The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-021-07659-2

APPLICATION

Gestures-teleoperation of a heterogeneous multi-robot system

Kevin Braathen de Carvalho[1] · Daniel Khede Dourado Villa[2] · Mário Sarcinelli-Filho[2] · Alexandre Santos Brandão[1]

Received: 4 March 2021 / Accepted: 6 July 2021

Abstract
This work presents a solution for the teleoperation of a heterogeneous team of mobile robots. Regarding the team of robots, two possibilities are considered which are UAV-UGV (Unmanned Aerial Vehicle-Unmanned Ground Vehicle) and UAV-UAV. To execute this task, high-level gesture patterns are made in a remote station, and we proposed an easy-to-train Artificial Neural Network (ANN) classifier to identify the skeletal data extracted by an RGB-D (Red, Green, Blue-Depth) camera. Our classifier uses custom data to build the gesture patterns, allowing the use of smooth and intuitive gestures for the teleoperation of mobile robots. To validate our proposal, experiments were run using two off-the-shelf *Parrot AR.Drone 2* quadrotors and the differential drive platform *Pioneer 3-DX*. The results of such experiments allow concluding that the proposed teleoperation system is able to accomplish inspection/surveillance tasks, and it can be easily modified to similar applications, as emergency response or load transportation.

Keywords Gesture recognition · Teleoperation · Multi-robot systems · Heterogeneous squads · Mobile robotics

1 Introduction

Human-robot interaction can be defined as "a field of study dedicated to understand, design, and evaluate robotic systems for use by or with humans" [1]. The interaction can occur in two different ways: remote and proximate. They are largely influenced by the proximity of the human and the robot. Works like [2–5] bring good examples of how remote communication can be exploited in several applications, such as navigation in unhealthy or potentially dangerous environments [6], robot-assisted medical procedures [7], surveillance and inspection [8]. When robots and humans are in the same place, the interaction can be classified as proximate communication. Such a kind of interaction leverages different concerns such as trust between agents, studies of proximity, safety and other challenges [9–14]. Thus, this interaction expands the possibilities of applications, being useful in labors with domestic-service robots or during a human-robot cooperation in manufacturing tasks.

The communication between human and robot can be done by voice commands [15–17], peripherals [18–20] or high-level gestures [21–23]. In more recent years, the interest in control problem based on gesture-inputted commands has appeared more intensively in the literature. This enables the commands to be established directly by a human, without needing any input device, thus allowing the control to be done directly, through predefined gestures [24]. In order to capture relevant information to perceive and classify gestures or actions, and then to define which action should be executed, some works use depth image [25–27]. Such approach is a more affordable one due to the low cost of the suitable sensors. Good examples are the Microsoft Kinect and Intel RealSense visual sensors. Other works use pre-processed information or pre-process the depth image to extract skeletal data [28–30], in order to represent each

✉ Kevin Braathen de Carvalho
kevin.carvalho@ufv.br

Daniel Khede Dourado Villa
danielkdv@gmail.com

Mário Sarcinelli-Filho
mario.sarcinelli@ufes.br

Alexandre Santos Brandão
alexandre.brandao@ufv.br

[1] Graduate Program on Computer Science, Federal University of Viçosa, 36570-900, Viçosa, Minas Gerais, Brazil

[2] Graduate Program on Electrical Engineering, Federal University of Espírito Santo, 29075-910, Vitória, Espírito Santo, Brazil

Published online: 22 September 2021

gesture with less information, in comparison with full depth images.

The main contribution of this work is a practical solution for teleoperating a team of mobile robots accomplishing high-level predefined tasks. The operator in a remote station uses intuitive gestures which are related to high-level actions to be performed by the robot team. We predefined high-level missions for a team composed of two UAVs (Unmanned Aerial Vehicle) or a UAV and a UGV (Unmanned Ground Vehicle), resembling actions used in surveillance and inspection. In specific, the highlights of our proposal contributions are:

- A neural network classifier for gesture recognition that uses custom datasets to classify skeleton data extracted from an RGB-D camera. Since we avoid the use of open-access datasets for training, our proposal allows a high degree of customization, which is crucial from a practical perspective to build intuitive and easy-to-remember gestures to be translated to high-level actions for the robots.
- The proposed gesture recognition algorithm uses features with a small number of joints, which simplifies the ANN (Artificial Neural Network) classifier. So, due to its small size, it results in a classifier that is easier to add/modifier layers and with a short-time demanding for training new users, or retraining. Therefore, the developed framework can incorporate new tasks and gestures, according to the user's design.
- The proposal is validated in real-world experiments, and the results allow concluding the success in accomplishing the teleoperation of robots emulating inspection and surveillance tasks. It is worth mentioning that the gesture-teleoperation framework proposed here can be customized for agro-industrial applications or even for educational purposes, both serving as a laboratory for human-robot interaction with practical feedback for the end-user.

1.1 Related works

Machine Learning techniques have been used quite often to recognize gestures and actions. Hidden Markov Models can easily model the time evolution of a set of parameters to classify actions, using two different feature sets: one being based on projection histograms, which contains the pixels that moved in each row and column, and the other based on shape descriptors [31]. The attention model proposed in [32] explores the spatial and temporal discriminants to address the recognition from skeleton data. Such an approach relies on a Recurrent Neural Network (RNN) framework with Long Shot Term Memory (LSTM) units for classification. Following the neural network approaches, in [33] an attention-based hybrid Convolutional Neural Network (CNN) uses the full extent of the spatial and temporal features of images of a surface electromyography to classify gestures. Similarly, but now for action classification, a 3D-based CNN is implemented in [34] to extract features of depth image sequences and input them to a Support Vector Machine (SVM). Another approach is presented in [22], where inertial measurement units (IMU) get accurate information of the upper body's motion. Furthermore, the obtained data are segmented in static and dynamic blocks in a unsupervised sliding window approach and then compared with an ANN and a LSTM based classification. In such a context, this paper uses an ANN for classification. In our case, the images are replaced by the skeleton data which represent them. In other words, the sequence of input images become a simple feature vector, used as the input for the network.

Regarding robot in real-world applications, in several scenarios a single agent may not be able to acquire all the information required to properly execute a task. Therefore, multi-robot systems (MRS) have been researched and applied in different scenarios, such as surveillance, monitoring, inspection and rescue [35]. Indeed, MRS have different advantages when compared to single-robot systems. In particular, MRS can be more time-efficient, less prone to single-points of failure, and generally offer multiple capabilities, which can lead to a more effective solution to a given problem [36].

Multi-robot systems provide more flexibility and potentially more efficiency to solve certain problems. In particular, a heterogeneous squad makes profit from the capabilities of different kind of robots in the same task. A UAV-UGV squad has the perk of being able to provide a broader view of the location, thanks to the UAV being an agile flying vehicle, with high-maneuverability and the ability to hover. In such a situation the UGV can act as a movable recharging station for the UAV, since, it is common sense that flying robots has low autonomy in comparison with terrestrial ones. Another possible scenario is the UGV serving as a central station for the UAV, once UGVs can carry much more payload. Thus, additional sensors, storage and processing power can be onboard the UGV. In turn, an aerial vehicle can yield faster completion and more effective execution of surveillance/inspection tasks.

For safety reasons of human operators, a multi-robot system may be controlled from a remote station. Such technique is called teleoperation. In our case, it is performed by gestures, which allows an easy and intuitive way to guide robots. The main concern regarding teleoperation is the time delay imposed by the communication channels and its effects on feedback and control. The concepts of stability and transparency are normally used to discuss such effects, being the transparency the kinematic correspondence perceived by the operator of the robot and its environment

[37]. Ideally, it is desired that a teleoperated system should be fully transparent, giving to the remote operator the same experience of controlling the system locally. However, it is known that the stability and the transparency of a system are opposing characteristics [38]. One way to contour this problem is the use of high-level control actions instead of direct real-time operation. Actually, as the high-level control is a preset of control actions, as it can benefits from local feedback to its execution; in contrast to direct real-time operation, which depends on remote feedback. In such a context, the proposed framework performs the correspondence of an operator gesture and a task accomplished by an autonomous robot, addressing the remote assignment of surveillance missions for a squad of two robots. Once high-level control actions are used, the control feedback loops are locally closed, allowing improved robustness and performance compared to conventional teleoperated systems.

In the context of gestures being used for high-level control actions, the authors in [39] use logistic regression to classify up to 10 gestures to be assigned to high-level tasks on a small UAV fleet. The authors achieve over 95% overall accuracy and over 90% accuracy for each gesture class. Multimodal sensoring was used in [40] through IMU and mechanomyography to classify up to 8 gestures using a Convolutional Neural Network. The achieved accuracy of 5 out of 8 gestures is suitable for real-world applications. In the work [41], the authors use a collection of 8 static and 2 dynamic gestures and assigns them to high-level control actions to a UAV in a Search and Rescue scenario having over 94% accuracy for test data. The authors further improve the work in [42] adding 5 hand gestures to be used for more specific commands, rather than just the 10 body gestures. The classification is done by DNN (Deep Neural Networks). As we did in our proposal, all of these works use their own self-built datasets due to the specificity of their application.

