Drone Human-Robot Interface

LIDIC: Live interactive drone image control

Group 2

by

Olav Ausland Onstad, Joachim Thorsen Larsen, Jeanett Wesstøl

in

IKT 213

Mashine Vision

Faculty of engineering and science

University of Agder

Grimstad, November 21, 2022

Abstract

Human-robot interaction is how a human can interact directly with a robot, for instance through speech, a controller, or gestures. In this case a computer or drone will recognize hand gestures and use them as input to control the drone. Data points from gestures are fed to a neural network model to predict which gesture are being performed. When the predicted value is over a threshold the corresponding command for that gesture is transmitted to a Tello EDU drone to control it. That way the drone can be controlled by a user either from a webcamera or from the drone camera.

Contents

List of Figures

List of Tables

Listinger

1 Introduction

Goodrich and Schultz state that “Human-robot interaction is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans” [3]. One of the applications can be to make a drone interact with a human using machine vision. Machine vision is the field of making a computer get some context of what is in an image. For example extracting cars, people or other objects from an image [14].

1.1 Project idea

For this project we wanted to create a human-robot interface using a Tello EDU drone. We will create software that can translate hand gestures, "seen" by the drone, into commands for the drone to execute. Commands will include moving to the left, right, up, and down by pointing in those directions. The drone should also be able to rotate to the left and to the right, and stop on given commands. Another idea is to pair the drone with the "driver", making it possible to control the drone without interference from the surrounding people.

1.2 Use Cases

There is a lot of use cases for this problem, our main label is: human-robot interaction, which is to control or interact with a robot through human gestures. For example this can be further improved by creating interactive system in which you can create your own data sets and apply your own actions to a gesture. This can help to create an entry way for students and children into machine learning and artificial intelligence.

Another example is to make the drone more autonomous. Lets say you are hired to detect leaks or cracks, instead of using a controller you can signal the drone to take a picture or start a video feed by gestures. Then you can signal the drone to follow the operator.

1.3 Key problems to solve

The key problem in this project is to detect the hand, train a model on our gestures, recognize it and make the drone perform the command.

For the operator the key problems will be to recognize the operator, and connect the hand to the operator. One problem is also to make the drone follow the operator

2 Theoretical background

2.1 Human-robot interaction

Human–Robot Interaction is using robotic systems by or with humans. Interaction is defined as communication between robots and humans. The communication can be divided into two categories; remote and proximate interaction. Remote interaction is when the robot and the human are at different locations, while proximate interaction is when the robot and the human are colocated. In this project we use proximate interaction when controlling a drone with gestures [3].

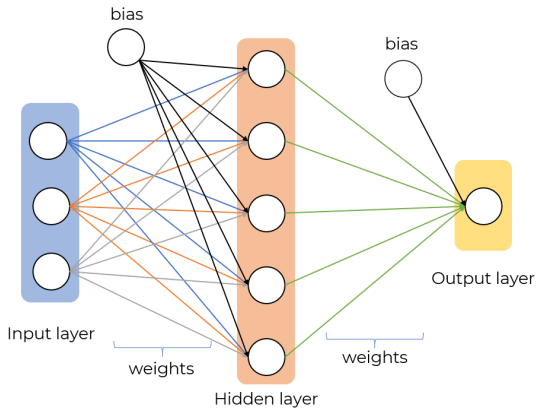
2.2 Machine Learning

Machine learning can be a lot of things, but more often is it referred to as a deep neural networks and similar. Machine learning can be things like simple linear regression and algorithms that learn "themselves" in some way or another. Essentially, it is an algorithm which takes in data and imitate the way a human learn, progressively getting better and more accurate[18].

Machine learning derives from artificial intelligence, meaning that every machine learning algorithm is also classified as artificial intelligence, but not the other way around. Artificial intelligence is different to machine learning in the way that it necessarily does not improve and self learn the way machine learning does, but rather behave similar / as well as a human or better. Examples of this could be a behaviour tree, or a path finding algorithm like A*.

2.3 Neural Networks

When we talk about neural networks there is a lot of different variations, but most of them have the same baselines, the neuron and back-propagation. A neuron is a concept taken from nature, you can look at a neuron like a messenger. The neuron receives some input (message) and outputs it to other recipients. A human brain contains billions of these neurons which takes in a lot of inputs and outputs as an electrical signals based on the inputs and weights. This is where the idea of creating a generalized model which can converge different data and output accurately came to be. These models (neural network) tries to recognize a series of patterns from the input they are given and tries to learn[17].



