Article

Fuzzy Gain-Scheduling Based Fault Tolerant Visual Servo Control of Quadrotors

Tolga Yuksel

Department of Electrical and Electronics Engineering, Engineering Faculty, Bilecik Seyh Edebali University, Bilecik 11100, Turkey; tolga.yuksel@bilecik.edu.tr

Abstract: When military and civil missions such as transportation increase, fault tolerant control of unmanned aerial vehicles will be an obligation. Although onboard sensors provide information about the status of a quadrotor, the camera is not included in the list. In this study, visual servo control of quadrotors as a popular method for motion control is addressed. we address a visual servo control system for quadrotors as a popular method for motion control. The feature motions in the image plane are analyzed to reveal the relation between the actuator faults and these motions. Four AI fault approximators, a neural network, an extreme learning machine, a linear support vector machine, and a long short-term memory are used to approximate actuator faults of a quadrotor while using feature inputs. The results are convincing and the approximation results are used by a fuzzy logic unit to provide gain-scheduling based fault tolerant control. The proposed system shows sufficient results as a visual servo system for fixed and moving feature targets while providing fault tolerance.

Keywords: visual servoing; fault tolerant control; neural network; fuzzy logic

check for updates

Citation: Yuksel, T. Fuzzy Gain-Scheduling Based Fault Tolerant Visual Servo Control of Quadrotors. *Drones* **2023**, *7*, 100. https://doi.org/10.3390/drones7020100

Academic Editor: Andrey V. Savkin

Received: 31 December 2022
Revised: 20 January 2023
Accepted: 22 January 2023
Published: 31 January 2023

1. Introduction

Visual servoing (VS) is a motion control method for robotic systems that utilizes visual feedback obtained from mono or stereo cameras [1]. It is often deployed in applications where a robot needs to move according to given visual goals, such as assembly tasks. With the increase of cameras in robot manipulator control, VS draws more attention and advanced approaches show up Increased cameras in robot manipulator control has drawn more attention and resulted in advanced approaches [2]. The use of these approaches in different fields, applications, and systems such as humanoid robots or unmanned vehicles is also becoming widespread [3,4].

Classical VS starts with obtaining meaningful data defined as features from an image which is obtained by using a camera, as shown in Figure 1 [5]. These k features are arranged in vector form $\boldsymbol{s}$ according to their metrics such as coordinates in the image plane. The main goal is to reach the desired $\boldsymbol{s}^*$ and the VS control law in terms of the velocities of the robotic system, especially the Cartesian velocities of the end effector, defined by the error vector between $\boldsymbol{s}$ and $\boldsymbol{s}^*$. After this generalized definition of VS, the approaches can be classified into two main types: position-based visual servoing (PBVS) and image-based visual servoing (IBVS) [1]. On the one hand, in PBVS, the control law uses the information about the pose of the end effector relative to the desired pose to obtain $\boldsymbol{s}^*$. On the other hand, IBVS utilizes $\boldsymbol{s}$ obtained from the instant image without any operations. PBVS is affected by pose estimation errors and IBVS is subject to keeping the image features in the field of view (FOV); therefore, hybrid approaches such as partitioned and 2½ D VS [6,7] or featureless approaches such as kernel-based VS [8] have been derived to avoid these problems. The advantage of robustness against depth estimation errors makes IBVS worthy for practical applications and it is preferred in this study.

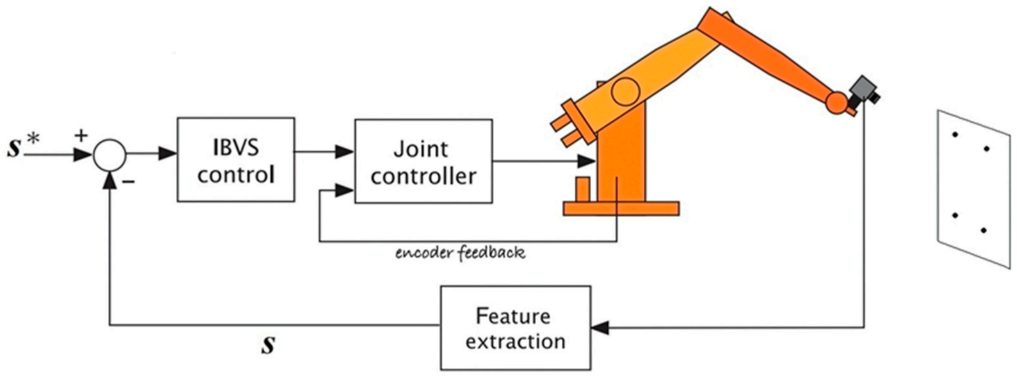


Figure 1. Classical visual servo control for robot manipulators [5].

The main platform of this study is a quadrotor which can be defined as an unmanned aerial vehicle (UAV) that is propelled by four independent rotors and can be used in a variety of applications, including aerial photography, mapping, or surveillance. While applications of VS on UAVs are becoming popular, the first application of VS on quadrotors can be considered to be the study by Mahony and Hamel [9]. The study used linear features but feature noise was neglected. Ceren and Altug proposed a VS system for quadrotors based on spherical image projection [10]. Although spherical camera projection is a potential replacement for VS image projection, investigations on VS control of quadrotors with spherical imaging have shown that this leads to inappropriate motion characteristics in Cartesian space. [11,12]. De Plinval et al. utilized the homography matrix obtained from frontal camera motion and gyrometer measurements to propose VS stabilizer laws [13]. Mebarki et al. used nonlinear observers to estimate translational velocity and spherical image features and designed a control law by utilizing an integral backstepping approach [14]. Abdessameud and Janabi-Sharifi proposed a VS system based on similar translational velocity estimates, but the velocity signals were neglected [15]. Zhang et al. focused on the visibility problem of features by optimizing VS control inputs under visibility constraints [16], while neglecting velocity signals. Control methods such as model predictive control (MPC) are also adapted to VS control of quadrotors. Zhang et al. proposed a robust nonlinear MPC VS scheme for quadrotors under disturbances, but the sudden changes in control inputs which could be dangerous for practical applications were neglected [17]. Quadrotor team formation with IBVS was discussed in [18], but the velocity signals and image feature trajectories were neglected. Xie and Lynch presented an input saturated controller for UAVs to regulate the relative pose between a vehicle and a planar horizontal visual target [19]. They simplified the control law design by using a virtual camera to define a set of image moment features and proposed another input saturation law to keep features in FOV. They mentioned it was a generalized law for UAVs, but only features of two points were selected and different dynamics between fixed-wing and rotary-wing UAVs were neglected. Zhao et al. focused on PBVS control of quadrotors with sensor fusion [20]. They utilized an IMU, an ultrasonic sensor, and a vision sensor and designed a robust compensator to enhance the robustness against nonlinearities, couplings, and uncertainties. The position of the quadrotor was estimated using a visual feature and the designed controller was compared to the classical PID controller, but details such as feature trajectory were not given. Cao et al. proposed an IBVS controller for quadrotors to stabilize hovering and tracking [21]. Backstepping controllers were designed to stabilize the VS of the quadrotor and a trajectory observer based on a nonlinear tracking differentiator was integrated into the proposed system. The system was verified under disturbance such as a sudden change of goal image feature locations, but motion in 3D, rotor torques, and image noises were neglected.

Other hot topics about motion control systems are fault detection and diagnosis (FDD) and fault tolerant control (FTC). FDD is the process of identifying and diagnosing the type, the altitude, and the time of a malfunction or fault in a system [22]. This process is followed by FTC which can be defined as the ability of a control system to continue operating under defined operation constraints in the presence of a fault [23]. It must be noted that temporary faults that show their presence at a time interval can be encountered. To ensure fault tolerance, critical control systems are designed with hardware redundant components, such as triple computer redundancy in airplanes, but the FTC design focuses on reconfiguring the controller with or without using FDD information.