To address the topics involved, the paper is hereinafter split in a few sections, starting with the gesture recognition strategy detailed in Section 2, from dataset creation to how the ANN classifier works. In the sequel, Section 3 describes the robots' models and controllers. Whereas Section 4 explains how each gesture class relates to the robots and their planning tasks, Section 5 shows some experiments run and discuss their results. Finally, Section 6 highlights our concluding remarks.

2 Gesture recognition

2.1 Dataset

To recognize action or gesture using machine learning techniques, one has often to rely on open datasets, such as [43], which commonly have two implications. First, they can have really large amount of data to process, making it unfeasible in some situations. Second, they provide low (or even none) flexibility if we want to use it for a specific proposal. In other words, the dataset might not have the actions or gestures that are interesting for certain applications. This section details the reason to create our own dataset and how it was conceived.

Kinect v2.0 is a worldwide sensor for robotics application. It was primarily designed to extract information from RGB-D cameras and deliver the skeleton joints for game entertainment. Knowing this, we use such an intrinsic ability to observe a human body and to describe it as high-level descriptors (the human's joint positions). In our case, Fourteen gesture classes were chosen to fit the goals of this work. They are depicted in Fig. 1, and are (A) *raise both hands and then lower the right one*, (B) *rotate both arms, ending with them in front of the chest, with the right hand slightly above the left*, (C) *raise both hands and then lower the left one*, (D) *draw a circle in front of the face using the right hand, keeping the left arm aside the body*, (E) *draw a circle in front of the face with the right hand, while raising the left arm to the height of the shoulder*, (F) *draw a circle in front of the face using the right hand, while raising the left arm to the height of the head*. Continuing, gestures (G)–(I) are the mirrored version of (D)–(F), obtained switching the arms. Besides, the following gestures are (J) *cross both arms forming an X in front of the chest*, (K) *move the right arm forward and then slide it to the left*, (L) *raise both arms to the height of the stomach and then raise the right hand*, (M) *raise both arms to the height of the stomach and then raise the left hand up* and (O) *move the left arm forward and then slide it to the right*.

As a comparison, the Brazilian Ministry of Defense establishes 29 gestures that an assistant can do to help to maneuver an aircraft [44]. Therefore, using 14 different gesture can be considered a reasonable amount of classes to deal with our proposal of surveillance and inspection missions performed by a heterogeneous robotics squad.

Classes (A) and (C) represent two UAVs, associated to the user's hands, with one of them landing. Class (B) represents a UAV on the top of a UGV. As for the circles in classes (D)–(I), they represent a surveillance task, indicating that a closed route should take place, with the position of the other arm is used to switch between three different cases. Classes (K) and (N) represent the user sliding (as for choosing an option in a virtual frontal menu). Classes (L) and (M) represent two robots and one of them is taking off.

In order to detect the beginning and the end of a gesture, an energy trigger approach analyzes the movement of the arms joints in a fixed window of f frames. If the joints' velocity reaches an empirically threshold l, the last d previous windows are stored, as well the current frames

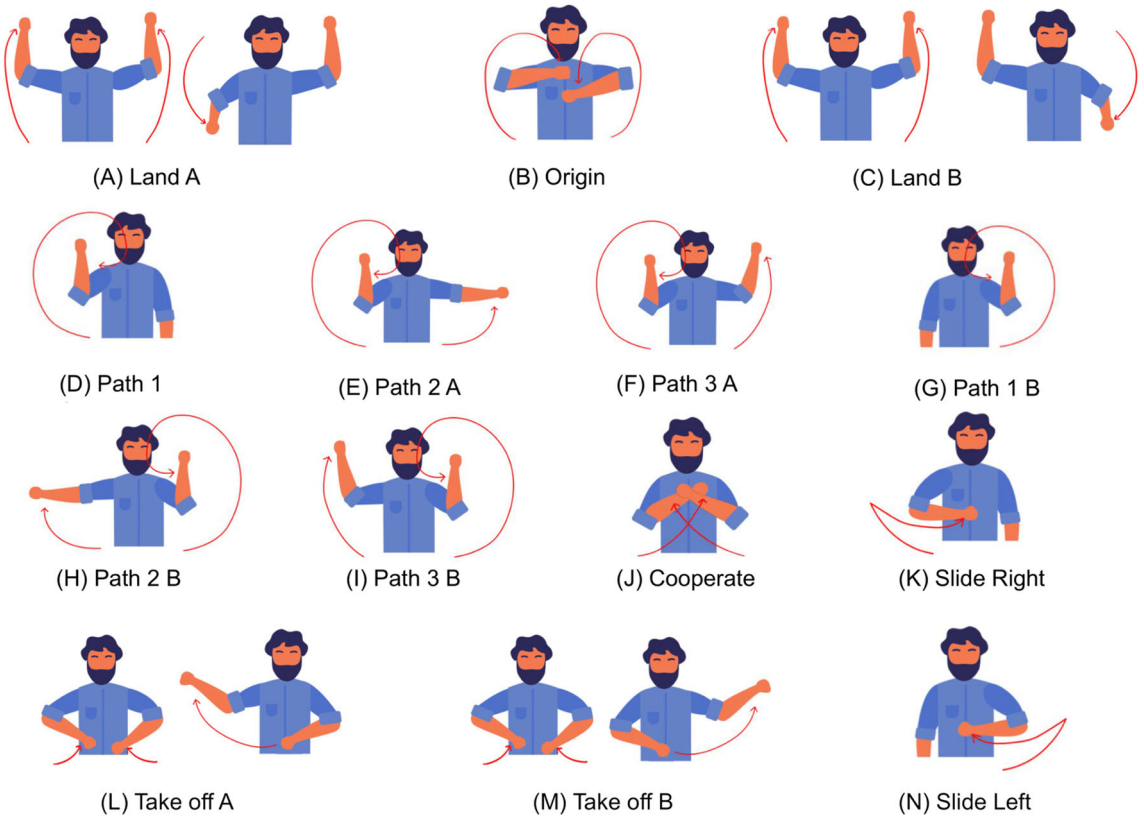
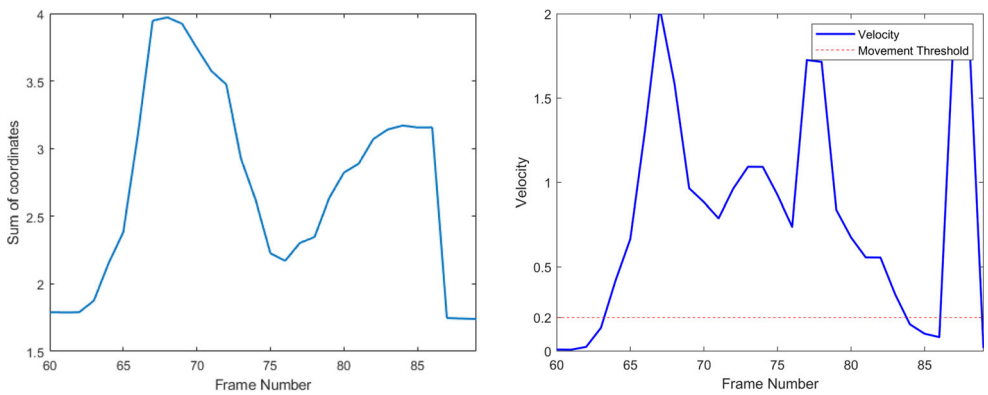


Fig. 1 Gesture classes on the dataset with a short description of their task functionalities

until the discrete-time derivative of the joints' position stays below the threshold for t ms. The values used for f, l, d and t were 3, 0.2, 0 and 100, respectively. Figure 2a shows the sum of all coordinates along the gesture (D) and Fig. 2b shows its discrete-time derivative, in absolute values, as well as the adopted threshold (red dashed line). The gesture is considered finished, if the absolute velocity is lower than the threshold l for t ms or more. The spike after 85th frame is due to the user moving his/her arms to the starting position. In order to avoid triggering a new gesture classification, in this case, a time interval is set to ignore the movements immediately after the classification of a gesture. Shortly before and after making a gesture, zones of silence are observed. In summary they are regions with movements of low amplitude.

Furthermore, an emergency shutdown gesture was added, based on the same energy trigger. If the velocity of the joints in this fixed-time window surpasses a higher threshold of 1.8 for 300 ms, then an emergency shutdown flag is activated.

Fig. 2 Gesture recognition strategy



(a) Sum of all arm joint coordinates

(b) Collective velocity, in absolute value.

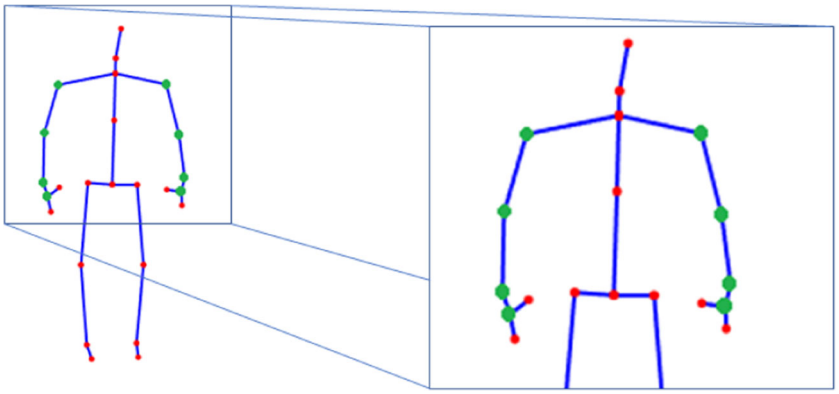


Fig. 3 Skeleton joints, highlighting those selected for classification

Kinect v2.0 sensor can operate at 30 FPS (frames per second), but here a 15 FPS capture rate was chosen. It prevents operations close to the sensor limits and mainly reduces the computation effort required by the gesture recognition algorithm.

