Research Article

Fiducial Markers Aided Position Estimation for Vertical Landing

Recep Behlul Sahin[1*], Dr. Muharrem Mercimek[2]

[1] Depart. of Control and Automation Engineering
Yildiz Technical University
Istanbul, Turkey, 0000-0001-5833-9515, rcpbehlul@gmail.com,
[2] Depart. of Control and Automation Engineering
Yildiz Technical University
Istanbul, Turkey, 0000-0001-8737-298X mercimek@yildiz.edu.tr

(First received February 10, 2023 and in final form May 11, 2023)

Reference: Sahin, R., B., Mercimek, M. Fiducial Markers Aided Position Estimation for Vertical Landing. The European Journal of Research and Development,3(2), 29-45.

Abstract

Autonomous landing is a critical step in unmanned aerial vehicles (UAVs) and requires accurate position information. In cases where GPS signals are unavailable or obstructed, vision-based approaches can provide support for landing capabilities. In this study, a vision-based position estimation algorithm is being developed in conjunction with markers used in vertical take-off and landing (VTOL) systems of UAVs. The developed framework is designed to be compatible with various types of visual markers. The Kalman Filter is used to the calculated position to correct measurement errors and reduce the uncertainty of the position estimation. The developed algorithm is extensively tested in a simulation environment. The positions of a quadrotor aircraft are compared with real measurements to analyze the performance of the proposed vision-based position estimation algorithm. The results demonstrate an acceptable level of accuracy for the algorithm. This study discusses the potential of using visual markers and integrating Kalman filtering to improve the accuracy of positioning in vertical takeoff and landing systems of the UAVs. The development of a vision-based position estimation algorithm can enhance the reliability and precision of autonomous landing capabilities and enable successful landings in situations where GPS signals are limited or unavailable.

Keywords: UAV, Computer Vision, Fiducial Markers, Kalman, Autonomous Landing, VTOL

1. Introduction

UAVs are used in military and civilian applications such as reconnaissance, surveillance, intelligence, search, and rescue. UAV systems are produced in different types as fixed and rotary wings and are equipped with advanced camera systems [1]. However, in cases where there is no GPS or there is jamming, UAV systems may lose their autonomous mission capability. In the absence of GPS during landing or take-off, the risk of uncontrollability increases because the position estimation cannot be made. Therefore, beacon or image-based positioning systems can be used as alternative solutions. [2,3]

Autonomous landing is one of the most important problems in UAV systems. Navigation equipment and flight control systems play a crucial role in providing the necessary position information and controllers for the landing task. Other sensors such as light detection and distance sensors such as lidar, ultrasonic, and IR sensors can also be used for the mentioned problem, but these sensors may not be sufficient in real-time outdoor applications [4]. Computer vision studies which can detect the landing target using cameras offer an effective solution. In addition, some UAVs can land on moving targets with coordinated communication between the UAV and the moving target [5,6].

In this study, a computer vision algorithm is developed to perform a vision-based autonomous landing of a UAV. An algorithm that can use visual fiducial systems has been developed to determine the position of the UAV and make estimations. The markers or tags placed on stationary objects were detected and followed by the algorithm that processes the images from the camera on the UAV.

The developed position estimation algorithm is designed to be capable of working with various types of visual references, providing versatility and flexibility to the study. This enables the algorithm to concurrently operate with multiple tags from popular visual reference systems, expanding its applicability. The autopilot is designed to operate in a structure that can be integrated into the ROS environment. Additionally, it is designed as a system capable of working with an auxiliary sensor for autonomous landing missions [7].

2. Materials and Methods

2.1. Visual Fiducial Marker System

Fiducial markers are reference objects frequently used in image processing and computer vision applications. The camera or sensor can detect and track these reference objects in real time by utilizing their specific properties and characteristics. The markers, also known as tags, are specific labels with predefined geometric shapes. These tags, when viewed by a camera or sensor, are used to determine the position and orientation of the object using features on the tag [8].

They usually have two key features: a unique identification number and a locator. The ID number represents a predefined number of the object, thus enabling different markers to be distinguished from one another. A locator is a unique geometric pattern that helps determine the position of the object. This pattern usually consists of geometric shapes such as lines, circles, or polygons.

The operation of visual markers consists of a two-stage process: detection and tracking. In the detection phase, the camera or sensor uses improved algorithms to detect markers in the image. These algorithms help determine the location and orientation of the object by identifying the unique identification number and locator characteristics of the markers.

In the tracking phase, the location and direction of the markers are tracked in real time. This tracking is performed considering the movements of the object and keeps updating the position and direction of the marker continuously.