In the last two decades, FDD and FTC methods have been adapted, modified, and updated for quadrotors as the flight dynamics are quite different from fixed-wing UAVs. A detailed review on FDD and FTC for UAVs can be found in [24]. This study focused on actuator faults, AI-based FDD, active FTC for quadrotors, and the studies on these foci are summarized. Avram et al. designed an FDD system that involved a nonlinear fault detection estimator, a bank of nonlinear adaptive fault isolation estimators, and a fault accommodator which used fault estimation information [25]. The study showed sufficient results, but the partial actuator fault ratio was quite limited which could be passively tolerated by PID controllers of the quadrotor. Song et al. proposed an indirect neural network (NN)-based adaptive FTC scheme with virtual parameter estimating algorithms [26]. The proposed system provided robustness against model uncertainties, but sudden changes in the control signals that could be catastrophic for a real quadrotor were neglected in the study. Ren proposed a robust H_∞ observer to observe actuator fault and state estimation of a quadrotor in the presence of external disturbances, parameter uncertainties, and nonlinear terms [27]. The fault was defined as a failure factor in lift and trust torques, but the faulty actuator was not diagnosed in the study. Ma et al. also proposed an observer-based adaptive controller to estimate and compensate actuator and sensor faults of quadrotors [28]. The faults in the actuators were defined as biases in the form of a nonlinear function, but the physical equivalence was not defined and the results were given for a noiseless scenario.

From a practical point of view, actuator faults in UAVs may cause catastrophic results while performing dedicated tasks. To avoid these undesirable scenarios, UAVs should benefit from all the hardware that are reliable after the fault to diagnose and accommodate this situation. Quadrotors are equipped with IMUs for navigation but it is hard to diagnose actuator faults from the signals of IMUs since the inner and outer loop controllers are coupled according to RPY motions. Furthermore, using auxiliary signals from auxiliary sensors such as cameras may enhance the possibility of diagnostics without the need of hardware redundancy. Additionally, cameras can provide this assistance while implementing other tasks such as visual SLAM or visual autolanding. Cameras are also more robust than IMUs and they still can be active in the case of faulty sensors on the quadrotor. In this study, camera images and features are deployed to diagnose quadrotor actuator faults as the primary contribution to the literature. The proposed system, which combines detection and diagnostics, first approximates actuator faults. The contenders for this approximation include NN, ELM, linear SVM, and LSTM. NN had the best performance based on the RMSE measure, although the competition was fairly intense.

Active FTC is defined as the reconfiguration of controller parameters or switching between different controllers according to a fault diagnosis [23]. These options may cause sudden changes and discontinuities that may cause hard maneuvers which are undesirable for a reliable flight. As a solution, a fuzzy GS-based active FTC stage is proposed to tune the faulty actuator gain while providing convergence after the fault diagnosis.

The system's robustness is evaluated in the presence of feature noise. Despite the steady-state feature errors, the system exhibits sufficient tolerance to actuator faults and does not diverge. The proposed system is also tested for tracking moving feature targets under feature noise. The tracking performance of the proposed system is quite convincing while keeping the features in FOV.

The study is organized as follows: In Section 2, the visual servo approach for quadrotors, and the proposed system with fault approximators and the fuzzy gain scheduling mechanism are described in detail; in Section 3, the simulation results for the proposed system under feature noise are given; in Section 4, conclusions and future goals are summarized.

2. The Proposed Fault Tolerant Visual Servo Control System

The proposed fault tolerant visual servo control system is presented in depth in this section. As the first step, it should be emphasized that an IBVS system with point features and with an eye-in-hand configuration is proposed as the quadrotor is carrying a down-looking camera. The proposed system is shown in Figure 2.

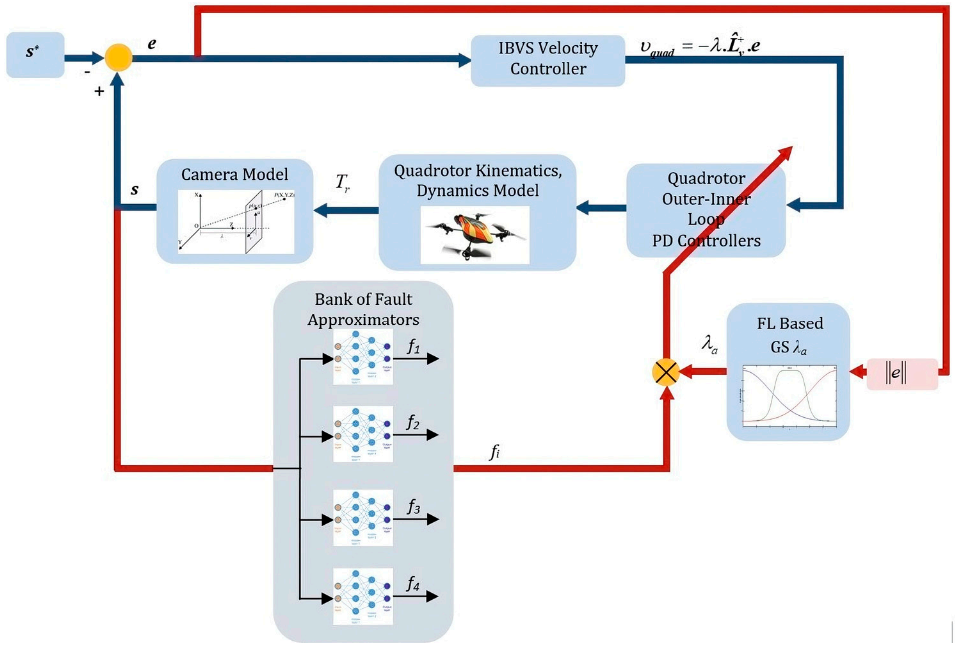

s^*
e
-
+
IBVS Velocity Controller
$v_{quad} = -\lambda.\hat{L}_s^+.e$
Camera Model
s
T_r
Quadrotor Kinematics, Dynamics Model
Quadrotor Outer-Inner Loop PD Controllers
Bank of Fault Approximators
f_1
f_2
f_3
f_4
f_i
λ_a
FL Based GS λ_a
$\|e\|$

Figure 2. The proposed fault tolerant visual servo system in blocks.

VS begins with the projection of point features to the image plane. Assume that a manipulator equipped with an eye-in-hand camera is seeing k feature points in 3D. These feature points' coordinates in the image plane of the camera are given as:

$$s_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T, \forall i \in \{1 \ldots k\} \tag{1}$$

where u_i, v_i are the coordinates in the u-v image plane. To characterize the behavior of the fixed-motionless feature points, these vectors are merged into a matrix in the form:

$$\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & \ldots & s_k \end{bmatrix}^T \in \mathbb{R}^{2\times k} \tag{2}$$

with $\boldsymbol{s}^*$ having the same dimensions as the desired fixed feature points matrix. Error convergence is the primary objective of all VS approaches:

$$\boldsymbol{e}(\boldsymbol{t}) = \boldsymbol{s}(\boldsymbol{t}) - \boldsymbol{s}^* \tag{3}$$

Following these notations, the linear velocity $\boldsymbol{v}$ and angular velocity $\boldsymbol{\Omega}$ of a point are determined in the world coordinates as:

$$\dot{\boldsymbol{P}}_{\boldsymbol{i}} = -\boldsymbol{\Omega}\times\boldsymbol{P}_{\boldsymbol{i}} - \boldsymbol{v} \tag{4}$$

where $\dot{\boldsymbol{P}}_i$ and $\boldsymbol{P}_i \in \mathbb{R}^{\boldsymbol{6}}$ are used to denote the point location and the velocity vectors, respectively. Subsequently, the focal length of the camera λ_f, the depth of the feature $\boldsymbol{Z}$, and $\boldsymbol{s}_i$ are used to define the projection of the linear and angular velocities in Equation (4) to the image plane.

$$\upsilon = \begin{bmatrix} \nu_x & \nu_y & \nu_z & w_x & w_y & w_z \end{bmatrix}^T, \begin{bmatrix} \dot{u}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} -\frac{\lambda_f}{Z} & 0 & \frac{u_i}{Z} & \frac{u_i \cdot v_i}{\lambda_f} & -\frac{{\lambda_f}^2 + {u_i}^2}{\lambda_f} & v_i \\ 0 & -\frac{\lambda_f}{Z} & \frac{v_i}{Z} & \frac{{\lambda_f}^2 + {v_i}^2}{\lambda_f} & -\frac{u_i \cdot v_i}{\lambda_f} & u_i \end{bmatrix} \begin{bmatrix} \nu_x \\ \nu_y \\ \nu_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \tag{5}$$

Equation (5) is shown as:

$$\boldsymbol{s}(\boldsymbol{t}) = \boldsymbol{L_s} \cdot \boldsymbol{v} \tag{6}$$

where $\boldsymbol{L_s} \in \mathbb{R}^{2k \times 6}$ is the interaction matrix (or named as image Jacobian). Since it is challenging to determine a feature's actual depth, its estimated value is presumed to represent the depth at $\boldsymbol{s}^*$ and this results in the estimated interaction matrix $_{\hat{L}_s}$.