Each frame consists in a discrete group of features, stored as

$$\mathbf{F_k} = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & z_i \end{bmatrix}, \tag{1}$$

where F_k is the feature matrix of the kth frame, the columns represent the 3-D Cartesian coordinates of the ith skeleton joint, with $i = 1, 2, \cdots, 25$ and $\mathbf{k}$ indicating a gesture sample.

To the eyes of an observer, a gesture done in different places in a room would look the same. However, it does not occur when these actions are captured by the Kinect v2.0 sensor. Indeed samples may have totally different coordinates for each skeleton joint, even though the gestures belong to the same class. To avoid such Cartesian displacement, all the captured skeleton data are centralized in reference to the coordinates of the left shoulder (adopted as a body reference for our research group, without loosing generality). After being centralized, the complete action is stored as the subsequent concatenation of the feature matrix, given by

$$\mathbf{A}_{m,n} = [\mathbf{F}_1\ \mathbf{F}_2\ \ldots\ \mathbf{F}_k] \tag{2}$$

where $\mathbf{A}_{m,n}$ is the action matrix containing the features, m is the class label, n is the sample's number, and k is the number of frames in the sample.

The dataset created contains 70 action samples of each gesture class, resulting in 980 samples.

2.2 Gesture recognition algorithm

This work uses a non-complex neural network, with a few number of neurons and layers. Despite simple, it can learn how to classify different gestures classes with high precision rates, since their inputs have been well preprocessed. Kinect v2.0 sensor provides the raw features, highlighted in the left part of Fig. 3, for the 8 joints used for classification. They are the shoulder, the elbow, the wrist and the hand joints. It is worth mentioning that these joints were chosen for being the body parts that move most in the selected gestures in this work, as illustrated in Fig. 1.

The next step is to reduce the dimension of the action matrix from $8 \times M$, where M is three times the number of frames of the sample under analysis, generating a set of just a few features. In other words, our approach consists in getting the principal components of $\mathbf{A} \cdot \mathbf{A}^\top \in \mathcal{R}^{8\times 8}$, given by the eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_8$.

Therefore, the vector $\Lambda = [\lambda_1\ \lambda_2\ \ldots \lambda_8]^T$ is the input of the ANN, whose configuration is one hidden layer with only 30 neurons, with all layers fully connected, and 14 outputs (one for each class). The activation function was the Sigmoid tangent for the hidden layer and softmax function for the classification layer. As for the training method, it was used the Bayesian Regularization, with Levenberg-Marquadt Optimization. Figure 4 illustrates a pipeline flowchart summarizing our classification subsystem.

3 Modeling and control of the robots

To perform the missions, reference control signals for navigation is delivered individually to the robots into the squad. The vehicles' dynamic model and the closed-loop

Fig. 4 Pipeline flowchart for the gesture recognition algorithm

feedback control law used to achieve the desired navigation for the UAV and the UGV is described in the following subsections.

3.1 UAV controller

The mathematical model for quadrotors has been extensively covered in the literature [45, 46]. Given the translational coordinates of the quadrotor as $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$ and its attitude described by the vector $\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$, which contains the roll, pitch and yaw angles, both related to $\langle w \rangle$, the Newton-Euler dynamics correspondent to the quadrotor can be written as

$$\begin{aligned} m\ddot{x} &= (c_\psi s_\theta + s_\psi c_\theta s_\phi)u_1 - d_1\dot{x} \\ m\ddot{y} &= (s_\psi s_\theta - c_\psi c_\theta s_\phi)u_1 - d_2\dot{y} \\ m\ddot{z} &= (c_\phi c_\theta)u_1 - mg - d_3\dot{z} \\ I_{xx}\ddot{\phi} &= u_{2,\phi} + (I_{zz} - I_{yy})\dot{\theta}\dot{\psi} - d_4\dot{\phi} \\ I_{yy}\ddot{\theta} &= u_{2,\theta} + (I_{xx} - I_{zz})\dot{\phi}\dot{\psi} - d_5\dot{\theta} \\ I_{zz}\ddot{\psi} &= u_{2,\psi} + (I_{yy} - I_{xx})\dot{\phi}\dot{\theta} - d_6\dot{\psi} \end{aligned} \tag{3}$$

where $s(\cdot)$ and $c(\cdot)$ represents $\sin(\cdot)$ and $\cos(\cdot)$, g is the gravity acceleration, m is the mass of the quadrotor, $\mathbf{I}$ is its matrix of moments of inertia, $\mathbf{d} = [d_1, ..., d_6]^\top$ represent drag coefficients, and $\mathbf{u} = [u_1\ \mathbf{u}_2]$ is the low-level thrust and torque control inputs.

As explained in [47], aerial vehicles with embedded autopilots use Eq. 3 to achieve attitude control, enabling the quadrotor to be controlled directly by high-level commands of desired roll angle u_ϕ, desired pitch angle u_θ, desired altitude rate $u_{\dot{z}}$, and desired yaw rate $u_{\dot{\psi}}$. Using this architecture and considering applications at moderate flight speeds, a near-hover linearization can be applied to the dynamics of the quadrotor in Eq. 3. Such a simplification performs well for roll and pitch angles up to approximately 30°, corresponding to linear velocities of approximately 1.5–2 m/s [48]. Upon such considerations, one can consider $\sin(\theta) \approx \theta$, $\sin(\phi) \approx \phi$, $\cos(\theta) \approx \cos(\phi) \approx 1$, and $u_1 \approx mg$, and the translational dynamics in Eq. 3 can be written as

$$\begin{aligned} \ddot{x} &= \left(c_\psi K_\theta u_\theta + s_\psi K_\phi u_\phi\right) g - \frac{d_1}{m}\dot{x} \\ \ddot{y} &= \left(s_\psi K_\theta u_\theta - c_\psi K_\phi u_\phi\right) g - \frac{d_2}{m}\dot{y} \\ \ddot{z} &= \frac{K_{\dot{z}} u_{\dot{z}}}{m} - g - \frac{d_3}{m}\dot{z} \end{aligned} \tag{4}$$

Exploiting the embedded autopilots, which is available for the quadrotor used in this work, the stabilization of the vehicle is left as responsibility of its firmware, thus modeling the UAV dynamic response just as a function of high-level translational control signals. Thus, a near-hover model for the quadrotor can be written in the linear form [47, 49]

$$\mathbf{u} = (\mathbf{F}\mathbf{K}_u)^{-1}(\ddot{\mathbf{x}}_{ref} + \mathbf{K}_{\dot{x}}\dot{\mathbf{x}}), \tag{5}$$

where $\mathbf{F}$ is a rotation matrix relating the coordinate systems $\langle w \rangle$ and $\langle b \rangle$, only dependent of ψ, $\mathbf{x} = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^\top$ are the displacements in frame $\langle b \rangle$ and yaw heading. The matrices $\mathbf{K}_u$ and $\mathbf{K}_{\dot{x}}$ are diagonal matrices containing, respectively, the dynamic and drag parameters for the model. The vector $\mathbf{x}_{ref}$ contains the reference control signal for a desired trajectory, and $\mathbf{u} = \begin{bmatrix} u_\theta & u_\phi & u_{\dot{z}} & u_{\dot{\psi}} \end{bmatrix}^\top$ is the vector of high-level commands, whose entries are all in the interval $[-1.0, +1.0]$.

The model parameters in $\mathbf{K}_u$ and $\mathbf{K}_{\dot{x}}$ are obtained through an identification procedure, as explained in [47, 49]. Knowing such parameters one can implement a feedback linearization controller, such that the reference signal $\ddot{\mathbf{x}}_{ref}$ can be obtained using PD feedback plus a feedforward term, which corresponds to

$$\mathbf{u} = (\mathbf{F}\mathbf{K}_u)^{-1}(\ddot{\mathbf{x}}_{des} + \mathbf{K}_d\dot{\tilde{\boldsymbol{x}}} + \mathbf{K}_p\tilde{\boldsymbol{x}} + \mathbf{K}_{\dot{x}}\dot{\boldsymbol{x}}), \tag{6}$$

where $\mathbf{x}_{des}$ is a trajectory to be tracked, $\tilde{\boldsymbol{x}} = \mathbf{x}_{des} - \mathbf{x}$, $\dot{\tilde{\boldsymbol{x}}} = \dot{\mathbf{x}}_{des} - \dot{\mathbf{x}}$, and $\mathbf{K}_d$ and $\mathbf{K}_p$ are positive definite gain matrices.