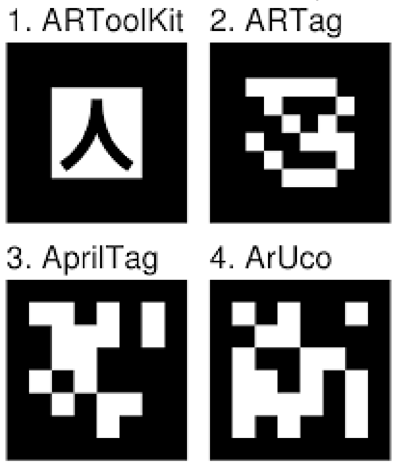

Figure 1 An Example of Marker Formats from ARTag, ArUco, ARToolKit, and AprilTag

Fiducial markers have many application areas. For example, they are used in many fields such as robotics, augmented or virtual reality, and air vehicle navigation. These markers are an important tool for precise positioning of objects and tracking their movements and have been used successfully in many applications. Some of the most popular tag types are ARTag, ArUco, ARToolKit, AprilTag, and CALTag. An example of marker formats from these types is shown in Figure 1.

2.2. Proposed Method

Visual fiducial marker systems have methods for label detection and identification, orientation, and position calculation in their libraries. These methods work in their own tag systems. The algorithm developed in this study works with any fiducial marker system and can perform detection and pose calculations. For this method, the 8x8 binary information and the edge lengths of the tag to are sufficient. Some filter coefficients and the camera matrix are adjusted to be parametric to work in different light conditions and different cameras. The steps of the algorithm which is developed for this work are shown in Figure 2.

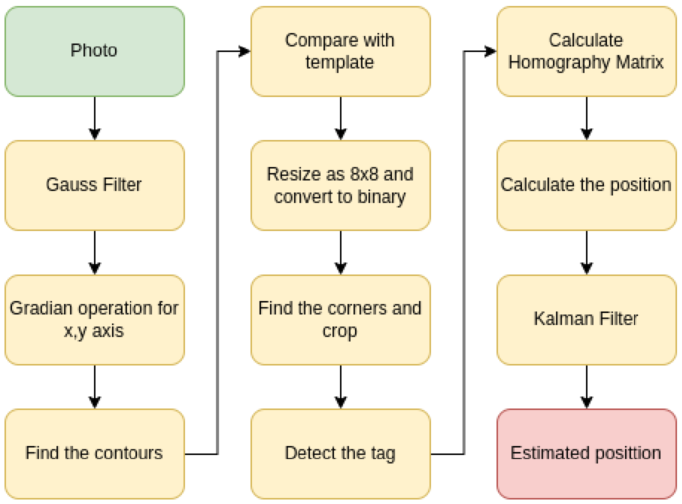

Figure 2 The Algorithm Diagram

The input photo is converted to a gray image and normalization is applied. Then gradient is applied on the x and y axis in Equation 1. 3x3 kernel matrix is used in the gradient operation. While applying the gradient process, filters were applied to control the line continuity in the x and y axis to detect the label more clearly. Thus, more clear results are obtained when detecting corner points. Gradient size is calculated in Equation 2 and 3.

Physical events such as different lighting conditions, shadowing, vibration, and shaking bring noise to the image. The noise can cause unacceptable results in these object detection-based applications. For this reason, gaussian blur is applied to the gray image to eliminate noise.

$$G_{mag} = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$f_G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-a)^2}{2\cdot\sigma^2}} \tag{2}$$

$$f_G(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-a)^2}{2\cdot\sigma^2}} \tag{3}$$

Perspective transformation is required for different viewing angles to detect the tag. For the transformation process, the four corner coordinates of the tag must be found. Therefore, all contours in the image are obtained. Then the contour corner coordinates are acquired. These contours are possible shapes that also contain the tag. The Douglas-Peucker algorithm is used to find the contour coordinates. After all the shapes in the image are found, the contour corner coordinates are sorted. Only shapes with four corner coordinates are filtered out.

A perspective transformation is applied to shapes with four corner points. The corner coordinates of the found shapes are listed. As depicted in Equation 4, the perspective transformation matrix is calculated by employing the coordinates of the upper left, upper right, lower left, and lower right corner points of the vertices.

$$[x' \quad y' \quad z' \quad w'] = [x \quad y \quad z \quad w] \times \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{4}$$

After applying the perspective transformation, the relevant area is cropped from the image. This area is resized to 8×8. The resized grayscale image is converted to a binary image. The 8×8 binary matrix of the template label image is compared with the values of the found area and when all values are equal, the detection of the tag is performed.

The second step is to calculate the position of the algorithm. Equation 5 is used to calculate the camera's position and orientation relative to the tag.

Figure 3 The Detection Algorithm

$$\begin{bmatrix} r_{00} & r_{01} & t_x \\ r_{10} & r_{11} & t_y \\ r_{20} & r_{22} & t_s \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ h_{10} & h_{11} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} & h_{23} \\ h_{30} & h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (5)$$

$$x' = H_n \times x \quad (6)$$

In Equation 5, the homography matrix is defined as the product of the camera matrix and the transformation matrix. The homography matrix is a projection in which images of a particular point are obtained from different angles [9].

$$\begin{bmatrix} x_i' \\ y_i' \\ c \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (7)$$

Equation 6 represents the relationship between the coordinates x of the reference image and the coordinates x' obtained from a different perspective. This relationship is expressed using the 3x3 section of the homography matrix, denoted as H_n. The H_n matrix contains information about perspective transformation and camera geometry, defining the coordinate relationship between the two images [10]. This relationship is provided in Equation 7.