Most VS systems make assumptions about camera formation and feature specs. Two assumptions for the proposed VS system are given below:

Assumption 1. *The camera is attached to the center of the quadrotor with the eye-in-hand configuration, and the camera frame ($\mathcal{F}_c$) and the center of the quadrotor ($\mathcal{F}_q$) intersect without any transformations:*

$$\mathcal{F}_c = T_e \cdot \mathcal{F}_e, \ T_e = I^{4 \times 4} \tag{7}$$

Assumption 2. *The depths of each feature are the same for each point, and all feature points are collinear. The characteristics adhere to the collinearity criteria outlined in [29] since A, B, and C are constants:*

$$\frac{1}{Z} = Ax_i + By_i + C \tag{8}$$

The Moore–Penrose pseudoinverse of the estimated interaction matrix $\hat{\boldsymbol{L}}_s^+$, the error vector, and a fixed gain are referred by classical IBVS as a kinematic velocity controller to exponentially reduce the error:

$$\left.\begin{array}{r} \dot{\boldsymbol{s}} = \boldsymbol{L}_s \cdot \boldsymbol{v} \\ \dot{\boldsymbol{e}} + \lambda \cdot \boldsymbol{e} = 0 \end{array}\right\} \Rightarrow \boldsymbol{v} = -\lambda \cdot \hat{\boldsymbol{L}}_s^+ \cdot \boldsymbol{e} \tag{9}$$

Here, it should be noted that $\hat{\boldsymbol{L}}_s^+$ and λ define the effectiveness of this velocity controller. There are two major differences when applying the conventional IBVS that was designed for fully actuated robot manipulators to quadrotors [5]. First, due to the underactuation of quadrotor systems, the Jacobian only includes the columns that correspond to $(v_x, v_y, v_z, \omega_z)$. Secondly, the features in the image will move as a result of the quadrotor's inability to translate without first tilting in the desired direction, increasing the image feature error. This may be disregarded for small amounts of roll and pitch, but it has to be considered for aggressive maneuvers. Here, it is ignored in this study.

The VS controller directly creates the necessary velocities for the velocity loops of these degrees of freedom in addition to performing the function of the outermost position loops for x- and y-position, altitude, and yaw. Here, it must be noted that the velocity loops still need rate information as an input, and quadrotors obtain this information through an inertial measurement unit. The inner loop PD controllers define attitude torque demands that are roll torque (τ_x), pitch torque (τ_y), yaw torque (τ_z), and height control force (T), with gravity feedforward. Then, Euler's equation of motion provides the rotational acceleration for the quadrotor as:

$$\boldsymbol{I} \cdot \boldsymbol{\omega} = -\boldsymbol{\omega} \times \boldsymbol{I} \cdot \boldsymbol{\omega} + \tau \tag{10}$$

where $\boldsymbol{I}$ is the inertia matrix, $\boldsymbol{\omega}$ is the angular velocity vector, and τ is the torque vector applied to the quadrotor. The motion of the quadrotor is defined by the relation between torque vector and height control force using the torque applied to each propeller:

$$\begin{bmatrix} T \\ \tau \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \omega_2^2 \\ \omega_1^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{11}$$

where $\boldsymbol{A}$ is the coefficient matrix including aerodynamic drag, distances of blades from the center of mass of the quadrotor, and lift constant defined in [5]. This closed loop is shown with blue signal flows in Figure 2.

In this study, we focus on actuator faults with partial loss of effectiveness (LOE) that may be a result of motor or propeller damage or component deterioration [25,26]. While diagnosing a fault, the LOE ratio should be approximated to prepare for the stage of FTC. The fault diagnosis stage of the proposed system starts with $\boldsymbol{s}^*$as the input of the bank of fault approximators. Each approximator approximates the LOE of a propeller as f_i.

Four AI candidates, neural networks (NN), linear support vector machine (SVM), and a deep NN, LSTM are chosen as function approximators.Four AI candidates, i.e., neural networks (NN), linear support vector machine (SVM), extreme learning machine (ELM), and a deep NN, i.e., LSTM are chosen as function approximators. As the first candidate, NN is a powerful classification and regression tool that can map input–output relations without any user interference as a black-box unit [30]. The architecture of the fault approximator NN is given in Figure 3.

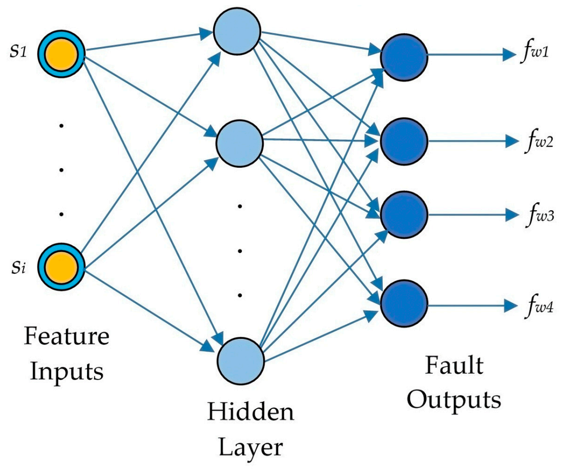


Figure 3. The architecture of fault approximator NN.

Haykin [30] provided all detailed information about NNs. Here, the learning algorithm, i.e., the Levenberg–Marquart (LM) algorithm, which is one of the essential indicators of the learning stage of NN, is briefly reviewed. NNs seek to reduce error and update their stated parameters with learning algorithms to provide acceptable outputs for appropriate inputs. The gradient descent approach, which is described in Equation (12), is the most popular learning algorithm used for this purpose:

$$w_{ij(n+1)=}w_{ij}(n) + \Delta w_{ij}(n) \tag{12}$$

$$\Delta w_{ij}(n) = -\eta \cdot \frac{\partial E(n)}{\partial w_{ij}} \tag{13}$$

where $E(n)$ is the n. step error function, w_{ij} is the weight from neuron i to neuron j, and η is the learning rate parameter. Instead of this parameter update, LM uses the sum of square errors for each input sample as the first step and its first derivative is defined in partial differential form as the Jacobian matrix in Equation (14). Then, the parameter update is implemented with μ learning rate using Equation (15):

$$J(w) = \begin{bmatrix} \frac{\partial E_1(x)}{\partial w(1,1)} & \frac{\partial E_1(x)}{\partial w(1,2)} & \cdots & \frac{\partial E_1(x)}{\partial w(1,j)} \\ \frac{\partial E_2(x)}{\partial w(2,1)} & \frac{\partial E_2(x)}{\partial w(2,2)} & \cdots & \frac{\partial E_2(x)}{\partial (2,j)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E_N(x)}{\partial w(i,1)} & \frac{\partial E_N(x)}{\partial w(i,2)} & \cdots & \frac{\partial E_N(x)}{\partial w(i,j)} \end{bmatrix} \tag{14}$$

$$\Delta w = \left[J^T(w) \cdot J(w) + \mu \cdot I \right]^{-1} \cdot J^T(w) \cdot e(w) \tag{15}$$

SVMs are a prominent and commonly used approach for machine learning classification issues by transferring the data to another hyperplane to classify linearly. In contrast, finding a function that roughly maps an input domain to actual numbers using a training sample is the goal of regression analysis. Linear SVM defines a linear equation under an error constraint:

$$\begin{gathered} \min \tfrac{1}{2} \|w_n\|^2 \\ s.t \left| y_n - \left(w_n{}^T x_n + b_n\right) \right| < \varepsilon \end{gathered} \tag{16}$$

where y_n is the output dataset, x_n is the input dataset, and ε is the maximum error, as shown in Figure 4. The line segments for the boundary linear equations are shown in red and the defined linear equation by linear SVM is shown in blue. Using basic quadratic programming approaches, this minimization problem may be stated in ordinary quadratic programming form. However, using quadratic programming approaches might be computationally costly. Therefore, approaches such as sequential minimal optimization (SMO) are referred to [31]. For input–output mapping for fault detection, four independent SVMs, each for one fault output, should be defined.

