AMERICAN UNIVERSITY OF BEIRUT

DEEP LEARNING AND MIXED REALITY FOR AUTOCOMPLETE TELEOPERATION

by
MOHAMMAD HUSSEIN KASSEM ZEIN

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Mechanical Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
January 2021

AMERICAN UNIVERSITY OF BEIRUT

DEEP LEARNING AND MIXED REALITY FOR AUTOCOMPLETE TELEOPERATION

by
MOHAMMAD HUSSEIN KASSEM ZEIN

Approved by:

[illegible]

Dr. Daniel Asmar, Associate Professor	Advisor
Mechanical Engineering	

[illegible]

Dr. Imad Elhajj, Professor	Co-Advisor
Electrical and Computer Engineering	

[illegible]

Dr. Elie Shammas, Associate Professor	Member of Committee
Mechanical Engineering	

Date of thesis defense: January 20, 2021

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT
RELEASE FORM

Student Name:	KASSEM ZEIN	MOHAMMAD	HUSSEIN
	Last	First	Middle

✓ Master's Thesis ○ Master's Project ○ Doctoral Dissertation

☑ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

[illegible] February 4, 2021
Signature Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

I would like to express my special thanks of gratitude to my advisor Dr. Daniel Asmar as well as my co-advisor Dr. Imad Elhajj who gave me the golden opportunity to do this wonderful project, which also helped me in doing a lot of research and I came to know about so many new things I am really thankful to them.
Secondly, I would also like to thank my parents and friends who helped me a lot in difficult and critical times. Without their support, I would not have become the person I am today.

An Abstract of the Thesis of

Mohammad Hussein Kassem Zein for Master of Engineering
Major: Mechanical Engineering

Title: Deep Learning and Mixed Reality for Autocomplete Teleoperation

Teleoperation of robots can be challenging, especially for novice users with little to no experience at such tasks. The difficulty is largely due to the numerous degrees of freedom users must control and their limited perception bandwidth. Although humans can become skilled teleoperators, the amount of training time required to acquire such skills is typically very high. To help mitigate these challenges, this thesis proposes a solution (named Autocomplete) which relies on artificial intelligence to understand user intended motion and then on mixed reality to communicate the estimated trajectories to the users in an intuitive manner. User intended motion is estimated using a deep learning network trained on a dataset of motion primitives. During teleoperation, the estimated motions are augmented onto a first-person live video feed from the robot. Finally, if a suggested motion is accepted by the user, the robot is driven along that trajectory in an autonomous manner. We validate our proposed mixed reality teleoperation scheme with simulation experiments on a drone and demonstrate, through subjective and objective evaluation, its advantages over other teleoperation methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The wide spectrum of applications for Unmanned Aerial Vehicles (UAVs) has led to the fast proliferation of these robotic systems in the fields of agriculture, surveillance, damage assessment, and amusement/sports. Drone racing, for example, has become a very popular event, where competitors perform challenging maneuvers to navigate their copters through an obstacle course. Damage assessment of bridges and electric poles is another relevant application, where a drone performs a specified trajectory about the test object while scanning it for damage.

In the field of Human Robot Interaction (HRI), teleoperation remains one of the dominant modes of interaction between humans and machines. This fact is especially true in applications which require a relatively high level of perception [1][2]; or where the implications of making mistakes are too high, such as in nuclear sources detection [3], or surgical robotics [4] [5]. Furthermore, it is not expected that in the near future many teleoperation tasks will be completely replaced by fully autonomous solutions [6].

The high dimensional, nonlinear, and under-actuated properties of drones make teleoperation difficult, and accidental crashes more frequent, even when they are manned by expert operators. Experience has shown [7] that remote controlling UAVs without any automated assistance (e.g., hover) is difficult due to the low perceptual bandwidth available at the remote end, as well as the relatively low number of degrees of freedom that the operator can control.

Moreover, in spite of the advantages of teleoperation, it also faces many challenges: first, teleoperation is difficult to perform well because of the required mapping of a relatively high number of degrees of freedom of actuators on the machine side to a lower number of input controls on the human side. This mapping from human interface to machine control is generally not intuitive and requires significant training. Users acquire the needed skills after they internalize how the controlled robot reacts dynamically to control commands sent using the teleoperation device. Second, research has shown that the effectiveness of remote teleoperation is affected by the bandwidth of sensory feedback of the user [8].

More specifically, research showed that poor perception impairs the situational awareness of the operator and influences the overall efficiency of required task. Third, the ability of a human to plan a trajectory from a remote location is not easy due to the requirement to convert high level objectives (driving the robot to inspect a bridge) into a sequence of low-level control commands that the robot understands [9]. Usually, operators follow a trial and error strategy until the target mission is executed successfully. Unfortunately, such an approach can affect the quality of the overall task on different levels including completion time, and accuracy of the followed trajectory.

Completely automated trajectory planning has previously been proposed [10], such as in the agricultural sector [11]. However, the programmed trajectories are dependent on the shape and size of the intended site, and need to be reprogrammed each time a new site is visited.

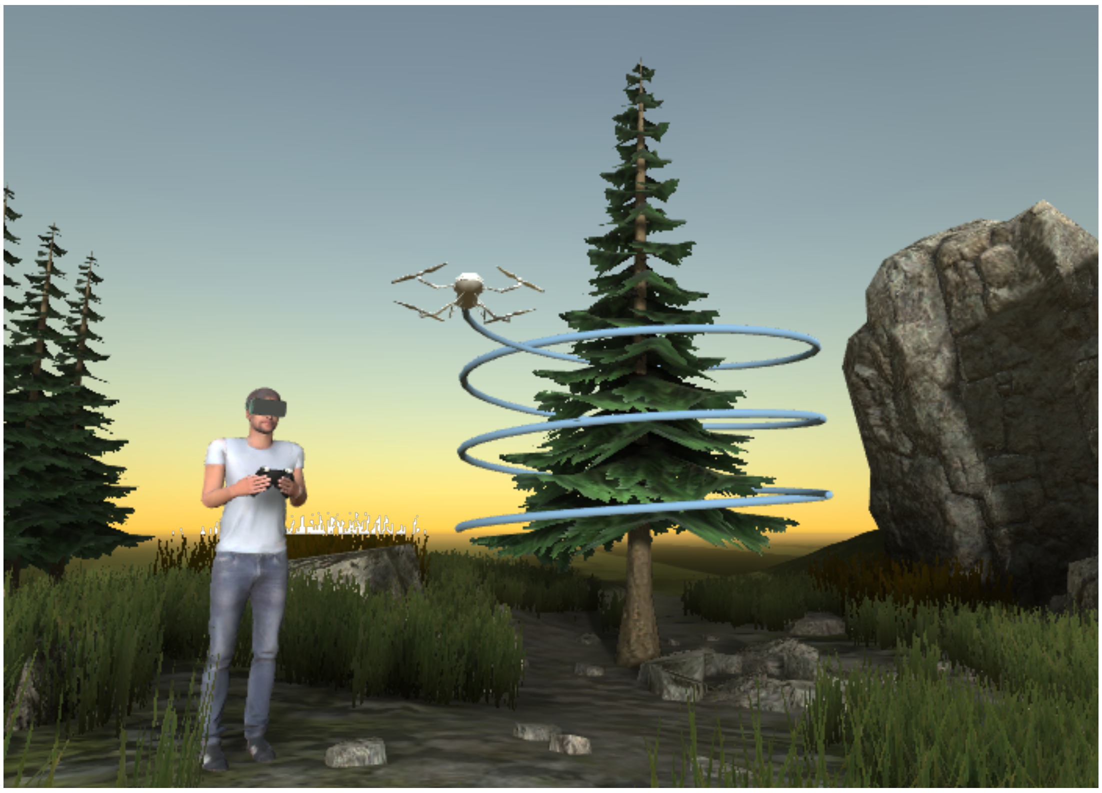


Figure 1.1: Autocomplete teleoperation supported by a Mixed Reality user interface.

We are proposing to mitigate the aforementioned challenges in teleoperation with the aid of Artificial Intelligence (AI) and Mixed Reality (MR) to address the needs for better perception and better control (Fig. 1.1). More specifically, we are proposing a system that (1) monitors user input, recognizes intended trajectories, and (2) suggests to the user through a Mixed Reality (MR) visual

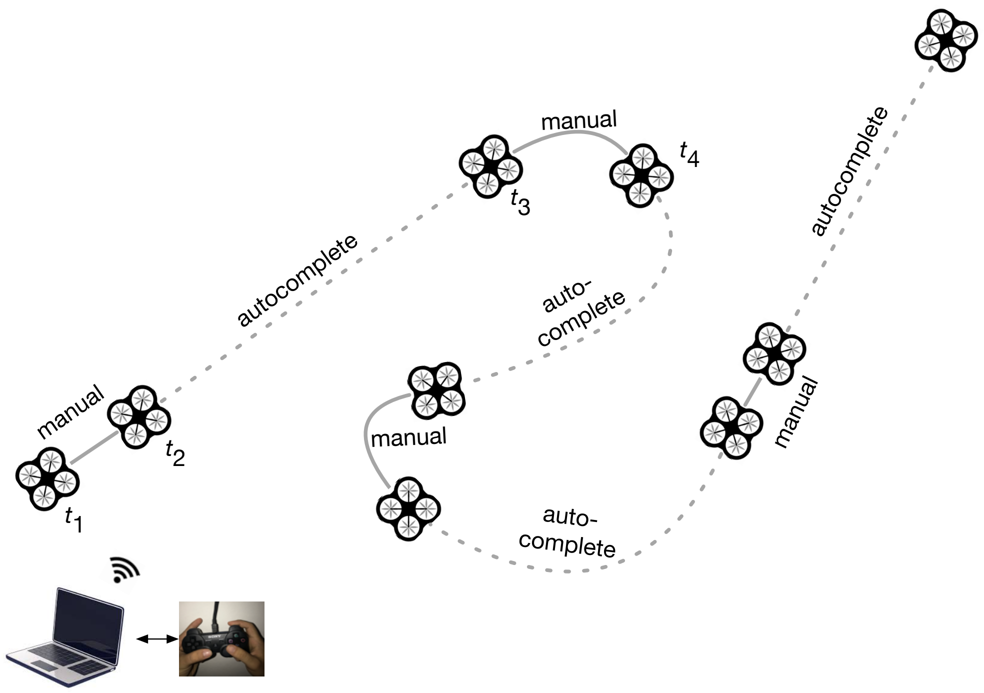


Figure 1.2: Autocomplete first classifies human intended motions, then synthesizes a proposed trajectory that the user can accept or not. If accepted, the automated system drives the vehicle along the synthesized path. For example, the system predicts a straight line between t_1 and t_2, prompts the user for a straight line, takes over and directs the UAV from t_2 to t_3 until the user changes the trajectory to a curve between t_3 and t_4.

interface a trajectory to follow, and (3) when instructed, autonomously executes (autocompletes) these trajectories for as long as the user desires.

