed and accuracy of the algorithm with this added filtering was then performed. Four cases were compared: 1) filtering with a rotation threshold of 60°, 2) a rotation threshold of 75°, 3) a rotation threshold of 90°, and 4) no filtering of visible features. The 111 initial images of the ship deck were used for this test. The images were provided in sequence to the algorithm, and the pose estimates from the algorithm were compared to the ground truth poses of the ship deck.

One goal of the visible feature filtering is to make slower feature detectors viable for real-time pose estimation. Because of this, the validation test used the feature detector ORB with nFeatures = 1000, which required long computational times in earlier tests.

The results for computational time are provided in Table 7. The visible feature filtering results in a significant decrease in computational time compared to the algorithm without added filtering. This computational time decreases with smaller rotation thresholds.

The RMSE results for position and attitude are provided in Table 8. The error in estimated position is similar to the baseline case for both the 75° and 90° thresholds but increase slightly for the 60° case. For the estimated roll and pitch, the error is modestly increased for the 60° and 75° cases. However, a larger increase in error is seen with the 90° rotation threshold.

From these results, visible feature filtering can be used to improve the vision algorithm's computational speed, but a tradeoff must be made with the accuracy of the results.

Table 7 Mean computational time for visible feature filtering with different rotation thresholds

Test case	Mean computational time, s
60° threshold	0.108
75° threshold	0.122
90° threshold	0.139
No filtering	0.199

Table 8 RMSE for visible feature filtering with different rotation thresholds

Test case	X, cm	Y, cm	Z, cm	Roll	Pitch
60° threshold	0.2	0.6	1.0	3.9	3.1
75° threshold	0.2	0.5	0.8	3.9	2.6
90° threshold	0.2	0.6	0.8	4.0	5.2
No filtering	0.2	0.5	0.8	3.7	2.6

XIII. Conclusions

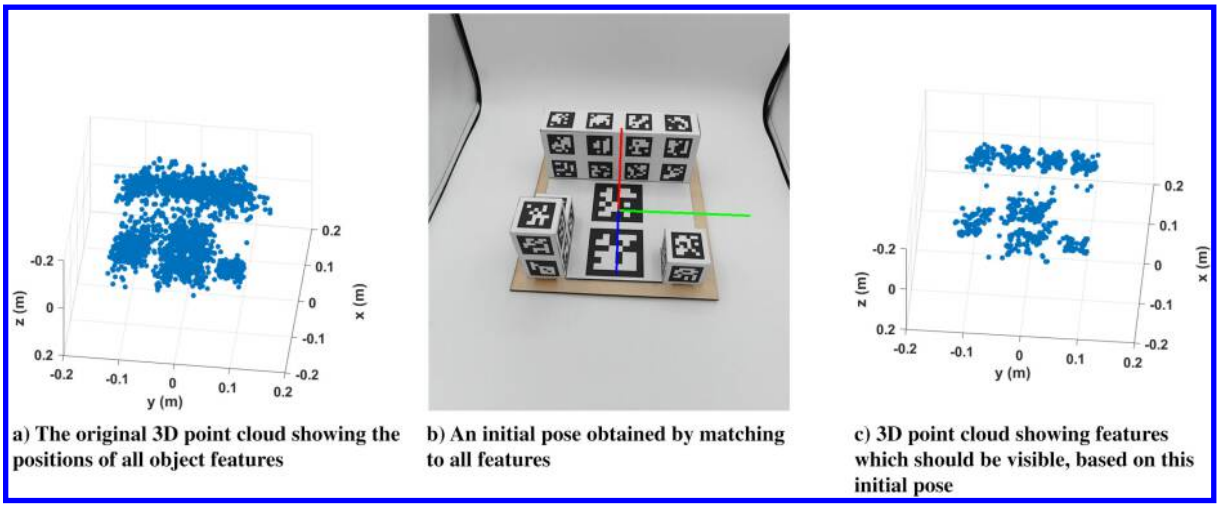
This work developed a new feature-based algorithm to estimate the pose of a 3D ship deck. The algorithm was incorporated into onboard quadrotor vision hardware and thoroughly validated using a simulated 3D ship deck with a Stewart platform. Algorithm performance with different feature detectors, specifically ORB and SIFT, was quantitatively evaluated. The algorithm was demonstrated to be capable of estimating ship-deck pose in real time under ideal conditions, Sea-State 6 ship-deck motion, and visually degraded conditions. Logged pose data were compared to Vicon ground-truth data. From these results, one can draw the following conclusions:

1) It is possible to use 3D computer vision to extract the pose of the ship deck using matched features.