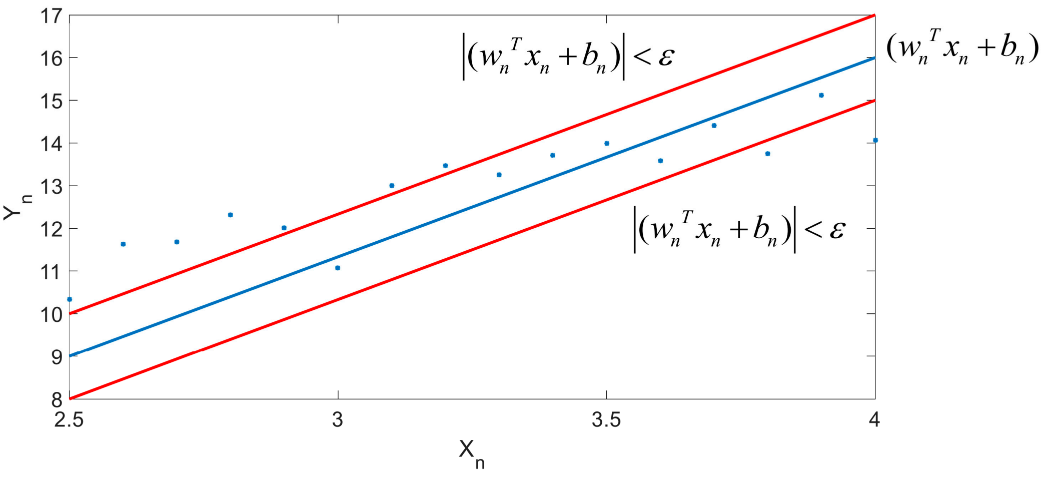


Figure 4. Linear regression and SVM constraints.

The most well-known regressors are NNs, SVMs, and hybrid structures such ANFIS, however, each of them has flaws. According to input–output mapping, the hidden layer parameters of SLFNs and the user-defined parameters of SVMs and ANFIS should be carefully set [32,33]. ELM generalizes single-layer feedforward NNs whose hidden layer does not require tuning as a member of the NN family. For this architecture, the output function of ELM as one output node scenario is:

$$f_{ELM}^{L}(\boldsymbol{x}) = \sum_{i=1}^{L} \beta_i \cdot h_i(x) = \boldsymbol{\beta} \cdot \boldsymbol{h}(x) \tag{17}$$

where $\boldsymbol{\beta}$ is the vector of the output weights between the hidden layer of L nodes and the output node, $\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} h_1(x) & h_2(x) & \cdots & h_L(x) \end{bmatrix}$ is the output vector of the hidden layer with $h_i(.)$ activation function. Input–output mapping of ELM is shown in Figure 5.

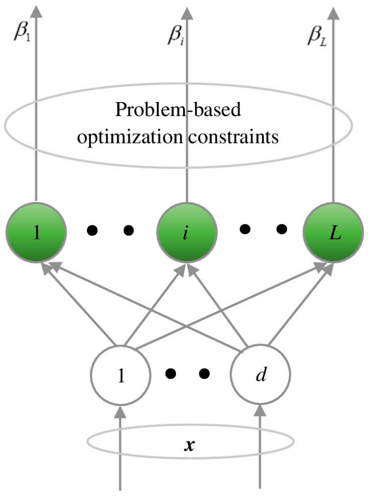


Figure 5. Input–output mapping of ELM.

The sigmoid, sine, or hard limiter functions are the definitions of the activation functions of the green hidden layer nodes in Figure 5. ELM tries to minimize the training error as well as the norm of the output weights, whereas conventional learning algorithms only frequently achieve the minimum training error between output and goal $\boldsymbol{T}$:

$$\left\|\hat{\boldsymbol{\beta}}\cdot\boldsymbol{H}(\boldsymbol{x}) - \boldsymbol{T}\right\| = \min_{\boldsymbol{\beta}}(\|\boldsymbol{\beta}\cdot\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{T}\|) \tag{18}$$

where $\hat{\boldsymbol{\beta}}$ is the minimum norm least square solution of $\boldsymbol{\beta}\cdot\boldsymbol{H} = \boldsymbol{T}$ with $\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{+}\cdot\boldsymbol{T}$ where $\boldsymbol{H}^{+}$ is the Moore–Penrose pseudoinverse as in Equation (9). In [33], it was stated that the orthogonal projection technique, orthogonalization method, iterative approach, and singular value decomposition (SVD) could all be used to obtain this pseudoinverse. Input–output presentation is the same as NNs.

The fourth regressor for fault function approximation is a deep NN architecture, i.e., long short-term memory (LSTM). An LSTM network is a kind of recurrent neural network (RNN) that can discover long-term relationships between sequence data's time steps [34]. Practically speaking, basic RNNs have a limited ability to learn longer term dependencies. RNNs are frequently trained via backpropagation, which can cause "vanishing gradient" or "exploding gradient" issues. These issues result in either extremely tiny or very high network weights, which limit the efficacy of applications that call on the network to learn long-term relationships. To solve this problem, LSTM networks employ extra gates to regulate which data from the hidden cell are transmitted as output and to the following hidden state, as shown in the LSTM cell in Figure 6. Here, f is the forgetting gate that decides what information is to be carried forward, g is the memory cell, i is the input gate that decides which values will be updated, o is the output gate, c_i is the cell state that is the memory of LSTM, h_i is the hidden state, and x_t is the input. The network can more successfully learn long-term associations in the data thanks to the extra gates. LSTM networks are superior to simple RNNs for evaluating sequential data because they are less sensitive to the time gap.

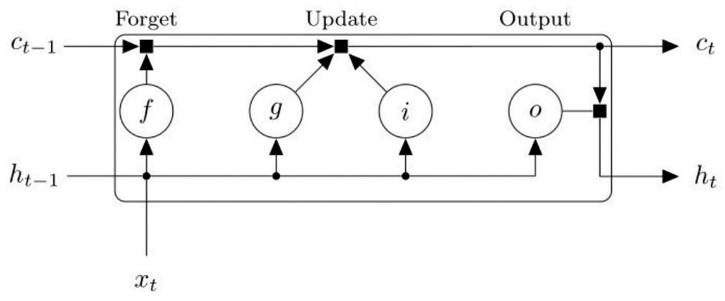


Figure 6. The architecture for an LSTM block.

A sequence input layer and an LSTM layer are the two main parts of an LSTM network. Data from time series or sequences are fed into the network using a sequence input layer. Sequence data's long-term relationships between time steps are learned by an LSTM layer. A regression output layer is also defined for regression purposes. It is not shown in Figure 6.

These AI regressor candidates are selected as fault approximators of the fault diagnosis system in Figure 1. The output of the approximator gives information about the LOE of the faulty rotor, and the quadrotor controllers should be updated according to this information which is defined as active FTC. The inner and outer loop controllers are coupled according to RPY motions, and changing the PD values of these controllers will not compensate the LOE of the faulty rotor directly. Therefore, torque and force signals in (11) should be scheduled by λ_a , the fault gain factor that is defined by fault approximators as gain scheduling (GS). However, this GS approach may cause a sudden change in the gain, resulting in sudden velocity changes in the motion, and concluding in divergence during the VS tasks. To avoid catastrophic results, an adaptive gain should be defined to provide a soft transition. Instead of an analytical definition, a fuzzy logic (FL) unit is deployed to obtain a soft transition and to also include user experience. The impact of error magnitude on IBVS features is taken into consideration while defining the linguistic rules of FL.

An FL unit's output depends on its type, input membership function (MF) types, MF aggregation, rulebase, and defuzzification type. A Mamdani-type FL unit is employed in this study, and therefore, the output functions are fuzzy MFs. As the most common fuzzy implication type, minimum is chosen and it must be noted that new implication types such as IFESI can be found in the literature [35]. Maximum is the aggregation type, and the centroid of area (COA), which is the weighted average of the centroids of output MFs with weighting factors (α_i) of input MFs $(\mu_i(\lambda_a))$, and ith rule is the defuzzification type [32]:

$$\begin{aligned} \alpha_i &= max\big(\mu_j(\|e\|)\big),\ \forall j \in \{1 \dots z\} \\ \lambda_a &= \frac{\sum_{i=1}^{n} \alpha_i \int \mu_i(\lambda_a) \cdot \lambda_a \cdot d\lambda_a}{\sum_{i=1}^{n} \alpha_i \int \mu_i(\lambda_a) \cdot d\lambda_a} \end{aligned} \tag{19}$$

After this FTC stage, the quadrotor receives the appropriate velocities for each rotor. Then, the quadrotor kinematics and dynamics provide the motion in 3D, completing the closed loop of the system presented in Figure 2.

