




# Personalized teleoperation via intention recognition

Serge Mghabghab, Imad H. Elhadj & Daniel Asmar


To cite this article: Serge Mghabghab, Imad H. Elhadj & Daniel Asmar (2018) Personalized teleoperation via intention recognition, *Advanced Robotics*, 32:13, 697-716, DOI: [10.1080/01691864.2018.1460619](https://doi.org/10.1080/01691864.2018.1460619)


To link to this article: <https://doi.org/10.1080/01691864.2018.1460619>

 View supplementary material [↗](#)

 Published online: 23 Apr 2018.

 Submit your article to this journal [↗](#)

 Article views: 316

 View related articles [↗](#)

 View Crossmark data [↗](#)

 Citing articles: 1 View citing articles [↗](#)

## Personalized teleoperation via intention recognition

Serge Mghabghab<sup>a</sup>, Imad H. Elhajj<sup>a</sup> and Daniel Asmar<sup>b</sup>

<sup>a</sup>Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon; <sup>b</sup>Department of Mechanical Engineering, American University of Beirut, Beirut, Lebanon

### ABSTRACT

One of the challenges of teleoperation is the recognition of a user's intended commands, particularly in the manning of highly dynamic systems such as drones. In this paper, we present a solution to this problem by developing a generalized scheme relying on a Convolutional Neural Network (CNN) that is trained to recognize a user's intended commands, directed through a haptic device. Our proposed method allows the interface to be personalized for each user, by pre-training the CNN differently according to the input data that is specific to the intended end user. Experiments were conducted using two haptic devices and classification results demonstrate that the proposed system outperforms geometric-based approaches by nearly 12%. Furthermore, our system also lends itself to other human-machine interfaces where intention recognition is required.

### ARTICLE HISTORY

Received 15 May 2017  
Revised 23 August 2017,  
26 November 2017 and  
17 February 2018  
Accepted 14 March 2018

### KEYWORDS

Quadrotor; personalized teleoperation; CNN; intention recognition

## 1. Introduction

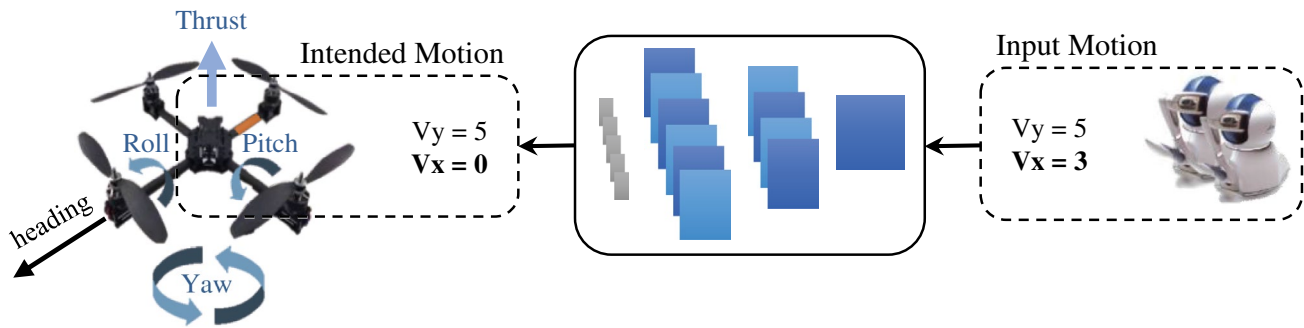
Teleoperation has numerous applications including tele-surgery [1], rescue [2] and the control of drones [3,4]. However, the teleoperation of drones is relatively difficult and challenging for novice users. One of the main reasons is the incompatibility between the degrees of freedom one can control and those of the controlled vehicle. With practice, a user learns to map these two different control spaces and the remote control becomes easier. The objective of the work in this paper is to use machine learning to create an 'interpreter' that can learn this mapping between spaces for any user. We propose to use a Convolutional Neural Network (CNN) for this learning task, and train the CNN using the 'user control' as input and 'true intention' as output. Figure 1 describes the problem with a picture.

Intention recognition systems have been explored before in the literature in human-machine interaction. In the work of Khokar et al. [5], motion intention is recognized for the sake of teleoperating a robotic hand; Hidden Markov Models (HMMs) were used to recognize objects of interest, select the desired configuration and aid the users in performing the required task. In an offline stage, an expert trained the HMMs to recognize user intention using as input the motion of a robotic hand end-effector. Afterwards, during testing, the learned system adjusts the human-machine input-output gains to mitigate for human intention. The result was an increase in efficiency

of the operators in performing a required task; however, the disadvantage of the proposed system is the requirement for a new learning stage for each new manipulated object of interest or new required motion.

Intention prediction with motion planning is combined in [6] to build an intuitive teleoperation system for a 6 DOF robotic arm controlled by a 2 DOF mouse. Their system performs two main tasks: first, the proposed teleoperation system should forecast the desired task by implementing a Gaussian mixture regression to learn and represent multi-modal transition models, afterwards, it should follow the forecasted path. The desired task is predicted using four main parameters:  $x$ , and  $y$  positions of the goal relative to the cursor, size of the goal, and velocity of the mouse cursor. Whereas the trajectory is followed by studying the position of the goal relative to the cursor, its velocity and its acceleration. This system showed promising results with the addition of the collision avoidance feature. Nevertheless, their work is only applicable to 2 DOF input devices, although their testing robot is capable of moving in the  $x$ ,  $y$  and  $z$  dimensions. Also, their work is based on predicting the desired trajectory using the combination of only three predefined shapes; which are a circle, a triangle and a figure-8. This approach prevents the system from performing freeform tasks, which is very common in the field of telerobotics.

In other work, intention recognition served as a means for selecting the proper control parameters to assist operators



**Figure 1.** The input from the operator is converted in real time by a CNN to a more accurate intended motion allowing for easier and more effective teleoperation.

in commanding a robotic suitcase [7]. During a learning phase, different motion patterns and corresponding user inputs are synthesized, and stored in a database; then during operation, user inputs are assessed and matched to the most similar motion pattern in the database. The major limitation of this method is that it is not possible to input new commands that were not considered in the training set.

In the work of Rabhi et al. [8], the performance of physically challenged users driving their wheelchairs was improved with the aid of machine learning. During an offline stage, an Artificial Neural Network (ANN) is trained with user commands as inputs, and intended controls as outputs. The ANN is then used during testing to aid in conveying the correct intention to the control of the wheelchair. However, in their work, the authors were only interested in modifying the rotational movements and their accelerations.

Gesture-based teleoperation of a mobile robot was investigated in Tzafestas et al. [9]; with the aid of a multi-layer perceptron neural network, it was possible to recognize shapes of hands. Multiple shape descriptors were used to represent the segmented hand shapes, and served as inputs to the neural network. With this architecture, the authors were able to control a mobile robot in real time, using multiple static hand postures. However, this system uses one-to-one mapping between each gesture and its corresponding movement, and the limited number of gestures constrains the robot's manoeuvrability.

Re-targeting function is used in [10] to map user inputs to the robot's activity. Here, the authors model the user-desired inputs as the actual given inputs, with the addition of an offset function. The challenge is in learning, in an online manner, the offset function that best fits every user's intention. Before making any manoeuvres, users have to teach the robot how to behave for every situation in order to achieve best performance. One of the challenges is that it is sometimes difficult or infeasible to track and capture the error between the desired motion and the performed one. Furthermore, it is not always practical to pause the robot and teach it how to behave for every new manoeuvre.

Semi-autonomous haptic teleoperation was investigated in [11], where multiple UAVs were controlled by one haptic joystick. Their threefold work is based on first teleoperating the UAVs Kinematic Cartesian virtual point (VP) using a haptic joystick with the assistance of an active collision avoidance system. Then, a controller is used to direct the UAVs towards their VP. Finally, the VP's velocity is controlled while sensing the UAV's surrounding area using haptics, making it easier to teleoperate in an unfriendly environment. The scope of this work is focused on teleoperating multiple UAVs safely through narrow and uneven places rather than teleoperating in a precise, and personalized manner. Due to the coordination between multiple UAVs, and the collision avoidance algorithm, the position of a single UAV is altered without input from the operator. In addition, the constants used for mapping the joystick commands to the VP's velocity are not customized per operator.

In our previous work [12], controller commands were also mapped to those of a drone's inputs; the control parameters were adaptive in a manner to enhance the performance of the pilots. The result was a more sensitive controller, with smoother transitions between different input commands. However, even with all these benefits, the system lacked flexibility, and the operator was restricted to operate using predefined gestures.

The modern CNN was originally introduced in 1998 in the paper of LeCun and Bottou [13]; nonetheless, its application was constrained by the available processing power at the time. However, with the increase in processing power in our day, and with the replacement of the traditional sigmoid activation function by the more successful rectified linear unit (ReLU), the applications of CNNs in the computer vision community has exploded.

In the paper of Simonyan and Zisserman work [14], a two-stream architecture was proposed to recognize the action of the targeted human, relying on both temporal and spatial components, each trained with its own CNN. Fusing the results of the CNN was attempted in two different manners: one involving a simple average of the results, and a second

relying on Support Vector Machine (SVM). Later, Park et al. [15] proposed a different approach for combining multiple CNN consensus. First, they rely on feature amplification optical flow, to perform spatially varying soft gating on intermediate CNN feature maps. Second, they use spatially varying multiplicative fusion for combining multiple CNNs trained on different sources. This fusion results in robust prediction, by amplifying or suppressing the feature activations based on the features agreement. The disadvantage of both of these systems is that they require complex computations and thus can only be processed at the end of a given input video.

The work presented in this paper is unique, in that the machine-learning algorithm learns, for different operators, their own weights, corresponding to their perceptual mapping of a desired motion to a given control. The training is done by prompting the user with known motions; then measuring the corresponding given commands by the user, and using them as inputs to a CNN. The contributions of this paper are as follows:

- A novel system for recognizing the intention of a human operator during the teleoperation of a remote-controlled device.
- A novel technique for mapping the motion trajectory of a joystick to an image, which is used to train the CNN.
- Our proposed system would allow inexperienced operators of UAVs to control their motion without any prior experience.

The remainder of this paper is structured as follows: Section 2 details the proposed Personalized Teleoperation System, hereafter referred to as PTS. Section 3 describes the experiments that were performed to validate PTS and compare it to prior art. Section 4 discusses the results and we conclude the paper in Section 5.

## 2. Personalized teleoperation system

Each time the PTS is trained for a different person, it uses input–output data that is specific to that particular user; the inputs to the PTS are the joystick motions and speeds, and the outputs are the corresponding ‘intended’ motions and speeds. In our work, we opted to use a CNN to learn this intended motion. The data-set is collected only one time per user in PTS, and once this is done, the input characteristics corresponding to each operator are analysed. As a result, each user will possess a single unique personalized network, which can serve him/her in teleoperating the UAV at any time. Two CNN architectures are proposed in this work, and both of them are evaluated before adopting the best of the two for our PTS. To compare the two architectures, the CNNs were used in the classification mode. Afterwards, the CNN architecture with the highest accuracy was adopted and used in the regression mode.