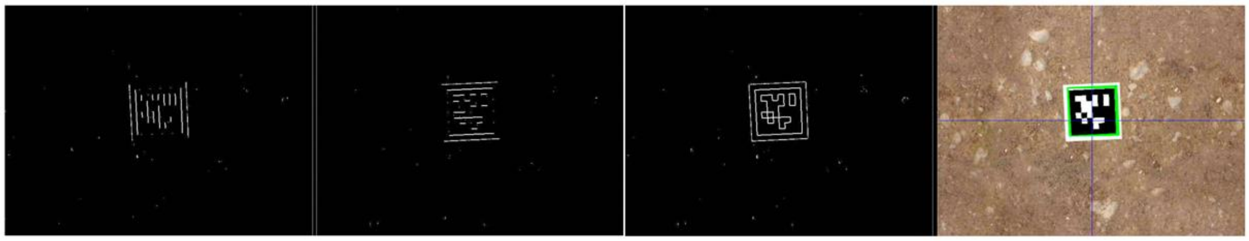
Figure 4 Detection Steps of The Algorithm

After completing the processes, the calculated coordinate data corresponds to the position of the label in the x, y, and z axes. This enables making a position estimation using the location information referenced from the tag. The detection steps of the tag in the algorithm are visually demonstrated in Figure 4.

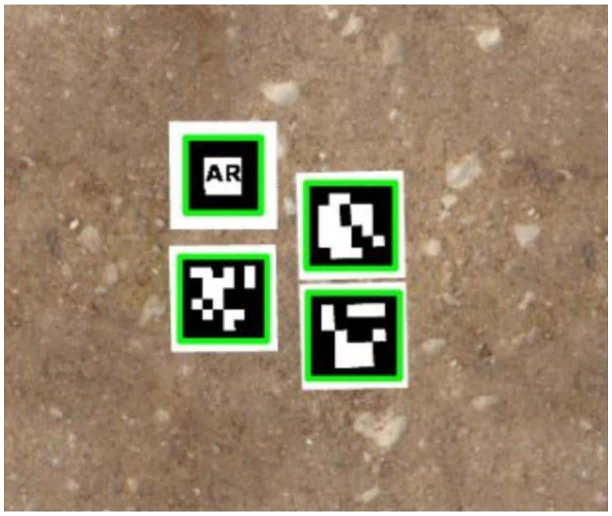

Figure 5 Detection of the Different Fiducial Markers

The Figure 5 illustrates the successful detection of four different fiducial marker types simultaneously because of the proposed algorithm. These results demonstrate the effectiveness of the algorithm in detection and its capability to detect multiple tags concurrently.

2.3. Kalman Filter

Calculated position information can sometimes contain errors due to rapid movement and distortions. In this study, a one-dimensional Kalman filter was applied to minimize the errors and deviations of the calculated position values. The proposed position estimation algorithm calculates local x, y, and z coordinates and these coordinates are given as input to the Kalman filter. Then the filter gives more probable and more stable results from these coordinates.

Equation 8 defines the equation of Kalman gain. K_n is the Kalman gain, $p_{n,n-1}$ is the estimation uncertainty, and r_n is the measurement uncertainty. The Kalman gain is given in Equation 8 and must be between 0 and 1 as seen in Equation 9.

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{8}$$

$$0 \leq K_n \leq 1 \tag{9}$$

In Equation 10, the state estimation update calculations are given. $\hat{x}_{n,n-1}$ is the system state estimation vector. $1 - K_n$ is the weight given to the estimated value, and K_n is the measurement length weight.

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n\left(z_n - \hat{x}_{n,n-1}\right) = (1 - K_n)\hat{x}_{n,n-1} + K_n z_n \tag{10}$$

The Kalman gain shows how much the estimate will be changed by a given measurement. It converges to zero when the measurement uncertainty is large, and the estimation uncertainty is small. When it converges to one, the measurement uncertainty is small, and the estimation uncertainty is large. The Kalman gain can be adjusted by changing the weights given for estimation and measurement.

$$P_{n,n} = (1 - K_n)p_{n,n-1} \tag{11}$$

The estimation uncertainty update equation is given in Equation 11. This equation updates the estimation uncertainty of the current state. This equation is called the "covariance update equation" [12]. Because of $(1 - K_n\ ) < 1$, the estimation uncertainty gets smaller with each filter iteration. When the Kalman gain is low, the convergence of the estimation uncertainty will be slow. However, if the Kalman gain is high, the estimation uncertainty will quickly converge to zero.