3. Simulation Results

To show the performance of the proposed system, the proposed fault approximators and FTC system are implemented using MATLAB Simulink, Robotics Toolbox, Machine Vision Toolbox [5], Deep Learning Toolbox, Fuzzy Logic Toolbox, and ELM codes from [33]. In this study, an X-4 flyer model is chosen as the quadrotor platform and the details of this platform can be found in [36]. Table 1 provides an overview of the quadrotor model's parameters.

Table 1. X-4 quadrotor platform parameters.

Parameter	Value
m (mass)	4.34 kg
I_{xx} (Diagonal inertial element)	0.0820 kg m^2
I_{yy} (Diagonal inertial element)	0.0845 kg m^2
I_{zz} (Diagonal inertial element)	0.1377 kg m^2
r (Rotor radius)	0.165 m
A (Rotor disc area)	0.0855 m^2

It is assumed that a camera is fixed to the quadrotor's center without being transformed, as mentioned in the assumptions in Section 2. The camera's resolution is 1024 $\times$ 1024 pixels, and the principal point's coordinates are (512,512). The system's control loop and video stream both run at a rate of 20 Hz. The four fixed collinear points of a square with a side length of 0.5 m are used to define the features as $\boldsymbol{P}^*$ in Cartesian coordinates. As the goal of the IBVS system, $\boldsymbol{s}^* \in \mathbb{R}^{2\times4}$, these points' centers should collide with the principal point. $\boldsymbol{P}^*$ and $\boldsymbol{s}^*$ are defined as:

$$\begin{aligned} \boldsymbol{P}^* &= \begin{bmatrix} 0.25 & 0.25 & -0.25 & -0.25 \\ -0.25 & 0.25 & 0.25 & -0.25 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{s}^* &= \begin{bmatrix} 612 & 412 & 412 & 612 \\ 412 & 412 & 612 & 612 \end{bmatrix} \end{aligned} \tag{20}$$

In Equation (9), the estimated value of the depth is needed for $\hat{\boldsymbol{L}}_{\boldsymbol{s}}^{+}$ and it is assumed to be 2 m. The performance of the system may be affected by this estimation, as in [1].

As the first step of fault diagnosis, a dataset for fault approximators is created. As mentioned in [30], regressors need a dataset that covers all the workspace for the best approximation. Therefore, a healthy IBVS system is simulated with initial linear and angular values as $(X_{0,}\ Y_0, Z_0) = \begin{bmatrix} -0.35 & 0.7 & 2 \end{bmatrix}^T$ m. and $(\phi_{0,}\ \theta_0, \psi_0) = \begin{bmatrix} 0 & 0 & -\pi/4 \end{bmatrix}^T$ rad., respectively. Here, it must be noted that IBVS drags the features in the image plane through a sliding surface mode which is reached in the sliding phase in sliding mode control. Five LOE percentages, i.e., 10%, 20%, 30%, 40%, and 50%, are defined for each rotor and the actuator faults are injected to the healthy system at six different time instants which may change the behavior of the system. Here, it must be noted that the healthy IBVS diverges after a fault bigger than 10% LOE. The fixed gain value in Equation (9), λ, for the healthy system is 0.3. The dimensions of the dataset for inputs and outputs are 8 $\times$ 27,613 and 4 $\times$ 27,613, respectively. The data are divided into two parts, 85% data for training and 15% data for testing. Furthermore, the inputs and outputs are normalized for better approximation performance.

The first candidate is NN with three hidden layers with 10 neurons in each layer. The activation functions of each layer, including the output layer, are logarithmic sigmoid and the learning algorithm is LM. The goal is 0 RMSE and 1000 epochs are performed. As the second candidate, the proposed system uses four different ELMs, each for a rotor fault, with sigmoid-type activation functions with 10 hidden neurons. The third candidate is linear SVM with five-fold cross validation without any dimensionality reduction methods. Here, it must be noted that other SVM types with different kernel functions such as cubic SVM are tested, but linear SVM gives the best result. The last candidate is LSTM with 200 hidden units with Adam optimizer [37]. Again, the goal is 0 RMSE and 250 epochs are performed. The RMSE results for each approximator and each rotor are given in Table 2.

Table 2. RMSE performances for fault approximators.

Fault Approximator	RMSE
NN	0.0120
ELM	0.0770
Linear SVM	0.2833
LSTM	0.0454

Table 2 shows that NN is the best approximator according to the RMSE results. While other approximators, as more recent AI architectures, are expected to show better performance, NN surpasses them with nonlinear regression capability. As an example, the first NN output of Rotor 1 fault approximator with 30% LOE for 6 time instants are shown in Figure 7 with blue as the real LOE and red as the approximation.

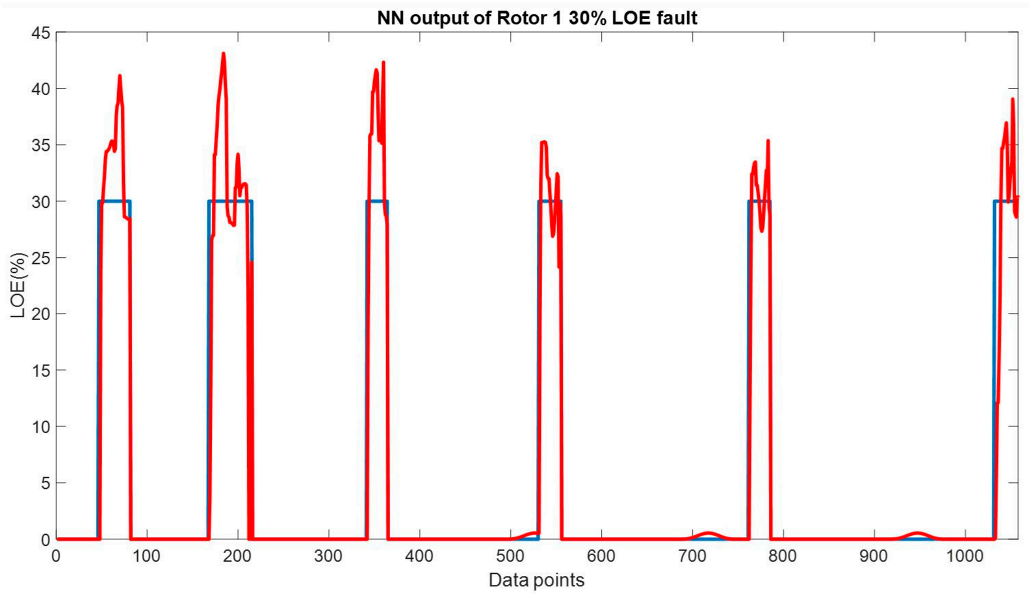
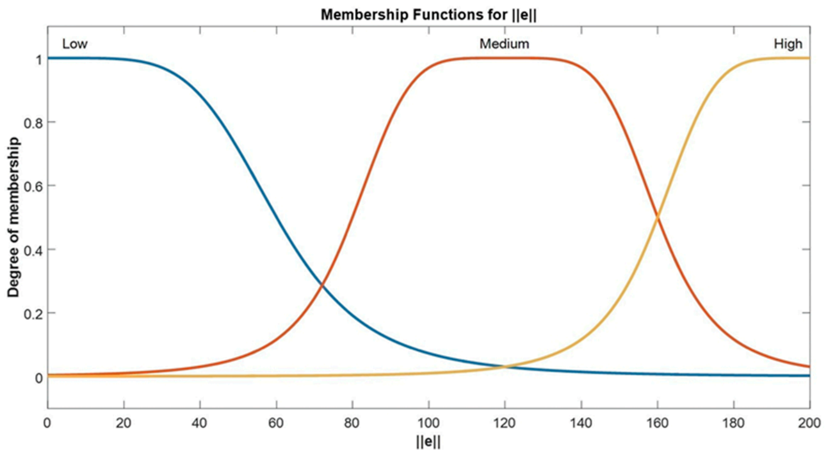


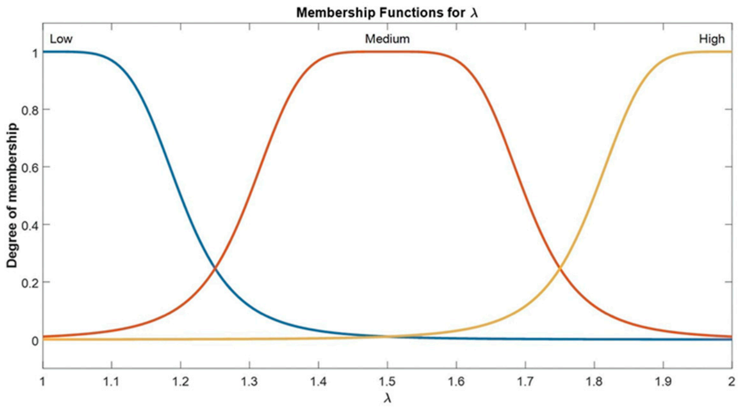
Figure 7. NN output of Rotor 1 30% LOE fault.