The proposed system results in each user having their own data-set and a personalized CNN that is used to translate personal commands to the desired UAV outputs. This personalized teleoperation results in improved performance as illustrated in the results section. The pseudocode below briefly presents the main steps of the PTS:

---

**Start**

```

Input: Operator.name, CNN.architecture (1 or 2)
Data collection: velocity + position vectors with time stamp from 2
joysticks
if personalized CNNs for Operator.name does not exist
  switch CNN.architecture
    case 1
      for all inputs in data-set
        inputs.preprocessed = matrices [5x6]: last 5 velocity vectors at
each instance
        outputs.preprocessed = vector [1x4]: values for the principle
motions that describe the final motion
      endfor
    case 2
      for all inputs in data-set
        inputs.preprocessed = matrices [(2n + 1) x 2*(2n + 1) x 3]: repre-
senting the 3D relative positions of the joysticks in an area of interest, the
3D relative positions are projected to the xy, yz, and zx planes (represented
in the 3 layers of the matrix) and the matrix is transformed to an RGB
image.
        outputs.preprocessed = vector [1x4]: values for the principle
motions that describe the final motion
      endswitch
    endif
    inputs/outputs.training = 80% of inputs/outputs.preprocessed (ran-
domly selected)
    inputs/outputs.testing = 20% of inputs/outputs.preprocessed (the
rest)
    train 4 CNNs: using each the same inputs.training but different portion
of output.training
    test 4 CNNs: accuracy using inputs/outputs.testing
    save the 4 personalized CNNs for Operator.name
  endif
  load the 4 personalized CNNs for Operator.name
  teleoperate the UAV using the 4 personalized CNNs
end

```

---

### 2.1. CNN introduction

A CNN is a deep learning technique currently used for visual and acoustic recognition. The main difference between CNNs and regular neural networks is the flow of signals between neurons. Typical, neural networks contain an input layer, output layer and multiple hidden layers. All the neurons between two consecutive layers are fully connected. Previously, these neural networks were successfully used to study images, texts and sounds. But the cost of this complex structure grew drastically with the use of both large data-sets and high-quality images. The cost of this architecture was reflected in the long training duration; thus, a new architecture was needed.

In 1998, the CNN was introduced in order to solve the training duration problem using a more efficient interconnection between its neurons [13]. CNN mainly relied on four layers; convolution layer, subsampling layer, activation layer and the fully connected layers. The use of these layers with the appropriate configuration and kernels allowed CNN to have the following characteristics:

- Ability to detect non-linear complex relationships between variables. The feature detection is done by convolution layers that study edges, curvatures, shapes and many other complex features.
- Ability to detect the feature in an image regardless of its location, due to the shared weights and shift-invariance architecture.
- Reduced number of parameters required in the training process due to the sharing of weights and biases in each feature map.

## 2.2. CNN architecture

CNNs are usually used to classify soundtracks, images and videos; however, in this application, we are interested in classifying joystick inputs. We architect and compare two different representations for the input feature vector: the first one is based on a temporal stratification of inputs, and the second is a mapping from joystick trajectory to a corresponding 2D image.

### 2.1.1. Temporal stratification representation

The first type of input to the CNN consists of layering, at each time stamp, five velocity vector inputs in chronological order – the current and past four. Table 1 shows an example of such an arrangement, where the first row displays the state of the current velocity, and each ensuing row represents the state immediately preceding the other in time. Each row represents the 3D velocities of the left and right haptic interfaces, for a total of six inputs per row.

The CNN architecture that best fits the stratified representation consists of an input layer, two convolutional layers, two fully connected layers, and a classification layer

**Table 1.** Example input for the stratified representation.

$VL_x(n)$	$VL_y(n)$	$VL_z(n)$	$VR_x(n)$	$VR_y(n)$	$VR_z(n)$
$VL_x(n-1)$	$VL_y(n-1)$	$VL_z(n-1)$	$VR_x(n-1)$	$VR_y(n-1)$	$VR_z(n-1)$
$VL_x(n-2)$	$VL_y(n-2)$	$VL_z(n-2)$	$VR_x(n-2)$	$VR_y(n-2)$	$VR_z(n-2)$
$VL_x(n-3)$	$VL_y(n-3)$	$VL_z(n-3)$	$VR_x(n-3)$	$VR_y(n-3)$	$VR_z(n-3)$
$VL_x(n-4)$	$VL_y(n-4)$	$VL_z(n-4)$	$VR_x(n-4)$	$VR_y(n-4)$	$VR_z(n-4)$

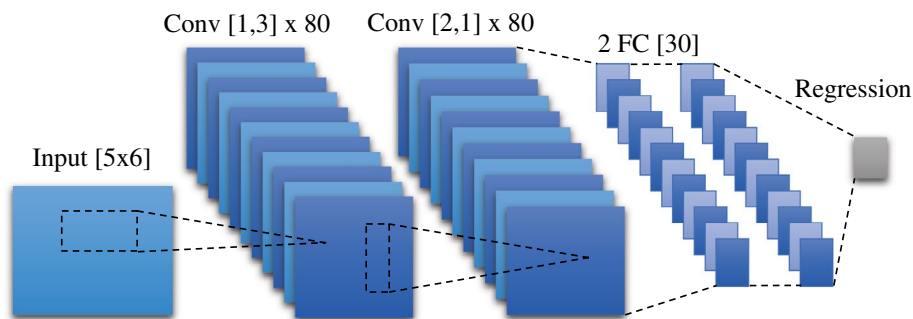
(see Figure 2). The first convolution layer uses 80 filters, each having a dimension  $1 \times 3$ . The purpose of these filters is to relate the velocity inputs of the two joysticks. The second convolutional layer, also uses 80 filters but with a dimension of  $2 \times 1$ . These filters relate the consecutive joystick inputs. The two fully connected layers consist of 30 neurons each. The final layer is composed of a regression layer mapping the input matrices to an output ranging from  $-1$  to  $+1$ , representing the desired direction and magnitude.

### 2.1.2. Representation by a three-channel input

Mapping the user inputs may be achieved by transforming the inputs of each joystick into a three-dimensional matrix, analogous to pixels in an RGB image. Each of the three channels represents the projection of the 3D trajectory of the haptic device on one of the three 2D spaces of an orthonormal coordinate frame, including the  $x$ - $y$ ,  $x$ - $z$ , and  $y$ - $z$  spaces. To ensure a compact representation of the data, the right and left inputs are concatenated side by side in each space, as shown in Figure 3. The displacements in the  $x$ ,  $y$  and  $z$  directions for the right and left joysticks are expressed as  $P_r x, P_r y, P_r z$  and  $P_l x, P_l y, P_l z$ , respectively.

To reduce the computational requirements of the CNN we focus, for each motion, on a local neighbourhood rather than the entire space. The size of the neighbourhood is dependent on the sought-after resolution. The proposed input matrix is composed of  $2n + 1$  rows, and  $2 * (2n + 1)$  columns, with a resolution of  $r$  mm. The input trajectory includes the displacement of the haptic device in the current time step, along with the previous displacements in a range of  $r * n$  mm in the positive and negative directions. These values could be selected differently for different applications with different inputs. By increasing  $r$ , the resolution increases, but requires a larger matrix to describe the same neighbourhood. Also, by increasing  $n$ , the system would account for a larger neighbourhood, but it would be more difficult to track sudden changes in the input.

The matrices are initialized such that they map part of the joysticks' workspace: at first the matrices are empty; as the operator moves the joysticks, the corresponding cells in the matrices are filled with the time elapsed to form this



**Figure 2.** CNN architecture for the stratified representation.

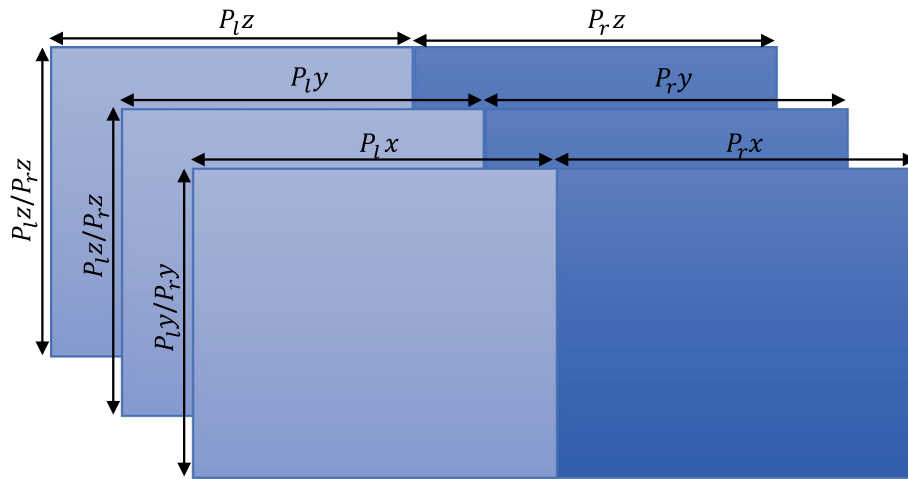


Figure 3. The three-dimensional input matrix with depth equal to 3.

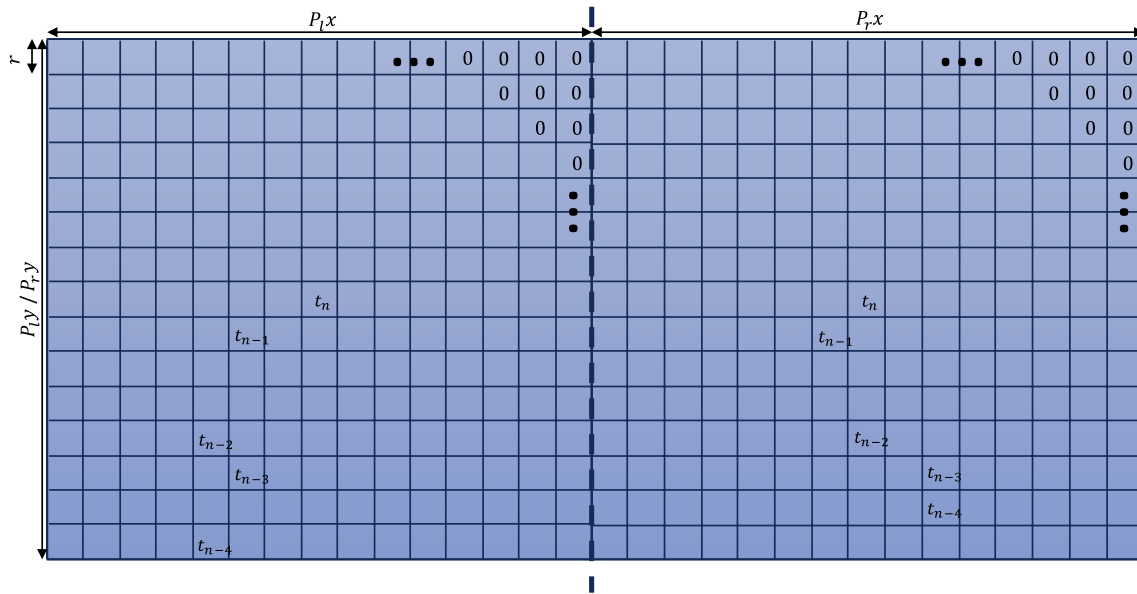


Figure 4. Representation of one depth of the input matrix.

manoeuvre. In other terms, each input matrix describes the most recent part of the input motions, the centre of the matrix is filled with the time stamp of the most recent input. Then, based on the position of the previous joysticks' inputs, the corresponding cells are filled with the timestamp. An example of one depth of the input matrix is shown in Figure 4.