$$\hat{x}_{n,n+1} = \hat{x}_{n,n} + \Delta t \hat{\dot{x}}_{n,n} \tag{12}$$

$$\hat{\dot{x}}_{n,n-1} = \hat{\dot{x}}_{n,n} \tag{13}$$

Extrapolation of estimation uncertainty is performed using a dynamic model equation, such as state extrapolation. In Equation 12, the estimated position information is equal to the sum of the current predicted position multiplied by the delta time and the current predicted velocity. The estimated velocity value, when assuming a constant velocity model in Equation 13, is equal to the current velocity prediction.

$$P_{n,n+1} = p_{n,n} + q \qquad (14)$$

The constant velocity dynamics are employed due to the fixed position of the tag. The calculated positions are considered as the aircraft's location, with the tag serving as the reference point. A delay error was observed in the results during the implementation of the Kalman filter. To eliminate the delay error and enhance the reliability of the prediction, process noise (q) was added to the system. Equation 14 demonstrates the inclusion of noise in the calculations of prediction uncertainty ($p_{n,n}$).

Figure 6 illustrates a simplified diagram of the Kalman method used in this study.

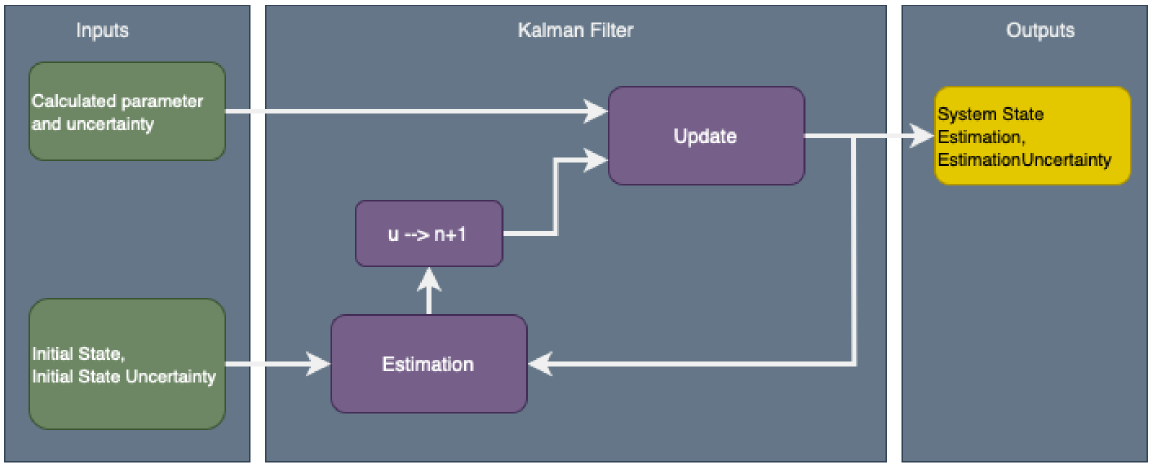
Inputs
Kalman Filter
Outputs
Calculated parameter and uncertainty
Update
System State Estimation, EstimationUncertainty
u --> n+1
Initial State, Initial State Uncertainty
Estimation

Figure 6 Kalman Filter Block Diagram

The inputs of the applied filter are analyzed in two parts: initial state and measurements. The initial system state ($\hat{x}_{1,0}$) and initial state uncertainty ($p_{1,0}$) are provided for the initial inputs. These inputs are determined through a trial-and-error method to yield the best results in the system. The measurement inputs consist of the measured system state (z_n) and its uncertainty (r_n). The outputs of the filter are the estimated system state ($\hat{x}_{n,n}$) and the prediction uncertainty ($p_{n,n}$). The prediction uncertainty ($p_{n,n}$) should gradually decrease, as $(1 - K_n) < 1$, where K_n represents the gain.

Table 1 Initial Values

Variables	$\hat{x}_{1,0}$	$p_{1,0}$	r_n	q
Values	5	1000	3	0.09

2.4. System Implementation

In this study, the Quadrotor model, developed based on mathematical models using dynamic and kinematic equations, was investigated within a Robot Operating System (ROS) [13] based framework in the simulation environment called Gazebo [14].

The algorithm was validated in the simulation environment. Initially, an image-based position estimation algorithm was executed, and the position information from the algorithm output was utilized for landing in the simulation environment.

For the simulation setup, the open-source ArduPilot [15] software was employed as the autopilot. Gazebo was used for mathematical modeling and visualization purposes. The image-based position estimation algorithm was developed in C/C++ within the Robotic Operating System (ROS) environment. The autopilot software, Gazebo, and the image-based position estimation code were executed in ROS. The system block diagram is illustrated in Figure 7.

