drones

Article

Unstable Landing Platform Pose Estimation Based on Camera and Range Sensor Homogeneous Fusion (CRHF)

Mohammad Sefidgar * and **Rene Landry, Jr.**

LASSENA Laboratory, École de Technologies Supérieure (ÉTS), Montreal, QC H3C 1K3, Canada; renejr.landry@etsmtl.ca
* Correspondence: mohammad.sefidgar@lassena.etsmtl.ca or seyedmohammadhossein.sefidgar.1@ens.etsmtl.ca

Abstract: Much research has been accomplished in the area of drone landing and specifically pose estimation. While some of these works focus on sensor fusion using GPS, or GNSS, we propose a method that uses sensors, including four Time of Flight (ToF) range sensors and a monocular camera. However, when the descending platform is unstable, for example, on ships in the ocean, the uncertainty will grow, and the tracking will fail easily. We designed an algorithm that includes four ToF sensors for calibration and one for pose estimation. The landing process was divided into two main parts, the rendezvous and the final landing. Two important assumptions were made for these two phases. During the rendezvous, the landing platform movement can be ignored, while during the landing phase, the drone is assumed to be stable and waiting for the best time to land. The current research modifies the landing part as a stable drone and an unstable landing platform, which is a Stewart platform, with a mounted AprilTag. A novel algorithm for calibration was used based on color thresholding, a convex hull, and centroid extraction. Next, using the homogeneous coordinate equations of the sensors' touching points, the focal length in the X and Y directions can be calculated. In addition, knowing the plane equation allows the Z coordinates of the landmark points to be projected. The homogeneous coordinate equation was then used to obtain the landmark's X and Y Cartesian coordinates. Finally, 3D rigid body transformation is engaged to project the landing platform transformation in the camera frame. The test bench used Software-in-the-Loop (SIL) to confirm the practicality of the method. The results of this work are promising for unstable landing platform pose estimation and offer a significant improvement over the single-camera pose estimation AprilTag detection algorithms (ATDA).

Keywords: pose estimation; color thresholding; autonomous UAVs; pose estimation; image processing

check for updates

Citation: Sefidgar, M.; Landry, R., Jr. Unstable Landing Platform Pose Estimation Based on Camera and Range Sensor Homogeneous Fusion (CRHF). *Drones* **2022**, *6*, 60. https://doi.org/10.3390/drones6030060

Academic Editor: Yangquan Chen

Received: 10 January 2022
Accepted: 22 February 2022
Published: 25 February 2022

1. Introduction

An introduction to this research is followed by a review of the recent pose estimation techniques for unmanned aerial vehicles (UAVs).

1.1. Problem Statement

UAV-enabled services already range from food delivery to moon landing. They are rapid, economical, and futuristic. Several tasks must work together to materialize drone control and flight. These tasks include takeoff, landing, and hovering. To make these tasks operational, a drone must have a full knowledge of its position (or "pose") in space. Pose estimation is an inseparable part of drone control, hence the plethora of research relating to these tasks over the past few years.

To develop the sensor fusion approach, the proposed algorithm includes the collection of distances from four sensors and their corresponding pixel coordinates at different altitudes and utilizes the interpolated version of these data to project the focal length by means of homogeneous coordinate equations. This information is then used for ToF sensor and camera calibration. The ToF–camera calibration is employed to acquire the focal lengths in the x and y directions. Knowing the focal lengths and planar equations of the landing surface, calculated by the mounted ToF sensors, and the landmark pixel coordinates, the world Cartesian coordinates of the landmark, AprilTag, is obtained. In the final step, the rigid body transformation of the landing platform and the camera is found by using the projected coordinates.

Recently employed techniques use sensory data to retrieve information about drone location and attitude angles. Some of the commonly used sensors are the Global Positioning Systems (GPS), Inertial Measurement Units (IMU), the Global Navigation Satellite System (GNSS), cameras, etc. The pose estimation task is even more challenging when the uncertainties of the environment increase. These uncertainties are linked to fogs, signal jamming, clouds, the unavailability of remote signals, and relatively fluctuating UAV stations, such as a ship's deck. However, the application of the GPS signal indoors and in some GPS-denied environments is unattainable, making the pose estimation imprecise and in some cases impossible [1]. Next, in defense-related design, signal jamming is considered, and alternative sensors and algorithms are included to tackle distorted signals [2]. Camera data are also reliable and usable when there is no occlusion and the light source is sufficient [3]. Furthermore, a downside to the INS inclusion in design is that they are heavy, bulky, and require high-power consumption. Moreover, the presence of radio frequency interference (RFI) can impose limitations in the system performance [4].

Few of the recent state-of-the-art studies focus on the more challenging positioning of non-GPS unstable areas, such as marine environments, where the UAV stations are unsteady. Moreover, the most reliable systems studied use multiple sensors to calculate the pose. Hence, we concentrate the current work on addressing the problem using sensor fusion. The advantages and novelties of the current research are manifold and can be listed as follows:

- The first contribution would be increasing the accuracy of altitude estimation during landing using range sensors, Time of Flight (ToF) sensors. This is a fit replacement for Light Detection and Ranging (LIDAR), which is expensive and heavy for installation on small-sized drones.
- Second, knowing the four ranges from the sensors, the equation of the descending platform is calculated. Hence, this ameliorates the calculation accuracy of the landing points in the real world in X and Y directions as well, letting X, Y, and Z be the world coordinate system.
- The data from the range sensors are also used for the calculation of landing platform Euler angles. Therefore, the proposed algorithm improves Euler angle estimation.
- Lastly, we employed sensor fusion that can operate completely in low light conditions. The camera contributes to the calculation of the headings of the landing platform; hence, the algorithm can be used in situations when the heading calculation is insignificant.

1.2. Literature Review

Truong et al. (2020) employed Skip Connection and Network (DCSCN) topology to create a super-resolution image-based posed estimation [5], while the authors of [6] applied LightDenseYOLO for tag tracking, leading to an accurate localization of the tag in the image. Another study concentrated on deep learning, utilizing SlimDeblurGAN architecture to de-blur images for landing tag detection, leading to a better-quality image when the image motion noise increased due to camera shaking or when gimbals are not available for stabilization [7]. A multi-layer perceptron and ultrasonic sensors mounted on the four arms of a quadrotor were used to estimate the landing surface suitability in [8].

UAVs are now being used in public services, and thus they must have safe landing measures in place, such as crowd detection. Castellano et al. (2020) took advantage of a fully convolutional network (FCN) to train UAVs to automatically distinguish between crowded and non-crowded areas [9]. UAV research is not limited to outdoor environments; indoor spaces are also covered in drone research. A work conducted in an indoor environment was able to detect the center of a gate and the gate's ROI based on a Single Shot Detector (SSD) algorithm for indoor drone racing [10]. While that algorithm uses deep learning, reinforcement learning, which includes reward- and penalty-based policy making, has progressed at the same rate. Chang et al. (2020) utilized ArUco markers as a reference to augment the accuracy of autonomous drones performing straight takeoff, flying forward, and landing, based on Deep Reinforcement Learning (DRL) [11]. Deep Q-Networks (DQNs) for navigation and Double Deep Q Networks (DDQNs) have been investigated for an environment with noisy visual data, improving the system performance to generalize efficiently in occluded areas [12].

As stated earlier, the complexity of UAV tasks increases significantly when the target is not stationary. One approach to this issue was investigated using a Deep Deterministic Policy Gradient (DDPG) algorithm, ameliorating the system practicality and continuous UAV landing maneuvers on a moving platform [13]. A more complex environment for testing algorithms is drone racing. Incorporating deep reinforcement learning and the relative gate observations technique to fly through complex gates has been investigated and appears promising for small-sized drones that can fly up to 60 km/h [14]. Although recent research has used artificial intelligence (AI) for drone control and state estimations, navigation-based techniques are at the center of most works. Several studies have addressed the issues of state estimation by removing the uncertainty of the system using filters such as the Kalman filter for linear motion, and nonlinear filters such as the extended Kalman filter (EKF), unscented Kalman filter (UKF), and particle filters (PF) [15–20]. A combination of the GPS, the nine degrees-of-freedom (DoF) IMU, and a barometer resulted in an accuracy level of approximately 2 m [21]. A subsequent effort applied ultra-wideband technology, low-cost IMU, and vision-based sensors to estimate the position of a mini drone. That same work used an extended Kalman filter to eliminate uncertainty from the proffered algorithm, resulting in a 10 cm accuracy [22]. A novel study compared the efficiency of the EKF and a variation of the EKF, the UKF. Their results showed that the EKF algorithm's accuracy drops when the vehicle dynamics are low, as the linearization of the EKF will fail, and proved that the UKF offers better accuracy with lower computational time [23]. Indoor drone navigation has also employed PFs to remove uncertainty from ultra-wideband (UWB)-based position estimation, with the results compared to those of an EKF. That work concluded that the nonlinear filter offered 14% better accuracy when drone dynamics were linearized through the extended Kalman filter (EKF) [18]. The work in [24] proposed infrared beacons along with a narrowband filter, making use of particle filtering to fuse both IMU and visual data to land a drone on the harsh situation of a makeshift runway in a GPS-denied situation. That combination achieved acceptable accuracy for long-range pose estimation.

2. Proposing Camera and Range Sensor Homogeneous Fusion (CRHF)

Figure 1 provides the full pathway followed to estimate the state of the AprilTag in the camera frame.

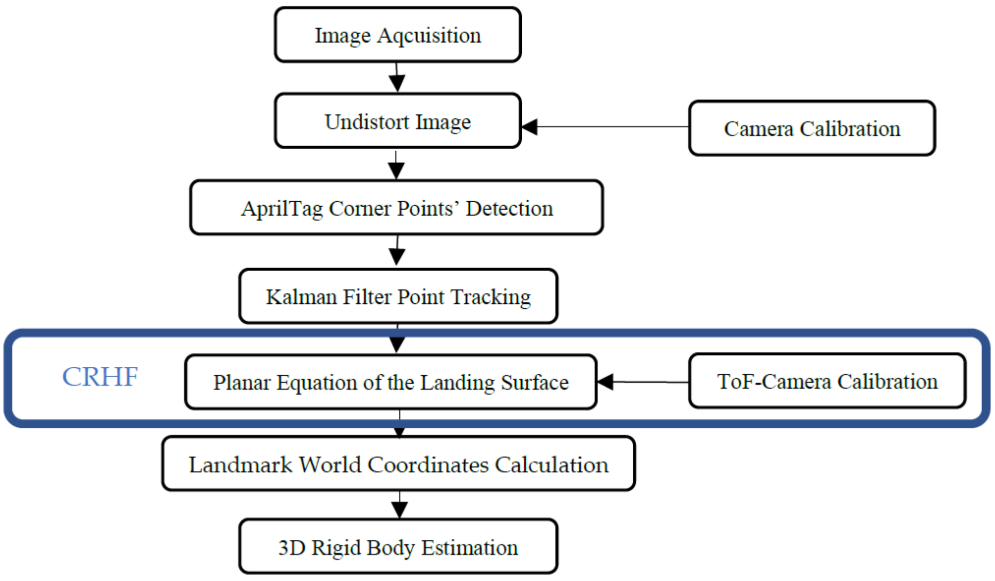


Image Aqcuisition
Undistort Image
Camera Calibration
AprilTag Corner Points' Detection
Kalman Filter Point Tracking
CRHF
Planar Equation of the Landing Surface
ToF-Camera Calibration
Landmark World Coordinates Calculation
3D Rigid Body Estimation

Figure 1. Overview of camera and ToF base pose estimation using CRHF.

2.1. Camera Calibration

Several open-source and closed-source applications are available to calibrate the monocular camera. Camera calibration theory uses the chessboard corners and size of the chessboard to calculate the distance of the intrinsic, extrinsic, and principal point parameters, or camera model, of a camera. A camera model can be written as follows:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1}$$

where u and v are vertical and horizontal pixel coordinates. f_x is the focal length in the x direction, f_y is the focal length in the y direction, c_x is the camera principle point in the x direction, and c_y is the camera principle point in the y direction. r_{ij} is the rotational component of the camera, t_x is the translational component of camera, X_w is the real-world location of the camera in the x direction, Y_w is the real-world location of the camera in the y direction, and Z_w is the real-world location of the camera in the z direction.

The camera calibration procedures follow the steps below:

- Collect images;
- Convert them into a grayscale image;
- Calculate the chessboard corners from the grayscale image;
- Compute the real-word chessboard corners. The chessboard has a square size of 7 cm and 7 × 10 corners; and
- Calculate the related camera projection matrix using Direct Linear Transformation (DLT), defined as follows [25–29]:

$$\lambda x = CX \tag{2}$$

where X represents the world points, C is the camera matrix, λ is a scalar value, and x is an image of the real world points in Equation (2). Figure 2 demonstrates the rover for camera calibration process.

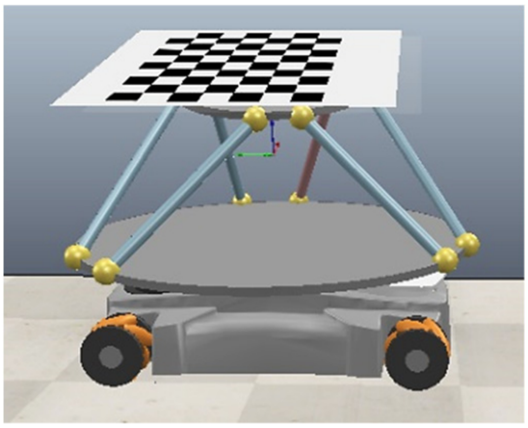


Figure 2. Camera calibration by Rover and Stewart table.

The DLT algorithm is based on the following five steps:

1. The input image point coordinates determined by transform "T" normalization;
2. Camera projection matrix C^N, camera matrix in frame number N, projected from the normalized image points;
3. Camera matrix C as the $C^N T^{-1}$ denormalization;
4. The reprojected image point coordinates, x^E, calculated by C and X multiplication;
5. The reprojection error calculation by Equation (3) [30,31]:

$$\text{Rep}_{Error} = \left| x - x^E \right| \tag{3}$$

A camera reprojection matrix can be represented as follows [32]:

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where C_x, C_y, f_x, and f_y are the camera principal points in x and y coordinates and the focal lengths in x and y coordinates, respectively.