Finally, each input is mapped to an 8-bit number, in a manner similar to the intensity value of a single image channel (1). This turns out to be helpful in using an off-the-shelf CNN that takes images as inputs.

$$V_{new_{x,y,z}} = \text{round} \left( \left( V_{old_{x,y,z}} - V_{minNZ} \right) * \left( 255 / (V_{max} - V_{minNZ}) \right) \right) \quad (1)$$

where  $V_{new_{x,y,z}}$  is the new value for a specific cell in the input matrix, and  $V_{old_{x,y,z}}$  is the previous value of that cell.  $V_{max}$  indicates the maximum value of the cells in the 3D input matrix and  $V_{minNZ}$  represents the non-zero minimum value of these cells. Figure 5 shows an example of such a mapping, where the most recent part of the input is analysed from the left and right joysticks. This figure shows how the velocity information is translated to RGB colour space.

The CNN that best fits this type of representation is constructed of two convolutional layers, each followed by a max-pooling layer, as shown in Figure 6. The convolutional layers consist of 40 and 80 filters, with 8-by-8 and 4-by-4 receptive fields, respectively. Each max-pooling

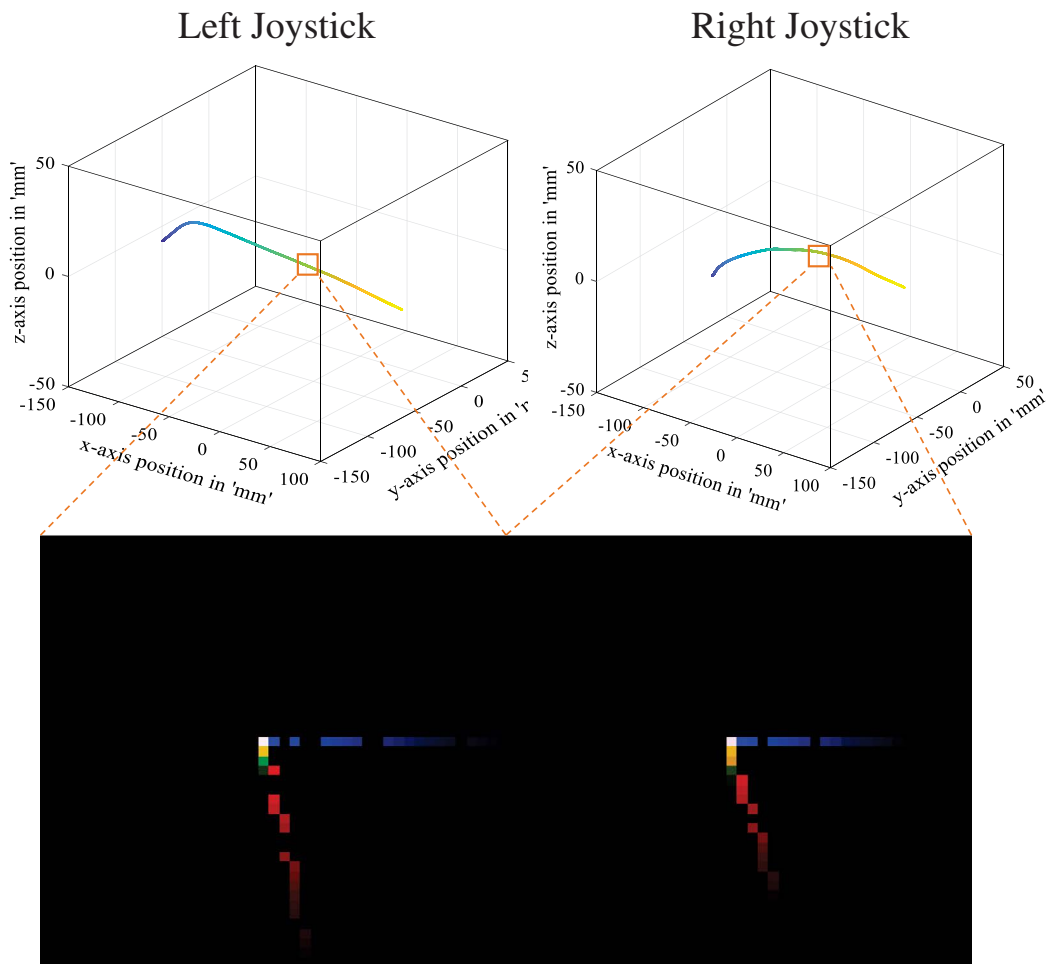


Figure 5. Sample of the RGB input matrix representation.

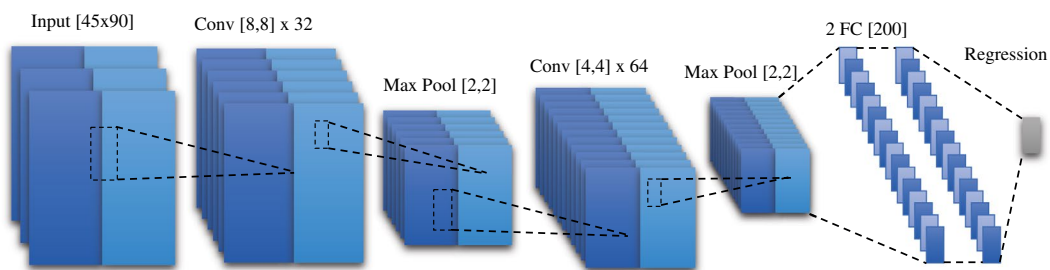


Figure 6. CNN architecture for three-channel input representation.

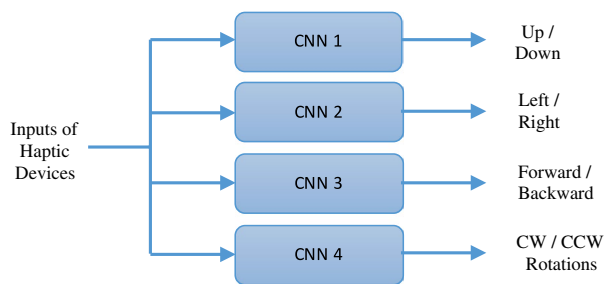
layer pools 2-by-2 regions at strides of two pixels. The convolutional layers are followed by two fully connected hidden layers, having two hundred units each. The final layer is composed of a regression layer mapping the input matrices to an output between  $-1$  and  $+1$ , representing the desired direction and magnitude. Both convolutional layers use the rectified linear activation function.

## 2.2. Prediction details

At the input, we consider 32 different motions, made up of combinations of principal movements in the up/down,

left/right, forward/backward directions, as well as rotations in the clockwise or counter clockwise directions. Each of these motions is performed at a slow and fast velocity, for a total of 64 different motions that each user is expected to input to train his CNN. The slow velocity is considered as 10% of the maximum possible velocity when training the CNN, while the fast velocity is considered as the maximum possible one.

In the prediction phase, the PTS relies on four CNNs, each corresponding to one of the four principal motions described above as shown in Figure 7; the output layer of each CNN selects a desirable output from the continuous



**Figure 7.** CNNs used to map user inputs to the UAV.

space that describes the five discrete classes, reflecting the direction and degree of motion in any direction. Using four versus one CNNs to predict the four sub-motions is more forgiving to errors, and offers the possibility of predicting part of the motion correctly. In other words, while using only one CNN the desired output is more prone to error since it is selected as one among 64 different classes (each class represents a compounded motion). On the other hand, using four different CNNs for classifying a move, each network classifies only a principal direction (five different options) and the resulting integration of the four network outputs determines the complete (in 6D) desired motion of the joystick.

Also, when an error occurs in the single network scenario, it is more likely that the error is affecting the four principal motions. While in the four CNN scenario, the networks are independent and when one of them is wrong the error is localized to that network, and the others could still predict their output correctly.

Sixty-four predefined motions are designed using the four principle motions, and are used to train each CNN independently from the remaining three. These 64 selected motions represent the most common motions used to teleoperate a UAV. While training the four CNNs, inputs are limited to the combination of principle directions, and users are asked to perform these inputs with the 10 and 100% of the maximum velocity. Using this dataset, the four CNNs learn the inputs characteristics of all the operators, where they will have their own perception of the main motions. Inputs of main motions could be curved and non-orthogonal due to the user input perception and other external factors related to the gravitational compensation problem and the joysticks' characteristics.

Once the training is complete, the operator is able to teleoperate the UAV using any desired motion, even if it was not part of the predefined motions that were used to train the CNNs. PTS assigns the desired outputs afterwards while allowing in-between motions with a velocity ranging from 0 to 100%. For example, for an output of  $[0.35 \ -0.1 \ 1 \ 0]$  from the CNNs, the UAV is directed to move upward, to the right and forward with 35, 10

and 100% of the maximum allowed velocities in these directions.

### 3. Evaluation of prediction accuracy

During the experiments, 12 test subjects used two haptic devices to perform their gestures. Different types of operators participated in the experiment, including nine right-handed and three left-handed male persons, with ages ranging from 20 to 27 years old. Four of the test subjects were video gamers and nine of them were familiar with the concept of thrust, roll, pitch and yaw motions; only three of them were amateur UAV operators. All the test subjects were selected randomly from the students at the American University of Beirut (AUB).

#### 3.1. Format of the input feature vector

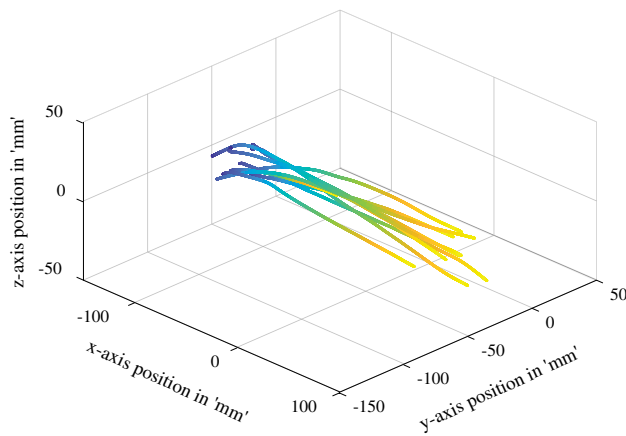
During the experiments, users had the choice to use either one or both haptic devices to perform any given input; the recorded motion is that of the tip of these devices, along with the time stamps of each collected point in 3D. The number of collected points is dependent on the sampling period of the computer, as well as that of the device itself. On average five hundred 3D points are captured from each haptic device for each performed gesture.