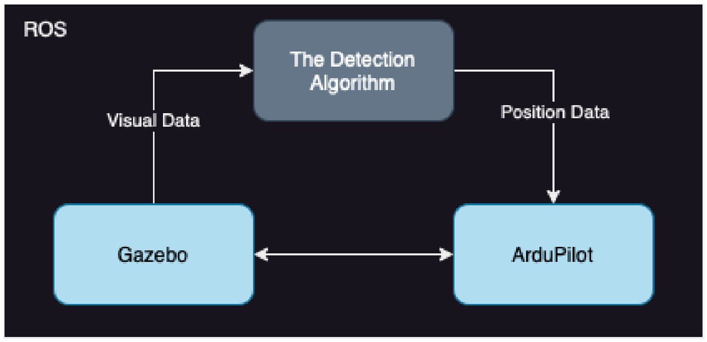

Figure 7 Simulation Environment System Architecture

3. Results

The proposed method has been validated with a quadcopter vehicle model in the simulation environment. The results of the experiments and the experimental setup are presented. The simulation was run at a frequency of 1000 Hz, while the algorithm outputs were recorded at a rate of 2 Hz.

Data obtained from 11 different flight tests conducted in the simulation were analyzed. The data from a sample test are shown in Figures 8, 9, and 10. The estimation data, represented by the blue line in the graph, corresponds to the position outputs of the developed algorithm. The orange-colored local data represents the actual position information calculated by the autopilot.