This is a simple neural network which has one input layer, one hidden layer and a output layer. The input layer has a matrix of 3x1 which defines that it can take in three values. The output is a 1x1 matrix which defines that it outputs a single value. The hidden layer is a $5x1$ matrix, each of the neurons will sum the values and pass it through an activation function to the output. The input of these neurons is defined by weights and biases.

Figure 1: Visualization of a neural network[5].

The weight and biases is changed based on the error from the output and the back-propagation algorithm. The back-propagation in this network will try to change these weights and biases to get a correct result. As we can see, every neuron is connected to each neuron in the next layer, this is called a fully connected neural network[11], fig. 1.

2.4 Gradient Descent & Stochastic Gradient Descent

The gradient descent algorithm is very often used in machine learning as it is very effective and quite intuitive. This algorithm is used in the neural network to minimize a loss function, although it has some restrictions. The functions needs to be convex or quasi-convex, as well as being differentiable.

When a function is differentiable means that a function has a property where it has a derivative for every point in its set[6].

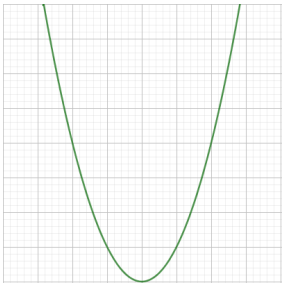


Figure 2:
Convex function,
$f(x) = x^2$

Figure 3:
Quasi-convex function,
$f(x) = x^4 + x^3 - x^2 - x$

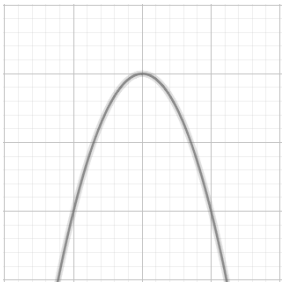


Figure 4:
Concave function,
$f(x) = -x^2$.

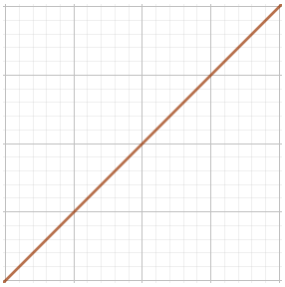


Figure 5:
Linear function,
$f(x) = x$.

Gradient descent can handle two of these functions, the ones that are convex or quasi-convex. A gradient descent can only handle functions that has a global minimum point. Gradient descent is an iterative function, this function l start with an initial value P_0 find the gradient of that position, multiply it with a learning rate (scalar) then it will subtract the calculated value from its position and make a step.
Its important to choose a good learning rate as a value to small or a value to high wont converge to the global minimum. If you have a value too small you might not reach the global minimum before you run out of iterations. If you go the other way around you might just jump over the global minimum and never reach it.
When we use gradient descent on a function which is **quasi-convex** (fig. 3) there is a chance that it wont find the global minimum as it might stumble upon a saddle point, or a local minimum. So although the gradient descent algorithm is frequently used in machine learning, there is no guarantee that it will find and converge.

Stochastic gradient descent is quite similar to normal gradient descent, the difference being stochastic in nature, which means random. The gradient descent uses all the data points available with a loss function to calculate a new step. Stochastic on the other hand uses a random subset of the all the data points to calculate the next step. This is quite useful for when you have a large dataset since you do not have to calculate as many terms. The number of data in the subset is called the batch size.

3 Choice of Technologies and tools

3.1 OpenCV

OpenCV is an open source computer vision and machine learning software library with more than 2500 optimized algorithms. These algorithms has many applications for example, it can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects,etc. It is widely used with more than 18 million downloads. OpenCV has interfaces for C++, Python, Java and MATLAB and supports Windows, Linux, Android and Mac OS [9]. OpenCV is used in this project to handle the video streams and convert the video frames to numpy arrays.

3.2 Python

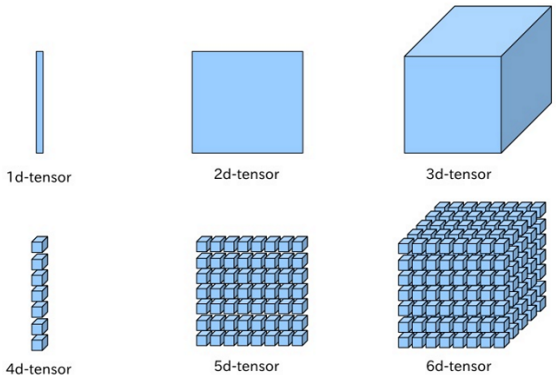
"Python is an interpreted, object-oriented, high-level programming language with dynamic semantics" [10]. according to the annual stack overflow survey, python is the 6th most popular programming language [13]. Python is chosen for this project because of the simple syntax, 3rd party tools, and popularity. The Python version used for this project is 3.8.