2) The algorithm demonstrates robustness to low illumination and occlusion and was able to estimate the pose of the moving ship deck accurately in these conditions. However, the algorithm performance degraded in the glare condition, as it failed to detect the ship deck for several seconds during the test.

3) In ideal visual conditions, the 3D algorithm is capable of estimating the pose of a stationary ship deck as well as a ship deck undergoing Sea-State 6 motion.

4) Different feature detectors present tradeoffs for real-time performance. SIFT is accurate even at longer distances from the ship deck but is very computationally slow; ORB performs poorly at longer distances but is much faster, making it viable for real-time use on the quadrotor computer hardware.



a) The original 3D point cloud showing the positions of all object features

b) An initial pose obtained by matching to all features

c) 3D point cloud showing features which should be visible, based on this initial pose

Fig. 29 Images illustrating the steps in filtering visible features.

5) The 3D feature extraction process produces a large number of features, which has a significant penalty to algorithm speed due to the large number of features that must be matched. By determining which features are visible based on an initial orientation, and matching only to these features, it is possible to improve algorithm update speed significantly. However, this comes at the cost of increased error in pose estimation.

6) Filtering pose estimates based on the number of inlier points can improve the quality of algorithm output by removing poor quality results.

Acknowledgments

This work was supported under the NAVAIR grant number N0042121S0001 with technical monitoring from James Bumbaugh, Daniel Shafer, and Peter Arslanian and partly under the U.S. Army/ Navy/NASA Vertical Lift Research Center of Excellence grant number W911W61120012 with technical monitoring from Mahendra Bhagwat.

References

[1] Shastry, A., Datta, A., and Chopra, I., "Feature-Based Vision For Stochastic Motion Tracking Under Partial Occlusion," *Vertical Flight Society 78th Annual Forum and Technology Display*, Vertical Flight Soc., Fairfax, VA, May 2022, pp. 1–20. https://doi.org/10.4050/F-0078-2022-17640

[2] Britcher, V., Shastry, A., and Chopra, I., "Exploration of Feature-Based Algorithm for Autonomous Ship-Deck Landing under Visually Degraded Conditions," *Vertical Flight Society 79th Annual Forum and Technology Display*," Vertical Flight Soc., Fairfax, VA, May 2023, pp. 1–14. https://doi.org/10.4050/F-0079-2023-18190

[3] Britcher, V., Chopra, I., and Datta, A., "Development and Validation of 3D Feature-Based Vision Algorithm for Autonomous Ship-Deck Landing," *Vertical Flight Society 80th Annual Forum and Technology Display*, Vertical Flight Soc., Fairfax, VA, May 2024, pp. 1–21. https://doi.org/10.4050/F-0080-2024-1339

[4] Badakis, G., Koutsoubelias, M., and Lalis, S., "Robust Precision Landing for Autonomous Drones Combining Vision-Based and Infrared Sensors," *2021 IEEE Sensors Applications Symposium (SAS)*, Inst. of Electrical and Electronics Engineers, New York, Aug. 2021, pp. 1–6. https://doi.org/10.1109/SAS51076.2021.9530091

[5] Lin, J., Wang, Y., Miao, Z., Zhong, H., and Fierro, R., "Low-Complexity Control for Vision-Based Landing of Quadrotor UAV on Unknown Moving Platform," *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 8, Aug. 2022, pp. 5348–5358. https://doi.org/10.1109/TII.2021.3129486

[6] Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J., and Scaramuzza, D., "Vision-Based Autonomous Quadrotor Landing on a Moving Platform," *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Inst. of Electrical and Electronics Engineers, New York, Oct. 2017, pp. 200–207. https://doi.org/10.1109/SSRR.2017.8088164

[7] Araar, O., Aouf, N., and Vitanov, I., "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform," *Journal of Intelligent & Robotic Systems*, Vol. 85, No. 2, Feb. 2017, pp. 369–384. https://doi.org/10.1007/s10846-016-0399-z