As the GS stage of the proposed system, a gain factor for torque and force signals in Equation (11), λ_a is defined using an FL unit as feature error norm, $\|e\|$ as input, and λ_a as output. The generalized bell MFs of this FL unit were selected to provide a smooth nonlinear surface, free of discontinuities. There are three MFs for each input and output. The rule base is modified in accordance with lessons learned through experiments that λ_a should be small, while the error norm is high to avoid discontinuities. The MFs and the surface between input and output are given in Figure 8.

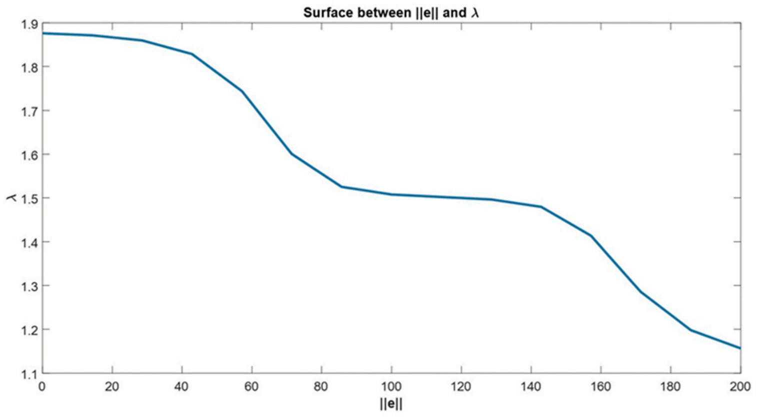
As stated in the section on practical disturbances for real-time VS systems, all systems can be exposed to feature noise and poor camera calibration [29,38]. When the system approaches convergence, which can be defined as very small feature errors, noise in the features may result in oscillations in the motion of the quadrotor. Two cases are discussed in the following subsections.



(**a**)



(**b**)

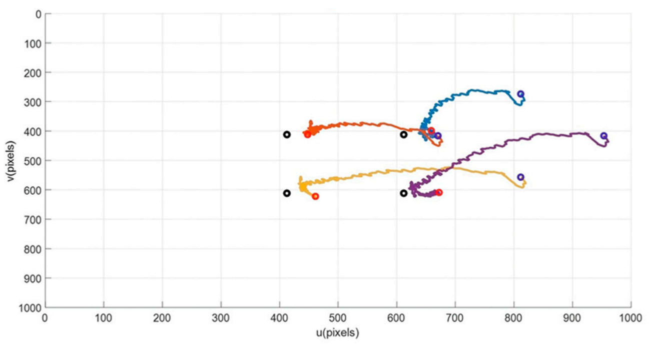


(**c**)

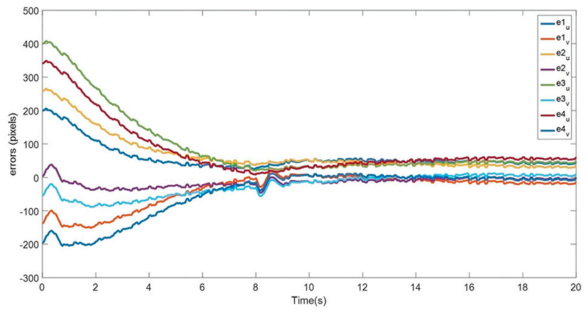
Figure 8. Membership functions and the surface of the gain factor FL unit: (**a**) $\|e\|$; (**b**) λ_a; (**c**) surface.

3.1. Case 1: Fixed Target Features under Noise

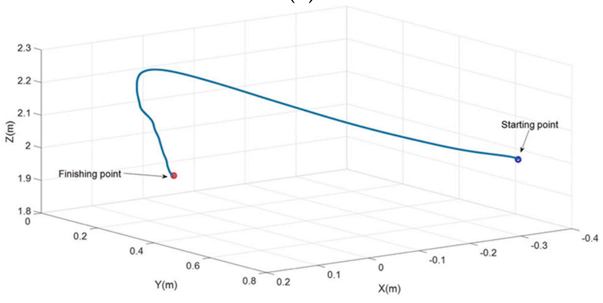
To show the performance and the robustness of the system, a fault at Rotor 1 with 30% LOE at t = 8 s and uniformly distributed random noise with five magnitude disturbing all feature points are considered. The results for this scenario are shown in Figure 9.



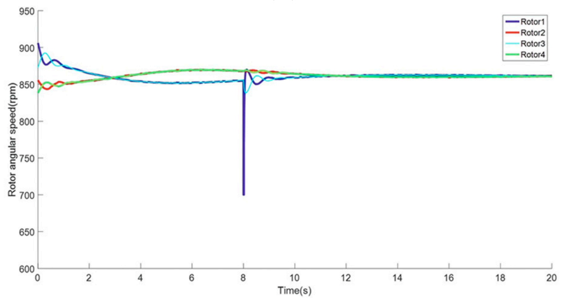
(**a**)



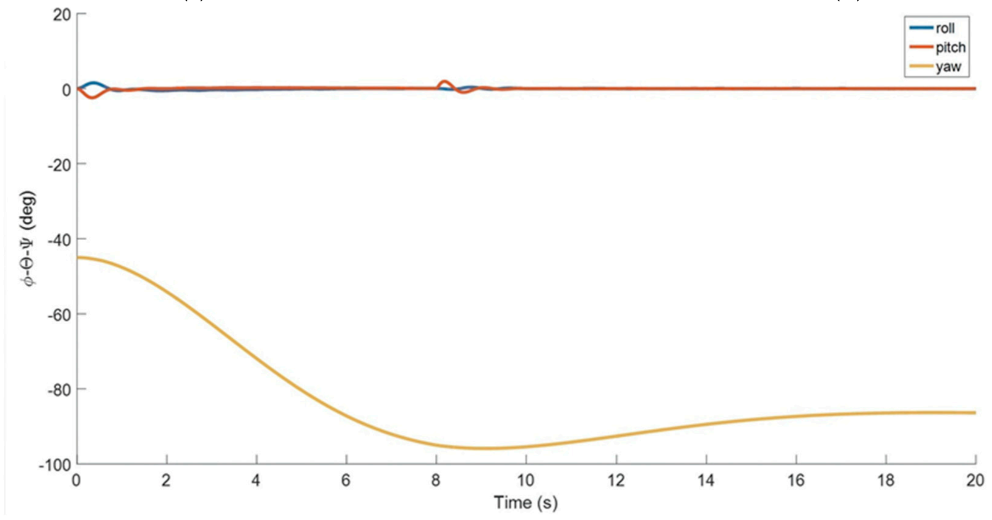
(**b**)



(**c**)



(**d**)



(**e**)

Figure 9. Result for the first case: (**a**) Feature trajectories; (**b**) feature errors; (**c**) trajectory of the quadrotor; (**d**) rotor speeds; (**e**) RPY.