- Use the two equations below, (5) and (6), to obtain radial distortion coefficients of the camera [33,34].

$$x_{\text{distorted}} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{5}$$

$$y_{\text{distorted}} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{6}$$

in which x and y are the coordinates of the normalized image—undistorted pixel position, $x_{\text{distorted}}$ is the coordinate of the distorted pixel in the x direction, $y_{\text{distorted}}$ is the coordinate of the distorted pixel in the y direction, and k_1, k_2, and k_3 are the lens's radial distortion coefficients. The magnitude of the radial distortion can be calculated as follows [35–38]:

$$r^2 = x^2 + y^2 \tag{7}$$

- Utilize following equations to obtain tangential distortions (8) and (9) [35–38]:

$$x_{\text{distored}} = x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right] \tag{8}$$

$$y_{\text{distorted}} = y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right] \tag{9}$$

where x and y are in normalized image coordinates—undistorted pixel position, and p_1 and p_2 are the tangential distortion lens coefficients.

2.2. Camera Calibration Implementation and Results

Camera calibration determines the inner camera parameter calculation, which is vital for practical operations and calculations such as world-to-camera coordinate mapping. The calibration guideline was followed using MATLAB Image Processing Toolbox User's Guide [39]. To implement camera calibration, a chessboard was mounted on a Stewart platform and the whole system was placed on the back of a rover. The rover traveled in a circular path.

Figure 3 illustrates the calibration process from below ground to some distance above ground.

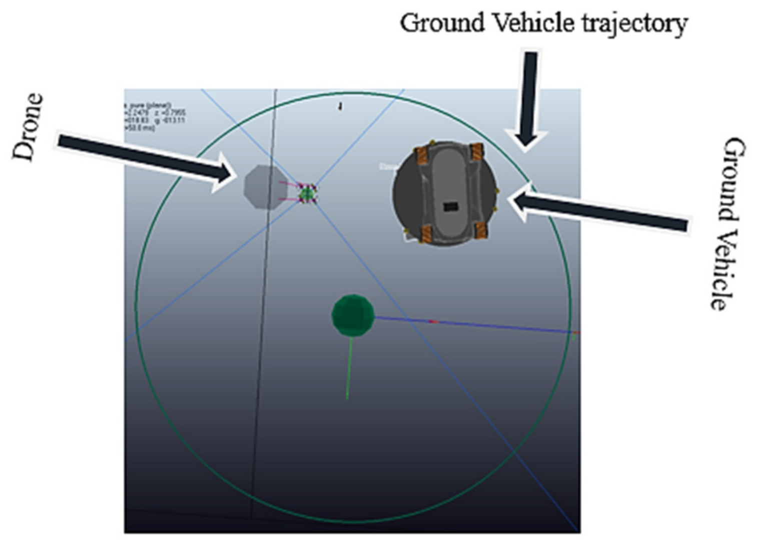
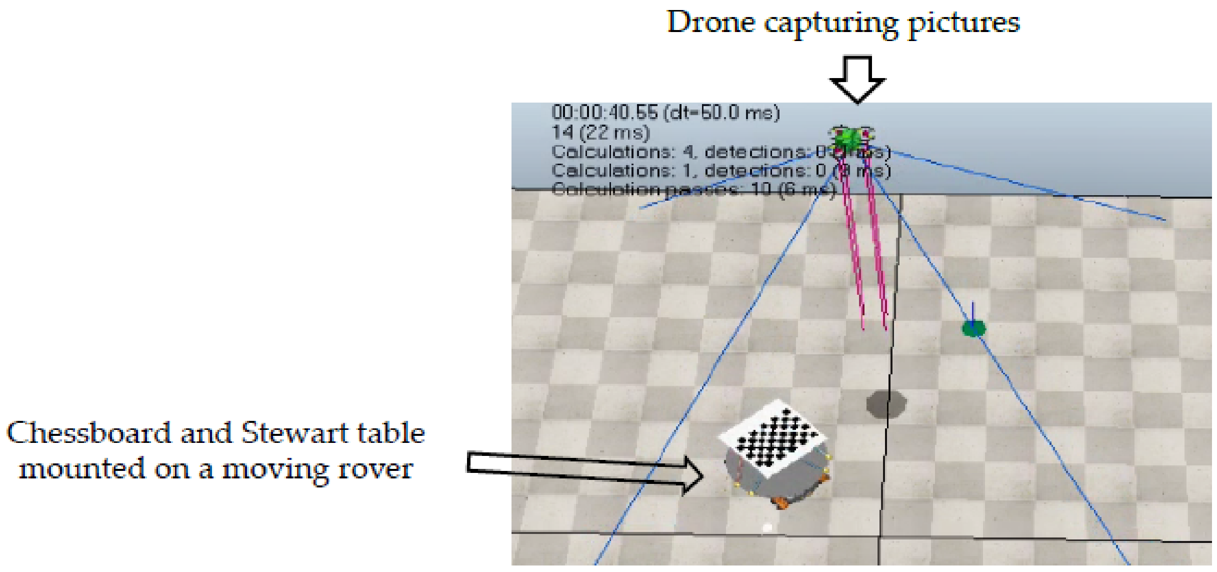


Figure 3. Drone, ground vehicle (rover), and circular path setup.

The calibration was developed in a MATLAB script that creates a scenario as follows, illustrated in Figure 4:

- The drone takes off.
- It hovers towards the surveillance point that includes all the rover paths and waits without any movement.
- The drone takes 20 photos when the rover is traveling along the circular path.



```
00:00:40.55 (dt=50.0 ms)
14 (22 ms)
Calculations: 4, detections: 0 (0 ms)
Calculations: 1, detections: 0 (0 ms)
Calculation passes: 10 (6 ms)
```

Figure 4. Downward camera calibration using a moving rover with a chessboard.

Figure 5 shows the detected points on the chessboard using the MATLAB camera calibration application.

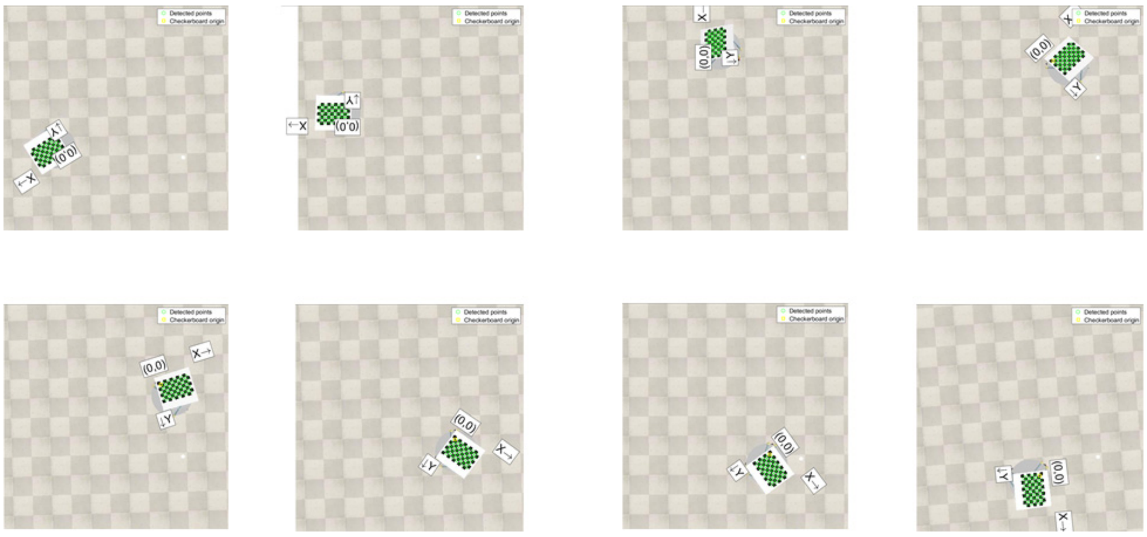


Figure 5. Corner detection for camera calibration in a circular path using the MATLAB camera application.

The chessboard Stewart table set on the rover moves along the circular path, covering all the coordinates of the camera's pixels. All the photos are processed by the MATLAB single-camera calibration application.

The camera calibration result in camera centric mode is illustrated in Figure 6.

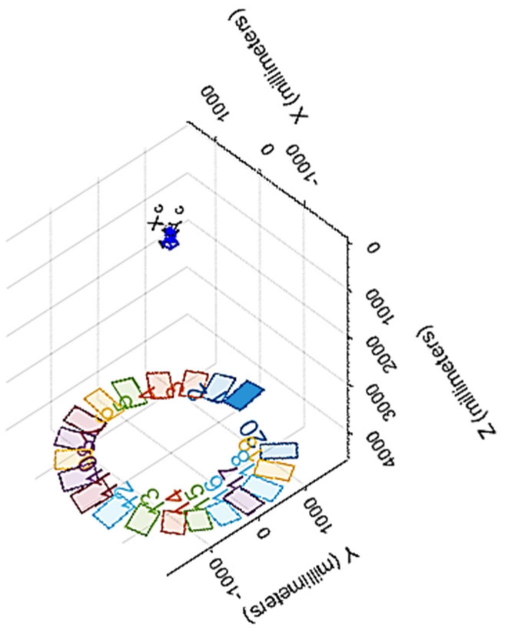


Figure 6. Camera centric mode calibration results.

While in some references, such as [40], it is highly recommended to conduct the camera calibration at the altitude that the camera is used, in the landing system camera calibration processes, there is no single distance for the best calibration point since the UAV must monitor behaviors of the landing platform until it touches the surface. In the current research, the maximum ToF sensors' range is selected as the calibration distance.

The intrinsic matrix results for a simulator camera with 512 × 512 resolution is given in Equation (10):

$$K = \begin{pmatrix} 458.5 & 0 & 0 \\ 0 & 457.855 & 0 \\ 258.7735 & 252.308 & 1 \end{pmatrix} \tag{10}$$

2.3. Camera ToF Calibration Implementation and Results

Figure 7 illustrates an overview of the camera and ToF calibration procedure.

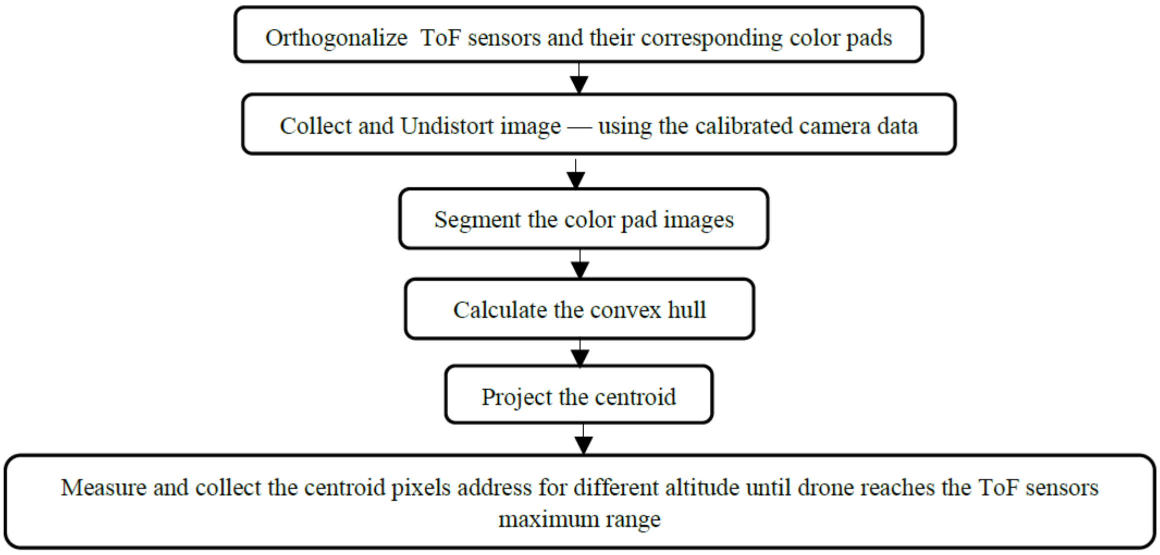

Orthogonalize ToF sensors and their corresponding color pads
Collect and Undistort image — using the calibrated camera data
Segment the color pad images
Calculate the convex hull
Project the centroid
Measure and collect the centroid pixels address for different altitude until drone reaches the ToF sensors maximum range

Figure 7. Camera ToF calibration overview.

The same radius color pads were used for each range sensor. The sensors are then orthogonalized to the center of the allocated color pads (refer to Figure 8).

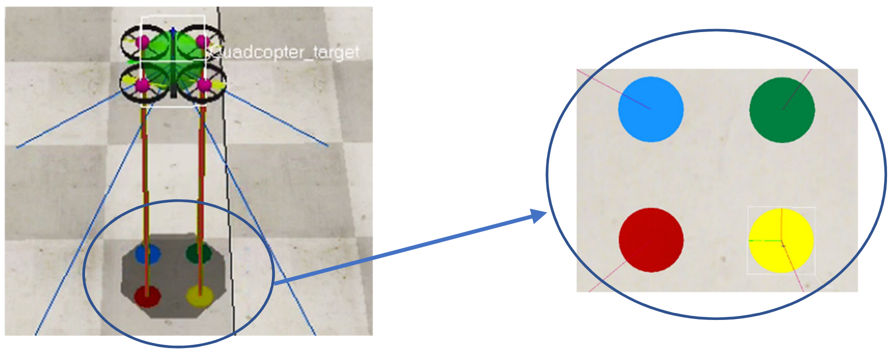


Figure 8. Orthogonalizing range sensors to allocated color pads.