[8] Truong, Q. H., Rakotomamonjy, T., Taghizad, A., and Biannic, J.-M., "Vision-Based Control for Helicopter Ship Landing with Handling Qualities Constraints," *IFAC-PapersOnLine*, Vol. 49, No. 17, Aug. 2016, pp. 118–123. https://doi.org/10.1016/j.ifacol.2016.09.021

[9] Sanchez-Lopez, J. L., Pestana, J., Saripalli, S., and Campoy, P., "An Approach Toward Visual Autonomous Ship Board Landing of a VTOL UAV," *Journal of Intelligent & Robotic Systems*, Vol. 74, Nos. 1–2, April 2014, pp. 113–127. https://doi.org/10.1007/s10846-013-9926-3

[10] Holmes, W. K., and Langelaan, J. W., "Autonomous Ship-Board Landing Using Monocular Vision," *American Helicopter Society 72nd Annual Forum*, American Helicopter Soc., Fairfax, VA, May 2016, pp. 1–15. https://doi.org/10.4050/F-0072-2016-11575

[11] Nicholson, D., Hendrick, C. M., Jaques, E. R., Horn, J., Langelaan, J. W., and Sydney, A. J., "Scaled Experiments in Vision-Based Approach and Landing In High Sea States," *AIAA AVIATION 2022 Forum*, AIAA Paper 2022-3279, 2022. https://doi.org/10.2514/6.2022-3279

[12] Shastry, A., Datta, A., and Chopra, I., "Vision-Based Autonomous UAS Landing on a Stochastically Moving Platform," *Vertical Flight Society 77th Annual Forum and Technology Display*, Vertical Flight Soc., Fairfax, VA, May 2021, pp. 1–18. https://doi.org/10.4050/F-0077-2021-16867

[13] Britcher, V., Datta, A., and Chopra, I., "Refinement and Characterization of Feature-Based Vision Algorithm for Ship-Deck Landing under Degraded Visibility," *Journal of the American Helicopter Society*, Vol. 70, No. 1, 2025, pp. 1–17. https://doi.org/10.4050/JAHS.70.012009

[14] Ma, Y., Soatto, S., Košecká, J., and Sastry, S. S., *An Invitation to 3-D Vision: From Images to Geometric Models*, 1st ed., Springer, New York, 2004, pp. 375–411. https://doi.org/10.1007/978-0-387-21779-6

[15] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, Nov. 2004, pp. 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[16] Tareen, S. A. K., and Saleem, Z., "A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Inst. of Electrical and Electronics Engineers, New York, March 2018, pp. 1–10. https://doi.org/10.1109/ICOMET.2018.8346440

[17] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., "ORB: An Efficient Alternative to SIFT or SURF," *2011 International Conference on Computer Vision*, Inst. of Electrical and Electronics Engineers, New York, Nov. 2011, pp. 2564–2571. https://doi.org/10.1109/ICCV.2011.6126544

[18] Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R., "Dino Data Set," *Multiview Stereo Evaluation*, Univ. of Washington, Seattle, Washington, D.C., Oct. 2005, https://grail.cs.washington.edu/projects/mview/.

[19] Harris, C. G., and Stephens, M. J., "A Combined Corner and Edge Detector," *Proceedings of the Alvey Vision Conference*, Alvey Vision Club, Manchester, U.K., Sept. 1988, pp. 23.1–23.6. https://doi.org/10.5244/C.2.23

[20] Schwartz, A., "SCONE—Standard Deck Motion Data for a Generic Surface Combatant," Office of Naval Research, March 2017.

[21] Lin, S., Garratt, M. A., and Lambert, A. J., "Monocular Vision-Based Real-Time Target Recognition and Tracking for Autonomously Landing an UAV in a Cluttered Shipboard Environment," *Autonomous Robots*, Vol. 41, No. 4, April 2017, pp. 881–901. https://doi.org/10.1007/s10514-016-9564-2

[22] Markley, F. L., Cheng, Y., Crassidis, J. L., and Oshman, Y., "Averaging Quaternions," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 4, July 2007, pp. 1193–1197. https://doi.org/10.2514/1.28949

R. Palacios
Associate Editor