The long-term objective of our work is to enhance the performance of human operators during machine teleoperation. While our current application is that of a UAV, our method is applicable to any type of land, aerial, or underwater robot. As a first step, this thesis presents a method for both classification and synthesis of motion primitives, thereafter leading in the future the application of the method to more complex compounded maneuvers. In this thesis, we present a solution (named Autocomplete) that exploits the advances of autonomous navigation while keeping the human as the **master** controller (Fig. 1.2). Autocomplete is adaptive in nature, capable of automatically detecting and completing intended user motions on robots. The system is collaborative in nature, taking inputs from humans, and then driving a robot the remainder of the intended motion in an automated manner. Autocomplete leverages the advantage of autonomous navigation while keeping the operator in-the-loop as the essential decision maker (Fig. 1.2). A Support Vector Machine (SVM) was trained using operator's input commands to classify intended trajectories in the form of a motion primitive (arc, 3D helix). However, the accuracy of the classical SVM model in terms of $F1 - score$ was **79%**. This relatively low accuracy affected the confidence of the operator in the predicted intended motion and thus influenced the overall teleportation performance. Hence, at a later stage (Fig. 3.1) the system is substantially improved. First, the SVM model is replaced with a Deep Learning Neural Network, which contributed to an increase in accuracy of approximately **10%**. Second, we develop a User Interface (UI) using Mixed Reality that aims at boosting the perceptive awareness of the operator while driving the robot. Finally, we validate the system through human-subject experiments in simulation on a UAV and demonstrate that the objective and subjective evaluations are improved over previous teleoperation methods.

The contributions of this thesis include:

- A system to detect intended human motions from control commands, based on a Deep Learning model.
- A mixed reality interface which allows the operator to visualize predicted motions augmented to the scene through an MR headset, rather than on a screen.
- Instead of always relying on the operator to drive the UAV, we propose a generalized task-independent system for automatically completing a desired motion.

Chapter 2

Literature Review

This section reviews the state-of-the-art in collaborative teleoperation frameworks, AI-based teleoperation, and in AR-supported human machine interfaces.

2.1 Collaborative Teleoperation

The idea of assisted teleoperation is not new [12], with prior work existing in assistive tele-manipulation [13] [14] and mobile robotics [15, 16, 17, 18, 19]. In these applications, the intent is to decrease the workload of the operators, while keeping them in the control loop to leverage their perceptual skills.

Gao et al. [16] propose a method for vehicle steer assistance, applied during the performance of specific tasks, such as object inspection, or door crossing. Their method is based on a Gaussian Mixture Regression model and a Recursive Bayesian Filter to first recognize the intended task of the user. Then, after inference is complete, an arbitration function blends the inputs from the user on one hand, and from the automated system on the other, to yield high quality steering control. The disadvantage of this approach is that it is task-centered, requiring unique data for each task, such as object shape and size in the case of object inspection. For each different task, different data is collected and the AI system needs to be re-trained. Our proposed approach is different in that it is user-centered rather task-centered; it recognizes intended user motion primitives regardless of the task, and accordingly guides the robot along the remaining synthesized path.

Battilani et al. [20] use bilateral shared control to estimate the 3D pose of a target goal during teleoperation. After the system recognizes the object of interest, and with the aid of haptic force feedback, the user is guided towards the optimal direction of actuation. The bilateral control architecture is also extended in Masone et al. [21], where the path is regulated to ensure robust convergence to the target goal. For tasks that do not involve targeting a goal, such as flights around a race track, these methods are ineffective due to the absence of the used features.

Yang et al. [22] propose task-independent adaptive teleoperation, where, instead of mapping velocity inputs to a desired motion using traditional methods, the action space is represented as a Motion Primitive Library (MPL), and a selector function is used to choose a specific motion primitive. User intent is modeled as a probabilistic distribution, and a set of available actions are adapted accordingly. Although their system produced promising results, it is completely dependent on the user. In contrast, our proposed system is collaborative and reduces workload on the operator when Autocomplete is activated.

Dynamic Motion Primitives (DMPs) [23] represent another approach for trajectory generation, where complex maneuvers are segmented into motion primitives that are learned using machine learning. Then, these learned primitives are presented to the dynamic system in the form of well-defined non-linear functions that guide the robot on a specific trajectory. Mueller et al. [24] represent the motion primitives of quadcopters by a combination of their initial state, the desired motion duration, and any combination of position, velocity, and acceleration at the motion's end. DMPs trajectory planning techniques are also employed in the tele-manipulation of robotic systems. Stulp et al. [25] apply model-free reinforcement learning to learn motion primitives for a task in manipulation. The system is able to learn shape and goal parameters of motion primitives that are robust to object pose uncertainty. The disadvantage of DMPs is the need to carefully choose the end point of the movement, which makes it difficult for the learned policies to generalize well to new situations.

2.2 Deep Learning in Teleoperation

A considerable amount of works in the literature have explored intention recognition systems in various domains, such as the teleoperation of robotic hands [26], driving wheelchairs [27], and in teleoperation of mobile robots [28]. However, only a few of these systems utilized Deep Learning in their approach. Laskey et al. [29] compared Human-Centric (HC) sampling and Robot-Centric (RC) sampling when dealing with deep learning in teleoperation. The comparison was done using a grid world environment and a physical robot object singulation task. Their simulation results showed that for linear SVMs, policies learned with RC outperformed those learned with HC but that when using highly expressive learning models (Deep Models) this advantage disappears. In another work, like TeachNet [30], an end-to-end teacher-student deep Convolutional Neural Network (CNN) was used for the vision-based teleoperation of dexterous robotic hands. Their network learns the kinematic mappings between the depth images of a human hand and the robot's joint angles. The results satisfied the high-precision condition, and imitation experiments teleoperated by novice users showed that TeachNet was faster and more reliable than the state of the art vision-based teleoperation methods. In a similar study, Zhang et al. [31] described how to make use of consumer-grade VR devices for the teleoperation of a PR2 robot. RGB and depth images are concatenated and fed into a single deep CNN architecture augmented with auxiliary prediction connections and then used to train deep visuomotor policies that directly map from pixels to actions using behavioral cloning, a method of imitation learning. Results showed that, for each task, less than 30 minutes of demonstration data was sufficient to learn a successful policy using the same hyper-parameters and model architecture across all tasks. Unlike the previously mentioned studies which captured the spatial features of their data through CNNs, we intend to capture both the spatial and temporal features of our data in order to produce a more robust model.

In this work, we aim to replace Autocomplete's intention recognition system [32] with a new Deep Learning model capable of achieving a more reliable performance. To make this attainable, a large dataset [33] was carefully collected and analyzed by following a well-documented procedure. This allowed us to create and deploy a Deep Learning model which learns to filter information by automatically extracting its own features from the input data rather than explicitly being told what features to use. These relevant features are then used to learn the long-term dependencies of the data thus allowing the model to capture the complete essence of our data.

2.3 Augmented Reality in Teleoperation

The features that Augmented Reality (AR) can add to a user interface (UI), such as the ability to overlay virtual objects to the real world, has shown potential improvements in different research areas [34, 35, 36]. Specifically, new AR interfaces are suggested to improve the teleoperation paradigm. Hedayati et al. [37] proposed an AR interface in the form of visual feedback to enhance quad-copter teleoperation. They created three AR designs that aim at supporting the user's perspective on how to convey information from the on-board camera. This support is in the form of augmentation to the robot, the environment, or to the user interface. They proved through subjective and objective evaluation that their system enhances the performance of teleoperation over traditional methods. In other works, such as in [38], they leverage the advancements of AR to create an effective teleoperation scheme that allows operators to better control aerial robots while reducing stress. They propose a system that prompts a virtual robot surrogate for the user to control instead of controlling the actual physical robot. Their ultimate goal is to provide the operator with a prediction of what might happen in the near future when a control action is taken. Lee et al. [39] proposed an intuitive method for telerobotic manipulation through virtual fixtures that are augmented in the scene. They hypothesize that this generation of virtual fixtures aims at improving operator's perception in complex environments and thus enables a precise teleoperation of dexterous manipulation in dynamic environments.

The problem with the previously mentioned AR interfaces is that they only focus on improving the perspective of the user while assuming that the robot is always in the field of view (FOV) of the operator. This assumption may render the proposed interfaces inapplicable in case of remote tasks where the robot is far from the control station. In our proposed visual interface, we leverage the advantages of MR to display for the user intended trajectories augmented inside the task scene. These candidate trajectories are generated using an intelligent agent and are meant to dynamically guide the user during teleoperation. Moreover, our interface is based on first-person view, where the users visualize video stream from the robot's on-board camera through a MR headset. This renders a more intuitive interface that aims at improving the perception of the operator in both remote and stationary tasks.

Chapter 3

Autocomplete System Overview

The proposed system is presented in Fig. 3.1. Joystick inputs are fed to the SVM model at a constant time interval ($T = 1s$); consequently, the predicted motion is classified from our library of motion primitives. Once identified, the system prompts the primitive through the UI. If the predicted motion is acknowledged by the user, it will be executed by the auto-pilot. All the elements of the proposed framework are discussed below in detail.

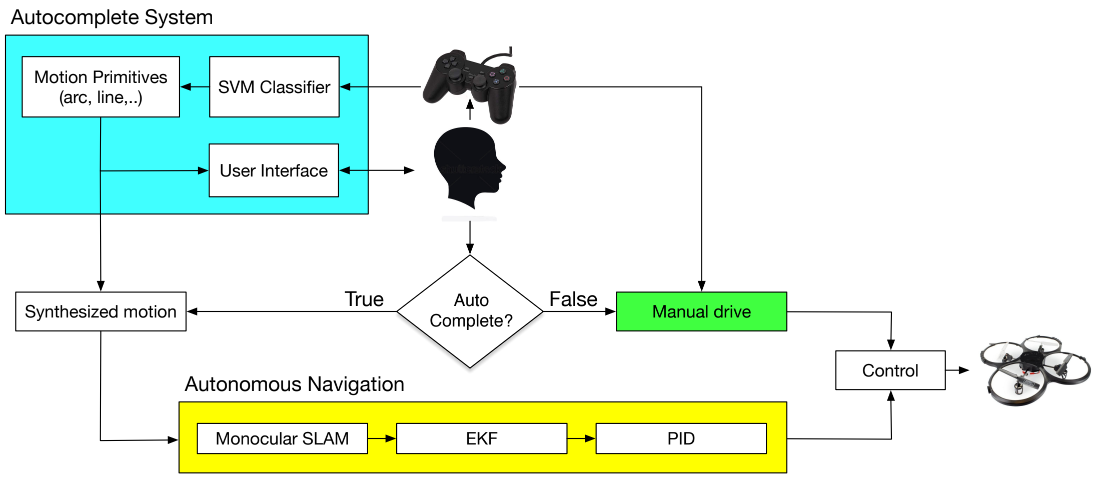


Figure 3.1: Flowchart diagram of the proposed Autocomplete framework.