3.2 UGV controller

Similar to the UAV, the UGV used in this work also has a embedded low-level controller that drives the two motors of the vehicle, enabling the use of high-level commands of linear and angular velocities. To deal with the dynamic effects caused by inertia and friction, a dynamic model first introduced in [50] and updated in [51] was used, which is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\psi - a\omega\sin\psi \\ v\sin\psi + a\omega\cos\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v \\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\theta_1}v_{ref} \\ \frac{1}{\theta_1}\omega_{ref} \end{bmatrix}. \tag{7}$$

The UGV translational coordinates and orientation are the entries of the vector $\mathbf{x} = \begin{bmatrix} x & y & \psi \end{bmatrix}^\top$. The control signals are the linear and angular velocities represented as the vector $\mathbf{u} = \begin{bmatrix} v & \omega \end{bmatrix}^\top$. The non-zero constant a is the distance from the origin of the body axes (the baseline linking the two driven wheels) to the point of interest for control. In addition, v_{ref} and ω_{ref} are the reference values for linear and angular velocities.

The model in Eq. 7 includes a set of parameters θ_i, $i = 1, \cdots, 6$, which are related to some physical parameters of the robot, such as its mass, moments of inertia, the electrical resistance of its motors, electromotive and torque constants,

radius of the wheels, and so on. All these parameters and their description can be found in [51], where one can also find a procedure to identify such parameters, once getting them through their mathematical description is quite difficult.

By rearranging Eq. 7, the kinematic model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\cos\psi & -a\sin\psi\\ \sin\psi & a\cos\psi\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} = \mathbf{Ku}, \text{ with } \dot{\psi} = \omega, \tag{8}$$

is obtained, as well as its dynamic model, which is given by

$$\begin{bmatrix}\theta_1 & 0\\ 0 & \theta_2\end{bmatrix}\begin{bmatrix}\dot{v}\\ \dot{\omega}\end{bmatrix} + \begin{bmatrix}\theta_4 & -\theta_3\omega\\ \theta_5\omega & \theta_6\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}v_{ref}\\ \omega_{ref}\end{bmatrix}, \tag{9}$$

or by the compact form

$$\mathbf{M\dot{u}} + \mathbf{Cu} = \mathbf{u}_{ref}, \tag{10}$$

where $\mathbf{K}$ is the direct kinematics matrix, $\mathbf{M}$ and $\mathbf{C}$ are the Inertia and Coriolis matrices, respectively.

Using inverse kinematics and considering a desired trajectory, the reference velocity command can be defined as

$$\mathbf{u}_{vel} = \mathbf{K}^{-1}(\dot{\mathbf{x}}_{des} + \mathbf{K}_p \tanh(\tilde{\mathbf{x}})), \tag{11}$$

where $\mathbf{K}_p$ is a positive definite gain matrix. The perfect velocity tracking assumption, i.e., $v \equiv v_{des}$ and $\omega \equiv \omega_{des}$, is normally made for low-speed applications or fast dynamic robots. However, this assumption does not hold in practice for our UGV under high-speed movements. To contemplate these applications and to evaluate a more realistic scenario, a dynamic compensation module is adopted.

Considering Eq. 10, the proposed dynamic compensation law given by

$$\mathbf{u}_{ref} = \mathbf{M}\left(\dot{\mathbf{u}}_{vel} + \kappa(\mathbf{u}_{vel} - \mathbf{u}) + \mathbf{Cu}\right) \tag{12}$$

guarantees the tracking of the desired velocities considering the modeled dynamic effects. The gain κ is a positive definite matrix and $(\mathbf{u}_{vel} - \mathbf{u})$ is the velocity tracking error. The control signal $\dot{\mathbf{u}}_{vel}$ is obtained by numeric differentiation.

It should be noted that the saturation of the physical actuators introduces unpredicted non-linearities in the system, which may cause instability. The function $\tanh(\cdot)$ is used in Eq. 11 to address this issue, saturating the control signals for large position errors. Although a non-linearity is inserted in the control law, it is considered in the stability analysis of the system, thus not introducing any unpredicted instability. The advantage of using it is to prevent the saturation of the robot actuators, guaranteeing a good behavior even for large errors $\tilde{\boldsymbol{x}}$. Furthermore, the choice for $\tanh(\cdot)$ over other saturation functions, such as $\text{signal}(\cdot)$, is because $\tanh(\cdot)$ is differentiable and smooth, avoiding fast commutations and chattering effect.

3.3 Stability analysis of the controllers

The controllers presented for the UAV and UGV were designed using the feedback linearization technique. Accordingly, perfect knowledge of the models of the vehicles were assumed in Eq. 6 for the UAV and in Eq. 12 for the UGV. Under these assumptions, the proposed controllers makes the system asymptotically stable when closing the loop.

Proof The dynamics of the error, when the designed controllers are used to close the loop, obtained inserting the controller Eq. 6 into Eq. 5 and Eq. 12 into Eq. 10, becomes

$$\ddot{\tilde{\mathbf{x}}} + \mathbf{K}_d\dot{\tilde{\mathbf{x}}} + \mathbf{K}_p\tilde{\boldsymbol{x}} = \mathbf{0}. \tag{13}$$

To demonstrate the system stability, the Lyapunov candidate function

$$V(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}) = \frac{1}{2}\tilde{\boldsymbol{x}}^\top\mathbf{K}_p\tilde{\boldsymbol{x}} + \frac{1}{2}\dot{\tilde{\boldsymbol{x}}}^\top\dot{\tilde{\boldsymbol{x}}} < 0 \quad \forall\, \dot{\tilde{\boldsymbol{x}}}, \tilde{\boldsymbol{x}} \neq \mathbf{0} \tag{14}$$

is adopted. Taking its first time derivative and replacing Eq. 13, one gets

$$\dot{V}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}) = \tilde{\mathbf{x}}^\top\mathbf{K}_p\dot{\tilde{\mathbf{x}}} + \dot{\tilde{\mathbf{x}}}^\top(-\mathbf{K}_d\dot{\tilde{\mathbf{x}}} - \mathbf{K}_p\tilde{\mathbf{x}}) \tag{15}$$

$$\dot{V}(\tilde{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}}) = -\dot{\tilde{\mathbf{x}}}^\top\mathbf{K}_d\dot{\tilde{\mathbf{x}}} < 0 \quad \forall\, \dot{\tilde{\mathbf{x}}} \neq \mathbf{0},$$

which demonstrates that the system is stable. Applying the theorem of LaSalle for autonomous systems, one can see that the largest invariant set contained in the set $\{\tilde{\boldsymbol{x}} : \dot{V} = 0\}$ is $\left[\dot{\tilde{\mathbf{x}}}\ \tilde{\mathbf{x}}\right]^\top = \left[\mathbf{0}\ \mathbf{0}\right]^\top$. Therefore, the tracking error origin is asymptotically stable, which means that $\dot{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}} \to \mathbf{0}$ when $t \to \infty$. □

Table 1 Surveillance paths

Path-shape	Description
Circle	$\mathbf{x} = v_{path}\left[0.75\cos(k)\ \ 0.75\sin(k)\right]^\top$, with $k(n) = 0\ldots 2\pi,\ n = 100$
Diamond	$\mathbf{x} = v_{path}\,0.75\left[\begin{array}{ccccc} 1-k & -k & -1+k & k \\ -k & -1+k & & 1-k \end{array}\right]^\top$, with $k(n) = 0\ldots 1,\ n = 100$
Lemniscate	$\mathbf{x} = v_{path}\left[0.75\sin(k)\ \ 0.75\cos(0.5k)\right]^\top$, with $k(n) = 0\ldots 4\pi,\ n = 100$

Table 2 Gesture classes and their actions

Gesture	Command	Action
(A)	Land/Stop robot A	Trigger the landing protocol for a UAV.
(B)	Return to origin	Call both robots to return to their starting position.
(C)	Land/Stop robot B	Triggers the landing protocol for a UAV or the stop command for a UGV.
(D)	Surveillance path 1A	Requests robot A to perform a circle-shape path using its current waypoint as reference.
(E)	Surveillance path 2A	Requests robot A to execute a diamond-shape path using its current waypoint as reference.
(F)	Surveillance path 3A	Requests robot A to follow a lemniscate-shape path using its current waypoint as reference.
(G)	Surveillance path 1B	Requests robot B to perform a circle-shape path using its current waypoint as reference.
(H)	Surveillance path 2B	Requests robot B to execute a diamond-shape path using its current waypoint as reference.
(I)	Surveillance path 3B	Requests robot B to follow a lemniscate-shape path using its current waypoint as reference.
(J)	Cooperative mode	Change the reference point of robot A and send it to a position above robot B, as regarding a UAV-UAG squad, or send it to the side of robot B, as regarding a UAV-UAV squard.
(K)	Slide to the right	Sends the robot A to its next waypoint.
(L)	Takeoff/Start robot A	Takeoff/Start robot A: triggers the takeoff command to robot A.
(M)	Takeoff/Start robot B	Triggers the takeoff command to robot B, as regarding a UAV, or the start engine command, as regarding a UGV.
(N)	Slide to the left	Sends the robot B to its next waypoint.
—	Emergency Gesture	Triggers a flag that shutdowns both robots simultaneously.