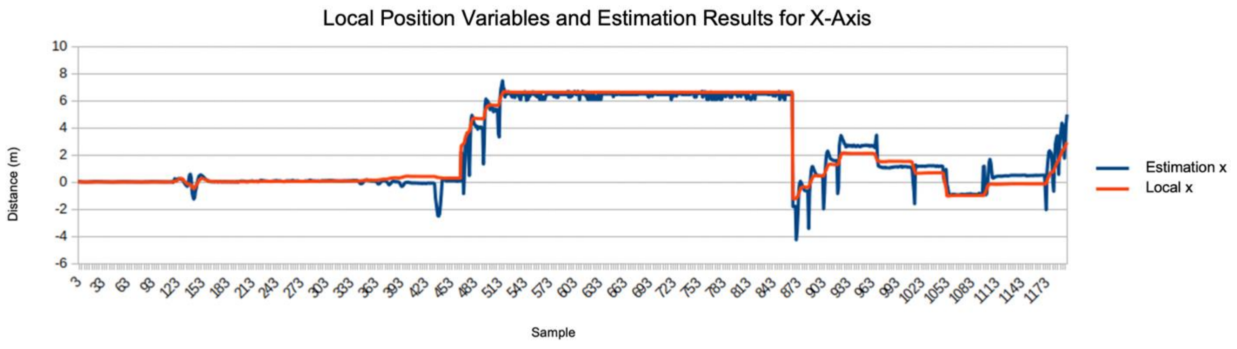

Figure 8 Local Position Variables and Estimation Results for X-Axis

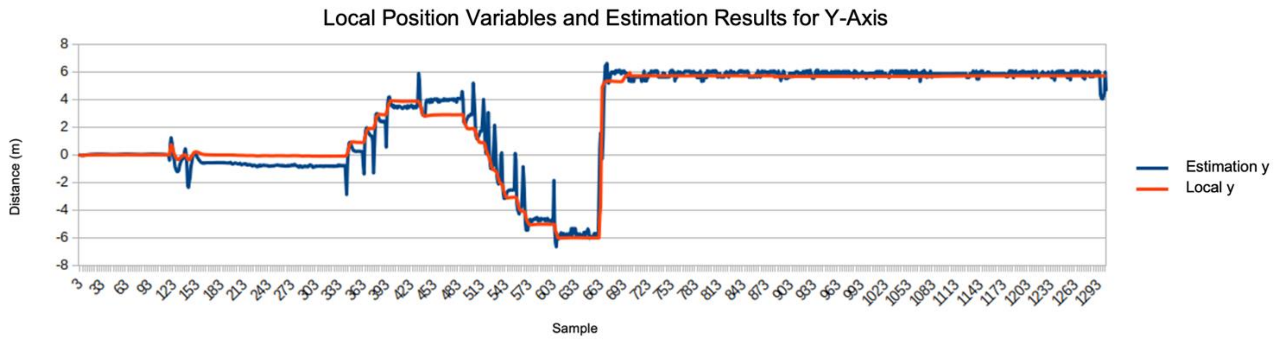

Figure 9 Local Position Variables and Estimation Results for Y-Axis

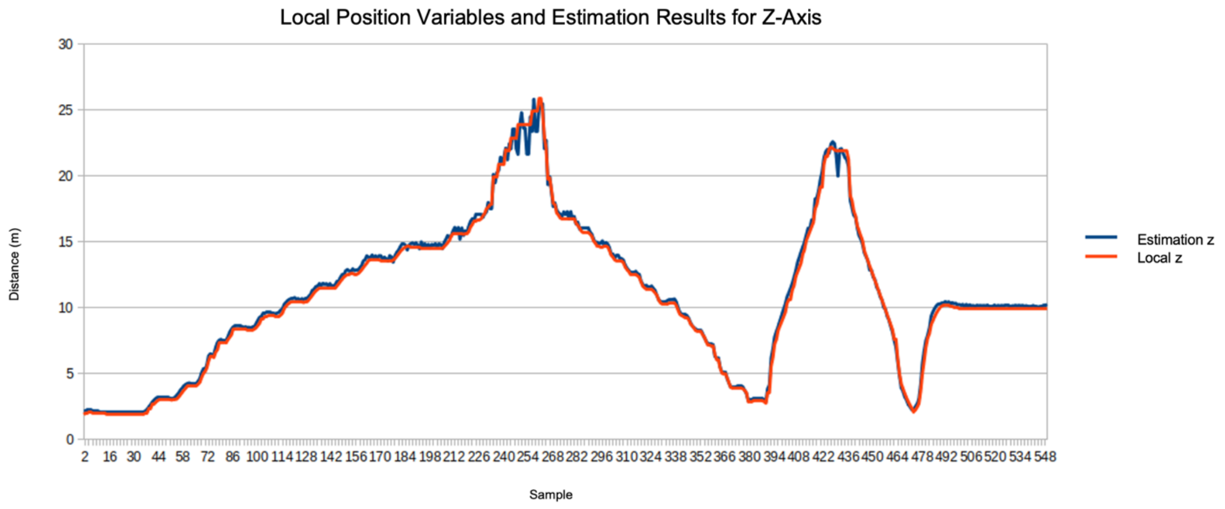

Figure 10 Local Position Variables and Estimation Results for Z-Axis

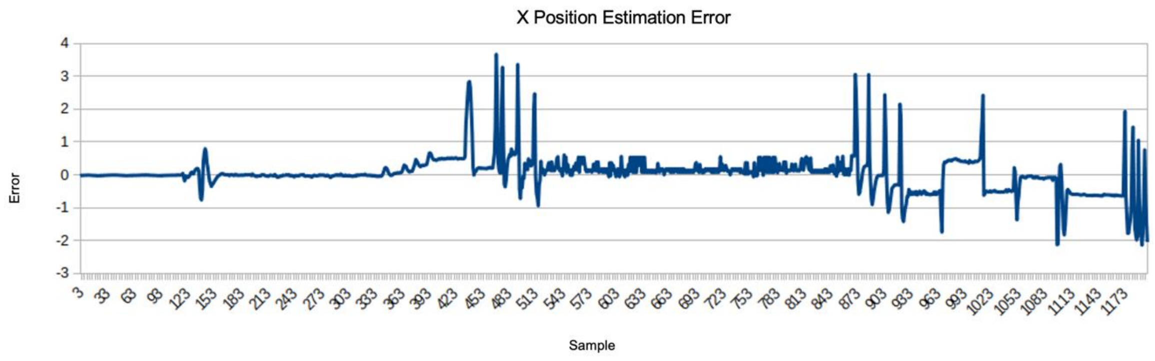

Figure 11 X Position Estimation Error Graph

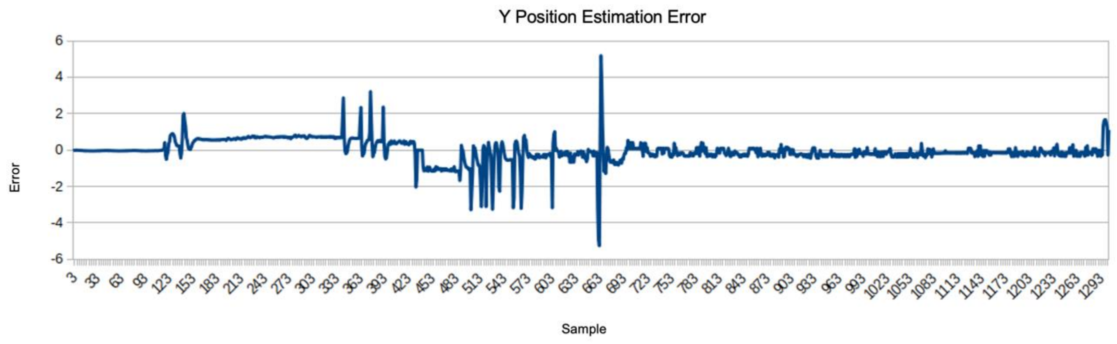

Figure 12 Y Position Estimation Error Graph

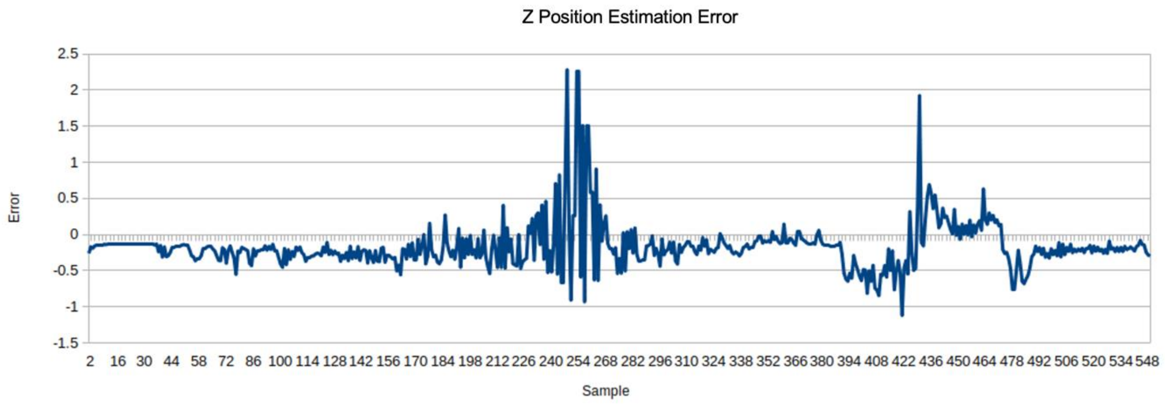

Figure 13 Z Position Estimation Error Graph

Figures 11, 12, and 13 illustrate the errors between the estimation data shown in Figures 8, 9, and 10 and the actual data. From these graphs, it can be observed that the algorithm makes inaccurate calculations up to a distance of 2.3 meters when providing position commands to the UAV. However, for constant or slow movements, the error in the algorithm's results decreases to below 5 cm.

The tests were repeated at different distances and rotations as seen in Figure 14. For the X and Y axes, the distance was limited to 6 meters to ensure that the tag remains within the camera's field of view. The results within this range are acceptable without sudden changes in velocity. The most suitable distance test was conducted along the Z axis. As the altitude increases, it can be seen in Figure 13 that the algorithm's error increases beyond 2 meters. The algorithm operates with an error of more than 2 meters up to a distance of 25 meters, but beyond this distance, it was observed that the algorithm fails to recognize the tag and produce accurate results.