3.3 Mediapipe

Mediapipe is a library from google which implements a lot of features, this can be face detection, pose detection and hand detection. Focusing on the MediaPipe's hand detection, they are using a machine learning pipeline which is utilizing multiple machine learning models. Firstly the pipeline goes through a palm detection model which will find the hands in an image and return the oriented bounding box of the hand. Then the pipeline will crop the hand based on the bounding box and send it to the next model. This model will take the cropped image and return a set of 21 keypoints with three dimensions, $[x, y, z]$. When we separate the tasks into different models, the need for data augmentation such as scaling, translation and rotation is drastically reduced[4]. This can also make it easier for the models to converge to a high accuracy as they focus on a single objective. For the dataset used as ground truth for this model, they manually annotated $30K$ images with 21, 3D ($[30000, 21, 3]$) coordinates, where the Z value is taken from the image depth map.

3.4 TensorFlow & Keras

Tensorflow is a machine learning library which allows us to create machine learning models without having to create everything from scratch, such as back-tracking and other. Keras is a submodule / sublibrary in Tensorflow with a higher abstraction, which makes it easier to take the step into machine learning.



When working with machine learning models and neural networks it is quite often you end up working with tensors, hence the name *Tensorflow*. A 1d tensor is just an 1 dimensional array. A 2d tensor is a matrix. From rank 3 and up the name tensor is used.

Figure 6: Example of different dimensions of tensors[1].

When thinking of tensors you can think of a collection or bucket of numbers with different ranks (dimensions). Higher dimensions of tensors are often used as a way to represents data, and different relationships between data. For a neural network model which takes in a black and white image with a size of 640x480, this could be represented as tensor with dimensions [640, 480, 1], since you only have a grayscale value. If your model needs some relationships between the colors in the image, the input could be a tensor of shape [640, 480, 3].

3.5 Tello EDU drone

The drone of choice in this project is the Tello Edu drone from DJI. TELLO EDU SINGLE COMBO Tello EDU is an programmable drone [15], using an API with a UDP protocol. This drone was chosen because of the experience with the simple API as well as the drone availability as school property.

4 System design and implementation

The system starts by connecting to the Tello drone and the computer. The Tello drone sends a video feed to the computer and the computer runs image processing on the video steam. After this the computer runs image classification, which will get the gesture shown. Then the computer sends a request to the Tello API, with a command based on the predicted gesture. The drone verifies if this is a correct command, and if it is, the command is executed. This process is done over and over in a loop for as long as the program is executing. Figure 7 is the system design shown as a diagram.

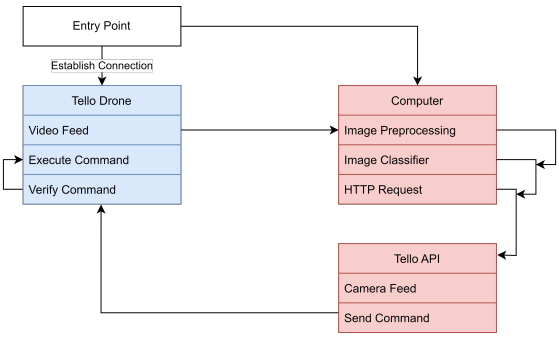


Figure 7: System Design

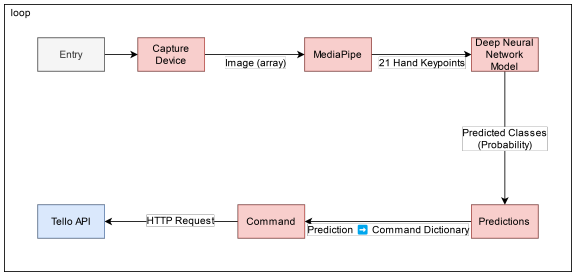


Figure 8: Processing Flow

Figure 8 shows the processing flow of the application and how the different components interact with each other to turn an image into a command for the drone. We start off by getting a capture device which outputs an image represented by a multidimensional array, MediaPipe takes in this image and outputs 21 keypoints which our DNN model takes in as input. After our model has processed the keypoints, it will output a class prediction which contains 5 different probabilities of each class. After this the application will pick the highest probability and check if its over a certain threshold, if it is send the mapped command to the drone and execute it.

4.1 Data Collection and datasets

For the machine learning process we need some training data that will be our gestures. This will be a large list of data points with a specific gesture connected to it. This way the model will know which gestures correspond with what data. To more easily create the data points we set up a video stream where we perform the desired gestures. 4 times per second a frame from the video stream is processed and the label provided prior will be set to the data points and appended to a CSV file.