In real-world applications, the assumption of perfect knowledge of the models and robots not subjected to disturbances is bold and unrealistic. In these cases, the use of adaptive or robust controllers for UAVs [52, 53] and UGVs [54, 55] is recommended. However, it is out of the scope of this work to discuss such controllers. Nevertheless, they can be straightforwardly incorporated to the proposed framework.

4 Task planning

A robot can accomplish a surveillance task after travelling by a set of waypoints or after following a predefined path. In this work, we choose a circle-, diamond- and lemniscate-shape paths. Each one is better described in Table 1. For all of them, we consider $z_{des} = 1$ m and $v_{path}^{UAV} = 0.7$ m/s if the path is followed by a UAV, and $v_{path}^{UGV} = 0.3$ m/s if the path if followed by a UGV.

The selected gestures represent high-level sub-tasks performed by either a UAV-UAV homogeneous squad or a UAV-UGV heterogeneous formation. For the heterogeneous squad, robot A is the UAV and robot B is the UGV. In turn, for the homogeneous team, robots A and B are determined beforehand. The gestures characterized in Section 2 have their associated commands detailed on Table 2.

5 Experiments and results

5.1 Experimental setup

The proposed algorithms were validated experimentally using the Pioneer 3-DX UGV, and the AR.Drone Parrot

Fig. 5 Illustrative representation of the motion capture system and robots used in the experiments [56]

2.0 UAV. The algorithms run in an off-board station, at a rate of 30 Hz, acquiring the poses of the vehicles through an *OptiTrack* motion capture system configured with eight cameras, and computing the reference control signals that are sent to the robots via ROS[1] (Robot Operating System). Figure 5 exhibits our experimental setup.

To control the UAV, its model parameters described in Eq. 6 are obtained by an identification method similar to the one adopted in [47, 57]. The obtained parameters are $\mathbf{K}_u = \text{diag}(14.72, 6.23, 2.65, 0.37)$ and $\mathbf{K}_{\dot{x}} = \text{diag}(0.27, 0.53, 2.57, 1.52)$. The UAV PD-controller gains are $\mathbf{K}_p = \text{diag}(2, 2, 3, 5)$ and $\mathbf{K}_d = \text{diag}(2, 2, 2, 5)$. For the UGV, the model parameters take from [51] are $\boldsymbol{\theta} = \begin{bmatrix} 0.53 & 0.22 & -0.01 & 0.95 & -0.08 & 1.05 \end{bmatrix}^\top$. The UGV PD-controller gains are $\mathbf{K}_p = \text{diag}(0.7, 0.7)$ and $\boldsymbol{\kappa} = \text{diag}(0.2, 0.2)$.

The robots use a carrot following algorithm to track the references. A feedback control guides them from their current position to the nearest point onto the path. After that the strategy sets their desired velocity v_{path} tangential to the predefined path (selected from Table 1). On the other hand, a positioning controller guides the robots when they must reach a set of waypoints.

5.2 Gesture classification

In order to validate the Gesture Classifier proposed in this work, two different tests where performed. The first one is the training using 10-fold cross validation on our dataset. This approach indicates how accurately the neural network learns the patterns of each class of gesture. The second one is an online test, in which the user prompts each gesture 30 times in a sequence previously unknown. This test aims to confirm that the dataset is not biased and leads to a network capable of accurately classifying gestures not included in the dataset. The results of the first and second tests are shown in Tables 3 and 4 respectively.

Using 10-fold cross-validation, a 98% overall accuracy was achieved, with a low percentage of false negatives and false positives. The highest percentage of false negative classifications occurred for gestures (A) and (C), which are symmetric. The highest rate of false positives occurred in the gestures B and C. In the first case classes (C),(G),(I) and (L) were falsely selected, and the latter had its false positive rate higher due to how much it is confused with gesture (A). The overall rate of right classifications, 98%, is a great accuracy, and suggests that the adopted classifier is suitable for real world applications.

The online tests show an overall accuracy of 96.6%, which is slightly worse than the results achieved in the offline training. The highest false negative rate was found in class (B). This could be due to the fact that the gesture ends with both hands in front of the user's chest, which leads to some oscillation on the joint locations due to how the Kinect v2 sensor extracts them, which could lead to the derivative surpassing the movement threshold when it should not. Therefore, more frames than usual are captured for the gesture and this could result to different eigenvalues after executing the mathematical manipulations. The false negatives on gesture contributed heavily on the false positive rates of gesture (L). However, an overall accuracy of 96.6% is still well suited for use in real world applications.

To endorse our findings, a comparison between relevant works is found in Table 5. Three out of four have over 94% average accuracy on test data. On practical tests, [40] had 94.38% using the 5 best gestures. Our work shows compatible results with these papers, as we had 98% average accuracy in offline tests and 96/6% in online tests. Regarding dataset size, it is possible to see that our proposal required substantially fewer samples to obtain efficient performance. All of these works use self-built datasets, therefore a smaller amount of samples is desired for the training of gestures and new users. Also, it is possible to see that neural network approaches are well suited for this kind of problem, given that three out of four works, as well as ours, used a neural network-based solution for gesture classification.

5.3 UAV and UGV experiments

Three experiments were run to illustrate the functioning human-robot interaction and the robot control performed using the system here proposed. In all the experiments the user performs the gestures in a separate room without an explicit visual feedback of what was happening in the room where the robots were. Such setup clarifies the idea of a Gesture-Teleoperation of a multi-robot system.

5.3.1 Experiment 1

In the first experiment a heterogeneous UAV-UGV squad was used. The UAV represents the robot A and the UGV the robot B. The video footage of the whole experiment can be found in https://youtu.be/fu5zbrB7HxE, where the sequence of actions can be checked. Table 6 indicates the user's commands, its aiming in the task and time stamp in which it starts in the experiment and the video separately. In Table 6 and the following ones, "Seq" indicates the sequence of the events, "Time" indicates the moment the action starts in the experiment and "Snapshot" indicates the time instant in which the action starts in the video.

This experiment shows that with the implemented gesture recognition and control algorithms one can successfully coordinate an heterogeneous squad to individually or jointly patrol and inspect an area, with the possibility that the

[1] See https://www.ros.org/

Table 3 Confusion matrix for the ANN using 10-fold cross validation

Desired classes															
Predicted classes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**FP**
A	92.8	-	1	-	-	-	-	-	-	-	-	-	-	-	1
B	-	97.2	2.8	-	-	-	1.4	-	2.8	-	-	1.4	-	-	8
C	7.2	-	90	-	-	-	-	-	-	-	-	-	-	-	7.4
D	-	1.4	-	100	-	-	-	-	-	-	-	-	-	-	1.4
E	-	-	1.4	-	100	-	-	1.4	-	-	-	-	-	-	2.7
F	-	-	-	-	-	100	-	-	-	-	-	-	-	-	-
G	-	-	-	-	-	-	98.6	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	-	98.6	-	-	-	-	-	-	-
I	-	-	2.8	-	-	-	-	-	95.8	-	-	-	-	-	2.8
J	-	-	-	-	-	-	-	-	-	100	-	-	-	-	-
K	-	-	-	-	-	-	-	-	-	-	100	-	-	-	-
L	-	1.4	1.4	-	-	-	-	-	1.4	-	-	98.6	-	-	4
M	-	-	-	-	-	-	-	-	-	-	-	-	100	-	-
N	-	-	-	-	-	-	-	-	-	-	-	-	-	100	-
FN	7.2	2.8	10	-	-	-	1.4	1.4	4.2	-	-	1.4	-	-	98

Table 4 Confusion matrix for the ANN online tests

Desired classes															
Predicted classes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**FP**
A	100	-	6.8	-	-	-	-	-	-	-	-	-	-	-	6.3
B	-	90	-	-	-	-	-	-	-	-	6.8	-	-	-	7.2
C	-	-	93.2	-	-	-	-	-	-	-	-	-	-	-	-
D	-	-	-	100	-	-	-	-	-	-	-	-	-	-	-
E	-	-	-	-	96.6	-	-	-	-	-	-	6.8	-	3.4	9.5
F	-	-	-	-	-	100	-	-	-	-	-	-	3.4	-	3.2
G	-	-	-	-	-	-	100	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	-	100	-	-	-	-	-	-	-
I	-	-	-	-	-	-	-	-	100	3.4	-	-	-	-	3.2
J	-	-	-	-	-	-	-	-	-	93.2	-	-	-	-	-
K	-	-	-	-	-	-	-	-	-	-	93.2	-	-	-	-
L	-	10	-	-	3.4	-	-	-	1.4	3.4	-	93.2	-	-	16.3
M	-	-	-	-	-	-	-	-	-	-	-	-	96.6	-	-
N	-	-	-	-	-	-	-	-	-	-	-	-	-	96.6	-
FN	-	10	6.7	-	3.4	-	-	-	4.2	6.8	-	6.8	3.4	3.4	96.6