Figure 8 shows an example for a trajectory performed by a user on the right joystick for a 'right' motion command. Note that the inputs are neither identical nor linear, which emphasize the importance of a personalized input recognition system.

#### 3.2. Data collection design and set-up

Prior to testing, every participant was introduced to the experiment, and then the operators took 5 min to familiarize themselves with the experiment so they could be ready to perform the required task accurately. Participants were free to apply any gesture on the left haptic device, the right one, or both of them. Experiments were held in a closed environment, and all the subjects were free to leave and stop the experiment, whenever they wanted to. The two haptic devices used in this experiment are the Sensable™ PHANTOM Omni.

Every user was required to perform 20 trials per class, for a total of 1280 gestures. The 20 trials were applied in four rounds, each containing five trials. When collecting the data-set, the order of the applied gestures was selected in a random manner. On average, each session of data-set collection lasted between 120 and 150 min. During each session, the participants were asked to take five-minute breaks between each 30-min session of data collection.



**Figure 8.** Input sample from the right joystick used to teleoperate the UAV to the right.

This data-set contains a large number of repeated motions and it requires substantial time for each user to acquire his data-set; accordingly, users are prone to both learning, and fatigue effects. While learning results in enhancing the quality of the collected data, fatigue deteriorates its quality. To reduce both of these effects on the acquired data, users were given regular breaks throughout the data-set collection session, and the displayed trials were randomized as mentioned above.

Once the data-set was acquired, it was processed as explained in Section 2.2 and then fed to the CNN. Eighty per cent of the data were used for training and validation, and 20 were used for testing the CNN. In other words, for each specific motion and velocity, 16 gestures were used for training and validation, and four gestures were used for testing.

While processing the data,  $n$  and  $r$  were selected equal to 20 and 0.3, respectively; accordingly, the dimension of the input matrix is equal to  $45 \times 90 \times 3$ , and the resolution between one cell and the other is equal to 0.35 mm. With these values, the area of interest in one matrix is equal to 7 mm. Input matrix parameters are summarized in Table 2.

### 3.3. Results

Once the data-set of a user was collected, 80% of it is used for training while 20% of the data is used for prediction. In Table 3, we show the prediction results for every test subject independently using the four trained CNNs. The table also includes the prediction results using the two different proposed architectures. Columns 3, 4, 5 and 6 display the accuracies per motion. The last column displays the result when all of the four CNNs yield the correct prediction. To better explain, if for example, if the output was predicted correctly for the first three CNNs and falsely for the last

**Table 2.** Summary of the input matrix parameters.

Parameters	Values
$n$	20
$r$	0.3
Matrix dimensions	$45 \times 90 \times 3$
Cell resolution	0.35 mm
Area of interest	7 mm

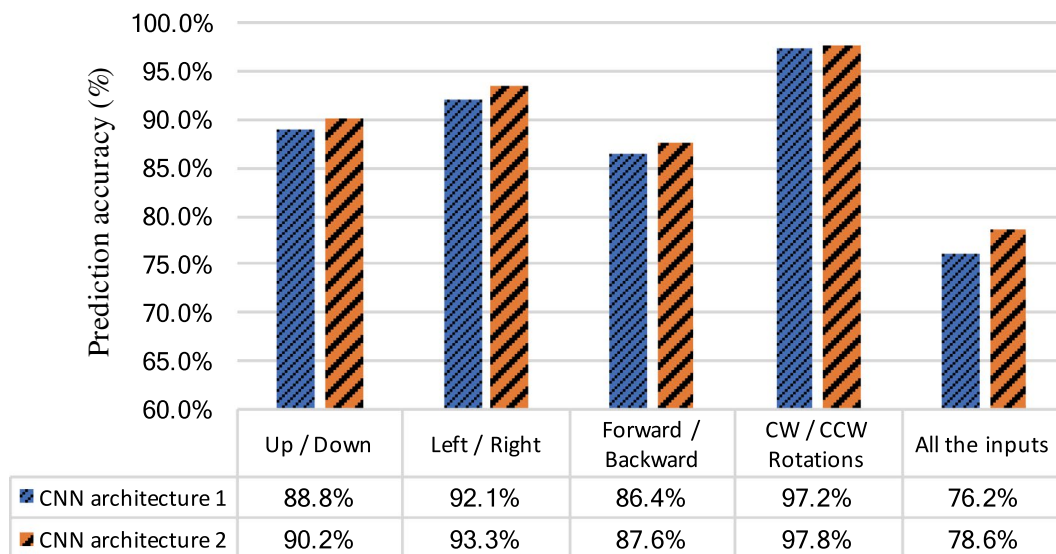
**Table 3.** Classification results for the participant subjects.

Subjects	CNN arch.	Up/Down (%)	Left/Right (%)	Forward/Backward (%)	CW/CCW rotations (%)	All the inputs (%)
1	1	94.3	97.3	95.5	94.0	86.9
	2	96.5	97.9	96.8	97.3	90.7
2	1	85.7	90.4	82.3	98.5	74.8
	2	86.4	91.7	86.1	98.9	79.3
3	1	80.4	87.6	84.3	97.1	68.7
	2	84.5	90.0	86.2	97.0	72.0
4	1	88.6	93.5	91.9	97.4	82.7
	2	91.8	96.8	93.5	97.7	86.1
5	1	94.2	96.7	94.6	95.4	85.0
	2	92.9	96.0	93.3	95.0	82.7
6	1	88.1	87.6	78.7	98.6	69.3
	2	88.2	90.6	79.2	98.3	70.8
7	1	84.5	92.0	81.5	98.4	69.2
	2	84.7	92.8	81.6	98.9	69.5
8	1	86.2	90.2	77.3	97.0	65.4
	2	85.4	87.7	75.4	97.8	65.1
9	1	88.2	88.1	85.1	97.9	73.7
	2	91.8	91.8	89.1	97.7	78.5
10	1	94.0	93.2	89.6	98.8	81.5
	2	95.0	94.1	91.0	99.1	84.1
11	1	99.3	98.7	99.0	99.9	97.8
	2	99.5	98.9	98.7	99.9	97.9
12	1	93.0	96.8	90.0	96.3	81.4
	2	95.0	97.3	91.1	97.8	85.4

one, the prediction is considered unjust in column 7. In the case where the classification accuracy is higher for the second CNN architecture, the results are highlighted in green, otherwise they are highlighted in orange. The average prediction accuracies of the two implemented CNN architectures are displayed in Figure 9.

A CNN lends itself to image-like inputs and is based on extracting the hidden features from an image, using its kernels. When the input is given as a table of previous velocity inputs, the features in the input matrix are reduced and altered, whereas in the 3D input matrix, a CNN is capable of detecting extra hidden features. This description is reflected in Figure 9, where the average classification accuracy is higher for the second CNN architecture, with 10 out of the 12 subjects having a higher classification accuracy for the image-like input architecture. On average, the accuracy in classification results increased from 76.24 to 78.56%.

The one-way Analysis of Variance (ANOVA) statistical test is used to analyse the relevance between the obtained means of the three modes. In order to accept or reject the null hypothesis, the resultant  $p$ -value is compared to the reference value  $\alpha = 0.05$ .



**Figure 9.** Average prediction accuracy for CNN architectures 1 and 2

The p-value relating these two CNN architectures is equal to 0.42, which implies that it is not definite that the second proposed CNN architecture will always outperform the first architecture by a percentage close to 2.32%. This is reflected in Table 3. But as the second CNN architecture gave overall better performance, it was the one adopted for the teleoperation experiments.

As can be seen in Table 3, User 8 has a 65.1% accuracy for all the inputs in the second CNN architecture, while User 11 has a 97.9% accuracy for all the inputs. It was observed that low accuracies were associated with users who performed similar motions, while intending to perform different outputs. It was also observed that better results were obtained when the input motions were close to straight lines, reducing ambiguity in the classification of an intention. As we are taking a small portion of the input at each instance to classify it, if the input is not a straight line, each portion of the input may be pointing to a different direction.

The first trial of each motion from the data-set is displayed in Figures 10 and 11, representing respectively the inputs of users 8 and 11. The inputs of user 11 are spread all over the area as straight lines. The different input motions could be differentiated from each other visually in most of the cases even when a small portion of the input is observed. That's why it was easier for the CNNs to classify the inputs and consequently obtained relatively high accuracy of 97.9% resulted. The inputs of Subject 8 are close to each other (Figure 10 right) and appear to be similar, in spite of the fact that each one belongs to a different class. Also, most of the inputs are far from representing straight lines. The combination of the curved and grouped inputs lead to the poor accuracy of 65.1%.

The classification accuracy displayed in Table 3 is only used to compare CNN 1 to CNN 2. Afterwards, the CNN architecture with higher accuracy is implemented in the continuous space using regression. Table 3 shows that the accuracy of some commands drops to low percentages; these drops are only harmful for classification but not for regression. This was further explained in Figure 12, where it is shown that most of the misclassified commands lie in the neighbouring classes. For example, if the user wants to go upward with a velocity equal to 60% of the maximum allowed velocity, the CNN should classify it as Class 5 (upward with fast velocity). CNN could assign this input wrongfully to Class 4 given that the desired output lies on the borders of Class 4 (upward with slow velocity). In the case of classification, the output will belong to a completely different class, but in the case of regression, the error will be minimal and the output could be 0.45 instead of 0.6 for example, resulting in a command with an upward velocity of 45% of the maximum velocity instead of 60%.