The exact data appended to the CSV will be the normalized x and y coordinates of every joint in the hand found by Mediapipe. That way the machine learning process will be able to train on the data points provided by Mediapipe and the relationship between them.

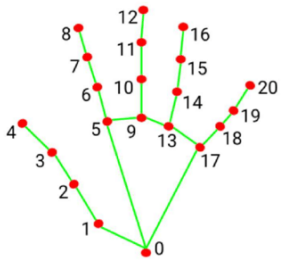
The reason we only collect 4 frames per second is mainly to avoid having a lot of nearly identical data. The difference between 2 frames in a video streaming at 30 FPS would be minuscule, not to mention the motion blurred frames would not be ideal. By providing more clear and specific data we can make sure the model will be as good as possible. One important thing is to also make sure we record the gestures at varying distances from the camera. This is to avoid unintentionally training the model on gestures at a specific distance from the camera, and instead focus on the relationship between the points.

4.2 Neural Network Model & Training

For the deep neural network model, we choose to have an input shape of $(42,)$, which corresponds a 42x1 matrix (vector) we can call $\vec{i}$. This vector contains every normalized x & y position of joints in the hand, fig. 9. Since Mediapipe outputs 21 points (0-20) the total amount of data we gather

respectively to the y & x position is $21 \cdot 2$, which gives us the 42 data points.

$$\begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ \vdots \\ x_{20} \\ y_{20} \end{bmatrix}$$



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 9: Mediapipe hand landmarks [4]

For the last layer we set the number of units (perceptrons) to be the number of gestures we want, in our case we train the model on 5 labels, which corresponds to 5 outputs. We use a **softmax** activation function to get a smooth percentage between $0 - 100\%$, this percentage indicates the probability of how confident the network is that the input corresponds to a class (label).

$$\begin{bmatrix} 0.074 \\ 0.026 \\ 0.009 \\ 0.12 \\ 0.78 \end{bmatrix}$$

Figure 10: Example output of our deep neural network.

For the optimizer we chose to use "Adam", this optimizer implements the **Stochastic Gradient Descent**2.4 method, which works like a normal gradient descent method but takes a random subset of the training data with a fixed size refereed to as the **batch** instead of the whole set[12]. For the loss function we use the **sparse categorical crossentropy** which is commonly used when dealing with two or more classes or labels, this is the recommended loss function[16]. For the metrics we keep track of the accuracy of the model, this way we can see how good the model converges, fig. 17.

Now that we have modelled what input the model expects and which output we want, we can start by modelling the hidden layers. Our model has a total of five fully connected layers (included input and output) and three hidden layers. In each of the hidden layers we have 20 perceptrons with a ReLU activation function, since we have a 21 hand joints / input data points. In between the first, second, and third layers we have a dropout layer which is there to prevent over fitting to our training data.

4.3 Implementation

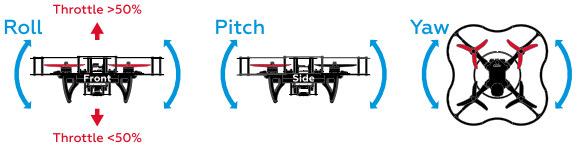
In our project we have two main files, one for creating our machine learning model and training it, as well as labeling new data. The other file is the actual system for handling the drone and predicting the users gestures to execute the commands.

Initially we start by creating two different threads, one for controlling and getting the camera feed from the drone. The other thread is for processing and predicting the hand gestures from an arbitrary camera feed, and then sending the commands to the drone through a UDP protocol.

In the controller we have implemented an enumerator, this enumerator chooses what type of control we are using, the options are: keyboard, xbox-controller, and gestures. For each iteration when using the gesture controller, we predict what the drone should do and if the prediction is over a certain threshold we send a new command to the drone. This is one of the main reasons we have the video and the controls in different threads, as continuously predicting and sending commands will make the video have a huge latency.

```
commands = {
    '1': f'rc {0} {0} {25} {0}',     # Up
    '2': f'rc {0} {0} {-25} {0}',    # Down
    '3': f'rc {-25} {0} {0} {0}',    # Left
    '4': f'rc {25} {0} {0} {0}',     # Right
    '0-1': f'rc {0} {25} {0} {0}',   # Forwards
    '0-2': f'rc {0} {-25} {0} {0}', # Backwards
    '0-3': f'rc {0} {0} {0} {-100}',# Rotate Left
    '0-4': f'rc {0} {0} {0} {100}'   # Rotate Right
}
```

Figure 11: Enumeration of commands accessible



rc $\{roll\}$ $\{pitch\}$ $\{throttle\}$ $\{yaw\}$

Figure 12: Drone Terminology[2]