3.1 Motion Classifier

User intention is predicted by training a Support Vector Machine with our own dataset of user motions. A key advantage of using this type of classifier lies in its ability to train quickly on collected data and provide accurate results for different classification problems [40]. During the training phase, the main objective of an SVM is to find an optimal hyper-plane which separates the data points corresponding to two different classes [41]. However, if more than two classes are to be recognized, the "one-against-one" implementation developed by Kner et al. [42] is used for multi-class classification. In the case of linearly inseparable data, the "kernel" method is used to obtain a highly non-linear separation margin with little additional computational cost to the linear separation method. Essentially, a kernel $\phi(x).\phi(y) = K(x, y)$ is used to map the vectors to a higher dimensional space where linear classification is more feasible. In our case, the kernel is defined as follow:

$$K(x_i, x_j) = \phi(x_i)^T.\phi(y_j) \tag{3.1}$$

For training, the feature vector is defined x by concatenating roll and pitch rotational speed commands $x = (\bar{\Phi}_1, ..., \bar{\Phi}_n, ...\bar{\Theta}_1, ...\bar{\Theta}_n)$, where $n = 100$, and each sample training vector corresponds to a label class y (line, curve, sine motion, etc.). Although most of the motions primitives are 2D in nature, we further classify 3D helical motions when both arc and vertical directions are detected.

794 were used as training examples to train an SVM, which is represented by the following function:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}, x_i) + b, \tag{3.2}$$

where x_i and y_i are training samples and α_i are the Lagrange multipliers. The parameters $\{\alpha_1 ... \alpha_n\}$ are estimated by the following optimization function:

$$\begin{gathered} \max_{\alpha} L = \max_{\alpha} (\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)), \\ s.t. \quad \alpha_i \geq 0, i = 1..., n \;\text{ and }\; \sum_{i=1}^{n} \alpha_i y_i = 0 \end{gathered} \tag{3.3}$$

and the parameter b is calculated as:

$$\begin{gathered} b = -\frac{1}{2} (\max_{j, y_j = -1} \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_j) + \\ \min_{l, y_l = 1} \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_l)) \end{gathered} \tag{3.4}$$

Since the system is task independent, as mentioned before, we do not use any visual information from the scene and we rely only on the joystick data as input to the SVM.

3.2 Motion Synthesis

Suggested paths for Autocomplete are generated by regressing to either a portion of a line, curve, or sine motion.

3.2.1 Line Motion

When the estimated path is that of a straight line, the collected data points are used to regress to a line $y_t = x_t * slope + b$. Thereafter, future desired positions are calculated by setting the look-ahead abscissa x_t to a large number ($x_t = 100$) and calculating the corresponding ordinate.

3.2.2 Arc Motion

Nonlinear regression can be solved either algebraically, or iteratively using a geometric fit by solving an optimization problem. Although, the latter approach produced more accurate results [43], the added accuracy comes at an additional computational cost. However, it is noteworthy that geometric solutions are usually initiated by an algebraic fit to place the solution closer to the global minimum. In this thesis, we rely on Levenberg-Marquard (LM) for the optimization solution, coupled with Taubin's [44] method to obtain a first estimate for the parameters of the desired curve.

In practice, after obtaining the center coordinates (a, b) and the radius R of the circle, a collection of target points $\{x_t, y_t\}$ are sampled by discretizing the circumference of the circle into N points. Where each target point represents the parameters of a command that makes the drone fly to a target position with respect to the current reference point. After that, N commands are queued and executed one after another to achieve an arc motion. Note that we chose N empirically to be big enough ($N = 64$) to ensure the smoothness of the executed arc trajectory. The pseudo-code of constructing queued commands is shown in Algorithm 1.

Moreover, to render a convenient autonomous motion for arc trajectories, we maintain the yaw angle by always pointing the roll axis of the drone in the direction of the tangent to the arc being executed.

3.2.3 3D Helix

3D helical motion is implemented in the same way as the arc motion but with the addition of an increment in the z-axis direction.

Algorithm 1: Construction of the Queued Commands for the Arc/Full-Circle Motion

```
Input: Circle Radius R, Center coordinates $(a, b)$
Output: Commands Queue
begin
    Initialize $A_p$ to angle from circle center, $N = 64$,
      angleIncrement=$\pi/N$;
    if counter-clockwise then
        angleIncrement = -angleIncrement;
    end
    for i=1 to N do
        $x_t = a + R * cos(A_p)$;
        $y_t = b + R * sin(A_p)$;
        cmd = goto($x_t, y_t$);
        CmdQueue.push(cmd);
        $Ap = Ap + angleIncrement$;
    end
    return CmdQueue;
end
```

3.2.4 Sine Motion

Fitting a sine function should be done according to a reference frame, which we take as the global reference frame of the initialized map. In general, after estimating the parameters of the sine motion, like the arc motion, a collection of target points are sampled. However, in the case where the sine wave is determined not to be aligned with the normal direction of the general coordinate frame the following transformations are required:

- First, the major axis of the robot's motion is found using Principle Component Analysis (PCA).
- Next, the angle α from the global x-axis to the major axis is computed and all points are rotated counterclockwise by α.
- Sine function parameters are estimated using LM, then target points are calculated and rotated again clockwise by α.

3.3 Design of the User Interface

To render teleoperation more intuitive, we designed a User Interface (UI), in which we augment the motions that are being estimated by the system on the

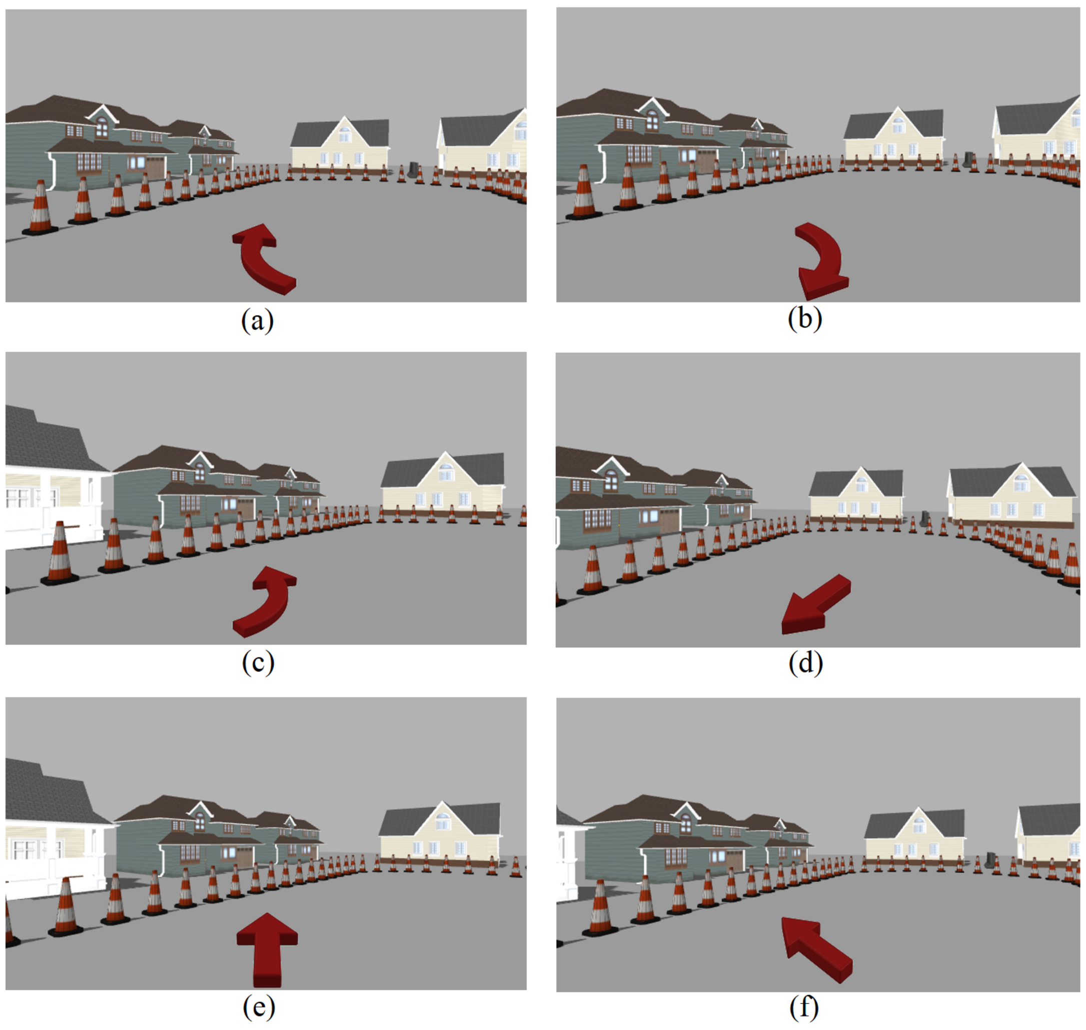


Figure 3.2: Screen shots of the User Interface (UI) with predicted motion primitives: (a)forward arc to the left clockwise, (b) backward arc to the right clockwise, (c) forward arc to the right counter-clockwise, (d) diagonal to the left backward, (e) forward, (f) diagonal to the left forward.

live video stream of the drone's camera. The predicted geometric motion primitives are represented in the form of a straight, curved, sine-shaped, and 3D helical-shaped arrows (see Fig.4.3). Since predictions are done every one second, augmenting the exact predicted angle of the line motion would be confusing to the operator. To solve this, we discretize the range of angles into four, including 0°, 45°, 90°, and 135°, so the user can perceive the general direction of the predicted line motion. As for the curved, sine-shaped, and 3D helical-shaped motions, we render the shapes without taking the corresponding parameters into consideration. Our goal here is to show the user the predicted motion not the exact trajectory.

3.4 Autonomous Navigation

To direct the UAV along a pre-set path, its actual motion needs to be estimated and control inputs need to be sent to it. Motion estimation is achieved using the Visual Simultaneous and Localization and Mapping (SLAM) proposed by Engel [45]. Once the pose of the robot is estimated, the robot is controlled along its path using a separate PID controller for each motion direction (in the xy-plane, along the z-axis, and yaw angle). Figure 3.1 shows the interaction between these components and how they are employed in Autocomplete.

It is worth noting that using Model Predictive Control (MPC) was investigated as an alternative to the PIDs for motion planning; however, the results we achieved were not superior and the PIDs were much simpler to implement.