In Figure 9a, the blue circles, the red circles, and the black circles show the starting, the finishing, and the target point features, respectively. The feature trajectories, also shown in Figure 9a, and the errors under feature noise, shown in Figure 9b, demonstrate the proposed IBVS system's convergence and resilience in the face of defined real-world problems. The fault approximator NN catches the fault at time instant 8.02 with an approximation of 36.28% LOE and fuzzy GS reconfigures the system in a small time interval, but a delay of 0.02 s in the fault diagnosis causes an abrupt change in Rotor 1 speed, as shown in Figure 9d. Furthermore, the fault causes steady-state errors in the features, as shown in Figure 9b, but the system converges under an actuator fault and the features are kept in FOV as an important practical positive. This is a tradeoff of the proposed FTC system but convergence is the main goal of an FTC system. The trajectory with the blue circle as the starting location and the red circle as the finishing location in Figure 9c and the RPY signals of the quadrotor in Figure 9e do not contain any abrupt changes which are good positives, especially for a reliable flight system.

3.2. Case 2: Moving Target Features under Noise

As a challenging scenario, it is assumed that the target features are moving in the *u*-*v* image plane with a sine shape. The starting feature points and feature noise characteristics are the same as in the first case and the goal features are defined as:

$$s_T^* = \begin{bmatrix} 612 + 50 \cdot \sin\left(\frac{t}{4}\right) & 412 + 50 \cdot \sin\left(\frac{t}{4}\right) & 412 + 50 \cdot \sin\left(\frac{t}{4}\right) & 612 + 50 \cdot \sin\left(\frac{t}{4}\right) \\ 412 + 10 \cdot t & 412 + 10 \cdot t & 612 + 10 \cdot t & 612 + 10 \cdot t \end{bmatrix} \quad (21)$$

The results for this scenario are shown in Figure 10. The feature colors in Figure 10a are the same as in Figure 9a. Additionally, cyan circles show the starting of goal features. The fault approximator shows its ability again and diagnoses the fault at 8.021 s with an approximation of 36.05% LOE.

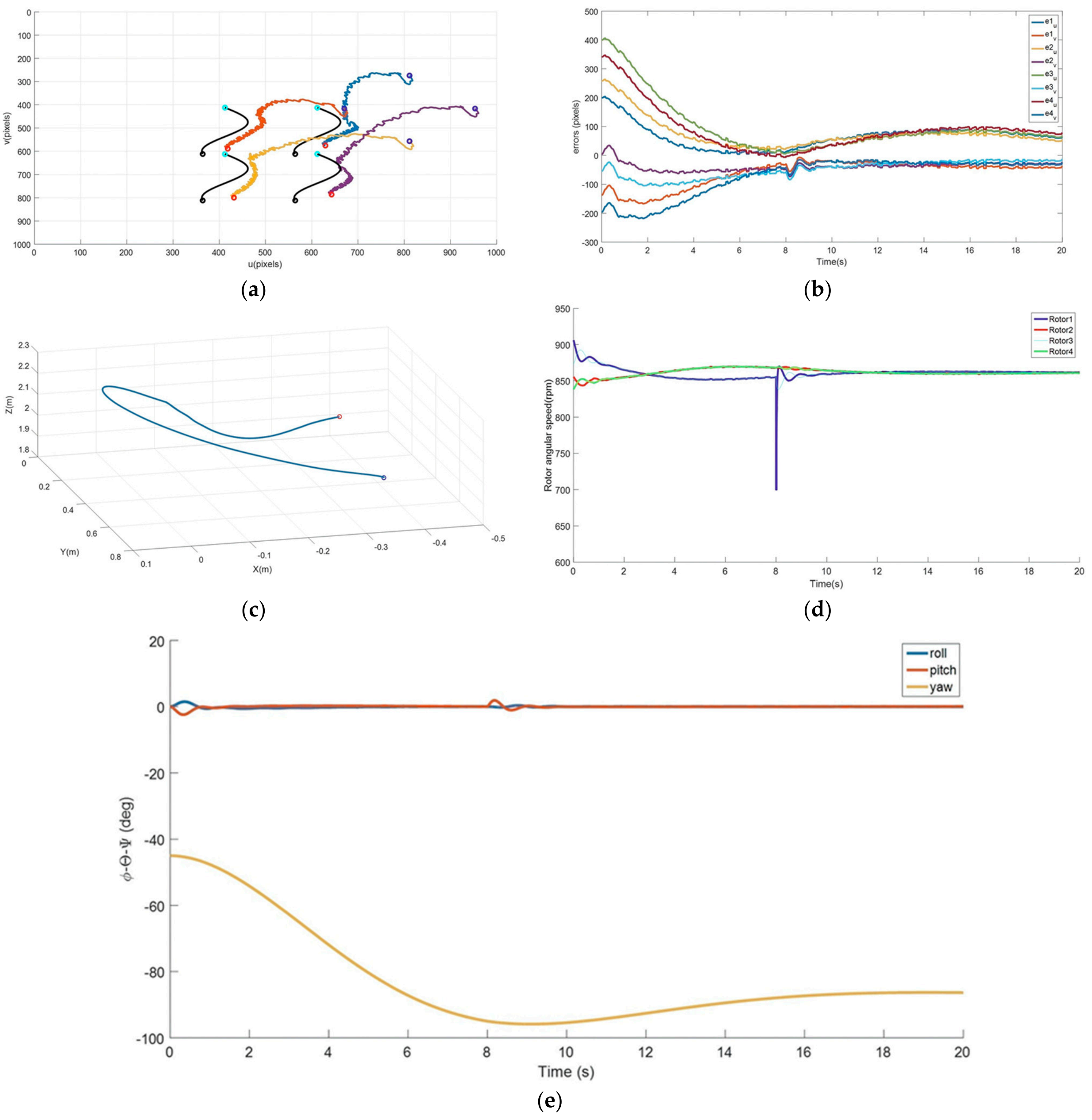


Figure 10. Results for the second case: (**a**) Feature trajectories; (**b**) feature errors; (**c**) trajectory of the quadrotor; (**d**) rotor speeds; (**e**) RPY.

It is clear that the proposed FTC system can track the moving features in Figure 10a, but the errors increase in an admissible amount, as shown in Figure 10b. Furthermore, the effect of rotor fault at Rotor 1 is obvious after the time instant at t = 8 s. After this time instant, it must be noted that the feature trajectories are kept in FOV while tracking moving targets, as in the first case, which is an important superiority. The trajectory followed by the quadrotor still does not contain any sudden maneuvers and this provides a reliable flight in realization. RPY motion and rotor angular speed characteristics are quite similar to fixed target results. It is a consequence of very low roll and pitch maneuvers with fixed yaw characteristics while tracking the moving targets.

4. Conclusions

Studies on fault detection and fault diagnosis issues often concentrate on quadrotor actuator faults by only using IMU or on-board sensor signals due to the fact that the inner and outer loop controllers are coupled according to RPY movements. As the main contribution to the literature, in this study, we utilize camera images and visual features to diagnose actuator faults of a quadrotor. Utilizing auxiliary signals from auxiliary sensors, such as cameras, may increase the likelihood of diagnosis without the requirement for hardware redundancy. Additionally, cameras can perform additional tasks such as visual SLAM or visual autolanding while also offering this support. The proposed system has two stages as an FTC system. First, the system approximates actuator faults that combine detection and diagnosis. NN, ELM, linear SVM, and LSTM are the candidates for this approximation. Although the competitors are quite strong, NN showed the best RMSE metric performance of 0.0120, which was four times smaller than the closest competitor. Then, an active FTC stage using fuzzy GS is designed and implemented to provide convergence. From the point of real-time field missions, a trained NN can be effective since embedded systems have superior instruction implementing speeds.

The system is tested for fixed and moving feature targets under feature noise to show robustness. Although steady-state feature errors arise, the system does not diverge and it shows sufficiency against actuator faults. Furthermore, the motion characteristics of the proposed FTC system under actuator faults are quite convincing, as high maneuvers do not occur and the features are kept in FOV (see Supplemental Files drones_fixed_targe.mat for fixed and drones_moving_target.mat for moving target scenarios). These two superiorities are required qualifications for an FTC-based VS system in practice.