The confusion matrices for the classification results of all the test subjects are displayed in Figure 12. The results displayed in this figure represent 20% of the data-set, equivalent to 1,438,598 input data points. The four confusion matrices assess the classification results of the four CNNs that predict the principal movements in the up/down, left/right, forward/backward directions, as well as rotations in the clockwise or counter clockwise directions. The columns represent the target classes (desired outputs), while the rows represent the output classes (actual outputs). In every matrix, the diagonal constitutes the correctly predicted inputs belonging to the five classes. Every cell displays the percentages and number of inputs

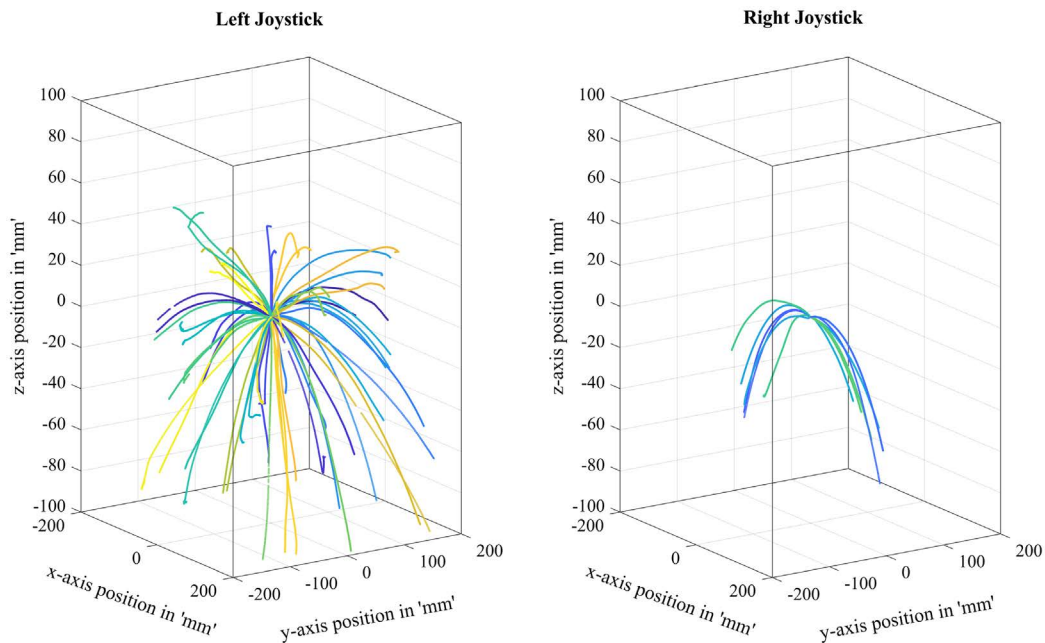


Figure 10. First trial of each motion for Subject 8.

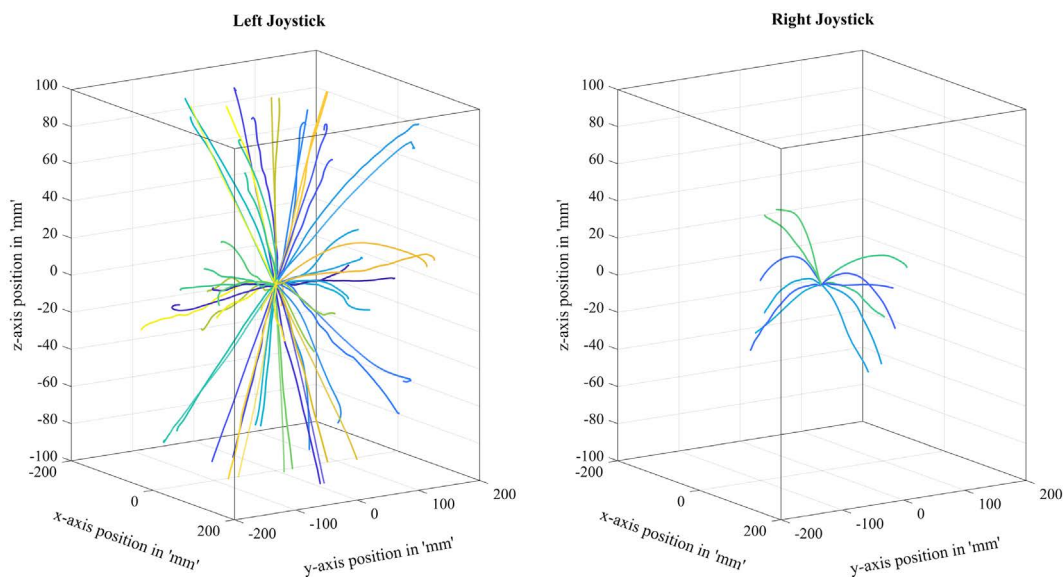


Figure 11. First trial of each motion for Subject 11.

predicted for the specified class. The five classes in this figure represent the following motions and velocities:

- (1) Negative motion with high velocity.
- (2) Negative motion with low velocity.
- (3) No motion.
- (4) Positive motion with low velocity.
- (5) Positive motion with high velocity.

As we can see in all cases, the majority of the inputs are predicted correctly, and most of the errors are only one class away in the speed, not in the direction. This

means that a misclassified input will not have a significant effect due to the small classification error. For example, in the confusion matrix representing the Up/Down motion, 98,583 inputs (constituting 79.8% of the total number of inputs that were intended for class 1) were predicted correctly for Target Class 1. Also, 17,620 inputs were misclassified as Class 2, 5725 inputs were misclassified as Class 3, 851 inputs were misclassified as Class 4, and 827 inputs were misclassified as Class 5. Class 1 represents an upward motion with a high velocity; Class 2 represents moving upward motion with low velocity, while Classes



**Figure 12.** Confusion matrices that describe the classification inputs of all the test subjects.

3, 4 and 5 represent staying still, and moving downwards with high and low velocities, respectively. In this example, most of the misclassified inputs lie in the second class, which means that 17,620 out of 123,606 predicted outputs were driving the UAV in the desired direction but with a slower velocity; such errors would have the least impact on the teleoperation performance. The significant prediction errors for this system are the ones two or more classes away from the desired class. In this example, this occurred for 7403 out of 123,606 inputs.

By further analysing the confusion matrix, we calculate the distance of the misclassified inputs and we compare it to the total number of classified inputs. If an input is classified correctly, the error distance is equal to 0, whereas an input classified with Class 1 instead of Class 3, has an error distance of 2. In our case, the average error distance of the four CNN outputs are equal to 0.1144, 0.0596, 0.1183 and 0.0249.

In order to assess the classification percentages of Table 3, it is compared to a geometry-based classifier, which is trained using the same input architecture as the one proposed in Section 2.2.1. The input is a vector containing 30 variables, representing 5 consecutive input velocities, each containing 6 variables. Every six variables consist of the linear velocities of the two haptic devices. Inputs are linked to their corresponding outputs and velocities. Also, the output of this classifier represents the states of the principal direction.

This classifier is trained by building a table containing rows of 30 variables. These 30 variables represent the last five velocity inputs in the  $x$ ,  $y$  and  $z$  directions for the left and right joysticks.

Therefore, there are in this Table 30 columns containing the variables that are going to be compared with each input, and there are 64 rows symbolizing the possible motions and velocities.

**Table 4.** Classification results of the geometrical-based classifier.

Subjects	Up/ Down (%)	Left/ Right (%)	Forward/ Backward (%)	CW/CCW rotations (%)	All the inputs (%)
1	89.4	92.45	85.2	97	79.1
2	79.2	85.2	78.3	95.7	66.7
3	78	84.2	80.9	94.3	62.9
4	91.2	92.8	93.1	97	83.6
5	85	97.7	87.6	88.8	65.4
6	77.8	83.5	77.1	96.6	59.5
7	80.2	88.6	77.3	97.7	62.2
8	80	87	70.8	95.3	58.7
9	82.3	86	77.8	91.1	61.5
10	75.1	84	73	97.1	55.9
11	94.7	91.7	94	98	83.7
12	85.89	89.4	79.1	88	64.8

In order to obtain the desired values for each row, all the training inputs are grouped by motions and velocities. Then, a vector containing the average values is calculated for each group and stored in the table and labelled with the desired classification classes. Afterwards, the classification of the new inputs is calculated by comparing the input vector to each row of this table. The row giving the lowest RMS error between all the table rows is chosen, and the label of this row is selected as the desired output. The classification results obtained using this method are displayed in Table 4. The average prediction accuracies of the geometrical classifier along with the prediction accuracies of the two implemented CNN architectures are displayed in Figure 13.

As shown in Figure 13, the average accuracy obtained using the geometry-based classifier is equal to 67%. The average accuracy of the CNN using both architectures is equal to 76.24% for the first architecture, and 78.56% for the second one. The ANOVA test shows that the p-values relating the CNN-based classifier to the geometrical based classifier are equal to 0.00067 in the case of the first CNN

architecture and 0.00045 in the case of the second CNN architecture. These p-values prove that the improved performance of CNN over geometrical based classifiers is statistically significant. This shows that it is more beneficial to assess the operator's inputs using the CNN rather than using geometrical based classifiers. Also, by implementing a CNN for output prediction, it is possible to determine the desired output in the continuous space without having reduced-DOF outputs. However, geometrical based classifiers do not possess this advantage.

## 4. Experimental set-up and results

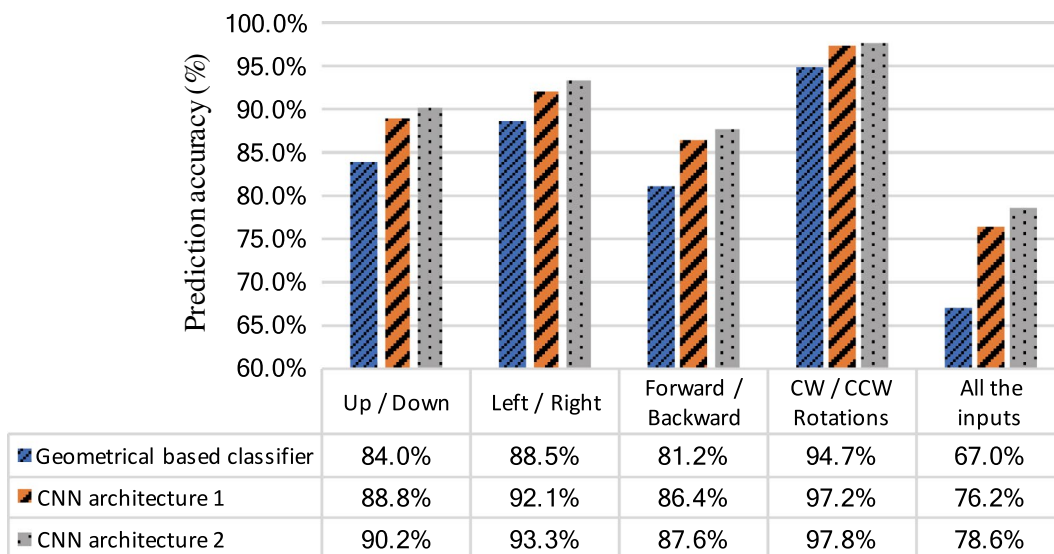
A statistical study of the participant's flight performance is presented in this section. Objective and subjective assessments are performed and the significance of the CNN-based teleoperation system is assessed.

### 4.1. Experimental design and set-up

To assess the proposed teleoperation method every participant had to perform 6 trials in 3 modes including the CNN-based teleoperation system. The test subject is required to teleoperate the UAV using these three modes in a predefined trajectory using a testing strategy that minimizes any bias effect. At the end of each trial, test subjects are required to fill a small questionnaire that describes their subjective opinion regarding their performance in each trial. Testing was done using the Parrot AR drone 2.0 quadrotor.