The model is implemented with a deep neural network with one input layer, one hidden layer and an output layer with 7 classes. Each class in this output layer will output a value between $0-1$, this represents the confidence of each class. 0% represents that the model does not think the input is the specific class, 100% represents that the class is very confident that the input should map to this class. If the output with the highest prediction is greater than 50%, this output is chosen as the command. The command is sent to the drone using the Tello-api, where the command corresponds to a function in the API, such as `up`. 'rc roll pitch yaw throttle' rc 0 0 0 100

For the implementation of creating a dataset, we have a continuous video stream 4.1. The user selects a label between $0-9$, and presses a key to toggle the recording. The user then goes through each of the desired gestures and labels while filming the corresponding video of the wanted gesture. The program will capture every 0.25 seconds, and save the keypoints and corresponding label to the desired output .csv file.

5 Results and discussion

Initially we started of by creating our gesture classifier by modelling the neural network in respect to grayscale images, in the shape of a 1d tensor ($[42, 1]$). When creating our dataset with this model, we created our own software which was able to extract frames from a video, label then and putting it in a respective folder. We also experimented with adding bounding box support to this software as shown in fig. 13, here we labeled (x_{min}, y_{min}) & (x_{max}, y_{max}) and saved the image with this name. A better way could be to create csv file which stores all of the information.

We tried making the hand about the same size and shape using the bounding box and process it into our model. This way we would get rid of a lot of noise and the data would be similar no

matter the distance from the camera. We got a problem whit the proportion of the hand, because some of the fingers became much longer than the other fingers and the hand became wider than its original width.

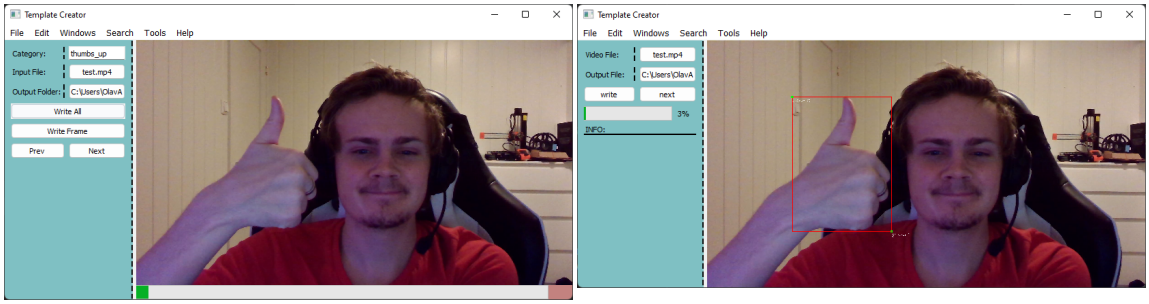


Figure 13: Custom template software.

When doing as described above we also tried a lot of different ways to preprocess the data, for example grayscaling the image, then applying a Gaussian filter, to get the edges and make the model focus on this. Although this gave decent results, there was a lot of overhead (noise) which could make it hard for the model to predict. Unfortunately this software is deprecated as we choose to go with keypoint detection for the gesture recognition. The keypoint detection is the hand recognition in mediapipe, and is traind on multiple hand shapes, color and distances. This make the hand detection more robust than creating our own. This is one of the reason we choose to use the keypoint hand detection from mediapipe.

Another reason we used Mediapipe for this project is that the built in hand detection saves us a lot of time and errors. Creating our own hand detector would take a lot of time which we could use on other parts of the project, it would also be hard to get enough representative data to work on multiple people and skin colors.

We also looked into using a projection-matrix and create our dataset out of this, unfortunately based on the time we had to finish this project this was not feasible, and an additional reason to choose mediapipe

5.1 The operator

To be able to control the drone, we wanted to detect an operator, thus the operator is the only one that will be able to control it without other people interfering. Our initial approach was to make the operator wear a specific QR-code, if this code is in our database we set this person to be the operator. This process would detect skeletons in a image, search for QR-code and check if there is a skeleton connected to this qr-code. If this is the case we would only send commands to the drone if a detected hand is connected to the skeleton. This method gave sub optimal results, The detection had problems detecting the QR code because of reflection from the light, and there was a difficulty finding the QR code if the shape got bent in some direction. We tried the same approach with a simple edge detection on a specific shape, this gave the same sub optimal results.

For the QR code and shape recognition we used Mediapipes functionality, which is already trained, as we use the hand detection. The detection works fine on a static image, but on a video with a

moving shape there was a major problem with getting the shape right for each frame. We chose to leave this idea to try something else.