At every altitude that the quadrotor rises to, the corresponding pixels are computed using the color segmentation technique. To segment the color, the RGB image is first converted to YCBCR format using Equation (11) [41]:

$$\begin{bmatrix} 601_Y \\ 219' \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \cdot \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{11}$$

where R', B', and G' are the normalized versions of the RGB matrices. C_B and C_R are the blue-difference and red-difference chroma components. The result of the summation and production is a 3 by 3 matrix. Hence, there are three channels for YCBCR representation. To segment different colors, the pixel value must be limited to the corresponding color range. To implement this limitation and segmentation, all the values in a specific range will be one and the rest will be considered as zero. The range specifications are presented in Table 1.

Table 1. Color channel segmentation ranges.

Color	Channel 1	Channel 2	Channel 3
Yellow	103 < Range1 < 255	0 < Range2 < 95	0 < Range3 < 255
Red	0 < Range1 < 160	0 < Range2 < 255	167 < Range3 < 255
Green	0 < Range1 < 255	0 < Range2 < 155	0 < Range3 < 90
Blue	0 < Range1 < 165	139 < Range2 < 255	0 < Range3 < 255

The color segmentation result is illustrated in Figure 9 in the masked format.

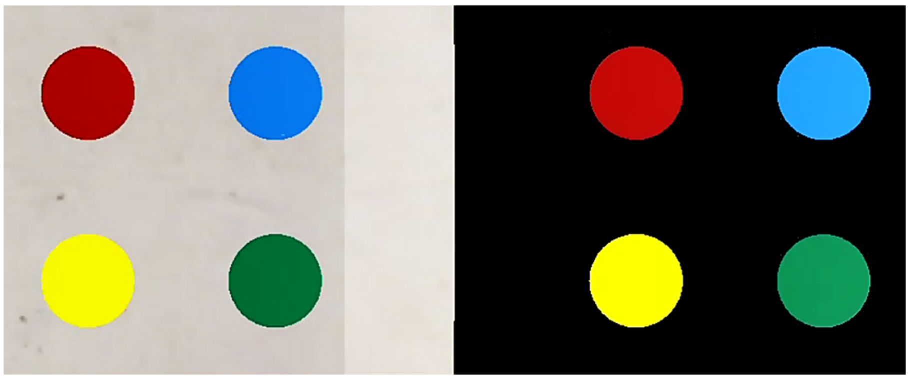


Figure 9. Image and masked image, from left to right, using color segmentation.

To ensure that there is no hole in the segmented color, the convex hull is calculated on the binary image. The convex hull is denoted as the smallest convex set that contains all the pixels of a region represented in the Euclidean plane:

$$X_k^i = \left(X_{k-1} \otimes B^i\right) \cup A \text{ i} = 1,2,3,4 \text{ and } k = 1,2,3,\ldots \tag{12}$$

where B^i is the structuring element of the binary image, $\otimes$ is element-wise multiplication operator, $\cup$ is the union operator, and X_k is the set of convex sets which are lines connecting points in the dataset, making a closed shape, expressed by Equation (13).

$$X_0^i = A \text{ and } D^l = X_{\text{con}}^i \tag{13}$$

where A is an arbitrary convex set, D is the converged convex subset, and subscript "conv" is defined as the convergence of the sets in the sense that satisfies Equation (14).

$$X_k^i = X_{k-1}^i \tag{14}$$

X_k^i and X_{k-1}^i are selected candidate subsets for the convex hull. Equation (15) can then be used to find the convex hull of the binary image [42–44].

$$\mathrm{C(A)} = \bigcup_{i=1}^{4} D^i \tag{15}$$

Figure 10 depicts the result for Segmented binary and convex hull process.

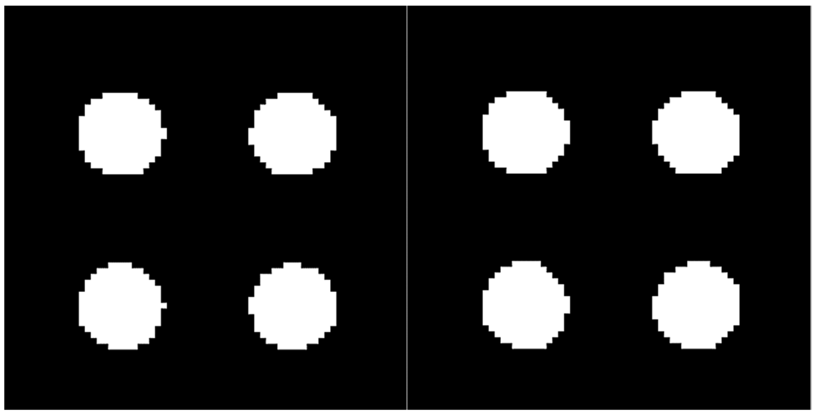


Figure 10. Segmented binary and convex hull results.

The convex hull is calculated to ensure that there is no non-uniform region of the binary image. The centroid of the binary image is then calculated using Equation (16):

$$\begin{cases} \vartheta x = \frac{1}{N}\sum_{i}^{N} x_i \\ \vartheta y = \frac{1}{N}\sum_{i}^{N} y_i \end{cases} \tag{16}$$

Figure 11 shows the centroid of each binary image, indicating the pixel correspondence with each sensor. The accumulation of all the points shows the 3D point correspondence of each pixel with respect to each sensor. The results are shown in Figure 12, where N is the number of points in the image [45,46].

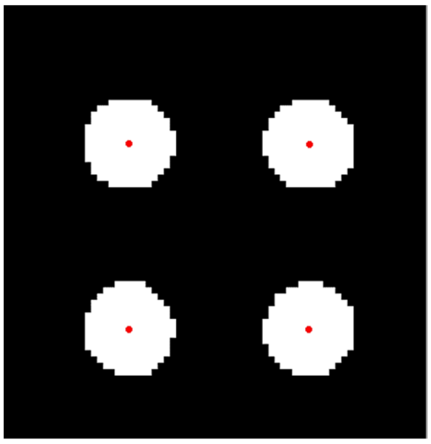


Figure 11. Convex hull centroid results.

Samples of the pixel and altitude data are summarized in Tables 2 and 3.

Table 2. First three frames' centroid calculations.

Color	C1—Frame1		C2—Frame2		C3—Frame3	
Blue	400.0070961	112.909046	392.6057903	120.4082739	383.8957617	129.1591336
Green	399.5563178	399.9410627	392.0877826	392.3846207	383.3468678	383.720831
Yellow	112.5299193	400.1530166	119.9678525	392.630475	128.6982725	384.0305636
Red	112.8321163	112.9362624	120.3259779	120.4101189	129.0014117	129.1613991

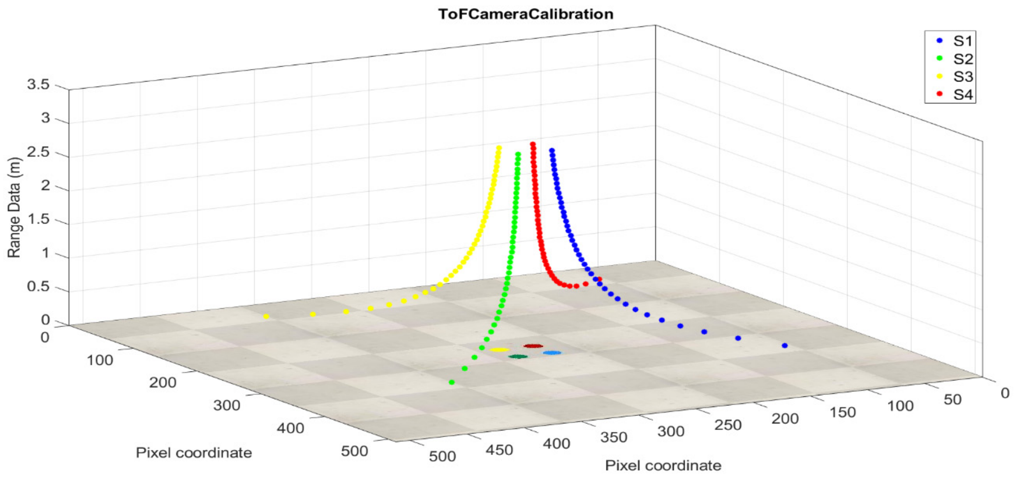


Figure 12. Camera pixels as a function of the sensor distance.

Table 3. First three frames' corresponding distances relevant to the sensors.

Color	C1—Distance Frame1	C2—Distance Frame2	C3—Distance Frame3
Blue	0.305179805	0.32637015	0.348936737
Green	0.304939508	0.32626918	0.34884578
Yellow	0.304848552	0.32633689	0.348892212
Red	0.304876596	0.326466531	0.349034697

There are two scenarios to pursue for pixel correspondent modeling:

- Curve fitting with the hyperbola parametric mode as "$f(x) = a/x + c$" and then finding a and c parameters; and
- Interpolating between the found points.

The hyperbola parametric mode was chosen because the model of the object projection in homogeneous coordinates is hyperbolic and can be represented as:

$$x = f\frac{X}{Z}, \; y = f\frac{Y}{Z} \tag{17}$$

where x and y are pixel coordinates, and X, Y, and Z are the real-world coordinates of the drone's altitude. The perspective projection overview can be illustrated as in Figure 13.

Knowing each altitude's corresponding pixel, it is easy to fit a hyperbola parametric model from the acquired data so that calibration equations, such as Equation (18) for all four range sensors and corresponding image pixels, can be obtained. Take as an example sensor 1 with its blue-colored pad. The parametric mathematical model is:

$$f(x) = a/x + c \tag{18}$$

Curve fitting seeks to find the "a" and "c" parameters from the provided data, where "a" is the focal length and "c" is the principal point of the camera calculated from each sensor. The Least Absolute Residuals (LAR) regression method, which finds the a curve that minimizes the absolute difference of the residual, is used in tandem with the Levenberg–Marquardt optimization algorithm. Figures 14 and 15 demonstrate the fitting results in image plan (x_c and y_c) and range data.

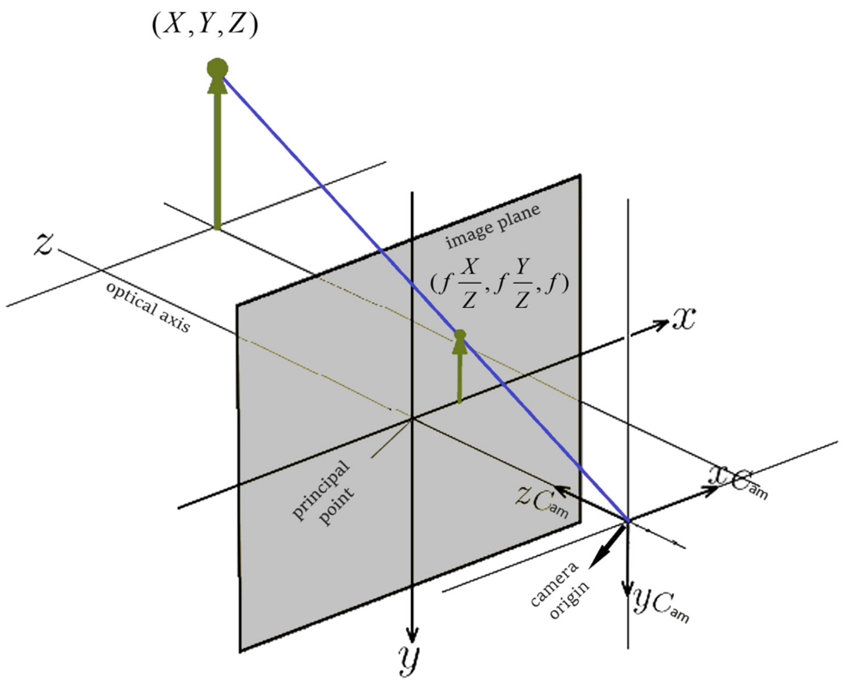


Figure 13. Real world to image plane projection.

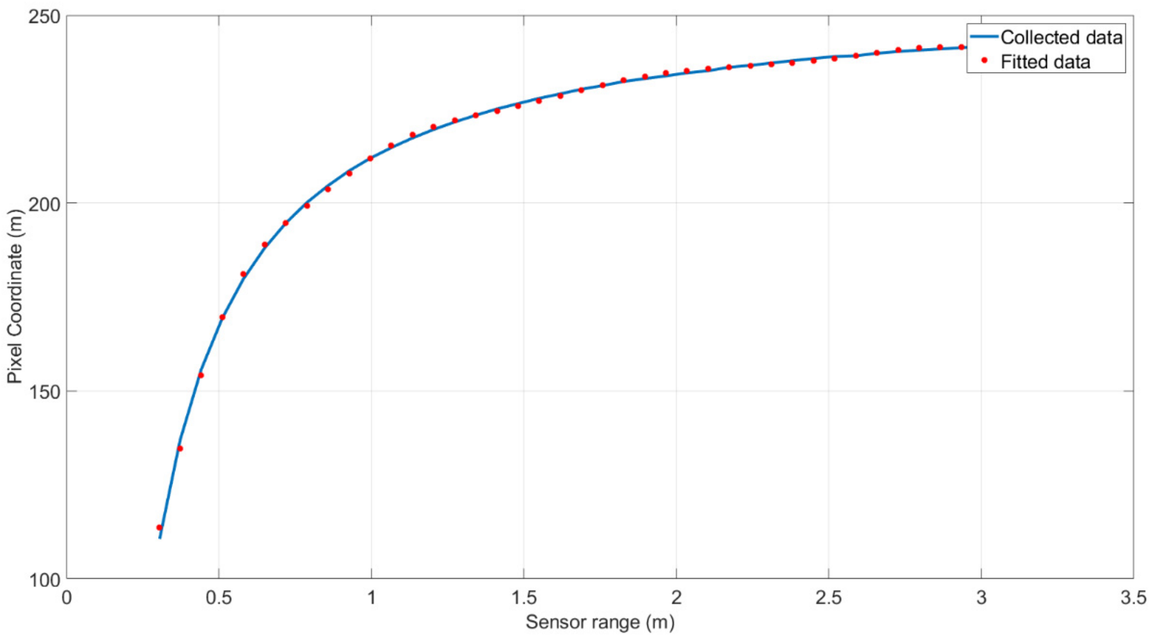


Figure 14. Range data versus pixel coordinate x_c—axis pixel curve fitting.