Table 5 Comparison of different works

Paper	Feasible gesture classes	Average accuracy	Dataset size	Classification technique	Sensoring
[39]	1 Static 9 Dynamic	95%	20 000	Logistic Regression	Accelerometer Gyroscope Magnetometer
[41]	8 Static 2 Dynamic	99%	9869	Deep Neural Networks	RGB camera + OpenPose
[40]	7 Static 1 Dynamic	81.5%	3240	Convolutional Neural Networks	Mechanomyography Inertial Measurement Unit
[42]	13 Static 2 Dynamic	94.7%	13 884	Deep Neural Networks	RGB camera + OpenPose
Our proposal	14 Dynamic Gestures	96.6%	980	Neural Networks	RGB-D Camera

Table 6 Sequence of events of the experiment 1

Seq	Time (mm:ss)	Gesture	Action	Snapshot (mm:ss)
1	00:02	(L) and (M)	Takeoff the UAV and turn on the UGV	00:11
2	00:07	(D) and (G)	Both robots start following a circular path around the reference point located where the experiment started	00:23
3	00:26	(J)	Change the reference point of the UAV and sends it to a position above the UGV	00:48
4	00:38	(C) and (A)	Stops the UGV and lands the UAV	00:48
5	00:48	(M)	Turns UGV on and moves it to the starting point	01:14
6	00:56	(L)	Takeoff the UAV	01:21
7	01:01	(D)	UAV starts following a circular path around using the UGV as reference	01:31
8	01:11	(K)	UAV goes to its next waypoint, which is the starting point	01:42
9	01:16	(D) and (G)	Both robots start following a circular path around the reference point located where the experiment started	01:49
10	01:31	Emergency	Both robots shut down simultaneously	02:07

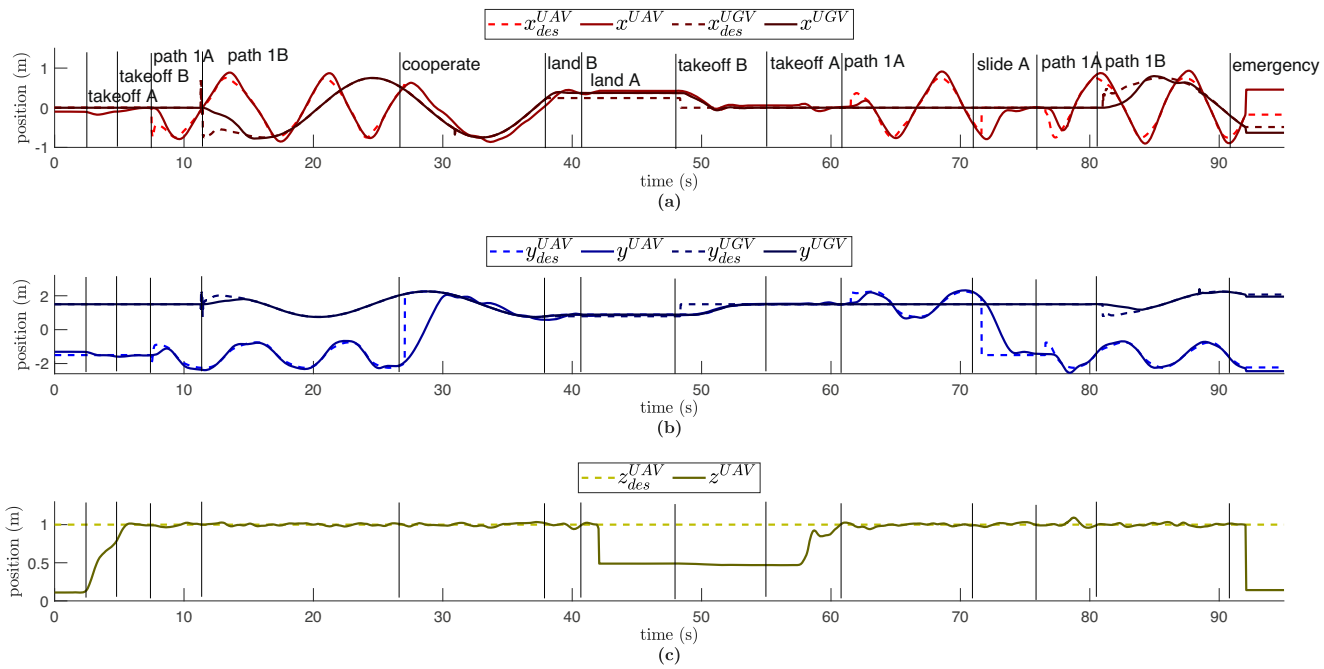


Fig. 6 Tracking performance of the UAV and the UGV in the experiment 1. The vertical dash lines represent the instant in which a new gesture is made. The gestures are represented as their class letters

Table 7 Sequence of events of the experiment 2

Seq	Time (mm:ss)	Gesture	Action	Snapshot (mm:ss)
1	00:01	(L) and (M)	Takeoff both UAVs	00:16
2	00:11	(D) and (I)	Robot A starts following a circular path and robot B a diamond path around the reference point located where the experiment started	00:29
3	00:32	(E) and (H)	Both robots start following a lemniscate path around the reference point located where the experiment started	00:53
4	00:46	(F) and (G)	Robot A starts following a diamond path and robot B a circular path around the reference point located where the experiment started	01:13
5	01:09	(A) and (C)	Lands both UAVs	01:43

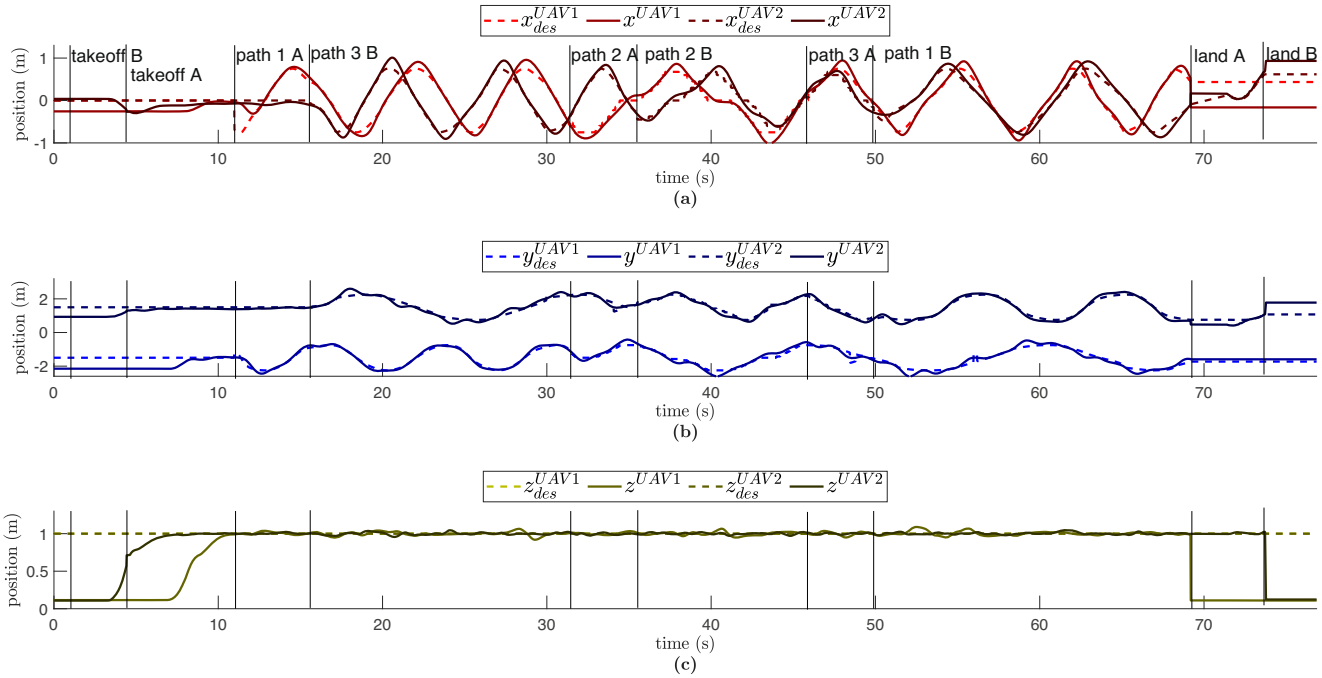


Fig. 7 Tracking performance of the UAV1 and UAV2 for the experiment 2. The vertical dash lines represent the timeframe in which a new gesture is made. The gestures are represented as their class letters

Table 8 Sequence of events of the experiment 3

Seq	Time (mm:ss)	Gesture	Action	Snapshot (mm:ss)
1	00:01	(L) and (M)	Takeoff the UAV and turn on the UGV	00:13
2	00:15	(N) and (K)	Both robots move to their next waypoint	00:27
3	00:24	(N) and (K)	Both robots move to their next waypoint	00:41
4	00:34	(N) and (K)	Both robots move to their next waypoint	00:52
5	00:43	(N) and (K)	Both robots move to their next waypoint	01:08
6	00:53	(A) and (C)	Land UAV and stop UGV	01:26

UAV land on the UGV. Also, the emergency gesture done at the end of the experiment had a quick response to shut down both agents, emulating an emergency situation. Furthermore, the tracking performance obtained by the UAV and UGV controllers for the given task can be seen in Fig. 6. It is possible to observe large errors occurring when a gesture commands a new reference for tracking. However, these errors quickly vanish due to the action of the feedback controllers.