Chapter 4

Deep Learning and Mixed Reality for Autocomplete

In this section, we summarize the Deep Learning model and Mixed Reality interface with implementation details.

4.1 Deep Learning Model

We first introduce the data collection procedure and then describe the implementation details of the chosen neural network architecture.

Data

Four motion primitives were examined in this study, including that of a straight line, an arc, a sinusoid, and a helicoid. The training data was collected in the Gazebo virtual environment [46] using the "tum_simulator" package to simulate the operation of an AR.Drone 2.0. along with the "joy_node" and "ardrone_joystick" packages which were used to control the drone via a PS3 joystick. Collected data included the joystick's velocity commands measured from the movement of its analog sticks along with their corresponding timestamps. The motion primitives of interest only make use of the drone's pitch, roll, and thrust, which correspond to movements along the x, y, and z axes. This data was retrieved by parsing all the echoed messages being published to the joy topic.
To begin the data acquisition process, a series of several primitives were performed using a stop indicator button, one for each primitive, to signify the end of that primitive. In order to ensure a diverse and complete dataset, directions were varied for each primitive as much as possible, i.e., clockwise, anticlockwise, forwards, backwards, downwards, upwards, etc.
For the second stage of the acquisition, each sequence of a motion primitive's velocity commands had to be grouped separately. Since each primitive had a stop indicator, each new sequence is separated by this indicator. Each of the v_x, v_y,

and v_z values corresponding to the same message were stored in a list as $[v_x, v_y, v_z]$ and will be referred to as a sample. Meaning that each primitive is structured as a list of these samples, $[[v_{x1}, v_{y1}, v_{z1}], [v_{x2}, v_{y2}, v_{z2}], \ldots, [v_{xN}, v_{yN}, v_{zN}]]$ where N represents the sequence size. It can be seen that the data is represented as a three-channel sequence, where each channel represents the sequence of velocity commands corresponding to movements along a certain axis. To determine the required length, i.e. input size, this data was analyzed by examining the number of samples N in every sequence.

When collecting the data we aimed to have each primitive last for at least 2s,

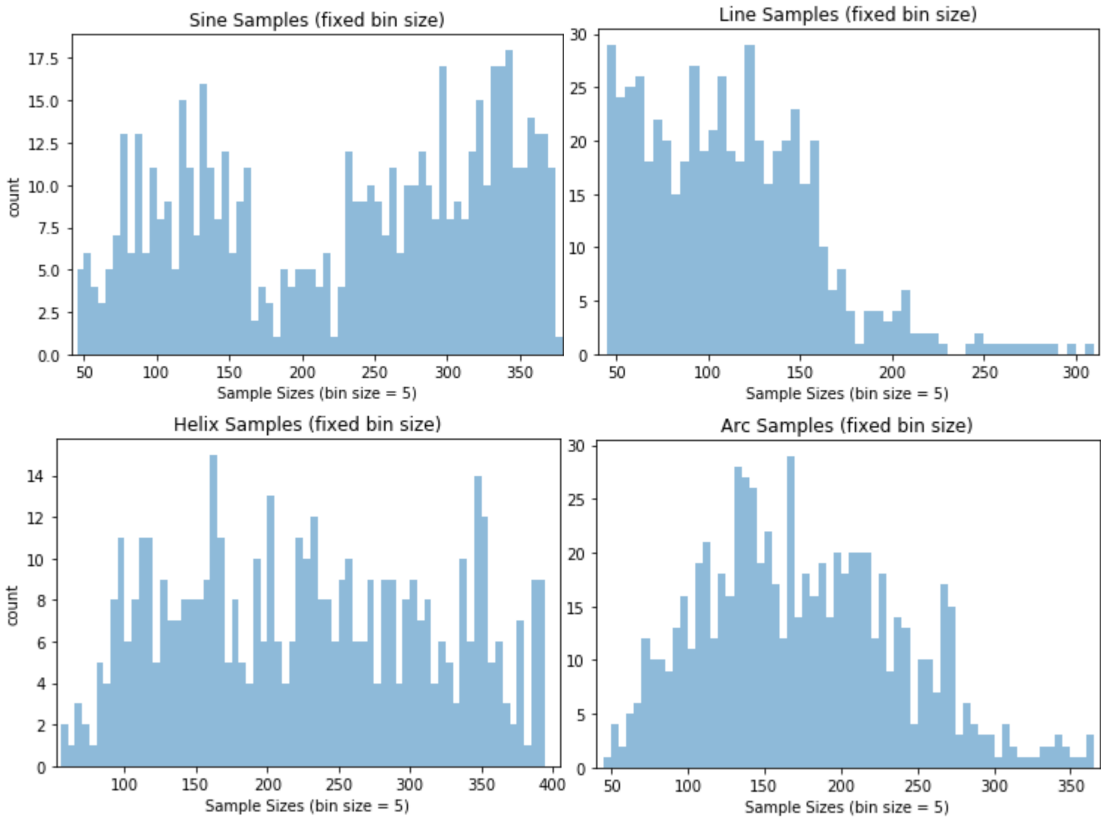


Figure 4.1: Histograms of Primitive Sizes

therefore any primitive found to last less than 1s was considered as an anomaly and discarded. The sampling rate was measured and found to be around 45 Hz, therefore any primitive with a sequence size less than 45 was discarded. The few primitives with a sequence size larger than 400 were attributed to missed stop indicators and also discarded as seen in Fig. 4.1. It is important to note that sinusoidal primitives can be seen as two arcs, thus it is important that our data capture the sinusoids' inflections points in order to make this distinction. The sinusoids have a mean size of 229 and a maximum size of 375, therefore 200 samples is sufficient to capture the inflection point. Finally, all the data was padded because each sequence is of a variable length and truncating this data any further so that they are all of equal sizes would cause a loss of essential information. The resulting data is padded sequences of size 200. The final dataset is made available online at the IEEE*DataPort* [33] and it consists of 2399 primitives with 597 Line, 716 Arc, 617 Sinusoid, and 469 Helicoid. This data was split into 80% training, with 20% of the training data used for validation, and 20% testing.

Model Architecture

In order to account for the class imbalance in the obtained dataset, class weights were incorporated into the learning objectives as it is a good strategy to account for class imbalance. Since the obtained dataset was structured to be made up of sequences, we propose a combination of 1D-CNNs and Gated Recurrent Units (GRUs). These two state of the art techniques are known to be very efficient when dealing with sequential data [47]. The model made use of the Adam optimizer, categorical cross-entropy loss function, and LeakyReLU activation functions in order to avoid exploding/vanishing gradients.

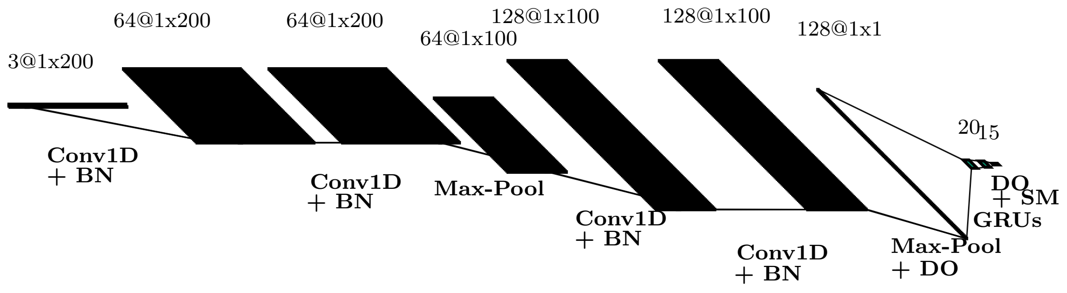


Figure 4.2: Deep Learning Model Architecture

The 1D-CNN layers are able to capture local/short-term dependencies and positional relations between neighboring samples. These low level feature outputs of the convolutional layers are then fed into the GRU layers which are able to learn global/long-term dependencies and outputs the high-level features of the sequences which are then used for classification. To construct this model, first only one convolutional layer combined with one GRU layer was tested. After obtaining the results, more combinations convolutional layers and GRUs were added, but limiting the number of GRU layers to less than three, until finally converging to the network architecture seen in Fig. 4.2 which consists of four 1D-CNN layers followed by two GRUs and finally the Softmax (SM) output. Two Dropout (DO) layers were used to minimize overfitting and Batch Normalization (BN) was used for standardizing the inputs to the network and to provide some regularization.

4.1.1 Mixed Reality Interface

Initially, the idea was to augment predicted trajectories, provided after motion synthesis, onto the live video stream from the drone's on-board camera. This augmentation was represented by the geometric shape that corresponds to each primitive: a straight, sine-shaped, curved, and 3D helical-shaped arrow. Since our main goal was to show the user the general motion, and not the exact trajectory, the range of the line motion's angles were discretized into 0°, 45°, 90°, and 135°. As for the other motions, their shapes were displayed without taking their exact variables into consideration[32].

In this new upgrade of the proposed Autocomplete system, we focused on enhancing the UI, as mentioned in the introduction, it highly impacts the performance of the operator.

To give the predicted motions a 3D texture and render a better intuitive display for the user, we augment the mentioned geometric shapes inside the operation scene using Unity software with Vuforia Engine SDK, rather than coloring the pixel values of the stream image. Moreover, we replace the simple arrow representing the primitive by a set of consecutive arrows that fades toward the vanishing line of eyesight as shown in Fig. 4.3 (right). This is demonstrated to give the user a better insight of where the drone would end up in the near future. Finally, we display the augmented video stream from the drone to the user through an Oculus Rift headset. So, instead of looking at the PC screen, the user will be wearing a headset during teleoperation and as a result the user is isolated from his surrounding. We hypothesize that this isolation will boost the cognition awareness of the user and result in increased focus on task execution. Fig. 4.3 shows the implementation details and flow of data for the UI system.