In future studies, we plan to realize the proposed FTC system on a real quadrotor that is controlled by visual servoing. Furthermore, the study would utilize point features. Using shape features, maybe features of an AR code will be more attractive as a popular marker in the field of VS. In addition to actuator faults, component faults such as carried mass variation and sensor faults such as IMU biases will be addressed, and newly trained AI structures such as the adaptive neuro-fuzzy inference system (ANFIS) which combines black box and white box [39] will be tested to deal with all these types of faults in one structure.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/drones7020100/s1.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. Chaumette, F.; Hutchinson, S. Visual Servo Control. I. Basic Approaches. *IEEE Robot. Autom. Mag.* **2006**, *13*, 82–90. [CrossRef]
2. Chaumette, F.; Hutchinson, S. Visual Servo Control. II. Advanced Approaches. *IEEE Robot. Autom. Mag.* **2007**, *14*, 109–118. [CrossRef]

3. Agravante, D.J.; Claudio, G.; Spindler, F.; Chaumette, F. Visual Servoing in an Optimization Framework for the Whole-Body Control of Humanoid Robots. *IEEE Robot. Autom. Lett.* **2017**, *2*, 608–615. [CrossRef]
4. Pebrianti, D.; Kendoul, F.; Azrad, S.; Wang, W.; Nonami, K. Autonomous Hovering and Landing of a Quad-Rotor Micro Aerial Vehicle by Means of on Ground Stereo Vision System. *J. Syst. Des. Dyn.* **2010**, *4*, 269–284. [CrossRef]
5. Corke, P.I. *Robotics, Vision and Control; Fundamental Algorithms in MATLAB*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2017.
6. Collewet, C.; Marchand, E.; Chaumette, F. Visual Servoing Set Free from Image Processing. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 81–86. [CrossRef]
7. Malis, E.; Chaumette, F.; Boudet, S. 2 1/2 D Visual Servoing. *IEEE Trans. Robot. Autom.* **1999**, *15*, 238–250. [CrossRef]
8. Kallem, V.; Swensen, J.P.; Hager, G.D.; Cowan, N.J. Kernel-Based Visual Servoing. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 1975–1980.
9. Mahony, R.; Hamel, T. Image-Based Visual Servo Control of Aerial Robotic Systems Using Linear Image Features. *IEEE Trans. Robot.* **2005**, *21*, 227–239. [CrossRef]
10. Ceren, Z.; Altug, E. Image Based and Hybrid Visual Servo Control of an Unmanned Aerial Vehicle. *J. Intell. Robot. Syst.* **2012**, *65*, 325–344. [CrossRef]
11. Odile, B.; Mahony, R.; Guenard, N.; Chaumette, F.; Hamel, T.; Eck, L. Kinematic Visual Servo Control of a Quadrotor Aerial Vehicle. *IEEE Trans. Robot.* **2007**, *25*, 833–838.
12. Hamel, T.; Mahony, R. Image Based Visual Servo-Control for a Class of Aerial Robotic Systems. *Automatica* **2007**, *43*, 1975–1983. [CrossRef]
13. Plinval, H.; Morin, P.; Mouyon, P.; Hamel, T. Visual Servoing for Underactuated VTOL UAVs: A Linear, Homography-Based Framework. *Int. J. Robust Nonlinear Control* **2014**, *24*, 2285–2308. [CrossRef]
14. Mebarki, R.; Lippiello, V.; Siciliano, B. Nonlinear Visual Control of Unmanned Aerial Vehicles in GPS-Denied Environments. *IEEE Trans. Robot.* **2015**, *31*, 1004–1017. [CrossRef]
15. Abdessameud, A.; Janabi-Sharifi, F. Image-Based Tracking Control of VTOL Unmanned Aerial Vehicles. *Automatica* **2015**, *53*, 111–119. [CrossRef]
16. Zheng, D.; Wang, H.; Wang, J.; Zhang, X.; Chen, W. Toward Visibility Guaranteed Visual Servoing Control of Quadrotor UAVs. *IEEE/ASME Trans. Mechatron.* **2019**, *24*, 1087–1095. [CrossRef]
17. Zhang, K.; Shi, Y.; Sheng, H. Robust Nonlinear Model Predictive Control Based Visual Servoing of Quadrotor UAVs. *IEEE/ASME Trans. Mechatron.* **2021**, *26*, 700–708. [CrossRef]
18. Liu, H.; Liu, D.; Lyu, Y. Completely Distributed Time-Varying Formation Target Tracking for Quadrotor Team via Image-Based Visual Servoing. *IEEE Trans. Veh. Technol.* **2022**, *71*, 21–32. [CrossRef]
19. Xie, H.; Lynch, A.F. Input Saturated Visual Servoing for Unmanned Aerial Vehicles. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 952–960. [CrossRef]
20. Zhao, W.; Liu, H.; Lewis, F.L.; Valavanis, K.P.; Wang, X. Robust Visual Servoing Control for Ground Target Tracking of Quadrotors. *IEEE Trans. Control Syst. Technol.* **2020**, *28*, 1980–1987. [CrossRef]
21. Cao, Z.; Chen, X.; Yu, Y.; Yu, J.; Liu, X.; Zhou, C.; Tan, M. Image Dynamics-Based Visual Servoing for Quadrotors Tracking a Target with a Nonlinear Trajectory Observer. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 376–384. [CrossRef]
22. Jiang, J. *Robust Model-Based Fault Diagnosis for Dynamic Systems*; Kluwer Academic Publishers: Amsterdam, The Netherlands, 2002; Volume 38.
23. Zhang, Y.; Jiang, J. Bibliographical Review on Reconfigurable Fault-Tolerant Control Systems. *Annu. Rev. Control* **2008**, *32*, 229–252. [CrossRef]
24. Fourlas, G.K.; Karras, G.C. A Survey on Fault Diagnosis and Fault-Tolerant Control Methods for Unmanned Aerial Vehicles †. *Machines* **2021**, *9*, 197. [CrossRef]
25. Avram, R.C.; Zhang, X.; Muse, J. Quadrotor Actuator Fault Diagnosis and Accommodation Using Nonlinear Adaptive Estimators. *IEEE Trans. Control Syst. Technol.* **2017**, *25*, 2219–2226. [CrossRef]
26. Song, Y.; He, L.; Zhang, D.; Qian, J.; Fu, J. Neuroadaptive Fault-Tolerant Control of Quadrotor UAVs: A More Affordable Solution. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 1975–1983. [CrossRef] [PubMed]
27. Ren, X.L. Observer Design for Actuator Failure of a Quadrotor. *IEEE Access* **2020**, *8*, 152742–152750. [CrossRef]
28. Ma, H.J.; Liu, Y.; Li, T.; Yang, G.H. Nonlinear High-Gain Observer-Based Diagnosis and Compensation for Actuator and Sensor Faults in a Quadrotor Unmanned Aerial Vehicle. *IEEE Trans. Ind. Inform.* **2019**, *15*, 550–562. [CrossRef]
29. Tahri, O.; Chaumette, F. Point-Based and Region-Based Image Moments for Visual Servoing of Planar Objects. *IEEE Trans. Robot.* **2005**, *21*, 1116–1127. [CrossRef]
30. Haykin, S. *Neural Networks and Learning Machines*, 3rd ed.; Pearson: London, UK, 2011.
31. Platt, J.C. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*; Microsoft: Redmond, WA, USA, 1999.
32. Jang, S.R.; Sun, C.T. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*; Prentice-Hall: Hoboken, NJ, USA, 1997.
33. Huang, G.; Zhou, H.; Ding, X.; Zhang, R. Extreme Learning Machine for Regression and Multiclass Classification. *IEEE Trans. Syst. Man Cybern. B Cybern.* **2012**, *42*, 513–529. [CrossRef]
34. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]

35. Tang, Y.M.; Zhang, L.; Bao, G.Q.; Ren, F.J.; Pedrycz, W. Symmetric Implicational Algorithm Derived from Intuitionistic Fuzzy Entropy. *Iran. J. Fuzzy Syst.* **2022**, *19*, 27–44. [CrossRef]
36. Pounds, P.E.I. *Design, Construction and Control of a Large Quadrotor Micro Air Vehicle*; The Australian National University: Canberra, Australia, 2007.
37. Kingma, D.P.; Ba, J.L. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
38. Lizarralde, F.; Leite, A.C.; Hsu, L.; Costa, R.R. Adaptive Visual Servoing Scheme Free of Image Velocity Measurement for Uncertain Robot Manipulators. *Automatica* **2013**, *49*, 1304–1309. [CrossRef]
39. Reddy, S.; Panwar, L.K.; Panigrahi, B.K.; Kumar, R. Computational Intelligence for Demand Response Exchange Considering Temporal Characteristics of Load Profile via Adaptive Fuzzy Inference System. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 235–245. [CrossRef]