5.3.2 Experiment 2

The second experiment was run using and homogeneous UAV-UAV squad to illustrate the switching between different surveillance paths. The UAV1 and UAV2 represent robot A and robot B, respectively. The video footage of the whole experiment can be found in https://youtu.be/k9K02qUpgds, where the sequence of actions can be checked. Table 7 indicates the user's commands, its aiming in the task and time stamp in which it starts in the experiment and the video separately.

This experiment illustrates the use of a UAV-UAV team to follow three different pre-determined paths that the agents can take to survey an area, running on a circle, a lemniscate or a diamond-shaped route. The tracking performance for this task can be seen in Fig. 7, showing the good behavior of the proposed dynamic UAV controllers.

5.4 Experiment 3

The final experiment means to showcase the application of the *slide* commands (gestures (K) and (N)), which sends either the robot A or robot B to its next waypoint.The video footage of the whole experiment can be found in https://youtu.be/i9kU2GrcD9M, where the sequence of actions can be checked. Table 8 indicates the user's commands, its aiming in the task and time stamp in which it starts in the experiment and the video separately.

The experiment begins with the user issuing the take off/start robot command (gestures (L) and (M)) to both agents, indicating them to go to the starting point, then he takes turns between sending each robot to its next waypoint (gestures (K) and (N)). The robots cover four different waypoints in the room, converging to the same (x, y) coordinates at the end. The user finishes the experiment sending the land/stop robot command (gestures (A) and (C)) to both agents, making the UAV land on the UGV. The tracking performance for this task can be seen in Fig. 8, where the controllers essentially deal with a positioning task.

6 Concluding remarks

In this paper a solution to teleoperate a team of mobile robots to accomplish high-level pre-generated tasks was

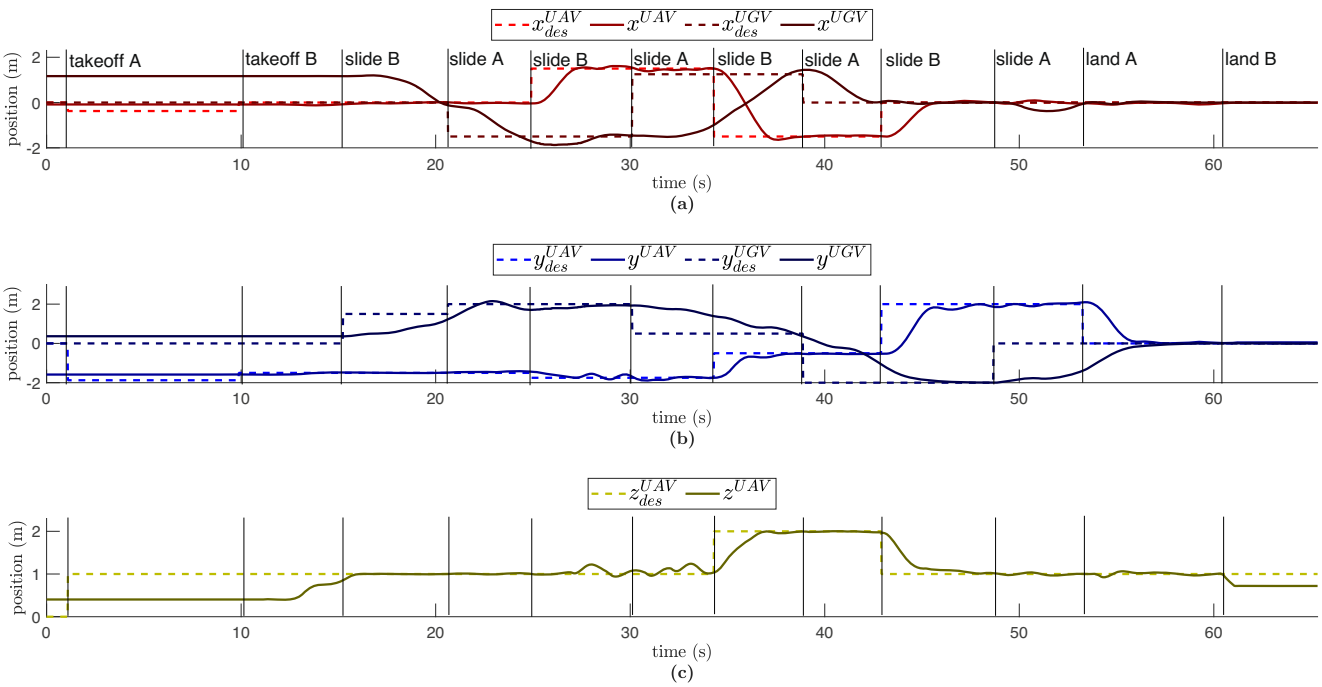


Fig. 8 Tracking performance of the UAV and UGV for the experiment 3.The vertical dash lines represent the instant in which a new gesture is made. The gestures are represented as their class letters

proposed. To specify the task, an ANN classifier was employed to recognize each of fourteen different gesture classes done by a user in a remote station. Skeleton information was used to represent the spatial-temporal data related to the movement of the user. The classifier showed a reliable performance with a 96.6% overall accuracy, for online tests. The task-intuitive gestures assigned high-level control actions for the robots, avoiding common teleoperation difficulties, such as time-delay. To validate the proposed solution, three experiments were run to showcase the possible applications in surveillance and inspection missions. In the first case, the robots traveled changing waypoints or following different path patterns. In the second case, the agents work cooperatively, according to the gesture executed and classified. Besides being effective as a system to allow commanding agents in surveillance/inspection tasks, the proposed framework allows high degree of customization, and it can be easily adapted to real-world applications, such as agricultural, industrial and education environments, in which a teleoperated control is more intuitive and efficient.

Acknowledgements This work was supported by FAPEMIG - Fundação de Amparo à Pesquisa de Minas Gerais, an agency of the State of Minas Gerais, Brazil, for scientific development, and FAPES - Fundação de Amparo à Pesquisa e Inovação do Espírito Santo, an agency of the State of Espírito Santo, Brazil, for scientific, technological and innovative development. The authors would also like to acknowledge Vitor Thinassi, for helping with the gesture images.

Funding This work was funded by CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, a Brazilian agency that supports scientific and technological development, CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, a Brazilian federal government agency under the Ministry of Education, responsible for quality assurance in undergraduate and postgraduate institutions in Brazil, FAPEMIG - Fundação de Amparo à Pesquisa de Minas Gerais, an agency of the State of Minas Gerais, Brazil, for scientific development, and FAPES - Fundação de Amparo à Pesquisa e Inovação do Espírito Santo, an agency of the State of Espírito Santo, Brazil, for scientific, technological and innovative development.

Availability of data and materials The authors have not provided the materials used in this paper aside from the experiment uncut video footage.

Declarations

Ethics approval The authors followed the COPE guidelines, which include but are not limited to: Not submitting the paper to multiple journals; the work being original and not split in multiple smaller papers for the sake of number of publication; not fabricating or manipulating data or results of this work by any means.

Consent to participate All the involved people in this paper, including who provided samples to the dataset, consented to participate in this work.

Consent for publication All the individuals that are having their information disclosured in this paper consent to have them published.

Conflict of interest The authors declare no competing interests.

References

1. Goodrich MA, Schultz AC (2008) Human-robot interaction: a survey Now Publishers Inc
2. Qian K, Niu J, Yang H (2013) Developing a gesture based remote human-robot interaction system using kinect. International Journal of Smart Home 7(4):203–208
3. Liu H, Wang L (2020) Remote human–robot collaboration: a cyber–physical system application for hazard manufacturing environment. Journal of manufacturing systems 54:24–34
4. Chivarov N, Chikurtev D, Chivarov S, Pleva M, Ondas S, Juhar J, Yovchev K (2019) Case study on human-robot interaction of the remote-controlled service robot for elderly and disabled care. Comput. Informatics 38(5):1210–1236
5. Jing X, Gong C, Wang Z, Li X, Ma Z, Gong L (2017) "Remote live-video security surveillance via mobile robot with raspberry pi ip camera," in International Conference on Intelligent Robotics and Applications, Springer, pp 776–788
6. Trevelyan J, Hamel WR, Kang S-C (2016) "Robotics in hazardous applications," in Springer handbook of robotics. Springer, pp 1521–1548
7. Legeza P, Britz GW, Loh T, Lumsden A (2020) Current utilization and future directions of robotic-assisted endovascular surgery. Expert Review of Medical Devices 17(9):919–927
8. Murphy RR, Steimle E, Hall M, Lindemuth M, Trejo D, Hurlebaus S, Medina-Cetina Z, Slocum D (2011) Robot-assisted bridge inspection. Journal of Intelligent & Robotic Systems 64(1):77–95
9. Washburn A, Matsumoto S, Riek LD (2021) "Trust-aware control in proximate human-robot teaming," in Trust in Human-Robot Interaction. Elsevier, pp 353–377
10. Washburn A, Adeleye A, An T, Riek LD (2020) Robot errors in proximate hri: how functionality framing affects perceived reliability and trust. ACM Transactions on Human-Robot Interaction (THRI) 9(3):1–21
11. Leichtmann B, Nitsch V (2020) "How much distance do humans keep toward robots? literature review, meta-analysis, and theoretical considerations on personal space in human-robot interaction," Journal of Environmental Psychology, vol. 68, p 101386
12. Pérez L, Rodríguez-jiménez S, Rodríguez N, Usamentiaga R, García DF, Wang L (2020) "Symbiotic human–robot collaborative approach for increased productivity and enhanced safety in the aerospace manufacturing industry,' The International Journal of Advanced Manufacturing Technology 106(3):851–863
13. Saenz J, Behrens R, Schulenburg E, Petersen H, Gibaru O, Neto P, Elkmann N (2020) Methods for considering safety in design of robotics applications featuring human-robot collaboration. The International Journal of Advanced Manufacturing Technology, pp 1–19
14. Bruno G, Antonelli D (2018) Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells. The International Journal of Advanced Manufacturing Technology 98(9):2415–2427
15. Liu R, Zhang X (2019) "A review of methodologies for natural-language-facilitated human–robot cooperation," International Journal of Advanced Robotic Systems, vol. 16, no. 3, p 1729881419851402
16. Hudson C, Bethel CL, Carruth DW, Pleva M, Juhar J, Ondas S (2017) "A training tool for speech driven human-robot interaction applications," in 2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE, pp 1–6