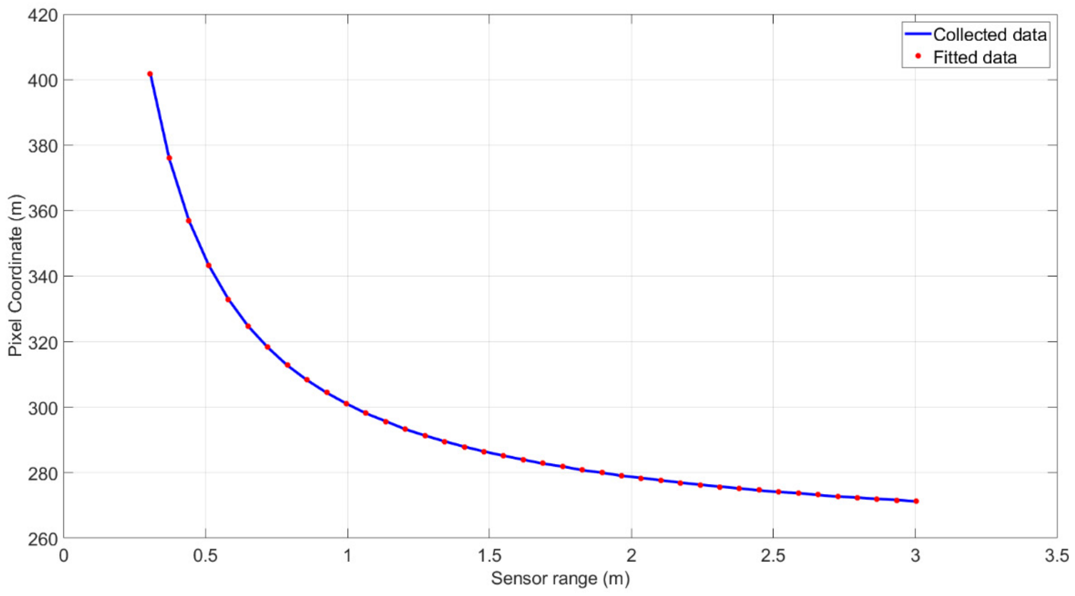


Figure 15. Range data versus pixel coordinate y_c—axis pixel curve fitting.

Using the curve fitting MATLAB application, the minimum error of the curve fitting reached by the tool was 0.1323. However, to not sacrifice any accuracy, the fitting algorithm was not used; instead, shape-preserving interpolation was utilized as a measure for calibration. The interpolation error is inconsiderable when the number of points is sufficient. In addition, the interpolation speed with the sample point, here 120 points, in each pixel coordinate does not reduce the performance. Surprisingly, the execution of the interpolation for one point is less than 0.03 s. Therefore, the second choice was selected, using the interpolation technique. Table 4 shows the results of the curve fitting.

Table 4. Sensor distance to pixel curve fitting results.

Parameters / Coordinate	X Coordinate	Y Coordinate
Coefficient trustworthiness	95% confidence	95% confidence
Coefficient value	a = −44.43 c = 256.6	a = 44.43 c = 256.6
Goodness of fit	SSE: 0.1409 R-square: 1 Adj-R-square: 1 RMSE: 0.03456	SSE: 0.1323 R-square: 1 Adj-R-square: 1 RMSE: 0.03348

Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) is a technique for interpolating between a set of points. The procedures listed below can be employed to calculate a PCHIP from a set of data. In the PCHIP algorithm, the value of f'_k at point x_k can be calculated by Equations (19) and (20). Assume $h_k = x_{k+1} - x_k$, $d_k = (y_{k+1} - y_k)/h_k$ is the slope between two interpolating points x_{k+1}, x_k and $w_1 = 2h_k + h_{k-1}$, $w_2 = h_k + 2h_{k-1}$ are PCHIP interpolation weights. The slope of the point between them has the equation:

$$f'_k = \begin{cases} \dfrac{(w_1+w_2)}{\left(\frac{w_1}{d_{k-1}} + \frac{w_2}{d_k}\right)} & d_k, d_{k-1} \neq 0 \; \& \; sign(d_k) \neq sign(d_{k-1}) \\ 0 & d_k, d_{k-1} = 0 \; \& \; sign(d_k) = sign(d_{k-1}) \end{cases} \quad (19)$$

$$sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \quad (20)$$

where *sign* implies the sign function of a number, and w_1 and w_2 are the interpolation weights [47]. Note that it is important to ensure that the end slopes are projected using a one-sided scheme [48].

2.4. Calculation of the Planar Equation of the Landing Platform Beneath the Drone

Four sensors are used to calculate the landing surface planar equation. Figure 16 illustrates the position of the drone, indicated by a cross, using the four sensors, S_1–S_4. Figure 16 depicts the platform at rest (a) and at a variable attitude (b).

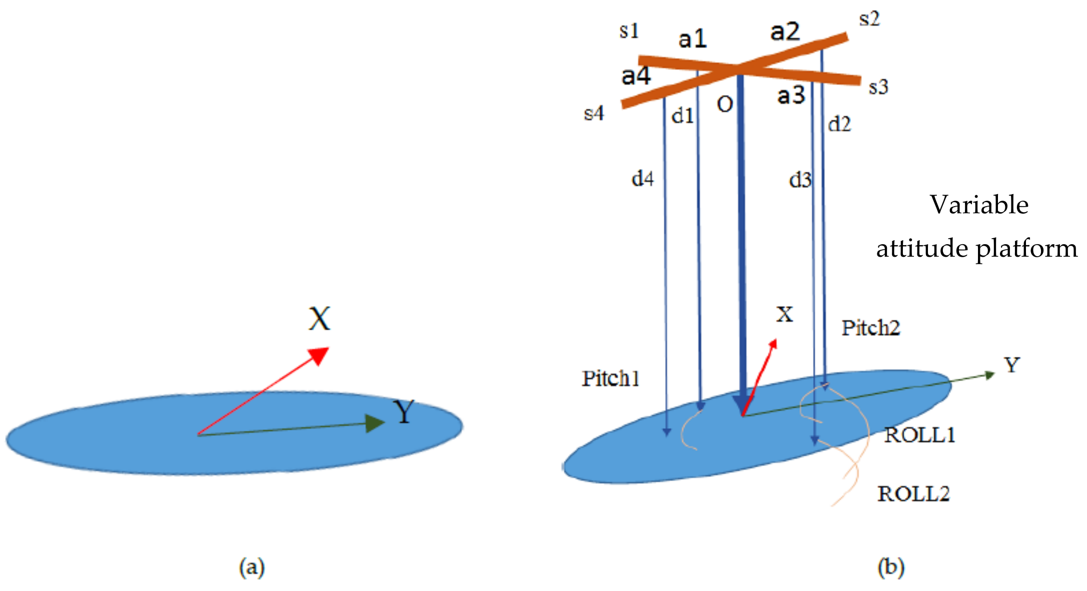


Figure 16. Landing platform and drone with related measurements and distances. The platform at rest (**a**) and at a variable attitude (**b**).

Equations (21)–(24) are utilized to transform the position of the sensors to the drone camera frame.

$$ps_1 = [-a_1 \times \cos(\text{AngleCam2Sensors}) - a_1 \times \cos(\text{AngleCam2Sensors})\, 0] \quad (21)$$

$$ps_2 = [-a_2 \times \cos(\text{AngleCam2Sensors})\, a_2 \times \cos(\text{AngleCam2Sensors})\, 0] \quad (22)$$

$$ps_3 = [a_3 \times \cos(\text{AngleCam2Sensors})\, a_3 \times \cos(\text{AngleCam2Sensors})\, 0] \quad (23)$$

$$ps_4 = [a_4 \times \cos(\text{AngleCam2Sensors}) - a_4 \times \cos(\text{AngleCam2Sensors})\, 0] \quad (24)$$

The Euclidean distance is the distance between each sensor and the origin of the drone, denoted as "o", which is the same as the camera frame. In Equations (21)–(24), "cos" implies the cosine function and "AngleCam2Sensor" is assumed to be 45°. To calculate the plane equation, it is essential to first project the plane normal vector, as in the following equations:

$$\hat{n} = (pst_1 - pst_2) \times (pst_2 - pst_3) \quad (25)$$

$$pst = [\ ps_{xy} \quad d\ _i] \quad (26)$$

where $\hat{n}$, pst_i, and ps_{xy} are the plane normal vector, the Cartesian coordinate of the touching point, and the sensor coordinate in the camera frame, respectively. Equation (27) demonstrates the descending surface planar equation:

$$P_{zI} = \frac{(d - \hat{n}_1 \cdot P_{XI} - \hat{n}_2 \cdot P_{YI})}{\hat{n}_3} \tag{27}$$

in which P_{XI}, P_{YI}, P_{zI}, and $\hat{n}_i$, where i implies the range sensor number, are the Cartesian coordinates of the points on the landing surface presented in the frame of the camera. The variable "d" can be calculated by averaging all the sensors' values. Figure 17 indicates the normal vector of the descending platform, which is time-dependent.

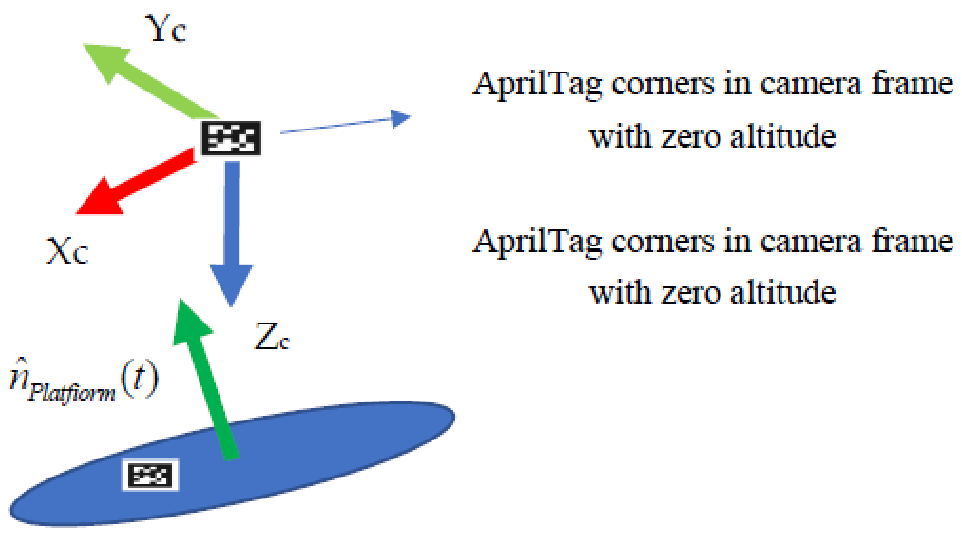

Yc
AprilTag corners in camera frame with zero altitude
AprilTag corners in camera frame with zero altitude
Xc
Zc
$\hat{n}_{Platform}(t)$

Figure 17. Landing platform normal vector expressed as a function of time.

2.5. AprilTag Corner Detection

The AprilTag detection algorithm (ATDA) was followed by the MATLAB built-in function. The algorithm was employed to obtain four corners of the descending landmark. The scope of the current work is beyond the ATDA process, and any set of corners can be used for pose estimation. The detection process can be itemized as listed below. For more information, please refer to reference [49].

Corner detection process:

- Detect the line segments;
- Quad detection;
- Homography and extrinsic estimation; and
- Payload decoding.

2.6. Kalman Filter Implementation

One more step, Kalman filtering, must be considered to track the corner points found by the corner detection process. This filter cannot track the corner points of the AprilTag for a long time-span occlusion, due to the highly complex motion model of unstable landing platforms such as the Stewart table used here. Hence, tracking algorithms including Kalman filtering or even Lucas Kanade point tracking for short-period occlusions are suggested.

There are two main equations for Kalman filters that are well known in navigation systems:

$$x(k) = Ax(k-1) + Bu(k-1) + w(k-1)\ \boldsymbol{w}_{k-1} \sim \mathcal{N}(0,Q) \tag{28}$$

$$z(k) = Hx(k) + v(k)\ \boldsymbol{v}_k \sim \mathcal{N}(0,R) \tag{29}$$

These relations are the process, Equation (28), and the measurement model, Equation (29), with $w(k-1)$, $v(k)$, $x(k-1)$, $x(k)$, and $z(k)$ representing the process noise, measurement noise, state at a previous time, the states of the system at the current time, and the measurements at the current time, respectively. A and B are process model coefficients, while H is measurement model coefficient. Two more phases are then executed in recursion over the tracking process, the prediction and update phases. The prediction steps are defined below in Equations (30) and (31):

$$\hat{\boldsymbol{x}}_k^- = F\hat{\boldsymbol{x}}_{k-1}^+ + B\boldsymbol{u}_{k-1} \tag{30}$$

$$P_k^- = FP_{k=1}^+F^1 + Q \tag{31}$$

where Q, P_k^-, $\hat{\boldsymbol{x}}_k^-$, F, and B are the Gaussian covariance and its prediction covariance error state prediction, and the state transition and control input matrices, respectively. The update phase follows the steps outlined in Equations (32)–(35):

$$\widetilde{\boldsymbol{y}}_k = z_k - H\boldsymbol{x}_k \tag{32}$$

$$K_k = P_k^- H^T\left(R + HP_k^- H^T\right)^{-1} \tag{33}$$

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k\widetilde{\boldsymbol{y}} \tag{34}$$

$$P_k^+ = (I - K_kH)P_k^- \tag{35}$$

where $\widetilde{\boldsymbol{y}}_k$, K_k, $\hat{\boldsymbol{x}}_k^+$, P_k^+, and H are the measurement residual, Kalman gain, updated state estimate, updated error covariance measurement matrices, and observation model, respectively. All steps of the Kalman filter, including the prediction and update phase, are repeated during the state estimation process [50,51].