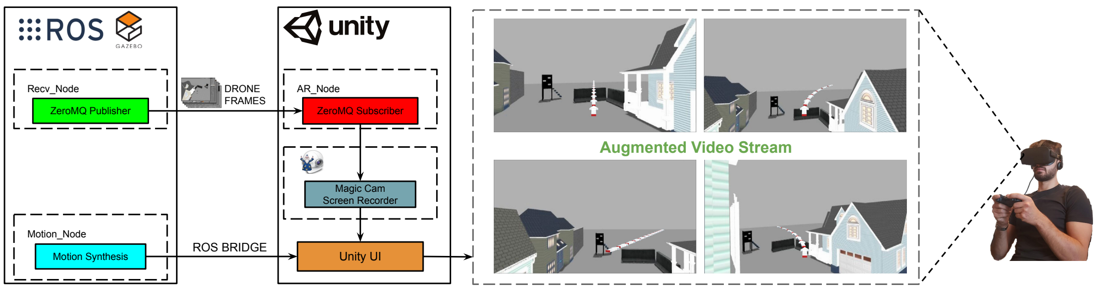


Figure 4.3: Mixed Reality User Interface

In general, augmenting an object inside the real world using Unity needs an *AR CameraObject* which can be added only if a physical camera is connected to the host PC. In our case, since the video feed is coming from a simulated drone camera inside Gazebo, we mimic a physical camera using ZeroMQ (ZMQ) network protocol and Magic Camera software. Frames from the simulated camera are sent from the *ZMQ publisher* on the ROS PC to the *ZMQ subscriber* which opens a video stream window using openCV on the Unity host PC. Then, this stream is captured in real time using Magic Camera screen recorder in which it turns the video feed into a *Virtual Webcam* detected by Unity. In addition to that, the motion primitives that should be augmented in the scene are sent directly from the motion synthesis node to Unity using ROS bridge.

Chapter 5

Results and Experiments

This section presents the experiments we performed to validate the efficiency of our proposed system. We first describe the hardware we used, then delve into the details of our experiments.

5.1 Simulated UAV

For the tests, we used a simulated model of a Parrot AR.Drone 2.0, which includes all the needed sensors for implementing a robust autonomous navigation system. The UAV measures 53cm x 52cm and weighs 420g; it is equipped with an HD camera (720p 30fps) for video recording with a field of view of 73.5°x 58.5°, and a vertical QVGA camera (60 fps) to estimate the ground speed. It is also endowed with a 3-axis gyroscope, accelerometer, magnetometer, and an ultrasound altimeter working at 25 Hz.

In the simulation environment running over ROS, all data from the virtual onboard sensors are used to control the vertical velocity $\dot{z}$ of the quad, along with its roll, pitch, and yaw rotational velocities (Φ, Θ, $\dot{\Psi}$). More specifically, teleoperation of the quadcopter is done using a standard six-axis PS3 joystick. The operator issues control commands $\mathbf{u}$=($\bar{\Phi}$, $\bar{\Theta}$, $\bar{\dot{\Psi}}$, $\bar{\dot{z}}$) $\in [-1, 1]^4$ using the dual analog sticks of the joystick (the left one for the roll and pitch commands and the right one for the yaw and vertical velocity commands); This command represents the reference value for the drone's next maneuver.

5.2 SVM Results

5.2.1 Training

Training the SVM model was done using scikit-learn library on Python [48]. The data-set we used is composed of 794 feature vectors with 217 samples for the line motion, 315 samples for the arc motion and 262 samples for the sine motion. This data is split into 80% for training and 20% for testing the model.

5.2.2 Model-Evaluation

As mentioned before, we chose the kernel trick method for building the SVM model and the available kernel functions for this purpose are: Radial Basis Function (RBF), polynomial function (Poly), and Linear function (Linear). Since our data is not perfectly balanced, we selected more than one performance measure to properly single out the best kernel function for our SVM model. The adopted evaluation metrics include Precision (PR), Recall (RC), F_1 Score, and Accuracy (AC). Note that the F_1 score is calculated using PR and RC according to the following equation:

$$F_1 = \frac{PR * RC}{PR + RC}, \tag{5.1}$$

We performed a grid search to find the optimal kernel and hyper-parameters. The RBF kernel with $C = 100$ and $\gamma = 0.001$ was chosen empirically since it achieved the highest accuracy (82.25%) and F_1 score (79.64%).

5.3 Experiments with SVM Model

For testing and validating our proposed method, we used the Gazebo simulation environment implemented on ROS [46]. The entire system runs on a MacBook Pro 2.3GHz octa-core Intel Core i7 processor and 16 GB of memory.

We evaluate our framework using hindsight bases and we compare the results of two methods of teleoperation: using manual steering and using Autocomplete. Specifically, We use the following metrics:

- *Total Distance* (d) covered by the UAV to approximately identify the deviation from the actual track.
- *Time* (t) taken to finish the track, this is a significant efficiency metric for most navigation tasks (e.g. racing).
- *Trajectory Similarity* between the actual flight trajectory and the ideal one. We are using Hausdorf Distance (HD) [49], a well known metric to evaluate the spatial similarity between different trajectories. The lower the HD value, the more similar are the trajectories.

In order to properly assess the efficiency of the system, experiments of 20 trials for 3 different scenarios using each method (manual and autocomplete) as described in the next 3 subsections.

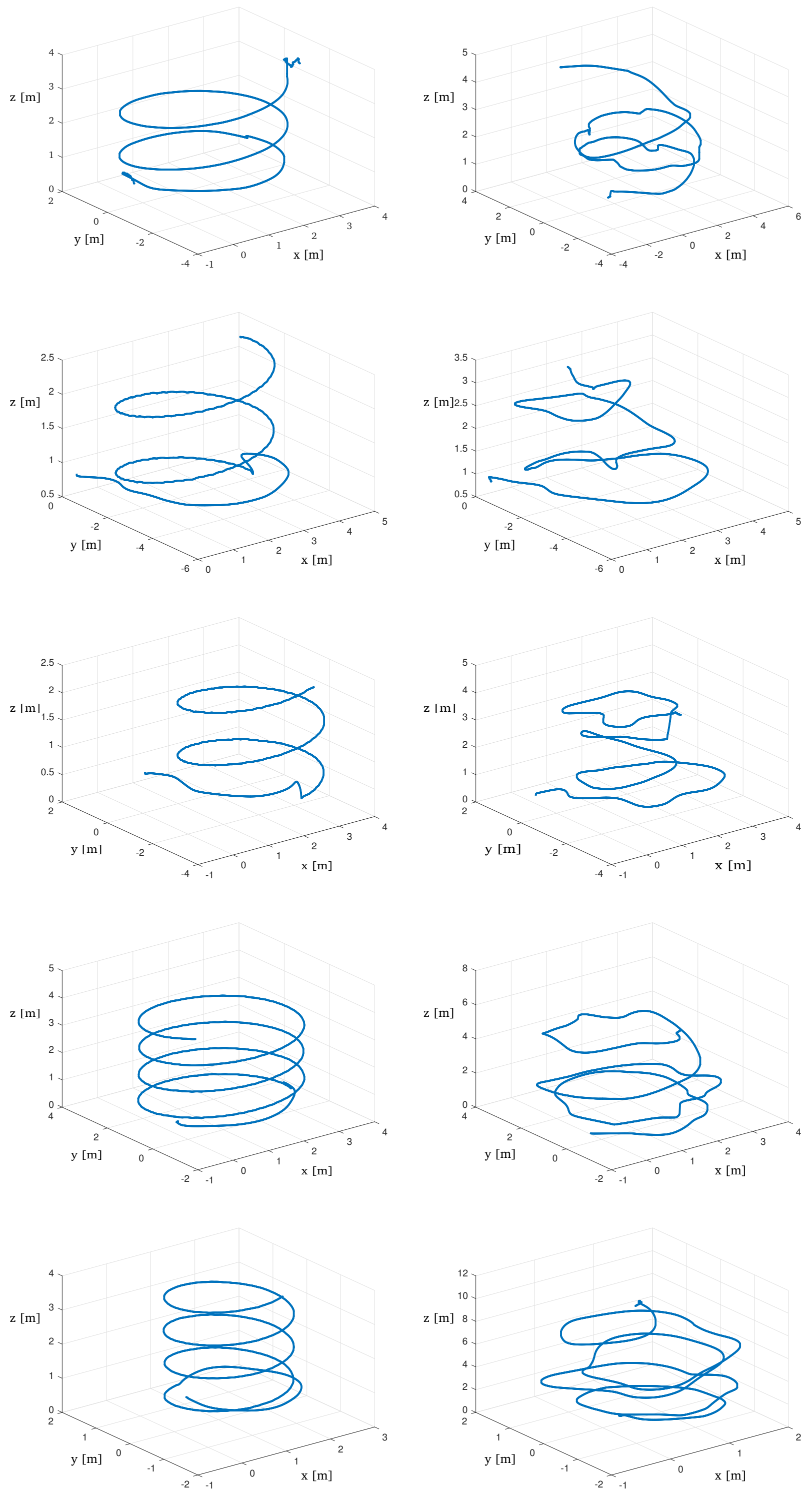


Figure 5.1: 3D helix trajectory using manual steering (right); and autocomplete (left)

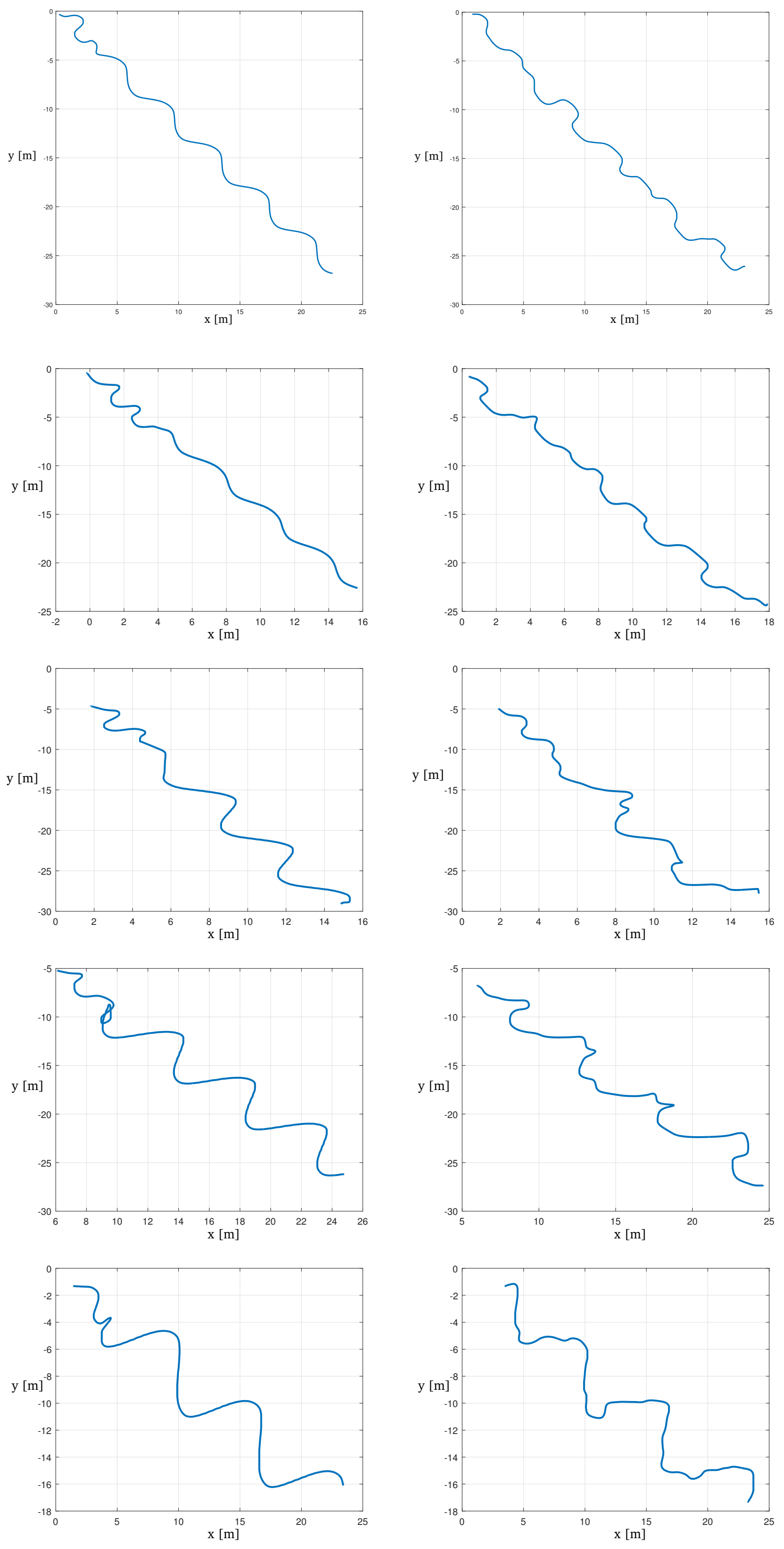
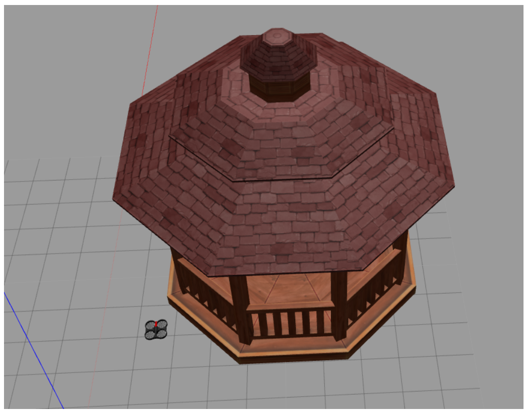
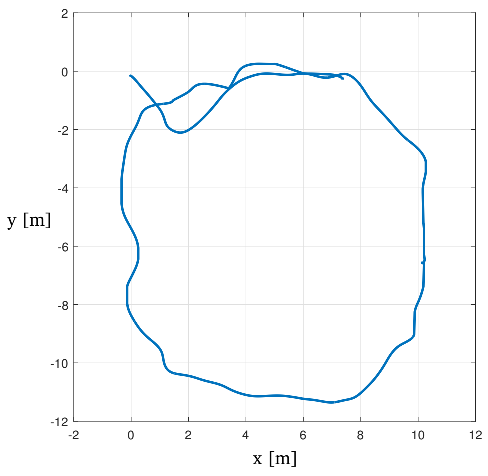


Figure 5.2: Sine trajectory using manual steering (right); and autocomplete (left)





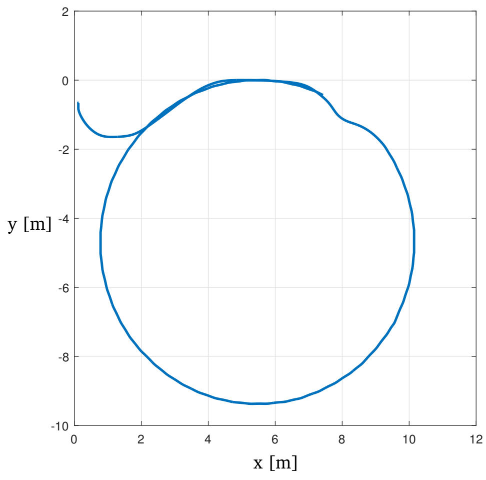


Figure 5.3: A simulated world of a gazebo (top); trajectory around the gazebo using manual steering (left); and autocomplete (right)

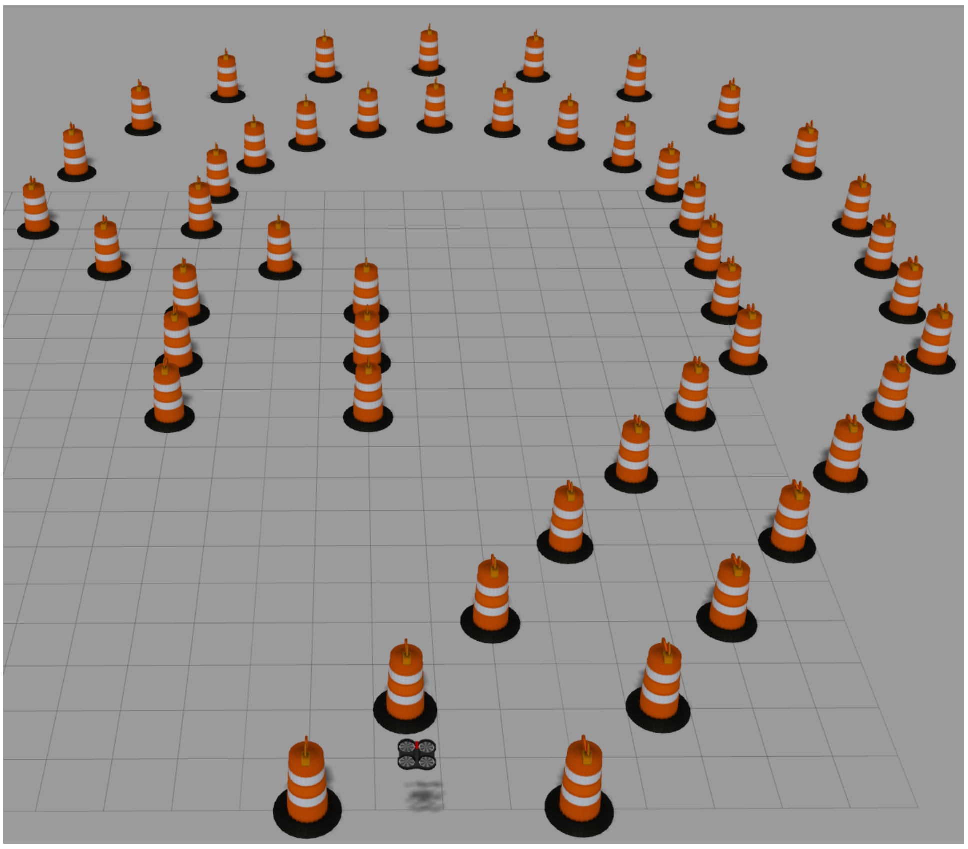


Figure 5.4: A simulated world of the track

5.3.1 3D Helix and Sine Motions

3D Helix type of motion is typical in the inspection of electric pole or wind turbine while Sine type of motion is commonly found in the agricultural applications such as seeding or pesticide spraying . Fig. 5.1 and Fig. 5.2 show the results of 5 different sets of simulations for each type of motion. For each set, the operator was asked first to start with the intended motion and allow Autocomplete (upper figures), then try to mimic the same trajectory manually (lower figures). Results show that manual control could not lead to the intended motions, while Autocomplete feature helped in achieving smoother trajectories regardless of the radius, pitch, amplitude, and total distance of the trajectory.

5.3.2 Motion around a gazebo

In this experiment, the operator was asked to fly the drone around a simulated gazebo model. The best run using manual steering reported $d = 33.11$ meters with $t = 50.27$ seconds, whereas the best run when flying using autocomplete reported $d = 30.15$ meters with $t = 39$ seconds, see Fig. 5.3. Furthermore, the

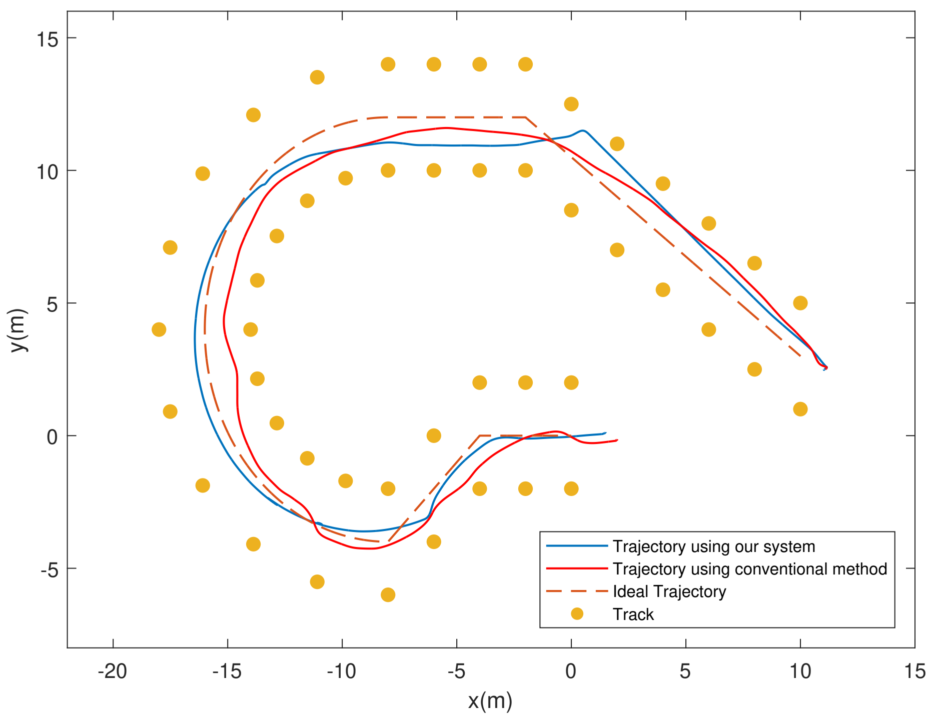


Figure 5.5: Trajectories taken by the quadrotor around the simulated track

average distance traveled while using Autocomplete was 31.005 meters which is less by 9.96% than that of the manual steering. As for the flight time, manual steering recorded an average of 60.53 seconds compared to 40.14 seconds for the flights with Autocomplete enabled.

Table 5.1: Results of Motion around the Track using SVM

	Autocomplete	**Conventional**
Average Distance [m]	58.27916	68.458
Average Time [sec]	56.418	67.5006
Average HD	1.154	1.251

5.3.3 Motion around a track

In this scenario, the operator was asked to execute a flight inside a track. Fig. 5.5 shows the ideal trajectory(the path equidistant from both boundaries of the track), in addition to both trajectories obtained from the best run during manual and Autocomplete modes. Results presented in Table 5.1 show that our system enhances teleoperation in terms of time and distance of flight. Furthermore, the obtained average HD value indicates that trajectories executed using Autocomplete are closer to the optimal trajectory.

5.4 Deep Learning Results

5.4.1 Training

First, the training and validation data were used in order to tune the necessary parameters (DO, number of filters, and filter sizes). Early stopping was used to monitor the validation loss and halt the model's training before it begins to overfit. It can be seen from Fig. 5.6 that the model does not overfit and converges to a training/validation accuracy of 88/87%.

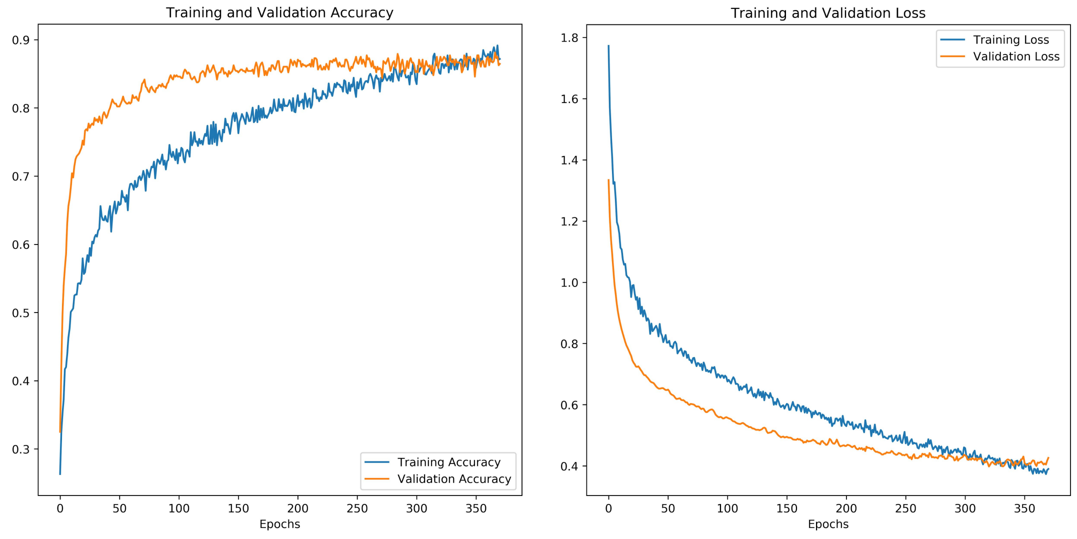


Figure 5.6: Training/Validation Loss

5.4.2 Evaluation

Next, using the same tuned parameters, the validation data was added to the training data for the final evaluation of the model. Since the dataset is unbalanced, the weighted average $F_1 - score$ was used as the metric for evaluating our model. The final model achieved an 89% $F_1 - score$ and its results can be seen in Fig. 5.7. It is apparent from the confusion matrix that the main source of error is from the misclassification of sinusoids and arcs, this misclassification is due to the two primitives being very similar as mentioned earlier.

Although Deep Learning models tend to be bulky and computationally demanding, our model's inference time was measured and found to be around 20ms which is generally considered to be very fast. This model also surpassed the SVM's [32] $F_1 - score$ by about 10%.

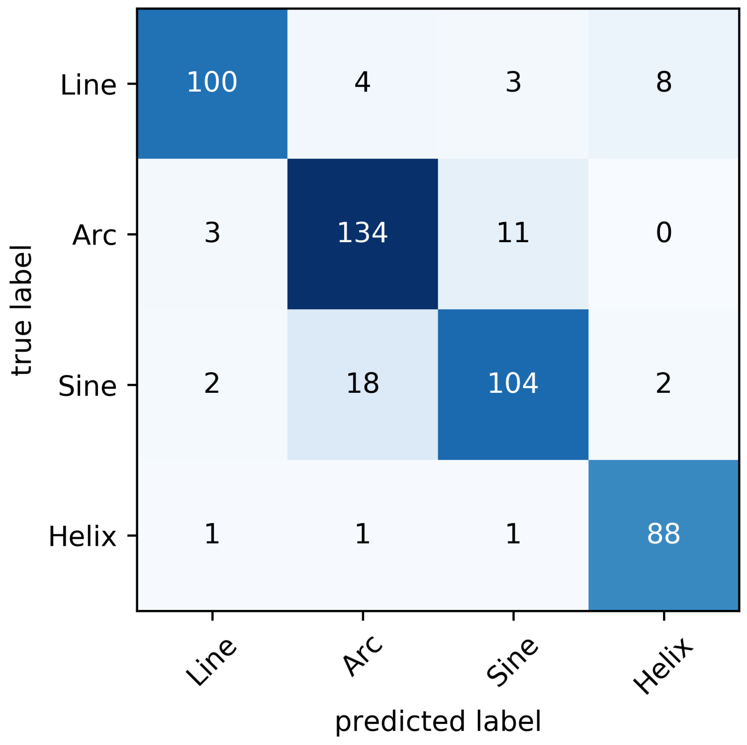


Figure 5.7: Confusion Matrix

5.5 Experiments of MR Autocomplete

We validate the proposed MR Autocomplete framework using human subject testing to evaluate its efficiency over both the **no MR** interface with the SVM model, and traditional teleoperation methods. For this purpose, 9 operators (5 males vs 4 females) were recruited for the experiments after getting approval from the university's Institutional Review Board (IRB). All operators reported that they have little experience driving a drone, and have previously tried Oculus VR headset.

5.5.1 Experimental Procedure

Testing was done using two scenarios: Around a predefined track and a free-hand 3D helix around a simulated building. For each scenario, the operator was given two trials for each of the three teleoperation methods.
For consistency, the same protocol is adopted in all experiments: (1) The operator

is briefed about the tasks and is given 10 minutes to familiarize him/herself with the system, (2) a shuffle of the tasks is done to choose which method with which task is to be tested next, (3) after the experimentation is done, the operator is asked to fill a survey for each method. We present below the objective and subjective results along with the criteria used in the evaluation process.

5.5.2 Subjective Evaluation

Since the main goal of the presented system is to reduce the workload on the user during teleoperation tasks, we subjectively assessed the three systems using the widely used NASA Task Load Index **(NASA-TLX)** [50]. More conveniently, the NASA-TLX software developed by Cao et. al [51] was utilized and operators were asked to report the workload demanded from them for each system based on six sub-scales that represent the mental demand, physical demand, temporal demand, performance, effort, and frustration level during task execution. Fig. 5.8 shows the average scores of the NASA-TLX questionnaire and demonstrates that the proposed Mixed Reality based Autocomplete is superior to the other systems in terms of reducing all the demands of the aforementioned subscales while maintaining a high performance level. This is due to the fact that the proposed Deep Model predicts human intended trajectories with high fidelity and most of the task can be handed to the autopilot, thus reducing operator's overall effort. In addition to that the MR interface offers a more reliable perception for the user allowing him/her to focus on task execution, and as such producing a good performance level during teleoperation.

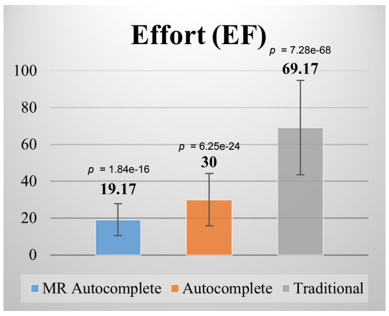
5.5.3 Objective Evaluation

We compare the average distance covered by the drone and time taken to finish the task. The mentioned criteria were considered since, for instance, the distance is an indicator of how much the operator diverges from the track, and the time is crucial for the efficiency of most teleoperation missions. As shown in table 5.2, the Mixed Reality autocomplete framework outperforms both methods in terms of average distance covered: **6%** and **19%** less average distance covered in all runs. Moreover, it is clear that the results of the covered distance have a direct impact on the time aspect. As such, average time taken to cover the track

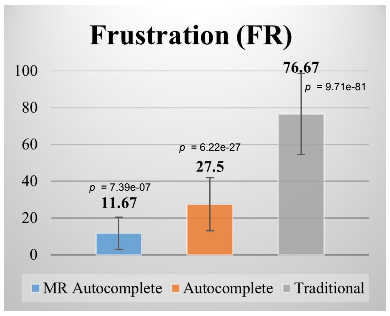
Table 5.2: Results of Motion around the Track using MR Autocomplete

	Average Distance [m]	**Average Time [sec]**
MR Autocomplete	106.37 ± 2.03	95.44 ± 3.67
Autocomplete	113.21 ± 4.34	108.35 ± 3.20
Conventional	127.39 ± 2.35	117.67 ± 1.97

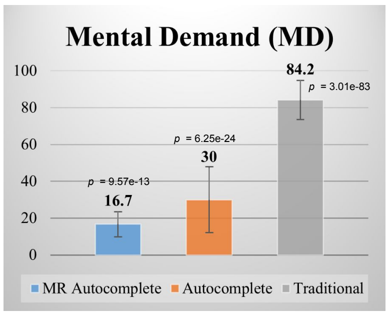
using the proposed method is less than average time taken using autocomplete



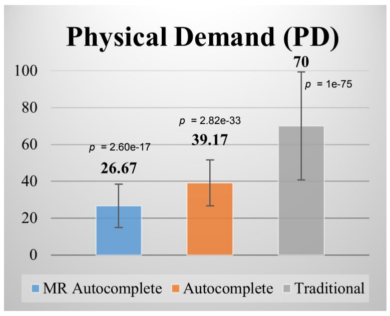
(a)



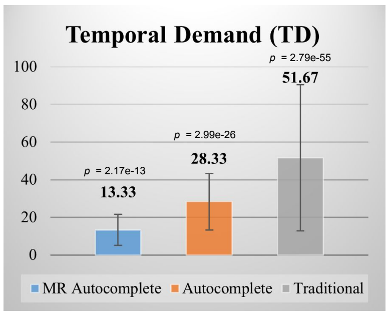
(b)



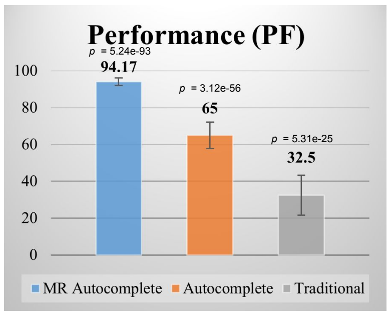
(c)



(d)



(e)



(f)

Figure 5.8: NASA-TLX Survey Scores

and conventional method by **19%** and **22%** respectively. We also note that, in all cases, the standard deviation is small which indicates the consistency of the results reported. Moreover, the calculated *p-value* which is <<< 0.005 in all cases, indicates a strong evidence in favor of the null hypothesis. Finally, Fig. 5.9 shows position data of the drone around the track recorded for the best 3 runs and it obviously demonstrates that trajectories executed using our framework (Fig. 5.9a) are superior in terms of smoothness and closeness to optimal trajectory.

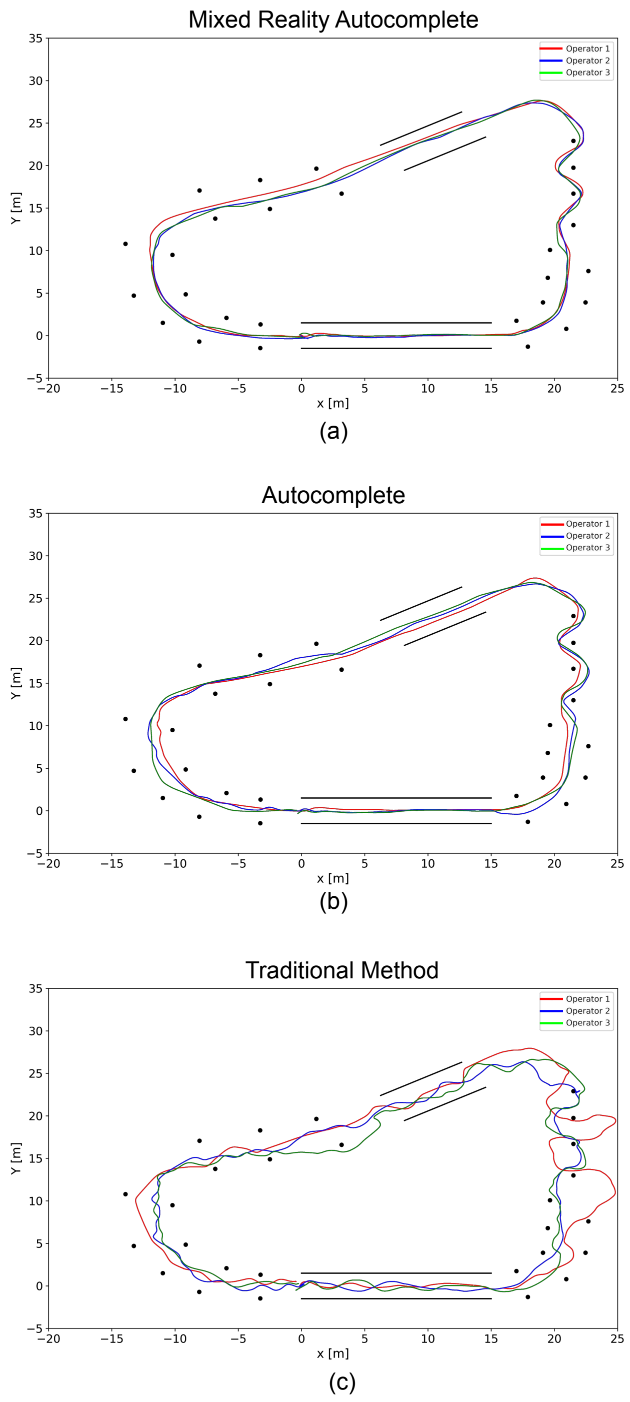


Figure 5.9: Trajectory taken by the UAV in the track experiments. Results represent the best three runs performed by some of the operators using (a) MR Autocomplete, (b) Autocomplete, (c) Traditional Method.

Chapter 6

Conclusion and Future Work

In this thesis, a new framework of teleoperation that aims to assist operators during teleoperation of robots was developed. Our method leveraged the advantages of mixed reality, deep learning, and autonomous navigation to reduce the workload on the user during task execution, while producing optimal trajectories. Machine learning was successfully applied to recognize human intents and then suggest autocompleted trajectories, hence leveraging the advantages of autonomous navigation while keeping the user as the main decision maker. The proposed system aims at reducing workload on humans during the remote control of UAVs and can be potentially implemented on other robots (e.g. Unmanned Ground Vehicles). Human subject experiments of manning a simulated drone proved that our method outperformed the other methods in terms of user experience and in providing better trajectories. Moreover, it demonstrated the effectiveness of our method at improving speed, distance traveled, and smoothness of trajectories. Future work includes developing a hierarchical model to better distinguish between the arc and sinusoidal primitives and increasing the number of motion primitives to include more complex trajectories that could be autocompleted.

Appendix A

Abbreviations

UAV	Unmanned Aerial Vehicle
MR	Mixed Reality
VR	Virtual Reality
SLAM	Simultaneous Localization and Mapping

Bibliography

[1] P. Ramon-Soria, M. Perez-Jimenez, B. Arrue, and A. Ollero, "Planning system for integrated autonomous infrastructure inspection using uavs," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 313–320, IEEE, 2019.

[2] F. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero, "Semi-autonomous teleoperation of uavs in search and rescue scenarios," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1066–1074, IEEE, 2017.

[3] J. Aleotti, G. Micconi, S. Caselli, G. Benassi, N. Zambelli, M. Bettelli, and A. Zappettini, "Detection of nuclear sources by uav teleoperation using a visuo-haptic augmented reality interface," *Sensors*, vol. 17, no. 10, p. 2234, 2017.

[4] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario, "Medical robotics and computer-integrated surgery," in *Springer handbook of robotics*, pp. 1657–1684, Springer, 2016.

[5] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Industrial Robot: An International Journal*, 2004.

[6] D. Szafir, B. Mutlu, and T. Fong, "Designing planning and control interfaces to support user collaboration with flying robots," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 514–542, 2017.

[7] T. Fong, C. Thorpe, and C. Baur, *Collaborative control: A robot-centric model for vehicle teleoperation.* Pittsburgh: Carnegie Mellon University, The Robotics Institute, 2001.

[8] J. Y. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1231–1245, 2007.

[9] C. Abras, D. Maloney-Krichmar, J. Preece, *et al.*, "User-centered design," *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, vol. 37, no. 4, pp. 445–456, 2004.

[10] T. B. Bodin, "Behavior flexibility for autonomous unmanned aerial systems," 2018.

[11] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *2015 International Conference on Intelligent Environments*, pp. 140–143, IEEE, 2015.

[12] D. Aarno, S. Ekvall, and D. Kragic, "Adaptive virtual fixtures for machine-assisted teleoperation tasks," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2005.

[13] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Robotics and Automation (ICRA), IEEE International Conference on*, 1997.

[14] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *Transactions on Industrial electronics*, 2005.

[15] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, H. Van Brussel, and M. Nuttin, "User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving," *Autonomous Robots*, 2008.

[16] M. Gao, J. Oberländer, T. Schamm, and J. M. Zöllner, "Contextual task-aware shared autonomy for assistive mobile robot teleoperation," in *Intelligent Robots and Systems (IROS), IEEE International Conference on*, 2014.

[17] W. Yu, R. Alqasemi, R. Dubey, and N. Pernalete, "Telemanipulation assistance based on motion intention recognition," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2005.

[18] K. Hauser, "Recognition, prediction, and planning for assisted teleoperation of freeform tasks," *Autonomous Robots*, 2013.

[19] T. Carlson and Y. Demiris, "Human-wheelchair collaboration through prediction of intention and adaptive assistance," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2008.

[20] N. Battilani, P. R. Giordano, and C. Secchi, "An assisted bilateral control strategy for 3d pose estimation of visual features," in *Intelligent Robots and Systems (IROS), IEEE International Conference on*, 2017.

[21] C. Masone, P. R. Giordano, H. H. Bülthoff, and A. Franchi, "Semi-autonomous trajectory generation for mobile robots with integral haptic shared control," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2014.

[22] X. Yang, K. Sreenath, and N. Michael, "A framework for efficient teleoperation via online adaptation," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2017.

[23] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2002.

[24] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *Transactions on Robotics*, 2015.

[25] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning motion primitive goals for robust manipulation," in *Intelligent Robots and Systems (IROS), IEEE International Conference on*, 2011.

[26] K. Khokar, R. Alqasemi, S. Sarkar, K. Reed, and R. Dubey, "A novel telerobotic method for human-in-the-loop assisted grasping based on intention recognition," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4762–4769, IEEE, 2014.

[27] T. Carlson and Y. Demiris, "Human-wheelchair collaboration through prediction of intention and adaptive assistance," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3926–3931, IEEE, 2008.

[28] M. Gao, J. Oberländer, T. Schamm, and J. M. Zöllner, "Contextual task-aware shared autonomy for assistive mobile robot teleoperation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3311–3318, IEEE, 2014.

[29] M. Laskey, C. Chuck, J. Lee, J. Mahler, S. Krishnan, K. Jamieson, A. Dragan, and K. Goldberg, "Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 358–365, IEEE, 2017.

[30] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang, "Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 416–422, IEEE, 2019.

[31] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[32] M. K. Zein, A. Sidaoui, D. Asmar, and I. H. Elhajj, "Enhanced teleoperation using autocomplete," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9178–9184, IEEE, 2020.

[33] J. Elhalabi, M. Al Aawar, M. Kassem Zein, I. H. Elhajj, and D. Asmar, "Drone motion primitive dataset," 2020.

[34] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *International journal of advanced robotic systems*, vol. 5, no. 1, p. 1, 2008.

[35] S. A. Green, J. G. Chase, X. Chen, and M. Billinghurst, "Evaluating the augmented reality human-robot collaboration system," *International journal of intelligent systems technologies and applications*, vol. 8, no. 1-4, pp. 130–143, 2010.

[36] E. Ruffaldi, F. Brizzi, F. Tecchia, and S. Bacinelli, "Third point of view augmented reality for robot intentions visualization," in *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pp. 471–478, Springer, 2016.

[37] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 78–86, 2018.

[38] M. E. Walker, H. Hedayati, and D. Szafir, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 202–210, IEEE, 2019.

[39] D. Lee and Y. S. Park, "Implementation of augmented teleoperation system based on robot operating system (ros)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5497–5502, IEEE, 2018.

[40] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3-Volume 03*, pp. 32–36, IEEE Computer Society, 2004.

[41] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[42] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," in *Neurocomputing*, pp. 41–50, Springer, 1990.

[43] N. Chernov and C. Lesort, "Least squares fitting of circles," *Journal of Mathematical Imaging and Vision*, vol. 23, no. 3, pp. 239–252, 2005.

[44] G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1115–1138, 1991.

[45] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Intelligent Robots and Systems (IROS), IEEE International Conference on*, 2012.

[46] H. Huang and J. Sturms, "tumsimulator."

[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[49] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE transactions on intelligent transportation systems*, vol. 11, no. 3, pp. 647–657, 2010.

[50] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, pp. 139–183, Elsevier, 1988.

[51] A. Cao, K. K. Chintamani, A. K. Pandya, and R. D. Ellis, "Nasa tlx: Software for assessing subjective mental workload," *Behavior research methods*, vol. 41, no. 1, pp. 113–117, 2009.