#### 4.1.1. Mode of teleoperation

The three modes of operation consisted of the CNN-based teleoperation system, the velocity-based teleoperation system [16], and the conventional teleoperation.

**Figure 13.** Average prediction accuracy for the geometrical-based classifier in comparison with the two proposed CNN architectures.

**4.1.1.1. Mode 1: CNN-based teleoperation system.** In this mode, the UAV is teleoperated using the trained CNNs. As the representation by a three-channel input gave the higher accuracy, it was used to map the subject inputs to the UAV control inputs. While teleoperating the UAV, the four networks are operating in the regression mode. The magnitude of the output level is personalized based on the training data.

The user commands are applied using two haptic joysticks and the motions are assigned based on each subject data-set. The commands are activated by pressing a button on the joystick end-effector, to allow the hand to move freely between consecutive inputs. Whenever test subjects want to perform a motion, they are required to press this button for the entire period of input. Once the subject releases the button, inputs are considered as free motions, and are consequently discarded. In the proposed teleoperation framework, the button is used to eliminate the neutral state problem and reduce the gravity compensation requirement. Whenever the user presses the button, the input is neutral till the user starts moving the joystick end-effector. The location where the button was initially pressed is considered as the position of the neutral input. Another benefit to this approach is that the user would not have to compensate for gravity when they are not issuing a command. Another button is used to command the UAV to take off and land.

**4.1.1.2. Mode 2: velocity-based teleoperation system.** In the literature, it was shown that the performance in teleoperating a UAV was highly improved using haptic joysticks [16,17]. Multiple modes were implemented, and the modes with the highest success rate were the ones with position to velocity [11], and velocity to velocity commands. Users were able to modify and assign the desired velocities to the UAV based on the position of the joystick or its velocity. In [16], a comparison between the position and velocity based commands showed better performance for the velocity to velocity commands where the joystick movements along the  $x$ ,  $y$  and  $z$ -axes directly control the UAV velocity in the  $x$ -axis,  $y$ -axis and  $z$ -axis directions, respectively, according to (2) (3) and (4). In our work, we added an extra equation (5) for the yaw movement:

$$V_x = k_x * J_r V_x \quad (2)$$

$$V_y = k_y * J_r V_y \quad (3)$$

$$T_z = T_{UAV} + k_z * J_r V_z \quad (4)$$

$$\psi = k_{yaw} * J_l V_y \quad (5)$$

where  $k_x$ ,  $k_y$ ,  $k_z$ , and  $k_{yaw}$  are constants used to adjust the sensitivity of the velocity commands.  $T_{UAV}$  is the current thrust value that is used to maintain the UAV at its current altitude.  $J_r V_x$ ,  $J_r V_y$ , and  $J_r V_z$  are the components right joystick's velocity vectors. While  $J_l V_y$  is the left joystick velocity in the  $y$  direction.  $V_x$  and  $V_y$  are the UAV velocities in the  $x$  and  $y$  directions while  $T$  is the UAV collective thrust and  $\psi$  is the yaw command.

The sensitivity constants  $k_x$ ,  $k_y$ ,  $k_z$ , and  $k_{yaw}$  were selected in the same manner as in [16] using sensitivity analysis test. In this experiment, these constants are selected as follows;  $k_x = k_y = 0.2$  and  $k_z = k_{yaw} = 0.15$ . In fact, one of the main drawbacks of most of the current teleoperation algorithms is the use of sensitivity constants. A common strategy to determine these constants is done by performing sensitivity analysis, where multiple test subjects perform different representative tasks with a range of different constants combinations. The constants that suit most of the test subjects are selected.

Additionally, to detect only the intended commands, a button on the end effector is pressed whenever the operator wants to send a command to the UAV. This allows the users to move their hand freely between consecutive commands. The take-off and landing were activated using another button on the haptic joystick.

**4.1.1.3. Mode 3: conventional teleoperation system.** In this mode, the UAV is commanded using the phone interface, which is the conventional interface for Parrot AR drone 2.0. The application shown in Figure 14 is available on android and IOS. This mode is used to assess the obtained results in the first two modes [18].

#### 4.1.2. Test set-up

Two testing areas were built to assess the performance of the operators in all the modes. The first experiment consists of a circular trajectory, where users were asked to teleoperate the UAV while following the desired path. Three sided wooden frames with sides equal to 1.56 m (equal to three times the width of the UAV) were placed every 60 degrees on the trajectory to help users keep track of the desired trajectory, as shown in Figure 15. In the second experiment, every user had to take-off, move the UAV in between two wooden frames, and then at the end land on a targeted area, as shown in Figures 16 and 17.

Figure 18 shows the connections made to test our system, the AR drone 2.0 quadrotor is the UAV we used, along with the Sensable™ PHANTOM Omni haptic joysticks. The VICON is used to track the quadrotor during the experiments, its data are analysed to obtain the distance separating the UAV from its desired trajectory, the time elapsed, the distance travelled and the accuracy in

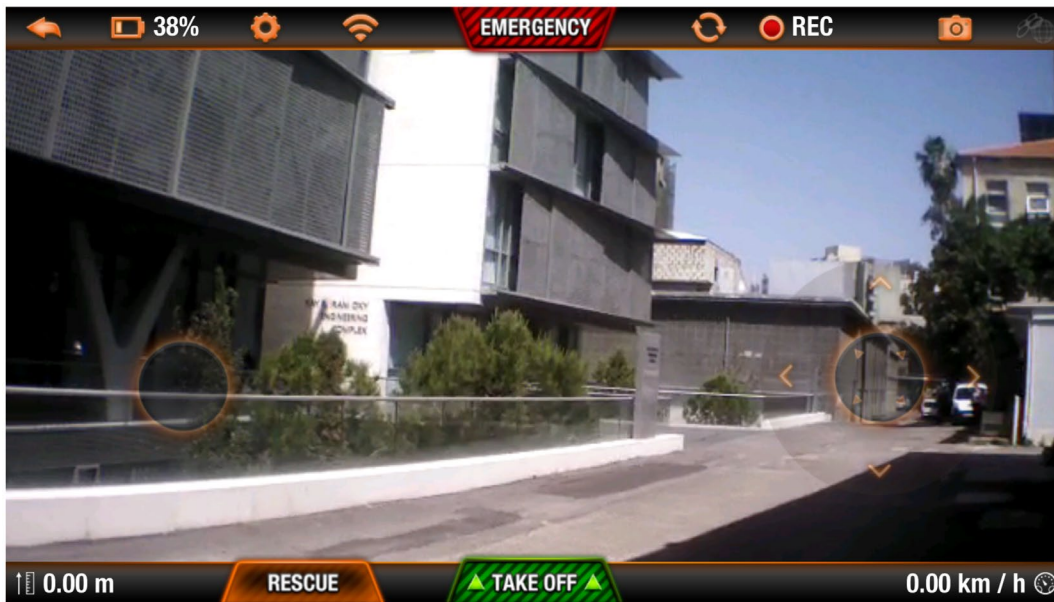


Figure 14. Parrot AR drone 2.0 remote control application.



Figure 15. Image showing the first test area.

landing. As for the connections, the two joysticks are connected to PC1 using a Firewire connection, the VICON system is connected to PC1 using an Ethernet cable. PC1 is connected to PC2 through an Ethernet cable and the connection is established using TCP/IP protocol.

As the prediction through the CNN requires high processing power, the tasks were divided between two computers in order to increase the frame rate of the system. While PC1 is responsible for collecting and logging the input data from the joysticks and the VICON system,

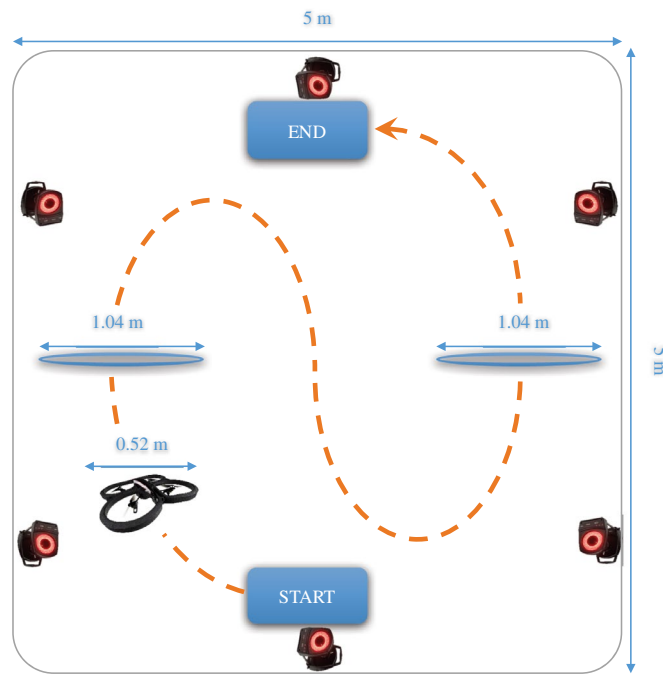


Figure 16. Testing trajectory of the second experiment.



Figure 17. Image showing the second test area.

PC2 is responsible for both mapping the inputs using the built CNN networks, and transferring the obtained command to the UAV using a Wi-Fi connection.

#### 4.1.3. Testing design and procedure

Prior to testing, every participant was introduced to the experiment, and then the operators took five minutes to

familiarize themselves with every mode of teleoperation. Experiments were held in a closed environment, and all the subjects were free to leave and stop the experiment whenever they wanted to.

**4.1.2.1. Experiment 1.** Every user is required to perform three trials per mode. The sequence of the first

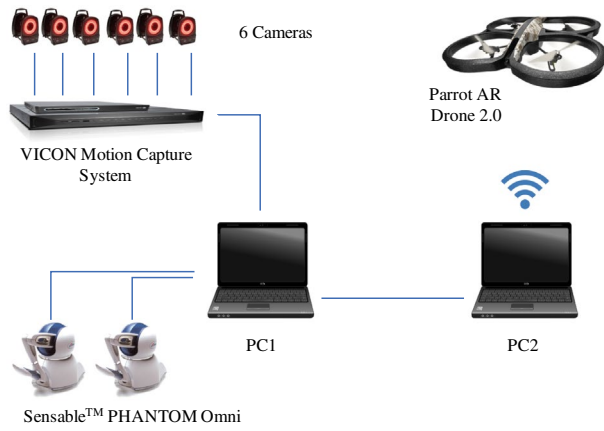


Figure 18. Image showing the testing set-up.

trial for each mode is selected randomly, and then the following trials were chosen with the same order. For every trial, we were interested in calculating the average distance separating the UAV path from the desired trajectory.