For the final approach we tried using face detection for detecting a operator in a frame, then go through the same process as before to check if the detected face is connected to a hand. With the time we had left this was unfortunately not feasible, but we do believe this is would yield good results as long as the person is facing the drone. A problem we have discussed with this approach is if the person is to fare away from the drone, then it will be harder to find and recognise the face.

5.2 Drone control

Similar to recognizing the face when the drone is far away from the operator, it is also a problem to detect the hand and the hand gestures when the distance is too large. For this project we then decided to use the web-camera on the computer to detect the hand gestures, such that the distance does not change as the drone move. Another aspect is that the drone's camera is of low quality, this makes it hard to predict since the feed can be noisy and the resolution is small.

The gestures detected and processed is right, left, up, down, turn right, turn left, forward, backward and stop. The commands are shown in fig. 14 and fig. 15

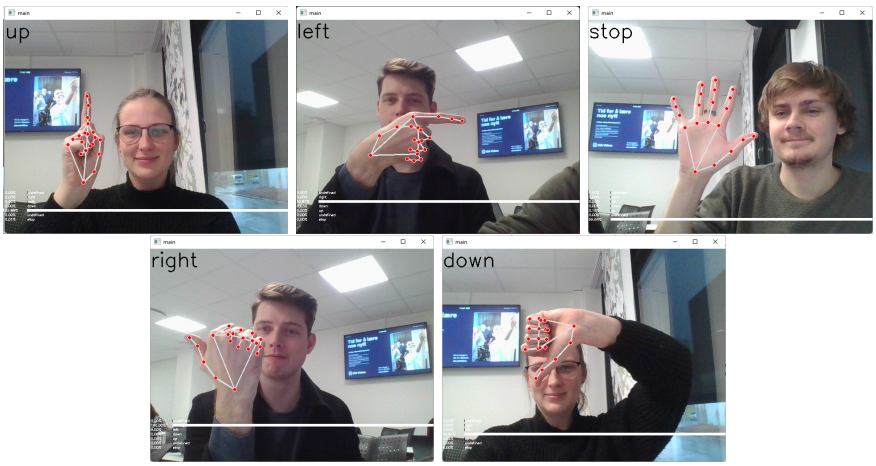
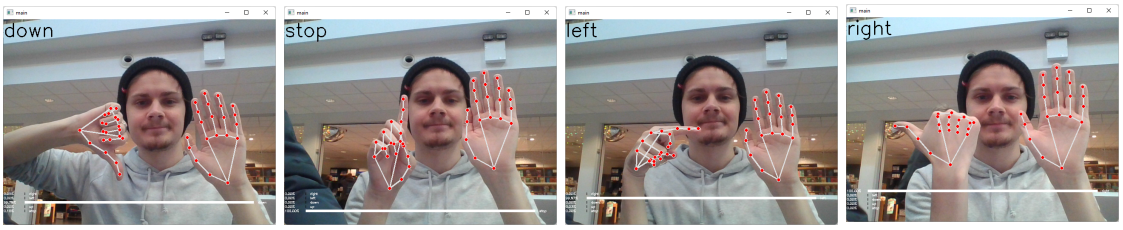


Figure 14: Gesture commands[8]



Gesture Combination - Backwards | Gesture Combination - Forward | Gesture Combination - Left | Gesture Combination - Right

Figure 15: Multi commands [8]

The stop command is used to start the drone. If it is not airborne, the drone will fly up when showing the stop command. If the drone is airborne the drone will land when the stop command is given. To rotate and move backward and forward, the stop command is shown together with one of the other four commands. For now this could be a problem if the stop command on one hand is shown alone for too long before adding the other hand. This will make the drone stop instead of executing the desired command. A solution could be to change some of the commands for further work, but for now we have at minimum to show both hands at the same time.

For a demonstration for the drone control using different gestures you can visit this link: Live demo.

5.3 Training

For our final dataset we got a model that converged to 96.05% accuracy for the training data and 99.38% accuracy on the validation data after 700 iterations out of 1000, fig. 17. This means that we can expect a accuracy of 99.38% for new data the model has not seen, as long as it is somewhat represented in the dataset. We tried out different batch sizes, but found out that a batch size of 64 gave us the best results.

For the loss function we got an epoch loss of 0.1205 on the training data, and a loss of 0.8986 for the validation data, fig. 16. The train-validation split is $80 - 20$, 80% of the dataset is set to be the training data and 20% of the dataset is used for the validation data.