17. Podpora M, Gardecki A, Beniak R, Klin B, Vicario JL, Kawala-Sterniuk A (2020) Human interaction smart subsystem—extending speech-based human-robot interaction systems with an implementation of external smart sensors. Sensors 20(8):2376
18. Wang P, Zhang S, Bai X, Billinghurst M, He W, Sun M, Chen Y, Lv H, Ji H (2019) 2.5 Dhands: a gesture-based mr remote collaborative platform. The International Journal of Advanced Manufacturing Technology 102(5):1339–1353
19. Huttenrauch H, Eklundh KS (2002) Fetch-and-carry with cero: Observations from a long-term user study with a service robot. Inproceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication. IEEE, pp 158–163
20. Montemerlo M, Pineau J, Roy N, Thrun S, Verma V (2002) Experiences with a mobile robotic guide for the elderly. AAAI/IAAI 2002:587–592
21. Chen B, Hua C, Dai B, He Y, Han J (2019) Online control programming algorithm for human–robot interaction system with a novel real-time human gesture recognition method. Int J Adv Robot Syst 16(4):1–18
22. Neto P, Simão M., Mendes N, Safeea M (2019) Gesture-based human-robot interaction for human assistance in manufacturing. The International Journal of Advanced Manufacturing Technology 101(1):119–135
23. Chen F, Lv H, Pang Z, Zhang J, Hou Y, Gu Y, Yang H, Yang G (2018) Wristcam: a wearable sensor for hand trajectory gesture recognition and intelligent human–robot interaction. IEEE Sensors J 19(19):8441–8451
24. Oyedotun OK, Khashman A (2017) Deep learning in vision-based static hand gesture recognition. Neural Comput & Applic 28(12):3941–3951
25. Li G, Tang H, Sun Y, Kong J, Jiang G, Jiang D, Tao B, Xu S, Liu H (2019) Hand gesture recognition based on convolution neural network. Clust Comput 22(2):2719–2729
26. Cheng W, Sun Y, Li G, Jiang G, Liu H (2019) Jointly network: a network based on cnn and rbm for gesture recognition. Neural Comput & Applic 31(1):309–323
27. Ma X, Peng J (2018) Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information. Journal of Sensors, vol. 2018
28. Devineau G, Moutarde F, Xi W, Yang J (2018) "Deep learning for hand gesture recognition on skeletal data," in 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). IEEE, pp 106–113
29. De Smedt Q, Wannous H, Vandeborre J.-P. (2019) Heterogeneous hand gesture recognition using 3d dynamic skeletal data. Comput Vis Image Underst 181:60–72
30. Liu X, Zhao G (2020) "3D skeletal gesture recognition via discriminative coding on time-warping invariant riemannian trajectories," IEEE Transactions on Multimedia
31. Vezzani R, Baltieri D, Cucchiara R (2010) "Hmm based action recognition with projection histogram features," in International Conference on Pattern Recognition. Springer, pp 286–293
32. Song S, Lan C, Xing J, Zeng W, Liu J (2018) Spatio-temporal attention-based lstm networks for 3d action recognition and detection. IEEE Transactions on image processing 27(7):3459–3471
33. Hu Y, Wong Y, Wei W, Du Y, Kankanhalli M, Geng W (2018) "A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition," PloS one, vol. 13, no. 10, p e0206049
34. Liu Z, Zhang C, Tian Y (2016) 3D-based deep convolutional neural network for action recognition with depth sequences. Image Vis Comput 55:93–100
35. Capitan J, Spaan MT, Merino L, Ollero A (2013) Decentralized multi-robot cooperation with auctioned pomdps. The International Journal of Robotics Research 32(6):650–671
36. Darmanin RN, Bugeja MK (2017) "A review on multi-robot systems categorised by application domain," in 2017 25th mediterranean conference on control and automation (MED). IEEE, pp 701–706
37. Hashtrudi-Zaad K, Salcudean SE (2002) Transparency in time-delayed systems and the effect of local force feedback for transparent teleoperation. IEEE Trans Robot Autom 18(1):108–114
38. Lawrence DA (1993) Stability and transparency in bilateral teleoperation. IEEE transactions on robotics and automation 9(5):624–637
39. Akagi J, Morris TD, Moon B, Chen X, Peterson CK (2021) Gesture commands for controlling high-level uav behavior. SN Applied Sciences 3(6):1–23
40. Ma Y, Liu Y, Jin R, Yuan X, Sekha R, Wilson S, Vaidyanathan R (2017) Hand gesture recognition with convolutional neural networks for the multimodal UAV control
41. Liu C, Szirányi T (2021) "Gesture recognition for uav-based rescue operation based on deep learning." in Improve, pp 180–187
42. Liu C, Szirányi T (2021) Real-time human detection and gesture recognition for on-board uav rescue. Sensors 21(6):2180
43. Li X, Zhang Y, Liao D (2017) Mining key skeleton poses with latent svm for action recognition. Applied Computational Intelligence and Soft Computing, vol. 2017
44. "Aviation signs, howpublished = https://www.pilotopolicial.com.br/Documentos/Legislacao/Portaria/ICA10012.pdf, pages =247-254 note = Accessed: 2021-02-16."
45. Lee KU, Choi YH, Park JB (2017) Backstepping based formation control of quadrotors with the state transformation technique. Appl Sci 7(11):1170
46. Muñoz F, Espinoza ES, González-Hernández I, Salazar S, Lozano R (2019) Robust trajectory tracking for unmanned aircraft systems using a nonsingular terminal modified super-twisting sliding mode controller. Journal of Intelligent & Robotic Systems 93(1-2):55–72
47. Santana LV, Brandão AS, Sarcinelli-Filho M (2016) "Navigation and cooperative control using the ar.drone quadrotor,' Journal of Intelligent & Robotic Systems 84(1):327–350
48. Tang S, Kumar V (2018) Autonomous flight. Annual Review of Control, Robotics, and Autonomous Systems 1:29–52
49. Pinto AO, Marciano HN, Bacheti VP, Moreira MSM, Brandao AS, Sarcinelli-Filho M (2020) "High-level modeling and control of the Bebop 2 micro aerial vehicle," in The 2020 International Conference on Unmanned Aircraft Systems. Athens, Greece:, IEEE, oo 939–947
50. De La Cruz C, Carelli R (2008) Dynamic model based formation control and obstacle avoidance of multi-robot systems. Robotica 26(3):345–356
51. Martins FN, Sarcinelli-Filho M, Carelli R (2017) A velocity-based dynamic model and its properties for differential drive mobile robots. Journal of Intelligent & Robotic Systems 85(2):277–292
52. Santos MCP, Rosales CD, Sarapura JA, Sarcinelli-Filho M, Carelli R (2019) An adaptive dynamic controller for quadrotor to perform trajectory tracking tasks. Journal of Intelligent & Robotic Systems 93(1):5–16
53. López-Gutiérrez R, Rodriguez-Mata AE, Salazar S, González-Hernández I, Lozano R (2017) Robust quadrotor control: attitude and altitude real-time results. Journal of Intelligent & Robotic Systems 88(2):299–312
54. Martins FN, Celeste WC, Carelli R, Sarcinelli-Filho M, Bastos-Filho TF (2008) An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Control Eng Pract 16(11):1354–1363
55. Boubezoula M, Hassam A, Boutalbi O (2018) Robust-flatness controller design for a differentially driven wheeled mobile robot. Int J Control Autom Syst 16(4):1895–1904

56. Rabelo MFS, Brandao AS, Sarcinelli-Filho M (2020) "Landing a uav on static or moving platforms using a formation controller," IEEE Systems Journal
57. Santos MCP, Rosales CD, Sarcinelli-Filho M, Carelli R (2017) A novel null-space-based uav trajectory tracking controller with collision avoidance. IEEE/ASME Transactions on Mechatronics 22(6):2543–2553