When examining the results, it can be noted that both the limited field of view of the camera and the algorithm's unacceptable readings beyond 25 meters are identified as weaknesses of this system. Considering that autonomous landing is expected to be performed more carefully and slowly compared to normal flight, it is observed that the algorithm provides satisfactory results, particularly in close distances and altitude estimation.

In this study, 11 different test scenarios were implemented, and their results were recorded to compare the estimation outputs with reference data obtained at different distances. For each test scenario, reference data and estimation outputs were obtained at specific distances. The collected data were statistically analyzed using average values calculated separately for each distance. The results of this analysis are presented in Tables 2, 3, and 4.

According to the results presented in Table 2, it is observed that as the distance in the Z-axis increases, the amount of error also increases. Particularly for distances up to 20 meters, there is a negative increase in error, followed by a positive increase. This indicates that the estimations are increasingly encountering distance errors and exhibiting a tendency to shift in the wrong direction. Beyond 25 meters, the algorithm fails to recognize the tag and cannot provide any results.

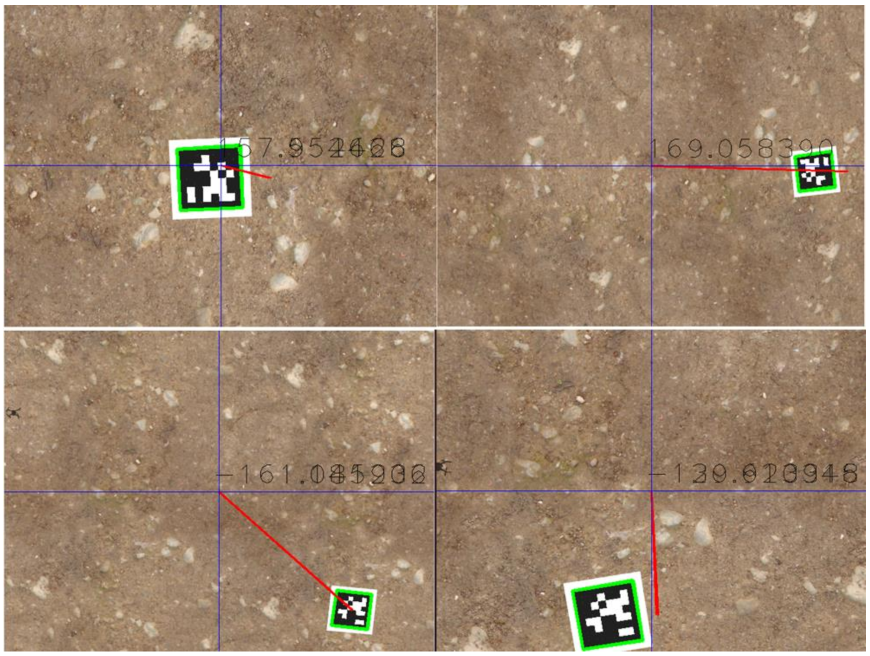

Figure 14 Tag Detections from Different Angles

Table 2 Estimation Output and Error Means on the Z Axis

Reference Z (cm)	**Estimation Z (cm)**	**Error Z (cm)**
500	502	-2.1
1000	1003	-2.6
1500	1505	-4.8
2000	1995	5.1
2500	2483	16.6

When examining Table 3, it can be observed that in the tests conducted along the Y-axis, the amount of error was negative at a distance of 1.2 meters, but it increased positively for subsequent distances. This indicates that as the horizontal distance of the tag along the Y-axis increases, the error between the algorithm's outputs and the reference data also increases, resulting in a decrease in the accuracy of the predictions.