**4.1.2.2. Experiment 2.** Every user is required to perform six trials per mode. The sequence of the first trial for each mode is selected randomly, and then the following trials were chosen with the same order. For every trial, we were interested in objective and subjective metrics. The objective metrics include:

- Time elapsed in each trial
- Total distance travelled
- Accuracy in landing

Subjective metrics are obtained at the end of each trial and are taken using a survey based on the NASA Task Load Index (NASA-TLX). This self-assessment survey is widely used in human factors research [15], the workload perceived by the test subject is spread over six dimensions [19]:

- Mental demands
- Physical demands
- Temporal demands
- Performance
- Effort
- Frustration

## 4.2. Testing results

Three subjects participated in the first experiment, while 10 subjects participated in the second experiment. Subjective and objective results are recorded and studied.

The purpose of the first experiment is to assess the ability of users in teleoperating the UAV on a predefined trajectory requiring precision. The second experiment provided the requirements for a more general task.

### 4.2.1. First experiment objective results

For illustration purposes, the third trial of every teleoperation mode for the three operators is displayed in Figures 19–21. Given the fact that even for a simple trajectory it is hard for the user to teleoperate manually the UAV exactly over the desired path, the three test subject were able to perform better in path tracking while using the proposed teleoperation mode. It is visible from the plots that the shape of the path travelled using PTS is closer to the desired circular trajectory. The average error in distance between the desired trajectory and the path travelled by the UAV using the CNN-based teleoperation system is equal to 0.269 m for all the testing trials, while it is equal to 0.317 m in the case of velocity-based teleoperation system, and 0.31 m in the case of conventional teleoperation system. These results show more reliability in path tracking in

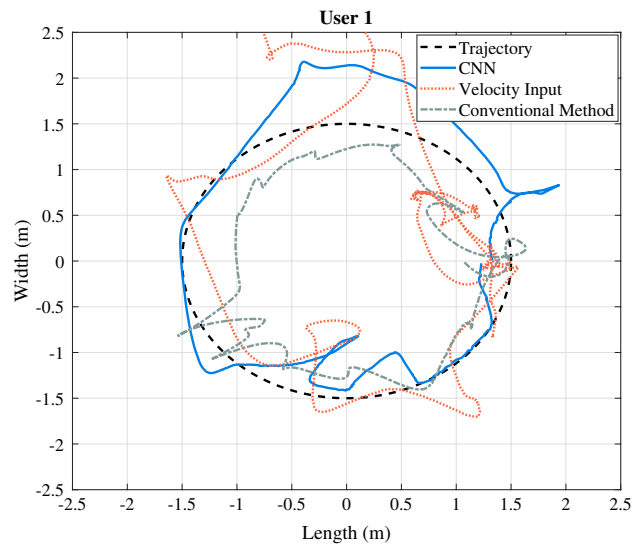


Figure 19. User 1 third trial using the three proposed modes.

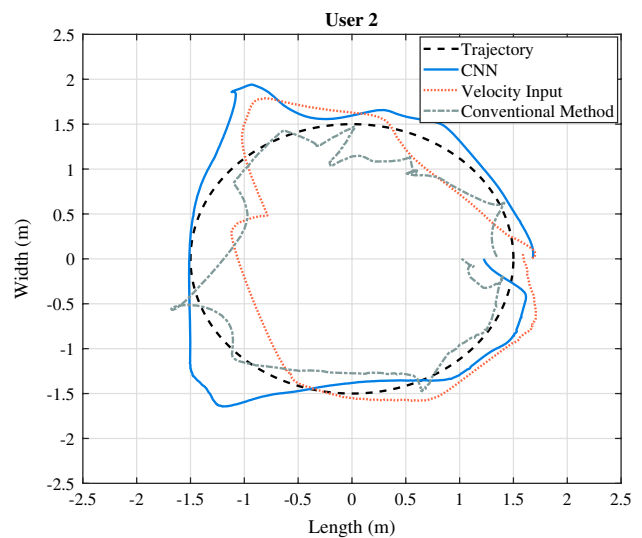
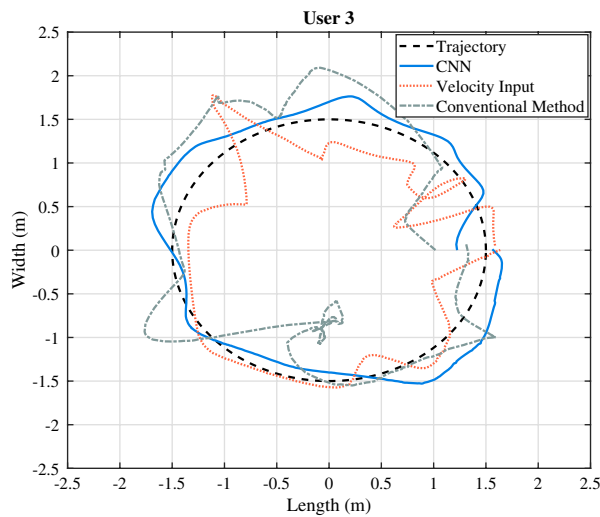


Figure 20. User 2 third trial using the three proposed modes.



**Figure 21.** User 3 third trial using the three proposed modes.

the case where the UAV is teleoperated using the proposed teleoperation system.

#### 4.2.2. Second experiment objective results

Figure 22 shows the mean and standard deviation for all the trials of the objective results (flight duration, accuracy in landing, distance travelled) for the three modes of operation. The bars represent the mean value for all the trials, while the standard deviations are displayed as vertical lines on top of the bars. The y-axis represents the flight duration in seconds, accuracy in landing in centimetres, and the distance travelled in metres.

In the first mode, when users were teleoperating the quadrotor using the CNN based teleoperation system, the mean of flight duration was equal to 35.4 s. In the second mode, using the velocity input-based teleoperation system, users finished the trials with a mean of 36.1 s. And in the third mode, using the conventional controllers, average trajectory time is equal to 47.5 s. Thus, first mode reduced the flight duration by 0.683 s or 1.91% compared to second mode and by 12.082 s or 28.94% compared to third mode (as shown in Table 5). As for the accuracy in landing, the mean landing distance from the target position was equal to 0.308 m in first mode compared to 0.356 m in second mode with a 14.2% improvement, as for the third mode the mean distance is equal to 0.376 m; thereby making Mode 1 better than Mode 3, with 18.36%. As well, the distance travelled was reduced from the second mode to the first mode, where it dropped from 16.3 to 15.7 m, with a 4.07% improvement. Also, the distance travelled was reduced from the third mode to the first mode, where it dropped from 18 to 15.7 m with a 13.83% improvement.

Moreover, the standard deviations are smaller in the first mode compared to the other two modes, which suggests that more consistent flights were obtained by

introducing the PTS. The standard deviation for the average time elapsed per trial was equal to 8.13 for the first mode, 8.24 for the second, and 14.21 for the third. We can infer from these results that subjects were more consistent time-wise for the first two modes compared to the third one, with a slightly better standard deviation for the first mode.

As for the accuracy in landing, the standard deviation for the three modes was equal to 9.06, 11.87 and 10.6, respectively. For the distance travelled, the obtained standard deviations for the three modes were 2.21, 2.29 and 2.9. Also, for the accuracy in landing and the distance travelled, better standard deviation results were obtained for the first mode, compared to the two others.

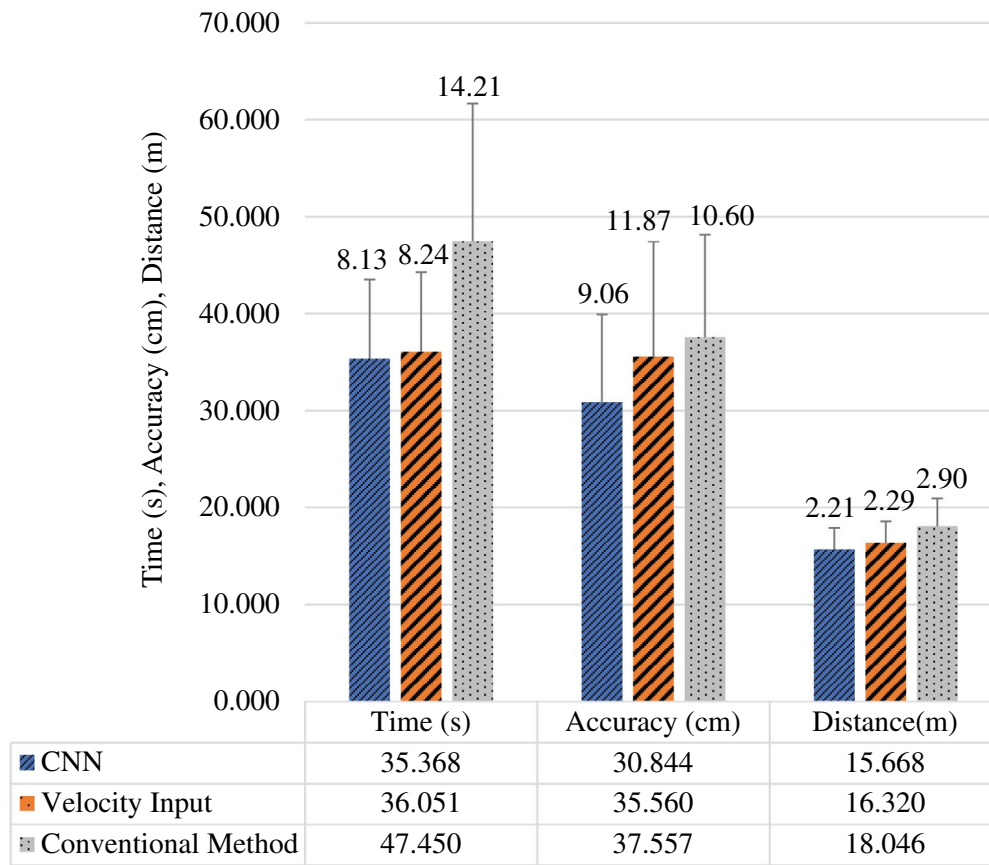
Table 6 displays the p-values related to each metric studied in the experiments. For all the comparisons, the p-values were less than 0.05 except for two cases; the flight duration for the CNN vs. Velocity-based inputs with a p-value of 0.6511 and the accuracy in landing for the Velocity-based inputs vs Conventional teleoperation method with a p-value of 0.3371. For the two cases where the p-value is high, we do not have sufficient statistical significance for the obtained results.

In our case, it is most likely that using the first mode of teleoperation would improve the distance travelled and the accuracy in landing compared to the second mode, but it is not proven for future trials that the flight duration is going to be reduced by 1.91%. The same case applies when comparing the second mode to the third one, there is no evidence against the null hypothesis when comparing the accuracy in landing for the two modes. But when comparing Mode 1 to Mode 3, it is clear that the teleoperation using the first mode is going to outperform the teleoperation using the third one in all the stated metrics with the displayed percentages of improvements.