2.7. Landmark World Coordinates Calculation

Using Equation (17), the 2D homogeneous equations of the landmark, here the AprilTag, are as follows:

$$x_1 = f_x\frac{X_1}{Z_1}\; y_1 = f_y\frac{Y_1}{Z_1} \tag{36}$$

$$x_2 = f_x\frac{X_2}{Z_2}\; y_2 = f_y\frac{Y_2}{Z_2} \tag{37}$$

$$x_3 = f_x\frac{X_3}{Z_3}\; y_3 = f_y\frac{Y_3}{Z_3} \tag{38}$$

$$x_4 = f_x\frac{X_4}{Z_4}\; y_4 = f_y\frac{Y_4}{Z_4} \tag{39}$$

where f_x, f_y, x_i, X_i, y_i, and Y_i are the focal length in the x direction, the focal length in the y direction, landmark x in the pixel coordinates, landmark X in the world coordinates, landmark y in the pixel coordinates, and landmark Y in the world coordinates, respectively.

The homogeneous coordinates of the sensors' touching points are represented as follows:

$$Stx_1 = f_{Stx1}\frac{StX_1}{StZ_1}\; Sty_1 = f_{Sty1}\frac{StY_1}{StZ_1} \tag{40}$$

$$Stx_2 = f_{Stx2}\frac{StX_2}{StZ_2}\; Sty_2 = f_{Sty2}\frac{StY_2}{StZ_2} \tag{41}$$

$$Stx_3 = f_{Stx3}\frac{StX_3}{StZ_3}\; Sty_3 = f_{Sty3}\frac{StY_3}{StZ_3} \tag{42}$$

$$Stx_4 = f_{Stx4}\frac{StX_4}{StZ_4}\; Sty_4 = f_{St4}\frac{StY_4}{StZ_4} \tag{43}$$

where f_{Stxi} and f_{Styi} are focal length calculated by the touch points of sensor number i in x and y directions, respectively. In each camera frame, a focal length will be calculated in both the x and y directions. Equation (44) shows the focal length calculations.

$$f = \begin{cases} f_x = \frac{\sum_{i=1}^{4} f_{Stxi}}{4} \\ f_y = \frac{\sum_{i=1}^{4} f_{Styi}}{4} \end{cases} \tag{44}$$

Calculation of the principal point, CP, of the homogeneous coordinates can be obtained using the following equation:

$$CP = \frac{d_1 \cdot Stp_1 + d_2 \cdot Stp_2 + d_3 \cdot Stp_3 + d_4 \cdot Stp_4}{d_1 + d_2 + d_3 + d_4} \tag{45}$$

where d_i and Stp_i are the sensors' measured distance and the corresponding pixel coordinates of the touching sensors. The above equation states that when the points' distances go to infinity, the CP moves toward that point, and when all distances are equal, the CP will be the average of the pixels. The real-world Z coordinates of each landmark point can be calculated using Equations (46)–(49).

$$Z_1 = \frac{d}{\hat{n}_1 X_1 + \hat{n}_2 Y_1 + \hat{n}_3} = \frac{d}{\hat{n}_1 \frac{x_1}{fx} + \hat{n}_2 \frac{y_1}{fy} + \hat{n}_3} \tag{46}$$

$$Z_2 = \frac{d}{\hat{n}_1 X_2 + \hat{n}_2 Y_2 + \hat{n}_3} = \frac{d}{\hat{n}_1 \frac{x_1}{fx} + \hat{n}_2 \frac{y_1}{fy} + \hat{n}_3} \tag{47}$$

$$Z_3 = \frac{d}{\hat{n}_1 X_3 + \hat{n}_2 Y_3 + \hat{n}_3} = \frac{d}{\hat{n}_1 \frac{x_1}{fx} + \hat{n}_2 \frac{y_1}{fy} + \hat{n}_3} \tag{48}$$

$$Z_4 = \frac{d}{\hat{n}_1 X_4 + \hat{n}_2 Y_4 + \hat{n}_3} = \frac{d}{\hat{n}_1 \frac{x_1}{fx} + \hat{n}_2 \frac{y_1}{fy} + \hat{n}_3} \tag{49}$$

$$X_1 = \frac{Z_1 \cdot x_1}{f_x} \; Y_1 = \frac{Z_1 \cdot y_1}{f_y} \tag{50}$$

$$X_2 = \frac{Z_2 \cdot x_2}{f_x} \; Y_2 = \frac{Z_2 \cdot y_2}{f_y} \tag{51}$$

$$X_3 = \frac{Z_3 \cdot x_3}{f_x} \; Y_3 = \frac{Z_3 \cdot y_3}{f_y} \tag{52}$$

$$X_4 = \frac{Z_4 \cdot x_4}{f_x} \; Y_4 = \frac{Z_4 \cdot y_4}{f_y} \tag{53}$$

Equations (46)–(49) demonstrate the landmark Z coordinate calculation in the camera frame, while Equations (50)–(53) demonstrate how the landmarks are calculated in the camera's X and Y coordinates. Figure 18 illustrates the focal length direction and principle point location in an image frame.

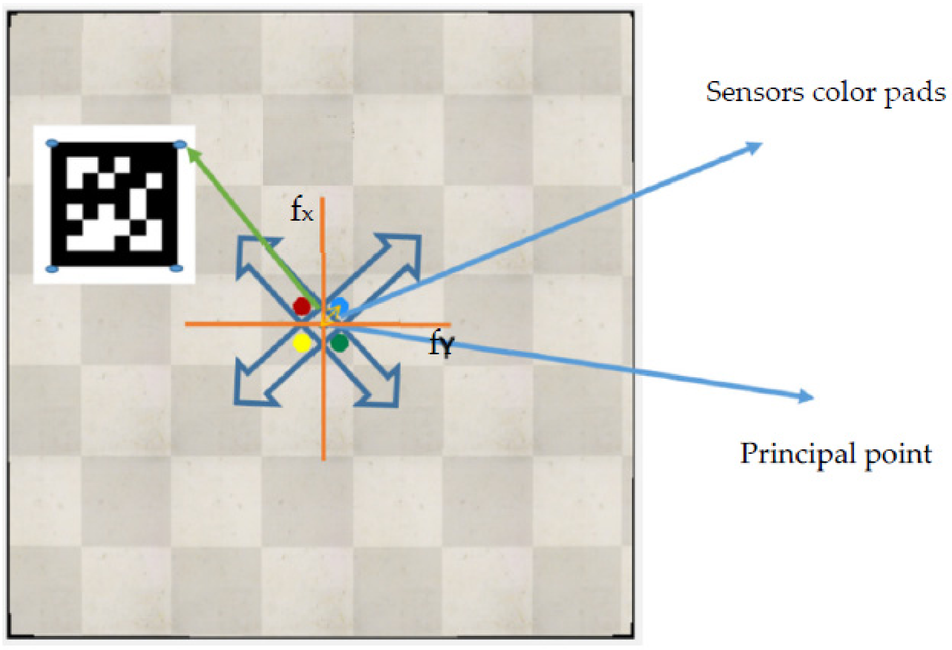


Figure 18. Focal length in x and y directions and homogeneous principal point.

2.8. Rigid Body Transformation Estimation of the AprilTag

Rigid body transformation estimation helps to find the rotation and translation of points in 3D space.

Assume that two sets of P_1 and P_2 points from two objects are known in their real-world coordinates. To calculate the 3D rigid body, the algorithm undergoes the processes given by Equations (54)–(59) [51].

$$C_{P_1} = ave(P_1) \tag{54}$$

$$C_{P_2} = ave(P_2) \tag{55}$$

$$n_{P_1} = C_{P_1} - P_1 \tag{56}$$

$$n_{P_2} = C_{P_2} - P_2 \tag{57}$$

where C_{P_1}, C_{P_2} and n_{P_1}, n_{P_2} are the centroid and normal points, respectively, of the two-point sets. Next,

$$Cov(P_1, P_2) = n_{P_1}^T \times n_{P2} \tag{58}$$

$$\text{Vect}(\text{U}, \text{S}, \text{V}) = svd(C) \tag{59}$$

where $Cov(P_1, P_2)$ and U, S, and V are P_1 and P_2 covariance, left singular, singular, and right singular vectors, respectively, and Vect is the vector representation of the data. C is a rectangular matrix. Next, in the projection step, the rotation matrix is obtained from left and right singular vectors, as follows:

$$R(t) = V \times diag([\ I(1,n) \quad sign(\det(U \times V^T)\ ]) \times U^T \tag{60}$$

where *diag* and det are representations of the diagonal and determinant, respectively, of the matrix, and where n is the size of the number of corresponding points minus one and I is identity matrix. Finally, $T(t)$ is calculated by Equation (61):

$$T(t) = C_{P_2}^T - R(t) \times C_{P_1}^T \tag{61}$$

in which superscript T indicates that the matrix is transposed. The transformation finds the tag between dynamic and static states. The dynamic points used to find the transformations are the current time location of the AprilTag points in the camera frame, and the static points are imaginary AprilTag points that are fixed in the camera frame and at zero altitude. The imaginary static AprilTag points' matrix is expressed as:

$$W^{C}_{Stat}(AP) = \begin{pmatrix} -tagSize/2 & tagSize/2 & 0 \\ tagSize/2 & tagSize/2 & 0 \\ tagSize/2 & -tagSize/2 & 0 \\ tagSize/2 & -tagSize/2 & 0 \end{pmatrix} \tag{62}$$

where the superscript C and subscript $Stat$ denote points in the camera frame and static imaginary points, respectively.

3. Experimental Setups

The simulation time step selected for this work is 50 ms with the physics engine of Bullet, with the version 2.78, and balanced speed. The data transfer protocol used is legacy remote API, which is lightweight and allows the transfer of data in languages including C/C++, Java, Python, MATLAB, and Octave. We utilize MATLAB R2021a, student version, for the computation side and CoppeliSim, EDU version 4.2.0 (rev. 5), for the simulation side.

The tests were performed using two experimental platforms: one static and the other highly dynamic, as shown Figure 19.

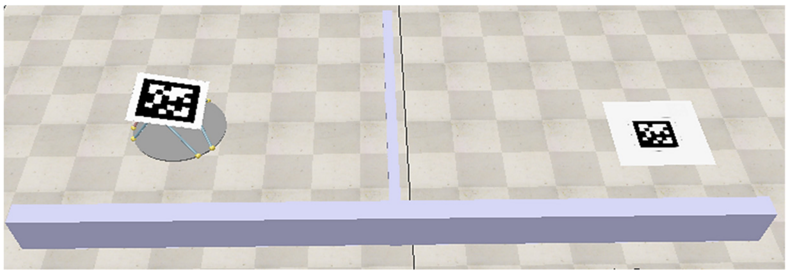


Figure 19. Static and dynamic platform tests bench from right to left.

The proposed method used MATLAB and a state-of-the-art simulator called CoppeliaSim in real-time Software-in-the-Loop (SIL) to prove the viability and practicality of the proffered algorithm. The proposed method used four range sensors and a camera for the calibration and the drone state estimation. The sensors are assumed to be under the propellers and distributed equally. Figure 20 indicates the four ToF sensors and a downward camera in the middle of a UAV, showing their installation on the drone.

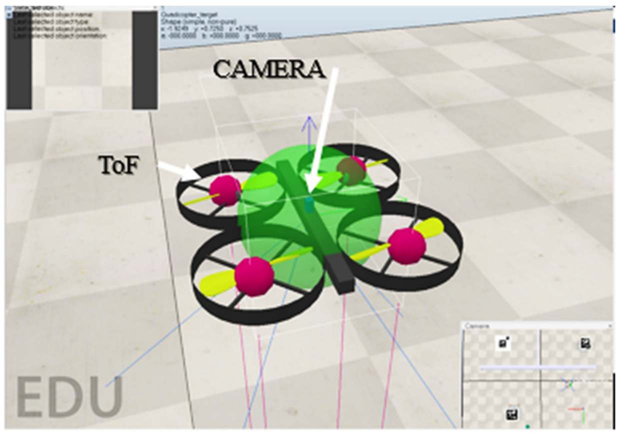


Figure 20. The quadrotor test configuration test.

All four sensors are used to obtain precise camera–Time of Flight (ToF) calibration data; the middle camera was used for the heading calculation in the drone pose estimation.

The landing bench is based on a Stewart platform with relatively highly unstable rotation and translation movement.

Figure 21 shows the landing platform setup with allowable movement, including rotations and translations.

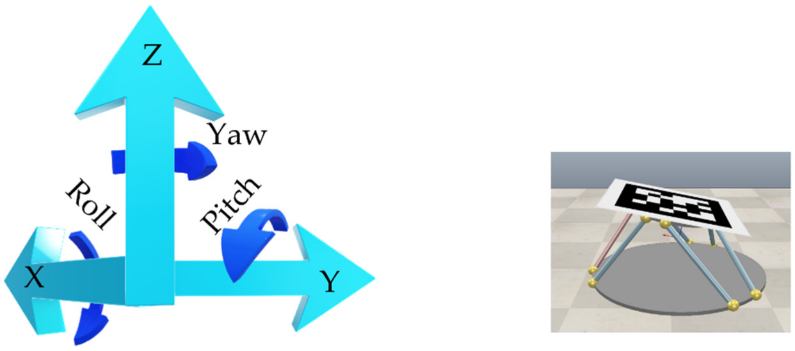


Figure 21. Unstable landing platform and possible directions of movements, imposed on AprilTag.

Before launching into the full algorithm exploration, it is worth mentioning that an important assumption was made about the immobile drone and variable attitude landing platform: during the landing, the drone is located over the platform and the pose is assumed to be as illustrated in Figure 22.

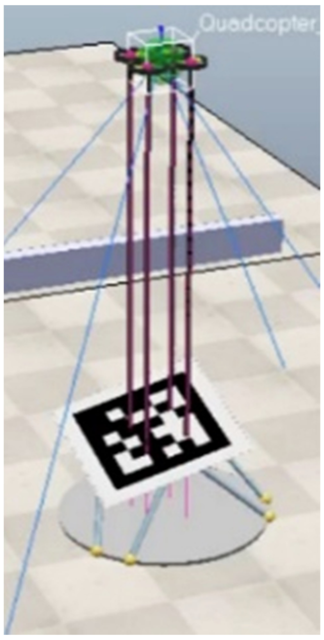


Figure 22. Stable flying quadrotor and variable attitude landing.