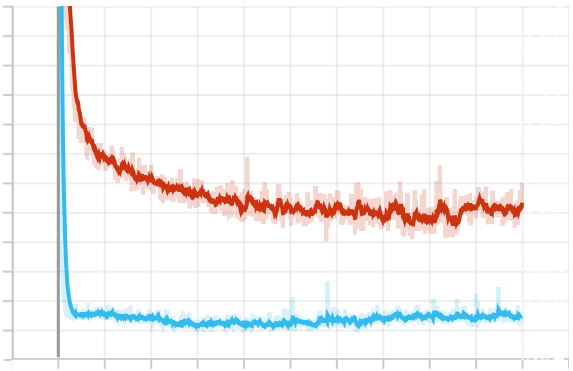


Figure 16: Loss

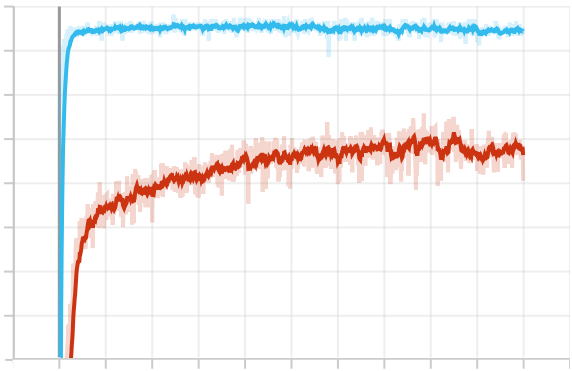


Figure 17: Accuracy

We have to keep in mind that we only have a dataset of 10000 data points, among three subjects. If we want to improve our model would should start off by gathering more, but just as important, good data! Since we want our model to work well for most people, we want to have more subjects in the dataset, this is because hands come in different shape and sizes, and we do not want the model to converge on a single subject.

We could also look into adding even more layers to our model so it could differentiate more features and converge to an even better accuracy, fig. 18.

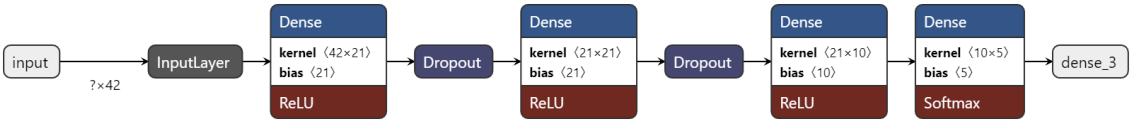


Figure 18: Model for detecting gestures made by the user[7].

6 Conclusion

Overall we managed to create a successful human-robot interface where we can accurately control the drone using a custom deep neural network model. We can use a video stream with predicted keypoints using mediapipe and send it to our model to accurately predict which gesture is being used. Then we can send the corresponding request to the drone which will perform the command.

The drone still has some struggle with the commands at certain distances, which would get better by using more training data. Our goal to connect the drone to an operator did not succeed this time. As explained we tried out different approaches, but with the time we got among other courses, this is something we have to consider as further work.

References

[1] Nooras Fatima Ansari. "10 most common Maths Operation with Pytorch Tensor - Nooras Fatima Ansari - Medium". In: *Medium* (Dec. 2021). URL: https://medium.com/@anoorasfatima/10-most-common-maths-operation-with-pytorchs-tensor-70a491d8cafd.

[2] Copter Express. *Manual flight · Clover*. [Online; accessed 17. Nov. 2022]. Nov. 2022. URL: https://clover.coex.tech/en/flight.html.

[3] Schultz AC Goodrich MA. "Human-robot interaction: a survey". In: *Foundations and Trends in Human-Computer Interaction* 1 (2007). URL: https://www.nowpublishers.com/article/Details/HCI-005.

[4] Google. *MediaPipe Hands*. URL: https://google.github.io/mediapipe/solutions/hands.html.

[5] *How To Build An Artificial Neural Network With Python - pyimagedata*. [Online; accessed 8. Nov. 2022]. May 2021. URL: https://www.pyimagedata.com/how-to-build-an-artificial-neural-network-with-python.

[6] Robert Kwiatkowski. "Gradient Descent Algorithm — a deep dive - Towards Data Science". In: *Medium* (July 2022). ISSN: 0481-1521. URL: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

[7] *Netron*. [Online; accessed 7. Nov. 2022]. Nov. 2022. URL: https://netron.app.

[8] OlavAusland. *LIDIC*. [Online; accessed 19. Nov. 2022]. Nov. 2022. URL: https://github.com/OlavAusland/LIDIC/tree/production.

[9] OpenCV. *About OpenCV*. URL: https://opencv.org/about/.

[10] Python.org. *What is Python? Executive Summary*. URL: https://www.python.org/doc/essays/blurb/.

[11] Bharath Ramsundar and Reza Bosagh Zadeh. *TensorFlow for Deep Learning*. Sebastopol, CA, USA: O'Reilly Media, Inc. ISBN: 978-1-49198045-3. URL: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html.