Table 3 Estimation Output and Error Means on the Y Axis

Reference Y (cm)	**Estimation Y (cm)**	**Error Y (cm)**

120	122	-2.8
240	234	6
360	350	9.7
480	462	18
600	575	25

Table 4 contains the results of the tests conducted for the X-axis. These tests were performed at the same distances as the Y-axis. According to the table, a positive amount of error was observed at a distance of 1.2 meters, while a subsequent increase in the negative direction was detected for the following distances. It was also observed that the amount of error between the algorithm's outputs and the reference data increased with the increase in distance along the X-axis.

Table 4 Estimation Output and Error Means on the X Axis

Reference X (cm)	Estimation X (cm)	Error X (cm)
120	117	3.1
240	245	-5
360	371	-11
480	496	-18.2
600	626	-26

To compare the results of these axes with the algorithm results in the Z-axis, an evaluation was conducted based on approximate results above 5 meters. According to this evaluation, the average error in the X-axis was determined to be -18.2 meters, while the average error in the Y-axis was 18 meters. For the Z-axis, the average error amount was calculated as -2.1 meters. Upon examining the results for the Z-axis, it can be observed that the error amount is lower compared to the other axes. Therefore, it can be concluded that the algorithm performs more accurate results on the Z-axis compared to the other axes.

4. Discussion and Conclusion

In this study, a vision-based position estimation algorithm was developed using fiducial markers. To provide more accurate outputs of the position information and mitigate the effects of disturbances, a one-dimensional Kalman Filter was used for the algorithm's results. The study was tested in a simulation environment. When examining the results, it was observed that the algorithm provided acceptable results at close distances. Based on this study, it can be suggested as an alternative or auxiliary system to commonly used sensors such as laser or lidar for autonomous landing.

Future studies can focus on utilizing systems such as gimbals to prevent the tag from going out of the field of view of the camera and increase the viewing distance. Additionally, using an advanced Kalman methods to reduce potential errors should be considered as a further research direction.

References

[1] Ariyur K B, Fregene K O. Autonomous tracking of a ground vehicle by a UAV[C] American Control Conference. 2008.

[2] Iwakura, D.; Wang, W.; Nonami, K.; Haley, M. Movable Range-Finding Sensor System and Precise Automated Landing of Quad-Rotor MAV. J. Syst. Des. Dyn. 2011, 5, 17–29.

[3] Gautam A, Singh M, Sujit PB, Saripalli S. Autonomous Quadcopter Landing on a Moving Target. Sensors. 2022; 22(3):1116.

[4] Arora, S.; Jain, S.; Scherer, S.; Nuske, S.; Chamberlain, L.; Singh, S. Infrastructure-free shipdeck tracking for autonomous landing. In Pro-ceedings of the IEEE International Conference on Robotics and

[5] Auto-mation,Karlsruhe, Germany, 6–10 May 2013; pp. 323–330. Gautam A, Singh M, Sujit PB, Saripalli S. Autonomous Quadcopter Landing on a Moving Target. Sensors. 2022; 22(3):1116.

[6] Herissé, B.; Hamel, T.; Mahony, R.; Russotto, F.X. Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow. IEEE Trans. Robot. 2012, 28, 77–89.

[7] M. UZUNOGLU, R. B. ŞAHİN and M. MERCİMEK, "Vision-Based Position Estimation with Markers For Quadrotors," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-6, doi: 10.1109/HORA55278.2022. 9800043.

[8] Zhang, Zhuming & Hu, Yongtao & Yu, Guoxing & Dai, Jingwen. (2021). DeepTag: A General Framework for Fiducial Marker Design and Detection.

[9] Ozuag, Ersin & Erturk, Sarp. (2014). A Homography Matrix Decomposition Based Video Synchronization Approach. 2014 22nd Signal Processing and Communications Applications Conference, SIU 2014 - Proceedings. 10.1109/SIU.2014.6830661.

[10] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," in IEEE Transactions on Automatic Control, vol. 25, no. 2, pp. 164-176, April 1980.

[11] Y. Ito and Y. Oda, "Estimation of Camera Projection Matrix Using Linear Matrix Inequalities," 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS), 2016, pp. 72-75.

[12] Charles K. Chui and Guanrong Chen. 2017. Kalman Filtering: With Real-Time Applications (5th ed.). Springer Publishing Company, Incor- porated.

[13] Stanford Artificial Intelligence Laboratory et al. (2018). Robotic Operating System. Retrieved from https://www.ros.org

[14] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 2004, pp. 2149-2154 vol.3, doi: 10.1109/IROS.2004.1389727.

[15] "ArduPilot", ArduPilot Documentation - ArduPilot documentation, [online] Available: https://ardupilot.org/ardupilot/.