#### 4.2.3. Second experiment subjective results

Figure 23 shows the individual averaging of each one of the NASA TLX subscales. The NASA Task Load Index uses 21 gradations on the scales, which we normalized to 100%. So, gradation 1 is 0% and gradation 21 is 100%. As shown, the load index for the first mode is always lower than the other two modes; also the load index is lower for the second mode compared to the third one. This proves that the use of velocity based inputs does not only enhance the performance of operators, it also reduces their average load index. However, this improvement is increased even more by using the CNN teleoperation system.

The total scores of the six subscales equally weighted gave a score of 25.7% for the CNN-based teleoperation system, compared to 34% for the velocity-based teleoperation system, and 42.2% for the conventional teleoperation system. This means that the load index was reduced by



**Figure 22.** Objective results of the three proposed modes.

**Table 5.** Objective comparison between the three proposed modes.

	Time (s) (%)	Accuracy (cm) (%)	Distance (m) (%)
CNN vs. Velocity input	-1.91	-14.20	-4.07
CNN vs. Conventional	-28.94	-18.36	-13.83
Velocity input vs. Conventional	-27.30	-5.46	-10.04

**Table 6.** *P*-values of the ANOVA test results describing the statistical significance of the objective results.

	Time (s)	Accuracy (cm)	Distance (m)
CNN vs. Velocity input	0.651	0.017	0.049
CNN vs. Conventional	3e-5	0.028	2e-5
Velocity input vs. Conventional	1e-4	0.337	6e-4

8.3% by switching from the third teleoperation mode to the second one. Furthermore, the load index was reduced by 16.5% when the user switched from the conventional teleoperation system to the proposed mode.

An ANOVA test was used to assess the obtained results; the *p*-value obtained for comparing the subjective results obtained from the first two teleoperation modes is equal to 0.015. By comparing the first and third modes together,

the *p*-value obtained was equal to  $2.8e-6$ . The *p*-value relating the second mode to the third one was equal to 0.032. The results of the ANOVA test show strong evidence against the null hypothesis, which proves that these results are statistically significant and they were not obtained by randomness (i.e. *p*-values  $< \alpha = 0.05$ ).

#### 4.2.4. Implications of the results

PTS was compared to the velocity-based teleoperation system, and the conventional teleoperation. Experimental results were assessed objectively and subjectively. Throughout the held experiments, PTS proved to be an enhanced version of two previously used teleoperation systems. The main advantage of PTS over the two other teleoperation techniques is the use of personalized mapping parameters that vary based on specific subjects and space. PTS eliminates the need for performing sensitivity analysis in order to select the mapping constants, and thus it accommodates for different teleoperation behaviours, and it works better for a wider set of users. Thus, this proposed CNN-based teleoperation system is capable of improving the quality of teleoperation. Hence, with PTS, more accidents are prevented, the teleoperation is safer, performance is improved, and beginners can now teleoperate more complex robots with limited or no training.

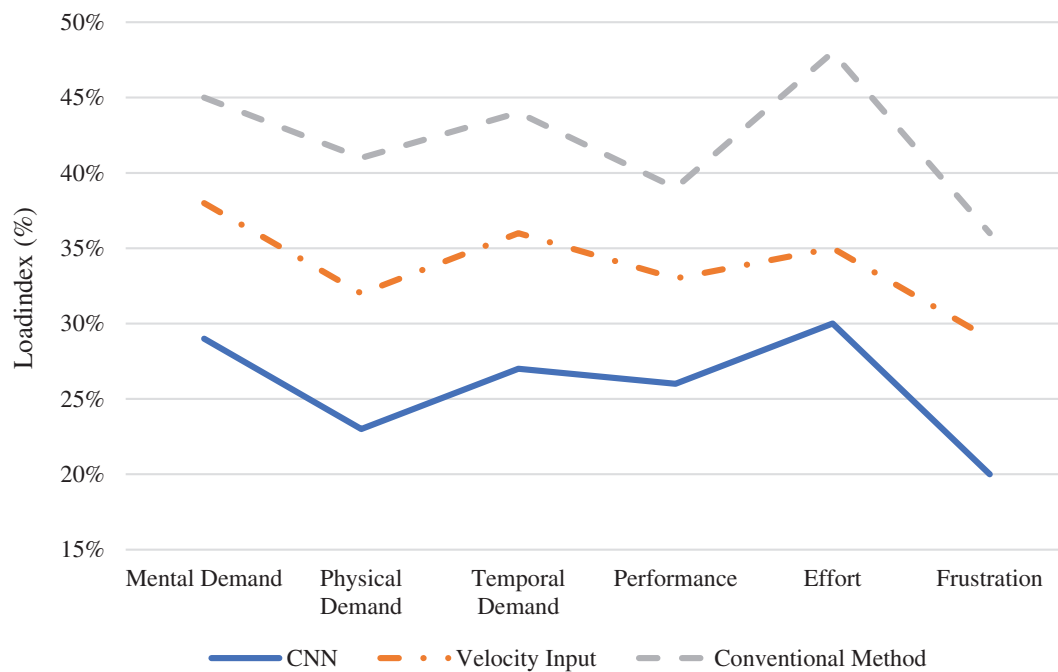


Figure 23. Subjective results of the three proposed modes.

## 5. Conclusion

This paper proposes a user-personalized teleoperation system based on the operator's desired motions. The user's intended motion is estimated using two CNN architectures, which were compared to a geometry-based classifier. The temporal stratification and three-channel input representations gave 9.24 and 11.56% higher accuracies than the geometry based classifier, respectively. The three-channel input representation gave higher prediction rate than the other proposed classifier architectures, and it was therefore chosen to classify the user inputs in the proposed teleoperation system. The performance of this method was compared to the velocity-based commands and to the conventional teleoperation device. Our method gave superior results both objectively and subjectively.

## Acknowledgement

This research was funded and supported by the AUB University Research Board at the American University of Beirut and the Lebanese National Council for Scientific Research.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Notes on contributors

*Serge Mghabghab* is currently a doctoral student at Michigan State University (MSU) in the ElectroMagnetics Research

Group (EMRG) under the supervision of Jeffery Nanzer. His current research is in radars, coherent distributed arrays and signal processing. Prior to joining the group, he earned his master's degree in Electrical and Computer Engineering from American University of Beirut (AUB) in 2017. He was a student in the area of control systems under the supervision of Imad Elhadj, where he worked on improving UAV teleoperation. He earned his BE in Electrical and Computer Engineering from Notre Dame University – Louaize (NDU).

*Imad H. Elhadj* is currently a professor with the Department of Electrical and Computer Engineering at the American University of Beirut. Elhadj is the past chair of IEEE Lebanon Section. He is a member of the World Economic Forum Global Agenda Council on Artificial Intelligence and Robotics. His research interests include robotics and instrumentation, cyber security, sensor and computer networks and multimedia networking. Imad received the Best Research Paper Award at the Third International Conference on Cognitive and Behavioral Psychology (CBP), the Best Paper award at the IEEE Electro Information Technology Conference in June 2003, and the Best Paper Award at the International Conference on Information Society in the twenty-first Century in November 2000. Elhadj is recipient of the Teaching Excellence Award at the American University of Beirut, June 2011 and the Kamal Salibi Academic Freedom Award 2014.

*Daniel Asmar* is an associate professor in Mechanical Engineering at the American University of Beirut. Daniel's research interests are in visual perception, autonomous robot navigation and mapping, environment representation and recognition, augmentation techniques in archaeology and segmentation methods in Computer Vision. Daniel is an ASME member, a senior member in IEEE, and the chair of the joint IEEE Lebanese chapter in Robotics and Automation, Instrumentations and Measurements and Control Systems.

## References

- [1] Maddahi Y, Zareinia K, Sepehri N, et al. Surgical tool motion during conventional freehand and robot-assisted microsurgery conducted using neuroArm. *Adv Robot.* **2016**;30(9):621–633.
- [2] Driewer F, Baier H, Schilling K. Robot–human rescue teams: a user requirements analysis. *Adv Robot.* **2005**;19(8):819–838.
- [3] Ruesch A, Mersha A, Stramigioli S, et al. Kinetic scrolling-based position mapping for haptic teleoperation of unmanned aerial vehicles. In: *IEEE International Conference on Robotics and Automation (ICRA)*; Saint Paul, MN; **2012**.
- [4] Nagi J, Giusti A, Gambardella L, et al. Human-swarm interaction using spatial gestures. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; Chicago, IL; **2014**.
- [5] Khokar K, Alqasemi R, Sarkar S, et al. A novel telerobotic method for human-in-the-loop assisted grasping based on intention recognition. In: *IEEE International Conference on Robotics and Automation (ICRA)*; **2014**.
- [6] Hauser K. Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Autonom Robots.* **2013**;35(4):241–254.
- [7] Nomura S, Inoue T, Takahashi Y, et al. Learning control based on intention recognition by inverted two-wheeled mobile robot through interactive operation. In: *Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on Soft Computing and Intelligent Systems (SCIS)*; **2014**.
- [8] Rabhi Y, Mrabet M, Fnaiech F, et al. A feedforward neural network wheelchair driving joystick. In: *IEEE International Electrical Engineering and Software Applications (ICEESA)*; Hammamet; **2013**.
- [9] Tzafestas C, Mitsou N, Georgakarakos N, et al. Gestural teleoperation of a mobile robot based on visual recognition of sign language static handshapes. In: *18th IEEE International Symposium on Robot and Human Interactive Communication; RO-MAN* **2009**.
- [10] Dragan AD, Srinivasa SS. Online customization of teleoperation interfaces. *IEEE RO-MAN*; **2012**.
- [11] Dongjun L, Franchi A, Il Son H, et al. Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *IEEE/ASME Trans Mechatron.* **2013**;18(4):1334–1345.
- [12] Mghabghab S, Elhadj I, Asmar D. I. Adaptive gain tuning for teleoperation of quadrotors. In: *IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*; **2016**.
- [13] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*; November. Vol. 86, No. 11; **1998**. p. 2278–2324.
- [14] Simonyan K, Zisserman A. Two-stream convolutional networks for action recognition in videos. *Adv Neural Inform Process Syst.* **2014**;27:568–576.
- [15] Park E, Han X, Berg TL, et al. Combining multiple sources of knowledge in deep CNNs for action recognition. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*; **2016**.
- [16] Kansa A, Elhadj I, Shamma E, et al. Enhanced teleoperation of UAVs with haptic feedback. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*; Busan; **2015**.
- [17] Hala R, Minh-Duc H, Hamel T, et al. Haptic-based bilateral teleoperation of underactuated Unmanned Aerial Vehicles. In: *18th IFAC World Congress*, Sophia Antipolis; France; **2011**.
- [18] Parrot AR. Drone 2.0 Elite Edition [Internet]. Parrot Store Official; [cited 2017 May 02]. Available from: <https://www.parrot.com/us/drones/parrot-ardrone-20-elite-edition>.
- [19] NASA Task Load Index (TLX) Instruction Manual, v. 1.0, Moffett Field. Moffett Field, CA: Human Performance Research Group, NASA Ames Research Center.