4. Results and Discussion

This section describes the system performance for each test platform in terms of the Euler angle and translation. An analysis of the results is then presented, followed by our recommendations for future work.

4.1. Static Platform

For the static platform, the proposed algorithm performed with remarkable accuracy. Figure 23 illustrates corner point coordinate estimation.

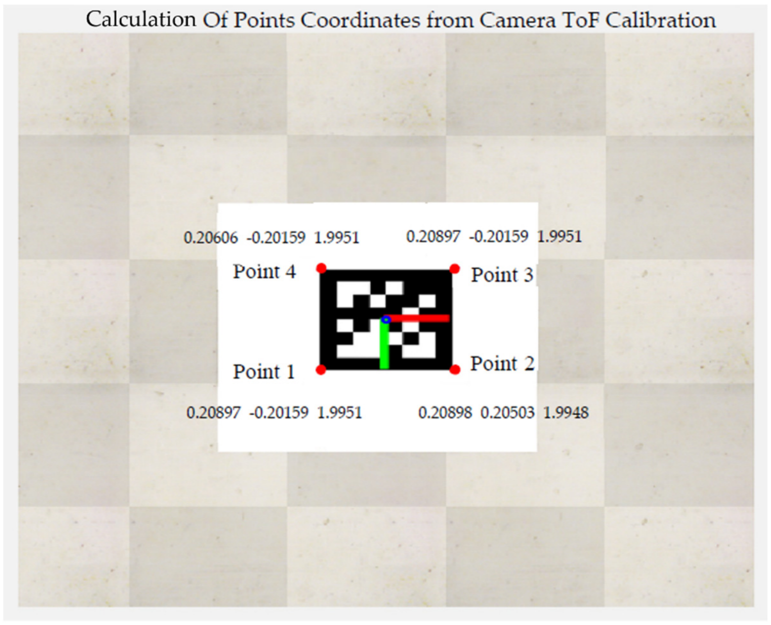


Figure 23. Real-world corner points in a camera frame using CRHF.

The ground trust (GT) for this simple tag had no rotation and translation only in the z-axis with a value of two meters. The results of the developed algorithm are as follows:

$$T^{c}_{Apr} = \begin{bmatrix} 0 & 0 & 1.995 \end{bmatrix} \qquad Rotm^{c}_{Apr} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The only error was posed in the z direction, with an error value of 0.005 m. It was quite a different case for the dynamic platform, as presented next. From the results T^{c}_{Apr} and $Rotm^{c}_{Apr}$ subscript and superscript, it can be inferred that it shows AprilTag's translation and rotation in the camera frame.

4.2. Dynamic Platform

The dynamic platform was tested for 120 iterations of simulations. The translational results are illustrated in Figure 24 in three subplots, showing the translation in the X, Y, and Z directions. In the X direction, in Figure 24, both algorithms followed GT closely and have similar tracking performance. However, regarding the translational error for ATDA and CRHF, indicated in mean absolute error Table 5, the proposing algorithm proved a slightly better performance. CRHF also indicated notable improvement in the Y direction. As it can be inferred from the Z direction subplot of Figure 24, using data from ToF sensors led to a significant amelioration of the pose estimation in the UAV downward camera frame. In Figure 25, the three subplots to assess the performance of the system in angle estimation for pitch, roll, and yaw, which are the rotation around the X-, Y-, and Z-axes, respectively. At 50, 140, and 180 s of simulation, in the Figure 25 pitch estimation subplot, ADTA demonstrated major inaccuracies, while this behavior was compensated to a great extent for the CRHF, having a slight deviation from GT at 140 s. The roll estimation result in the same Figure indicates a better estimation using CRHF technique for the last 70 s, between 130 and 200 s. Both techniques demonstrate an almost identical performance for the yaw calculation, with a negligible improvement by CRHF regarding Table 5.

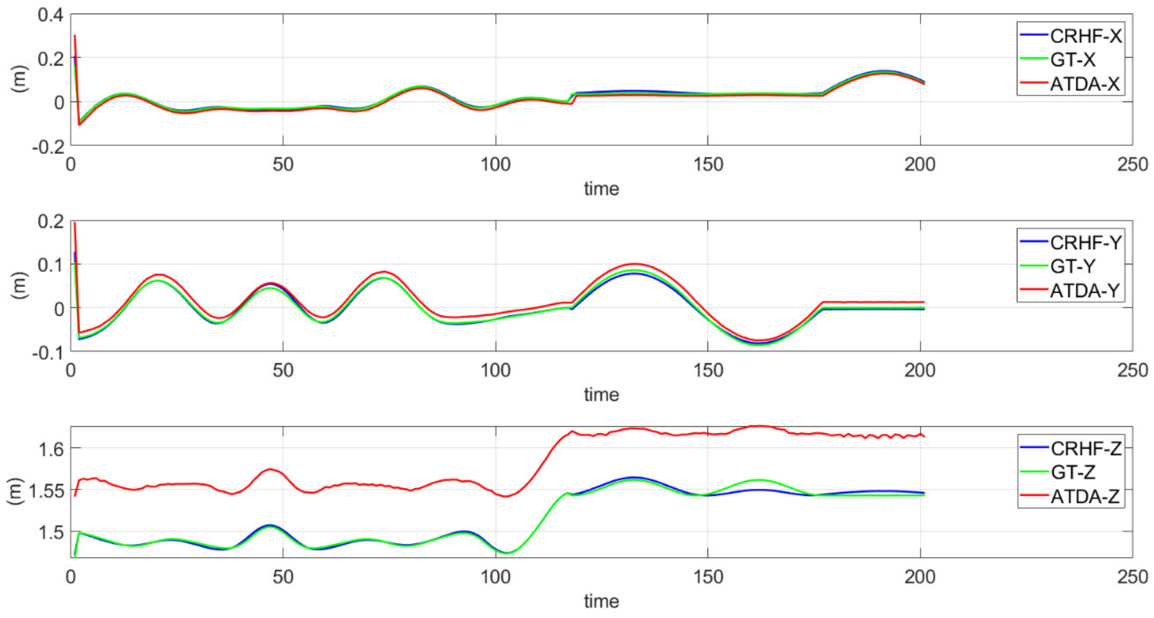


Figure 24. Translations in the X, Y, and Z directions of CRHF.

Table 5. Mean absolute error comparison of CRHF and ATDA algorithms.

Mean Absolute Error		**ATDA**			**CRHF**	
Translations (meters)	0.0157	0.0129	0.0687	0.0047	0.0042	0.0051
Angles (degrees)	3.9043	1.7684	1.7859	1.4239	1.5231	1.3290

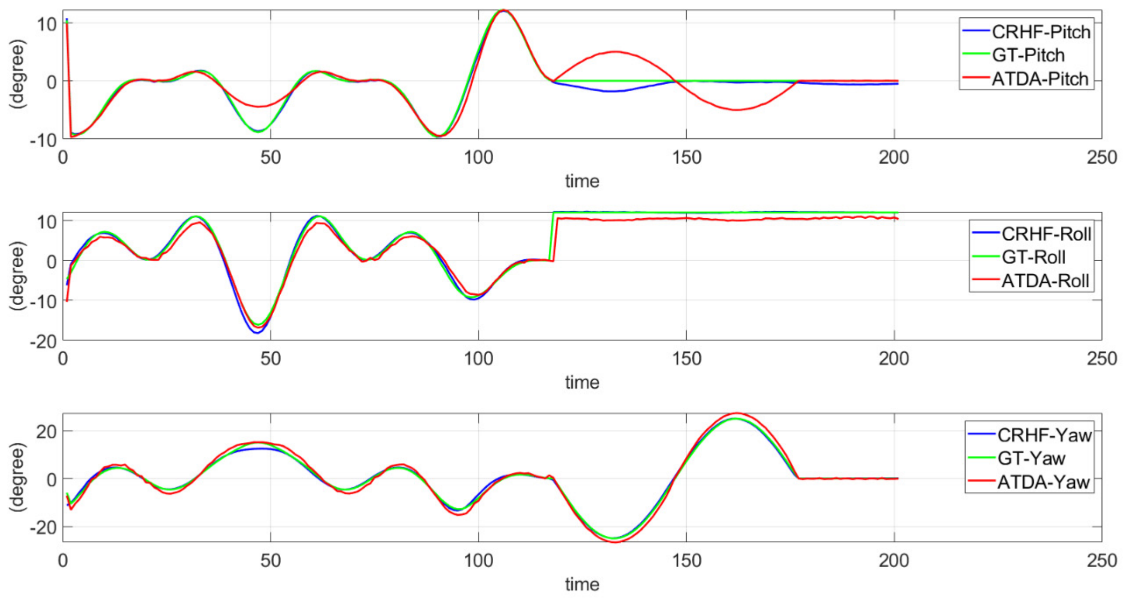
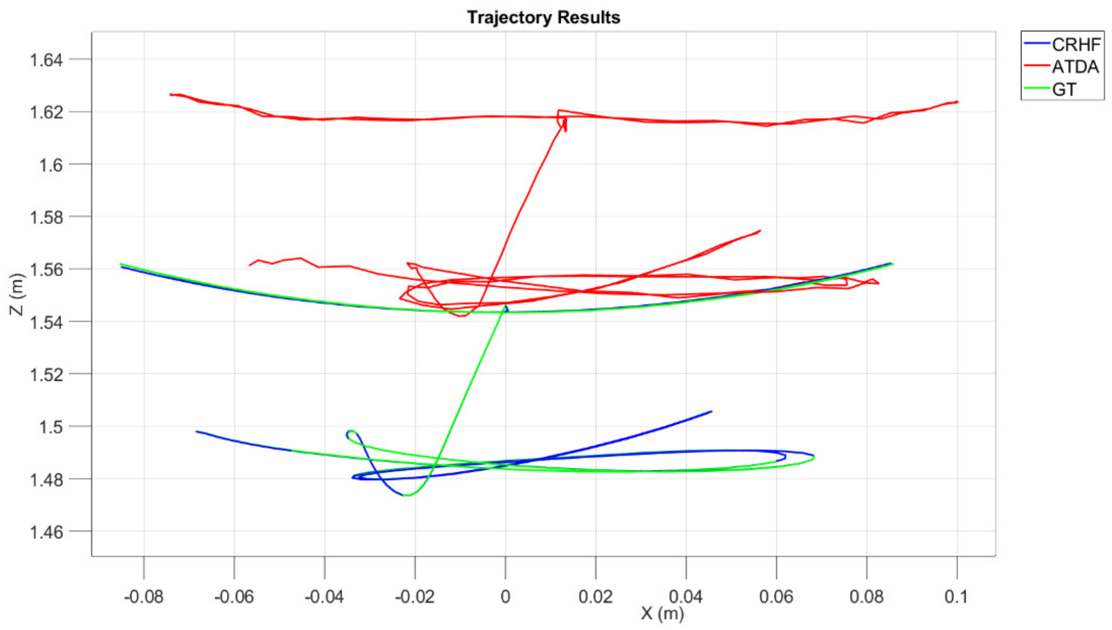
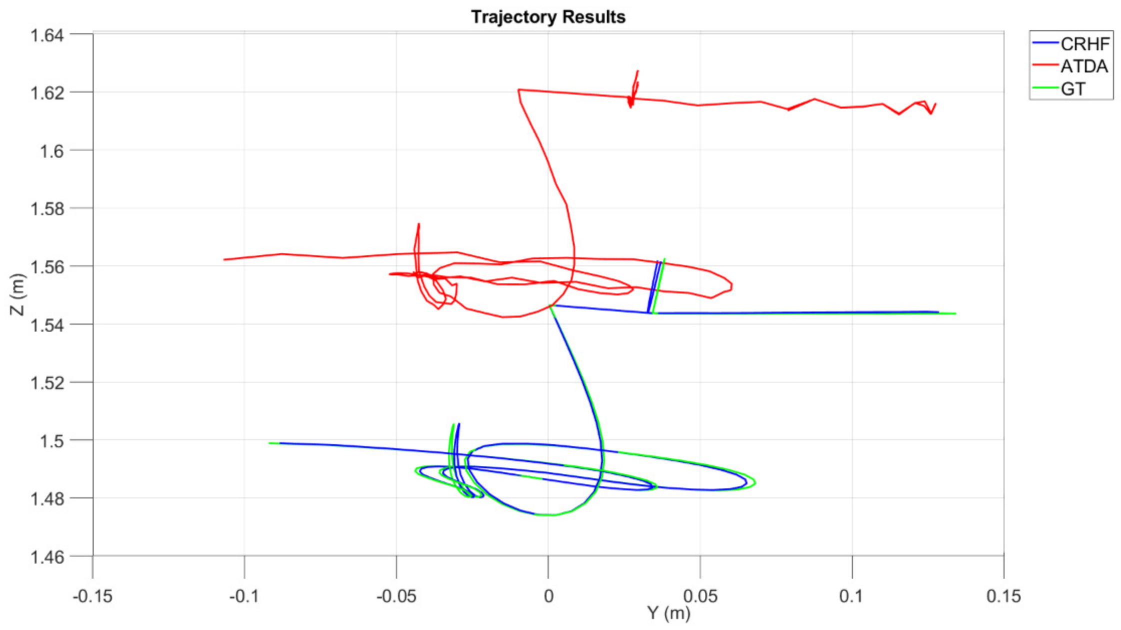


Figure 25. Pitch, roll, and yaw for 120 s simulation.

Finally, the trajectory of the platform motion estimation was calculated for the CRHF, ATDA, and GT. Figure 26 shows the 3D plots of the Stewart table trajectory in Cartesian coordinates of the drone downward camera frame. Figure 26a–d depict a multi-view trajectory plot in the XZ, YZ, XY, and XYZ directions, respectively. As it can be seen in the XZ view in Figure 26a, the proposing system significantly tackles the error in the X and Z directions. Figure 26b illustrates a noticeable improvement of landing platform pose estimation in the Y and Z directions. In bird's-eye view, or drone downward camera view, in X and Y directions depicted in Figure 26c, the plot verifies the better performance of the CRHF compared to the ATDA algorithm. Figure 26d depicts the 3D view of the landing platform movement and provides a better view for the improvement of altitude estimation.

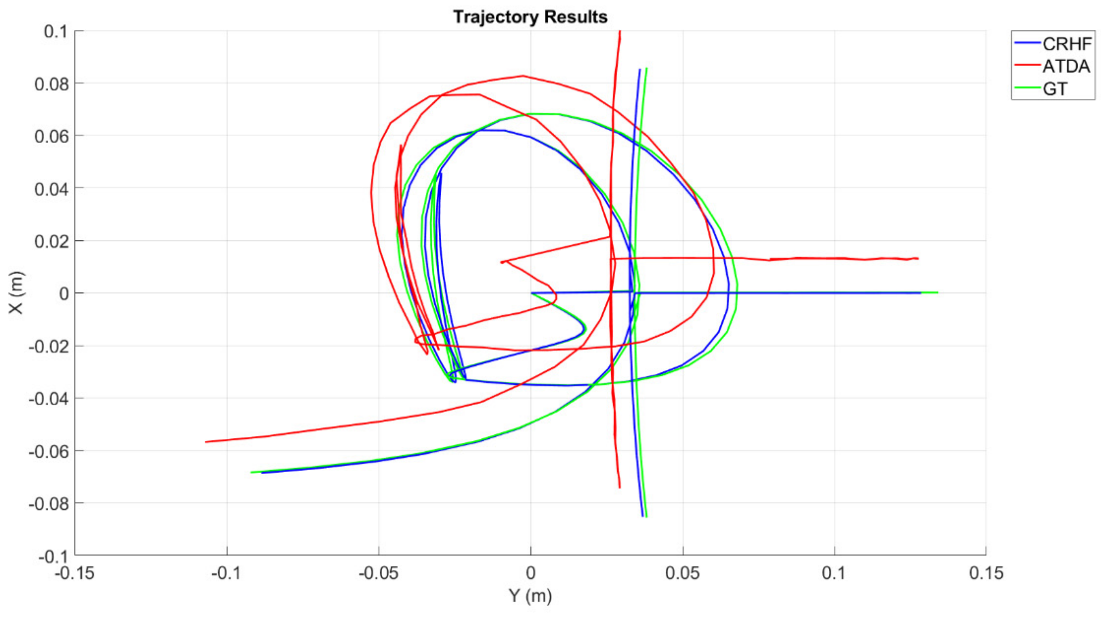


(**a**)

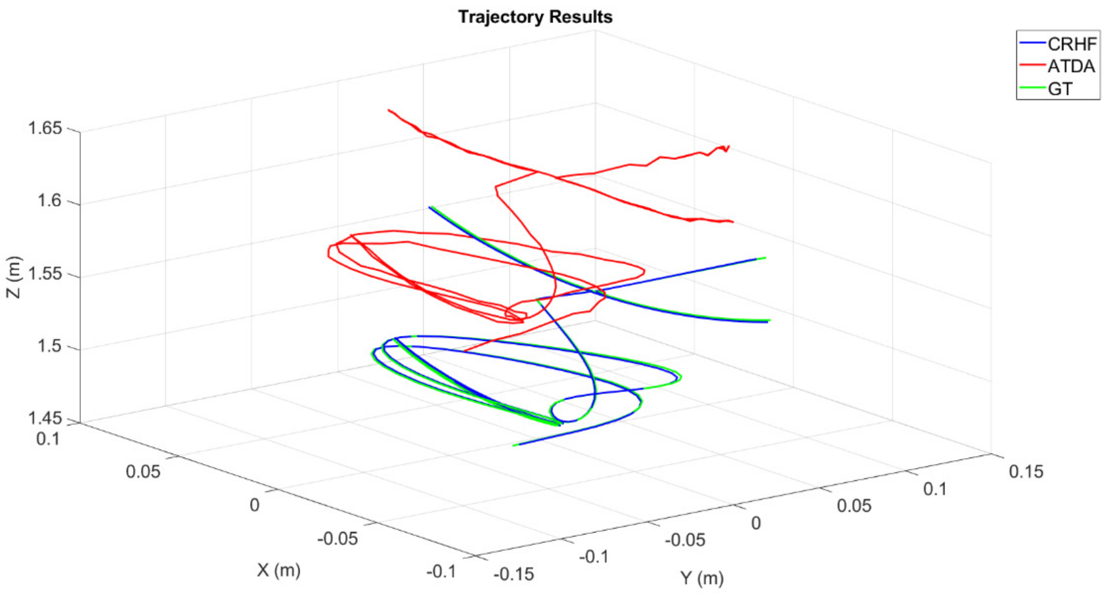


(**b**)

Figure 26. *Cont.*



(**c**)



(**d**)

Figure 26. Trajectory plots for pose estimation for the camera, CRHF, and GT-XZ view (**a**), YZ view (**b**), XY view (**c**), and XYZ view (**d**).

CRHF transitional errors for X, Y, and Z directions, in meters, are demonstrated in Figure 27. The highest error happens at 170 s and belongs to altitude estimation of the table. While the error was lower for the Z direction at 140 s, the X and Y directions demonstrate the highest errors. During the rest of the periods, the error stays less than 0.01 m, showing better and stable altitude estimation.

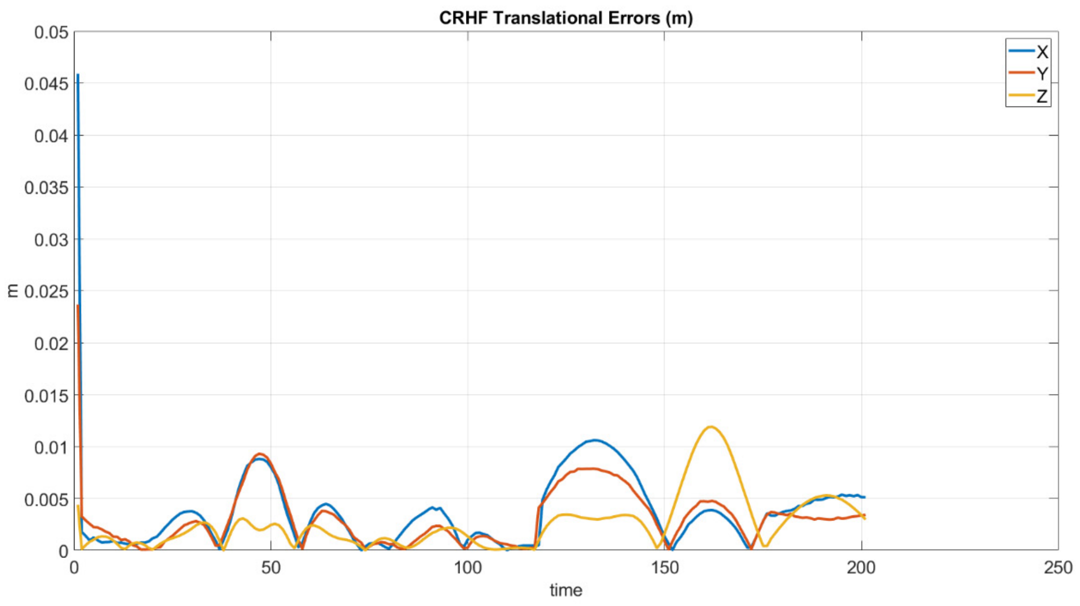


Figure 27. CRHF translational absolute errors.

Figure 28 plots the Euler angle estimation error of the CRHF for pitch, roll, and yaw in degrees. The highest error belongs to heading estimation with around 2.5 degrees and happened at 52 s. The second highest is for roll error, which peaked at around 2.3 at 50 s. However, disregarding the 0 s error, the plot demonstrates errors of less than 2 degrees in the remaining time.

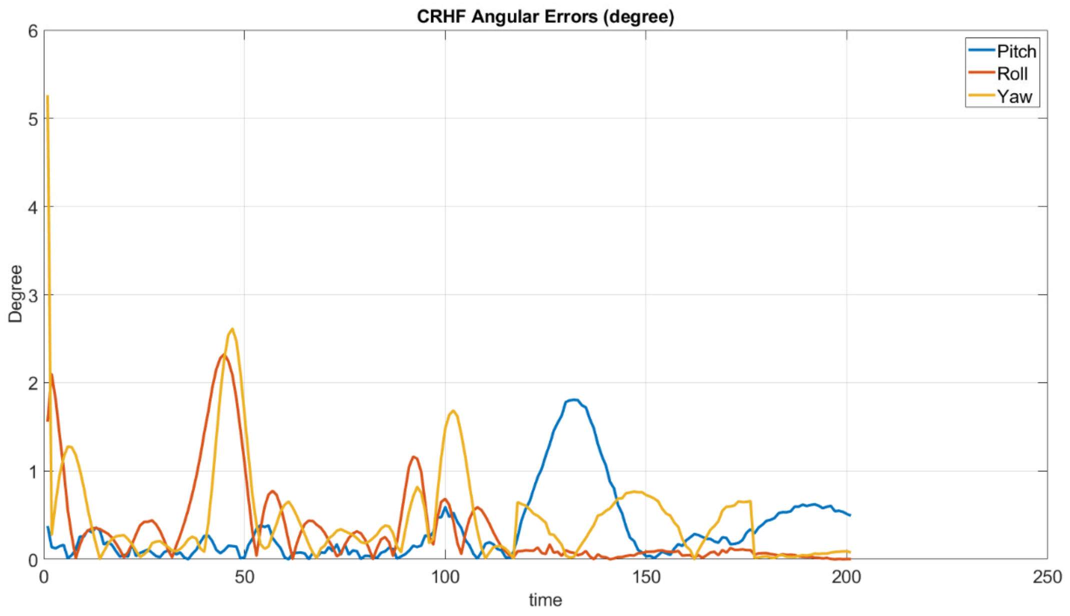


Figure 28. CRHF angular absolute errors.

Figure 29 provides the error plot of ATDA Euler angle estimation for the Stewart table. At 140 and 170 s, roll estimation error peaked toward 5 degrees and shows the most dramatic value. The next highest error was at 52 s with around 4.2 degrees. During the rest of the time, ATDA demonstrated error lower than 2.8 degrees, higher than that of CRHF's angular error value. Figure 30 shows ATDA algorithm translational errors, highlighting significant attitude estimation error. However, ATDA demonstrated errors of less than 0.02 in the X and Y directions.

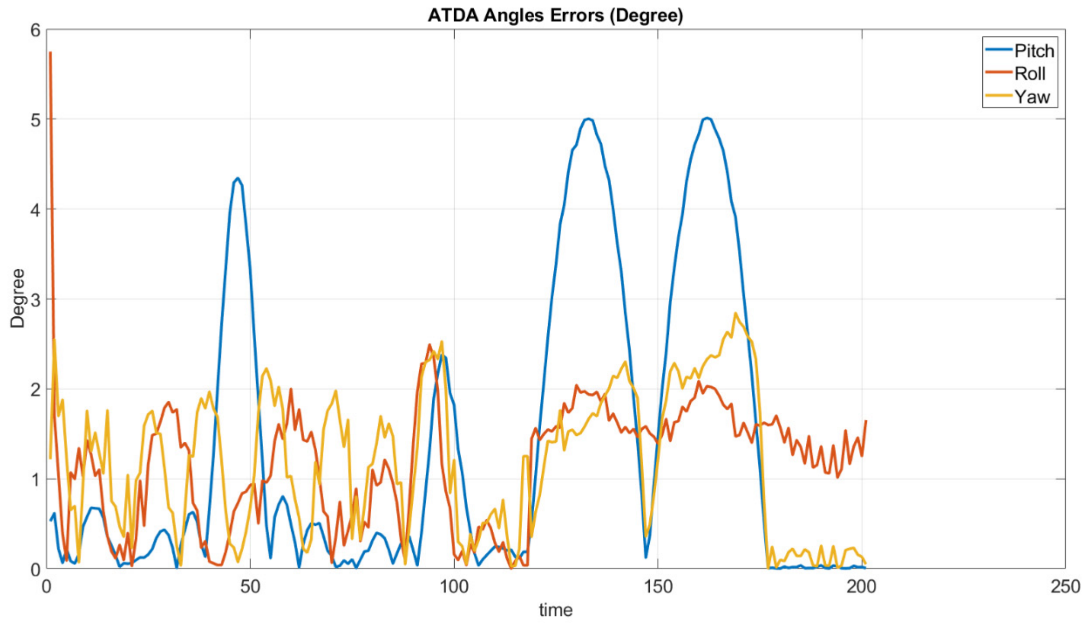


Figure 29. ATDA angular estimation errors.

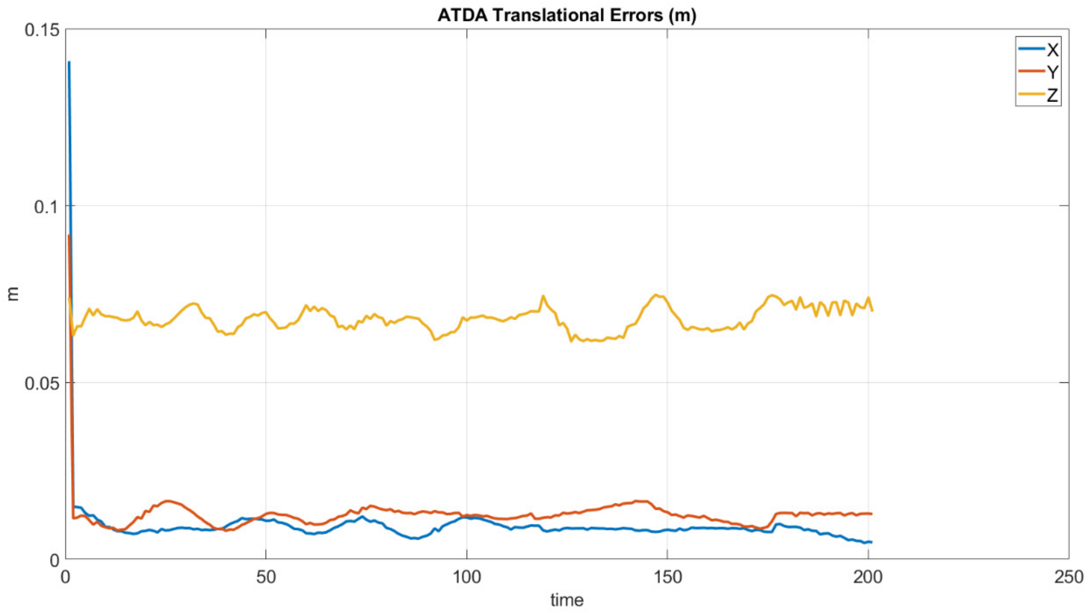


Figure 30. ATDA algorithm translational errors.

The averages of these errors over the whole simulation period are summarized in Table 5 for both algorithms. It is readily evident from the table that the proposing algorithm strikingly ameliorated the altitude estimation in the Z direction. In addition, Table 5 demonstrates a significant improvement in the X and Y directions. The Euler angles estimation summary for the landing table reveals a significant accuracy of pitch estimation and slightly lower error for the yaw and roll.

5. Conclusions and Future Work

The results of this work reveal a notable improvement in translational as well as angular pose estimation compared to the AprilTag navigation algorithm, but with the expense of adding range sensors. The main motivation for developing such a system development is the problem of highly unstable ship deck landing platforms. Their instability makes the landing of emergency medical services (EMS) and heavyweight drones quite problematic.

The results achieved here suggest that reducing translational error could be possible via alternative pose estimation techniques such as projective geometry. The current research addresses the issue of relatively unstable UAV landings. The proposed method considered the sensor fusion algorithm based on range sensors and a camera to address the accuracy of the pose estimation for the unstable platform using CRHF. The fusion technique uses a calibration method that makes sensor touch points available for calculation of camera focal lengths of the camera. Then, these focal lengths are employed to calculate the world coordinates of the landmark corner points. In the final stage, using rigid body transformation, the pose of the AprilTag is obtained in the frame of the drone downward camera. The research also considers static and dynamic landing platforms to assess the developed technique. The findings from the current work verify a better accuracy for pose estimation in rotational angles of roll, pitch, and yaw and translational estimations in the X, Y, and Z directions for the landing platform when compared to ATDA.

Different cameras, including fisheye and monocular cameras, have various radial and tangential distortion models. In our future work, we would like to evaluate the possibility of camera radial distortion model calculation through range sensor data to lower the projection errors. Next, an autonomous UAV design considers two phases for landing system development. These are rendezvous, when the UAV moves toward the landing point, and final landing, which we considered in our research. Further study is required to address the pose estimation for the rendezvous phase of the flight using sensor fusion by different sensors such as IMU and camera. The concept of artificial intelligence must not be neglected in machine vision algorithm developments. Neural networks, for example, have significant potential for black box modeling, which does not use mathematical models; hence, it is highly advisable to use it for the tasks including landing surface detection and roughness measurement of the landing area. Finally, our next targeted future work is to use reinforcement learning technique to teach a drone to conduct landings in more complicated scenarios, such as landing on a rover that is moving in a complex path.

Author Contributions: Conceptualization, M.S.; Investigation, M.S.; Methodology, M.S.; Software, M.S.; Supervision, R.L.J.; Validation, M.S.; Visualization, M.S.; Writing—original draft, M.S.; Writing—review & editing, M.S. Also, all authors have contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to acknowledge the Lassena lab and the Ecole de Technology Superieure for making resources available for conducting this research. In addition, we would like to acknowledge the contribution of Farnoush Falahatraftar for the first revision of this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wubben, J.; Fabra, F.; Calafate, C.T.; Krzeszowski, T.; Marquez-Barja, J.M.; Cano, J.-C.; Manzoni, P. Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition. *Electronics* **2019**, *8*, 1532. [CrossRef]
2. Yang, T.; Li, P.; Zhang, H.; Li, J.; Li, Z. Monocular Vision SLAM-Based UAV Autonomous Landing in Emergencies and Unknown Environments. *Electronics* **2018**, *7*, 73. [CrossRef]
3. Lin, S.; Jin, L.; Chen, Z. Real-Time Monocular Vision System for UAV Autonomous Landing in Outdoor Low-Illumination Environments. *Sensors* **2021**, *21*, 6226. [CrossRef] [PubMed]
4. Lee, D.-K.; Nedelkov, F.; Akos, D.M. Assessment of Android Network Positioning as an Alternative Source of Navigation for Drone Operations. *Drones* **2022**, *6*, 35. [CrossRef]
5. Shi, G.; Shi, X.; O'Connell, M.; Yu, R.; Azizzadenesheli, K.; Anandkumar, A.; Yue, Y.; Chung, S.-J. Neural Lander: Stable Drone Landing Control Using Learned Dynamics. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.

6. Nguyen, P.H.; Arsalan, M.; Koo, J.H.; Naqvi, R.A.; Truong, N.Q.; Park, K.R. LightDenseYOLO: A Fast and Accurate Marker Tracker for Autonomous UAV Landing by Visible Light Camera Sensor on Drone. *Sensors* **2018**, *18*, 1703. [CrossRef]
7. Truong, N.Q.; Lee, Y.W.; Owais, M.; Nguyen, D.T.; Batchuluun, G.; Pham, T.D.; Park, K.R. SlimDeblurGAN-Based Motion Deblurring and Marker Detection for Autonomous Drone Landing. *Sensors* **2020**, *20*, 3918. [CrossRef] [PubMed]
8. Hamanaka, M.; Nakano, F. Surface-Condition Detection System of Drone-Landing Space using Ultrasonic Waves and Deep Learning. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020.
9. Castellano, G.; Castiello, C.; Mencar, C.; Vessio, G. Crowd Detection for Drone Safe Landing through Fully-Convolutional Neural Networks. In *SOFSEM 2020: Theory and Practice of Computer Science*; Springer: Cham, Switzerland, 2020; pp. 301–312.
10. Jung, S.; Hwang, S.; Shin, H.; Shim, D.H. Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2539–2544. [CrossRef]
11. Chang, C.-C.; Tsai, J.; Lu, P.-C.; Lai, C.-A. Accuracy Improvement of Autonomous Straight Take-off, Flying Forward, and Landing of a Drone with Deep Reinforcement Learning. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 914. [CrossRef]
12. Polvara, R.; Patacchiola, M.; Sharma, S.; Wan, J.; Manning, A.; Sutton, R.; Cangelosi, A. Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 115–123.
13. Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; de la Puente, P.; Campoy, P. A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform. *J. Intell. Robot. Syst.* **2019**, *93*, 351–366. [CrossRef]
14. Song, Y.; Steinweg, M.; Kaufmann, E.; Scaramuzza, D. Autonomous Drone Racing with Deep Reinforcement Learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021.
15. Santos, N.P.; Lobo, V.; Bernardino, A. Two-stage 3D model-based UAV pose estimation: A comparison of methods for optimization. *J. Field Robot.* **2020**, *37*, 580–605. [CrossRef]
16. Khalaf-Allah, M. Particle Filtering for Three-Dimensional TDoA-Based Positioning Using Four Anchor Nodes. *Sensors* **2020**, *20*, 4516. [CrossRef] [PubMed]
17. Kim, J.; Kim, S. Autonomous-flight Drone Algorithm use Computer vision and GPS. *IEMEK J. Embed. Syst. Appl.* **2016**, *11*, 193–200. [CrossRef]
18. Khithov, V.; Petrov, A.; Tishchenko, I.; Yakovlev, K. Toward Autonomous UAV Landing Based on Infrared Beacons and Particle Filtering. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2016; pp. 529–537.
19. Fernandes, A.; Baptista, M.; Fernandes, L.; Chaves, P. Drone, Aircraft and Bird Identification in Video Images Using Object Tracking and Residual Neural Networks. In Proceedings of the 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 27–29 June 2019; pp. 1–6.
20. Miranda, V.R.; Rezende, A.; Rocha, T.L.; Azpúrua, H.; Pimenta, L.C.; Freitas, G.M. Autonomous Navigation System for a Delivery Drone. *arXiv* **2021**, arXiv:2106.08878. [CrossRef]
21. Benini, A.; Mancini, A.; Longhi, S. An IMU/UWB/Vision-based Extended Kalman Filter for Mini-UAV Localization in Indoor Environment using 802.15.4a Wireless Sensor Network. *J. Intell. Robot. Syst.* **2013**, *70*, 461–476. [CrossRef]
22. St-Pierre, M.; Gingras, D. Comparison between the unscented kalman filter and the extended kalman filter for the position estimation module of an integrated navigation information system. In Proceedings of the IEEE Intelligent Vehicles Symposium, Parma, Italy, 14–17 June 2004. [CrossRef]
23. Raja, G.; Suresh, S.; Anbalagan, S.; Ganapathisubramaniyan, A.; Kumar, N. PFIN: An Efficient Particle Filter-Based Indoor Navigation Framework for UAVs. *IEEE Trans. Veh. Technol.* **2021**, *70*, 4984–4992. [CrossRef]
24. Kraus, K. *Photogrammetry: Geometry from Images and Laser Scans*; Walter De Gruyter: Berlin, Germany; New York, NY, USA, 2007.
25. Gruen, A.; Huang, T.S. *Calibration and Orientation of Cameras in Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2021. Available online: https://www.springer.com/gp/book/9783540652830 (accessed on 29 September 2021).
26. Luhmann, T.; Robson, S.; Kyle, S.; Boehm, J. *Close-Range Photogrammetry and 3D Imaging*; De Gruyter: Berlin, Germany, 2019.
27. El-Ashmawy, K. Using Direct Linear Transformation (DLT) Method for Aerial Photogrammetry Applications. ResearchGate, 16 October 2018. Available online: https://www.researchgate.net/publication/328351618_Using_direct_linear_transformation_DLT_method_for_aerial_photogrammetry_applications (accessed on 29 September 2021).
28. Sarris, N.; Strintzis, M.G. *3D Modeling and Animation: Synthesis and Analysis Techniques for the Human Body*; Irm Press: Hershey, PA, USA, 2005.
29. Aati, S.; Avouac, J.-P. Optimization of Optical Image Geometric Modeling, Application to Topography Extraction and Topographic Change Measurements Using PlanetScope and SkySat Imagery. *Remote Sens.* **2020**, *12*, 3418. [CrossRef]
30. Morales, L.P. Omnidirectional Multi-View Systems: Calibration, Features and 3D Information. Dialnet. 2011. Available online: https://dialnet.unirioja.es/servlet/dctes?info=link&codigo=101094&orden=1 (accessed on 29 September 2021).
31. Panoramic Vision, Panoramic Vision—Sensors, Theory, and Applications | Ryad Benosman | Springer. 2021. Available online: https://www.springer.com/gp/book/9780387951119 (accessed on 29 September 2021).
32. Omnidirectional Vision Systems. Omnidirectional Vision Systems—Calibration, Feature Extraction and 3D Information | Luis Puig | Springer. Springer.com. 2013. Available online: https://www.springer.com/gp/book/9781447149460 (accessed on 29 September 2021).

33. Faugeras, O.D.; Luong, Q.; Papadopoulo, T. *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some*; MIT Press: Cambridge, MA, USA, 2001.
34. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
35. Heikkila, J.; Silven, O. A four-step camera calibration procedure with implicit image correction. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 1106–1112.
36. Learning OpenCV. *Learning OpenCV*; O'Reilly Online Learning; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2021.
37. Wang, J.; Shi, F.; Zhang, J.; Liu, Y. A new calibration model of camera lens distortion. *Pattern Recognit.* **2008**, *41*, 607–615. [CrossRef]
38. Poynton, C. *Digital Video and HD: Algorithms and Interfaces*; Morgan Kaufmann: Amsterdam, The Netherlands, 2012.
39. MathWorks, Inc. *Image Processing Toolbox: User's Guide*; MathWorks, Inc.: Natick, MA, USA, 2016.
40. Lens Calibration (Using Chessboard Pattern) in Metashape. 2021. Retrieved 27 January 2022. Available online: https://agisoft.freshdesk.com/support/solutions/articles/31000160059-lens-calibration-using-chessboard-pattern-in-metashape (accessed on 29 September 2021).
41. Liang, Y. Salient Object Detection with Convex Hull Overlap. *arXiv* **2016**, arXiv:1612.03284.
42. Lin, S.; Garratt, M.A.; Lambert, A.J. Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment. *Auton. Robot.* **2017**, *41*, 881–901. [CrossRef]
43. Yadav, A.; Yadav, P. *Digital Image Processing*; University Science Press: New Delhi, India, 2021.
44. Arthur, D.; Vassilvitskii, S. K-means++: The Advantages of Careful Seeding. Available online: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf (accessed on 29 September 2021).
45. Corke, P. *Robotics, Vision and Control—Fundamental Algorithms In MATLAB*®, 2nd ed.; Springer: Berlin, Germany, 2017.
46. Fritsch, F.N.; Butland, J. A Method for Constructing Local Monotone Piecewise Cubic Interpolants. *SIAM J. Sci. Stat. Comput.* **1984**, *5*, 300–304. [CrossRef]
47. Moler, C.B. *Numerical Computing with MATLAB*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2004.
48. Olson, E. AprilTag: A robust and flexible visual fiducial system. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3400–3407.
49. Welch, G.; Bishop, G. *An Introduction to the Kalman Filter*; TR 95—041; University of North Carolina at Chapel Hill, Department of Computer Science: Chapel Hill, NC, USA, 1995.
50. Blackman, S.S. *Multiple-Target Tracking with Radar Applications*; Artech House, Inc.: Norwood, MA, USA, 1986; p. 93.
51. Arun, K.S.; Huang, T.S.; Blostein, S.D. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. Pattern Anal. Mach. Intell.* **1987**, *9*, 698–700. [CrossRef] [PubMed]