[12] Aishwarya V. Srinivasan. *Stochastic Gradient Descent — Clearly Explained !! - Towards Data Science*. Towards Data Science, Sept. 2019. URL: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31.

[13] Stackoverflow. *Programming, scripting, and markup languages survey*. 2022. URL: https://survey.stackoverflow.co/2022/#section-most-loved-dreaded-and-wanted-programming-scripting-and-markup-languages.

[14] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2022.

[15] *Tello EDU Single Combo*. [Online; accessed 19. Nov. 2022]. Nov. 2022. URL: https://djioslo.no/produkt/tello/tello-edu.

[16] *tf.keras.losses.SparseCategoricalCrossentropy | TensorFlow v2.10.0*. [Online; accessed 7. Nov. 2022]. Oct. 2022. URL: https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy.

[17] *What Is a Neural Network?* [Online; accessed 7. Nov. 2022]. Nov. 2022. URL: https://www.investopedia.com/terms/n/neuralnetwork.asp.

[18] *What is Machine Learning?* [Online; accessed 7. Nov. 2022]. July 2022. URL: https://www.ibm.com/cloud/learn/machine-learning.

Universitetet i Agder
Fakultet for teknologi og realfag
Gruppe 2 IKT 213

A Group contract

Gruppekontrakt

1. Gruppe 2

Gruppe **IKT213-project work 2** består av:
Olav Ausland Onstad, mobiltlf 46907461, mail: olavao@uia.no (Gruppeleder)
Joachim Thorsen Larsen, mobiltlf 40200395, mail: joacla14@uia.no
Jeanett Wesstøl, mobiltlf 41582130, mail: jeanettwe@uia.no

Oppdrag
Vi er 3 bachelor studenter IKT-Data som viderefører det gode samarbeidet vi har hatt nå i flere emner. Vi ser ingen grunn til å endre på laget og har valgt å jobbe sammen i gruppe i emnet IKT-213 oppgavene i emnet.

Bakgrunn
Bakgrunnen vår er at vi alle har erfaring fra softwareutvikling arbeid og vi tar alle bechelorstudiet. Vi mener at vår erfaring og bakgrunn vil komme til nytte når vi nå tar fatt på IKT-213.

Mål
Vårt mål er å i felleskap utfylle hverandre slik at vi er i god stand til å utføre gruppeoppgavene. Vårt mål er også å tilegne oss kunnskap i emnet samtidig som vi vil anvende den erfaring vi har innen softwareutvikling til å bli gode innen Machine Vision aplikasjoner.

Gjensidige forventninger
- Vi forventer at vi alle tre skal gjøre vårt beste og bidra med kunnskap og erfaring slik at vi er best i stand til å løse våre gruppeoppgaver.
- Vi forventer at oppgaver som tildeles utføres etter avtale. Hvis en står fast på noe må en kontakte medlemmene slik at ikke tid går tapt.
- Vi forventer at vi skal opptre saklig og korrekt og ha respekt for hverandres meninger og synspunkter.
- Vi forventer at en melder fra hvis en blir syk eller forhindret i arbeidet slik at andre i gruppen kan om mulig tar over for å rekke frister.
- Vi skal i felleskap komme fram til løsninger som alle kan stå bak.

Rollefordeling
Vi vil organisere oss i en rimelig flat struktur hvor vi i felleskap fordeler oppgaver og ansvar fra gang til gang. Gruppeleder vil ha ansvar for å laste opp leveranser og kommunisere med emneansvarlig.

Arbeidsform, Arbeidsrutiner:

1. Felles gruppemøte på UiA etter forelesninger.
2. Oppgaver fordeles etter fellesgruppemøte
3. Arkiv: vi har laget mappe på Notion
4. Øvrig kommunikasjon: epost, og mobil og Discord

Hva gjør vi hvis noen ikke oppfører seg skikkelig
Vi setter opp et møte med vedkommende om oppførsel. Hvis det ikke skjer forbedring kontakter vi Nadia for veiledning og eventulet delt karakter ved behov.

Dette punktet forventer vi å la ligge, men det må være med for å sikre oss hvis det skulle oppstå uenighet og uoverenstemmelse. Vi har i et tidligere punkt beskrevet hva som forventes av hverandre.

Vi vil i felleskap ha en åpen dialog om dette og regner med at vi skal løse dette selv.

Hvis vi mot formodning ikke klarer å løse problemene internt i gruppe vil vi kontakte foreleser for å søke råd.

Hvorvidt et medlem skal tilsnakkes, utvises fra gruppen , ekskluders er ikke opp til gruppen og gruppen ber om at det er foreleser sammen med instituttet som vedtar sanksjoner.

Signatures: