UPPSALA UNIVERSITET

Project report

Intelligent Interactive Systems

Ahmet Can Turgut, Aliaksandra Kupreyeva, Egemen Yiğit Kömürcü,
Viktor Romedahl

June 2, 2020

Abstract

In this project, a system that takes an image with a hand gesture as input and makes a virtual agent respond to this gesture has been developed. This system is split into three subsystems: Landmark Extraction, Gesture Recognition, and Responsive Behavior. The first subsystem automatically finds the location of all landmarks present in the input image. The second subsystem is a neural network that takes as input a set of landmark locations and predicts the gesture. Finally, the last subsystem takes as input the name of a gesture and shows the response of a virtual agent to this gesture. To train and evaluate the first subsystem, the provided a set of videos was used, while the second subsystem was trained and evaluated on the provided .csv file containing annotations. Each of the three subsystems is functioning well on its own. The Landmark Extraction system extracts the positions of landmarks with an accuracy of 80% and the second subsystem has shown to achieve an accuracy of 100% on previously unseen data. Additionally, we attempted to develop a system based on transfer learning that takes as input a raw image with a hand gesture and predicts the hand gesture present in this image. This system achieved 68.53% on the validation data and 55.82% on the test data.

Introduction

We are required to create a system which from videos of users showing different hand gestures makes a virtual agent react to these gestures. This can be split into three subsystems: Landmark Extraction, Gesture Recognition and Responsive Behaviour. The output of each subsystem will be the input for the next one. We are also required to use a specific provided dataset which contains a .csv file with landmarks and corresponding videos. Egemen and A.Can were mostly responsible for Landmark Extraction system, Aliaksandra for Gesture Recognition system and specialization part, and Viktor for Responsive Behaviour system.

1 Landmark Extraction

1.1 Methodology

For the landmark extraction, two different approaches are used in this paperwork. Firstly, the landmark extraction is tried to be done from scratch. Before moving on to the algorithm implementation, the actual dataset is explored such that how many landmarks are needed to be predicted for each gesture of each user, and the information is displayed whether they are consistent or not. Despite many variations, the mod value of the values for the number of landmarks is predicted for each user. Consequently, 20 landmarks are needed for open palm and open dorsal gestures, whilst 12, 9, 18, and 16 landmarks are required for fist palm, fist dorsal, three fingers palm and three fingers dorsal gestures respectively.To begin with, we loaded videos and split them into the images. Secondly, we convert the images into the images having grayscale and HSV channels. After that, we tweak with the HSV values in the images and found an HSV value which gave the hand. Then, we draw and get the largest contour made by the above HSV values. We also draw a rectangle covering the largest contour which must be the hand. Finally, if y coordinates of the image increase as it goes down at the rectangle, the highest y coordinate corresponds to the root of the hand, whilst the lowest corresponds to the middle finger. Furthermore, the one found at the leftmost is considered as a thumb. After this, the index and the ring fingers are calculated by taking the means or some fractions of their right and left fingers. Also, we tried to create our haar cascade model for detection of the hand, but we got very bad accuracy. Then we decided to use an already created haar cascade which is aGast, but also this cascade file does not give us an optimal solution.

The alternative approach uses a pre-trained CNN model that is trained with lots of images as the file size of the model is 100 MB. The CNN model detects the landmarks by first using the hand bounding box detection and then multiview bootstrapping which extracts probabilities of each pixel to have behaved as landmarks, as the researchers at Carnegie Mellon University (CMU) first find out [1]. The model has a reasonable accuracy detecting landmarks and it detects 22 points as default for all images. This predicts 2 additional landmarks at the root of the hand and there should be much fewer landmarks in case of for instance fist hand images. This means that it is required to modify the code according to the project's images. So instead of detecting 22 points, 2 points are discarded in order to not make the program crash since the dataset has at most 20 landmarks.

1.1.1 Design

We kept the design as simple as possible. The model takes one video and outputs a landmarks.csv file which can be fed to the next subsystem.

1.1.2 Materials

Initially we planned to use only OpenCV and aGast haar cascade for landmark extraction and hand detection, but due to the problems mentioned in Section 1.1 , we skipped using aGast haar cascade. For this subsystem, we used the CNN model over OpenCV which is modified to fit our requirements. The CNN model has .caffemodel type of file and there is an additional file called protoTxt for prototype information of different landmarks on the hand.

1.1.3 Procedure

A python file is created for all processes. Running the command `python landmark_extraction.py FILENAME` gives a .csv file as a result. After the process ends, the next subsystem can be run with this file for gesture detection.

1.2 Results

Using the CNN model gives high accuracy for detecting landmarks. As mentioned earlier, we removed the extra two landmarks. In the end this subsystem is ready to provide data for the Gesture Recognition system. For evaluation, the generated dataset which is the output of this algorithm is compared with the actual dataset. After detecting at most 20 points for each image, some landmarks have coordinates equal to 0, so they are not considered as errors. After comparing the datasets, the evaluation has a margin parameter that the prediction is not counted as misclassified one if the prediction is equal to the actual value within that margin. Consequently, the landmark extraction algorithm achieves 60% and 80% accuracy with 50 and 100 pixels of acceptable margin respectively.

1.3 Discussion

We choose to use the second approach due to the problem with the first approach which is that not all of the images are standardized as the brightness and contrasts of the images may differ. The pre-processing techniques such as image histogram matching or gamma standardization are required so it is decided to move on with the alternative approach which is modified according to the scope of the project.

2 Gesture Recognition

2.1 Methodology

2.1.1 Design

In this sub-system, we have built a neural network that takes as input a set of landmarks that are received from the first sub-system and outputs one of the 6 possible hand gestures. Because there are lots of possible options for the hyperparameters, we decided to experiment with a few selected hyperparameters while keeping the other ones fixed. We used `Adam` optimizer with its default settings and `cross_entropy_loss` as a loss function, which is usually used for multiclass classification problems. Furthermore, the network was trained for 10 epochs with a batch size of 64. When choosing the network architecture, we started with a very small network to see how it performs on the validation data. Then we gradually built it up to see how the performance is affected. It turned out that a simple network with only two fully connected layers together with earlier mentioned hyperparameters yielded very high validation and test accuracy. The data is split into training, validation, and test sets manually, such that the data from frames of the same person always occur in the same part of the splitting to prevent the leakage of information. The splitting is done in such a way that approximately 80% of the data goes for training, 10% for validation, and the remaining 10% for testing.

2.1.2 Materials

For the implementation of our network, we used Keras, which is a popular deep learning framework. Other libraries that were used include Pandas, NumPy, Scikit-Learn, and Matplotlib.

2.1.3 Procedure

Our model is trained on data contained in a .csv file, where each row contains 40 x and y positions of landmarks, as well as the name of the gesture. In order to be able to pass the data to the model, it was converted into two arrays, where one array contains landmark positions and the other one contains the corresponding gesture names. The array with gesture names is then encoded using one-hot encoding. The next step was to transform the array containing landmarks by scaling each landmark to a given range. After that, network architecture and other hyperparameters were chosen as described in Section 2.1.1. The chosen network consisted of two fully connected layers. The first layer is an input layer containing 80 nodes, which is followed by `ReLU` activation function. The network's second layer is an output layer containing 6 nodes, one for each of the classes. The output layer is followed by `softmax` activation function which transforms the outputs into probabilities. This network is trained on the preprocessed data and at the end of each epoch evaluated on the data from the validation set. When the training is finished, the network is evaluated on previously unseen data contained in the test set.

2.2 Results

Given a set of landmarks in a frame, our model predicts the gesture represented by these landmarks with 100% accuracy on both validation and test sets. In Figure 1 in Appendix the loss and the accuracy are shown, and we can see that our model converges already after the first epoch. The model was also retrained multiple times on differently mixed train and validation sets and nearly 100% accuracy was achieved there as well.

2.3 Discussion

Our model achieved 100% accuracy on previously unseen data with very simple network architecture, the reason for which could be that our data is easy to learn. In some cases when the train and

validation sets were mixed differently, the train or validation accuracy could drop down to 96%, which might be because some parts of our data were not annotated correctly.

3 Responsive Behaviour

3.1 Methodology

Before we could start programming, we needed to define meaningful interactions for each of the hand gestures. The responses only needed to use a hand gesture as input; not audio nor emotion and therefore, the response only considered what the gesture represented as a unaccompanied symbol.

- An open palm is universally accepted as friendliness and a greeting, and a simple smile and "Greetings" was chosen as response.
- A dorsal palm is in contrast the opposite, an expression of sadness and a "Goodbye" was chosen as response.
- Open fist is used as a sign of solidarity, therefore a nod and "Solidarity!" was used as response.
- A dorsal fist could be considered rude and thus, an expression of anger and "Rude!" will be shown as response to this gesture.
- Individual fingers in order being shown are often used as a way of expressing numbers, and thus the response to open three fingers gesture will count up from 1 to 3 and express thoughtfulness.
- Response to the dorsal three fingers gesture is similar to the previous one, but counts up in reverse and it closes its eyes.

3.1.1 Design

Using the Furhat SDK agent, a skill is inserted while it is running. The skill contains pre-defined instructions for the agent to use on any user input. The agent will automatically wait for input from the previous sub-system when the skill is run, and respond accordingly.

3.1.2 Materials

We've used Furhat SDK as an agent and Intellji IDEA for Kotlin to run skills for it, following both Furhat's own documentation and lab 3 setup and instructions.

3.1.3 Procedure

The main.kt file is the controlling module and first runs the Idle state defined in general.kt. On default the agent would interact with a virtual user when it appeared within a distance, but since we will use a different input, the Idle state instead immediately switches to a state called Reaction defined in reaction.kt. Here it will open a TCP server and simply wait for input from the previous subsystem. If a string containing the name of the gesture is sent, a corresponding response is used on the agent. As a fail-safe, the input "break" can be sent to exit the server loop, and it will use the agent's default loop with questions instead, asking for the used gesture.

3.2 Results

The agent gives a response from the input over TCP. It runs the server until exited, however, there are some issues with Furhat flow queue getting full and this could cause delays in the response.

3.3 Discussion

For the agent to work a lot of applications are being ran at the same time which is not optimal for performance, and it being its own subsystem means that each subsystem could contain their own applications which could slow it down even more.

4 Specialization

4.1 Methodology

4.1.1 Design

In this part we attempted to implement an end-to-end system which combines the landmark extraction and the gesture recognition systems into one single system which predicts the shown gesture from a raw image. To achieve this, two approaches were tested. The initial plan was build and train a CNN from scratch. Many combinations of network architectures together with other hyperparameters and data augmentations have been tested, but the validation accuracy never got higher than 30 %. Therefore we have decided to proceed with another approach which was based on transfer learning. In this approach we take an network which was originally pretrained on ImageNet. From this network, the last layer is replaced by a new classifier that is trained from scratch. We "freeze" all layers in the network and only update the weights of the classifier. In this way, the network maintains the ability to extract features which it has learned from the dataset it was previously trained on, while it is also able to classify images from our dataset. Since the provided dataset is very small, we needed to do data augmentation to prevent overfitting. The data was split into training, validation and test sets in the same manner as in Section 2.1.1.

4.1.2 Materials

The same libraries as in Section 2.1.2 were used to implement this system.

4.1.3 Procedure

To begin with, the videos were split into images. All images were resized to 224×224 pixels. Before the images were fed to the network they were altered by using transformation techniques such as rotations, shifts, horizontal flips, zoom and brightness altering. We used `ResNet-50` network, where we removed the last layer by global average pooling layer, followed by fully connected layer with 128 output nodes, which was in turn followed by a dropout layer and output layer with 6 output nodes. During training, we used `SGD` optimizer with learning rate of 0.001 and momentum of 0.9. The network was trained for 10 epochs with batch size of 128. However, the network with the best validation accuracy is saved and used later to evaluate the network on previously unseen data.

4.2 Results

In Figure 2 in Appendix the loss and the accuracy of the final network are shown. As we would expect, we can see that the loss keeps decreasing and the accuracy keeps increasing on the training data throughout the whole training phase. The loss on the validation set decreases in the beginning, but after the certain amount of time it starts to increase which is because the network is overfitting the training data. Since we use early stopping, the network parameters from the epoch 7, which gives the best performance on the validation set, are used evaluate the network on the test data. In Figure 3 the confusion matrix for this network on the test data is shown. As we can see the network predicts open dorsal with 100% accuracy, fist palm with 98.06% accuracy and fist dorsal with 81.25% accuracy. However, the network thinks that open palm is open dorsal and that three fingers dorsal is three

fingers palm. For comparison reasons, the confusion matrix of the same model for the validation set is shown in Figure 1. In contrast to the confusion matrix for test set, here the network is also predicting three fingers palm with 100% and confusing three fingers dorsal with open dorsal.

4.3 Discussion

Our network achieved an accuracy of 68.53% on the validation data and 55.82% on previously unseen data from the test set. Poor performance of the network on the test data can be due to the fact that the test set was rather small. In fact, it only contained samples from one person. Thus, the test set was not really depicting how the real world data looks like and thus it might not give a correct evaluation of our network. To obtain a fair evaluation of the performance we would need more diverse samples in the test set, but this could not be afforded in this project because the dataset was already rather small and we needed as much data as possible for the training.

5 General discussion

5.1 Discussion of the overall pipeline

When we look at the overall pipeline our sub-domains seem to work well, but they are not communicating with each other automatically. We can say that our pipeline works manually, which implies that the data should be transferred by hand between sub-domains. One of the big problems in the pipeline is that every sub-domain should rely on the data correctness from the previous sub-domain. We don't have any mechanism to check whether the incoming data is correct or not.

5.2 Challenges faced

For the landmark extraction part the challenge was to find a way to obtain the correct side of the hand. It's nearly impossible to detect the hand side from the video which is taken by a notebook camera. It was also planned to detect the hand side by creating CNN model, but we thought its bad accuracy could affect the next subsystem's correctness negatively. We also faced some issues with sending inputs to the responsive behavior agent since the previous subsystem used Python and the agent used Kotlin. Another challenge was to obtain a good accuracy for the end-to-end system as we were not given enough data to work with. We also faced some difficulties in the group work as everything was done remotely without meeting in person.

5.3 Ethical issues

Most of the ethical issues are related to the detection of the correct gesture, since our gesture detection system is based on the model which trusts the correctness of the data set which is provided by the students. If this data is not correct our responsive behavior system may act unreasonable against found gestures.

6 Conclusion

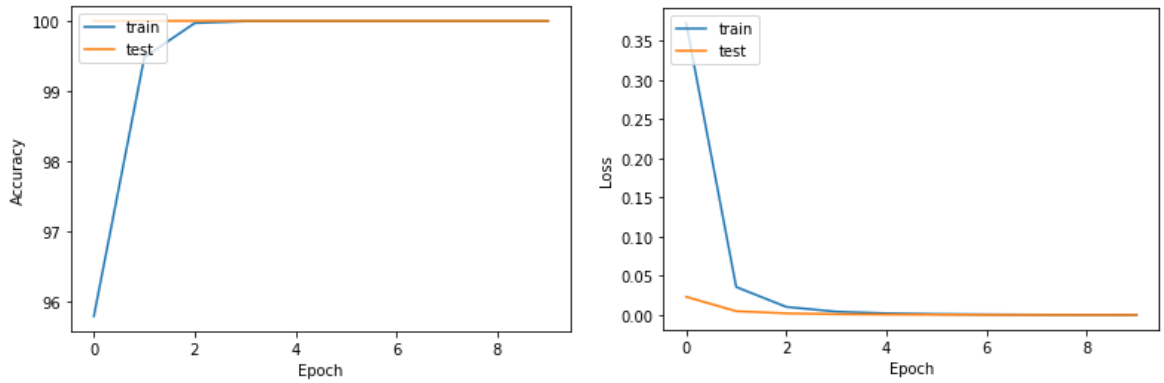
In conclusion, we successfully achieved the gesture recognition and responsive behavior parts individually, with the possibility of sending input over a TCP server.

References

[1] Simon, T., Joo, H., Matthews, I., Sheikh, Y. *Hand keypoint detection in single images using multiview bootstrapping.*. (2017). In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1145-1153).

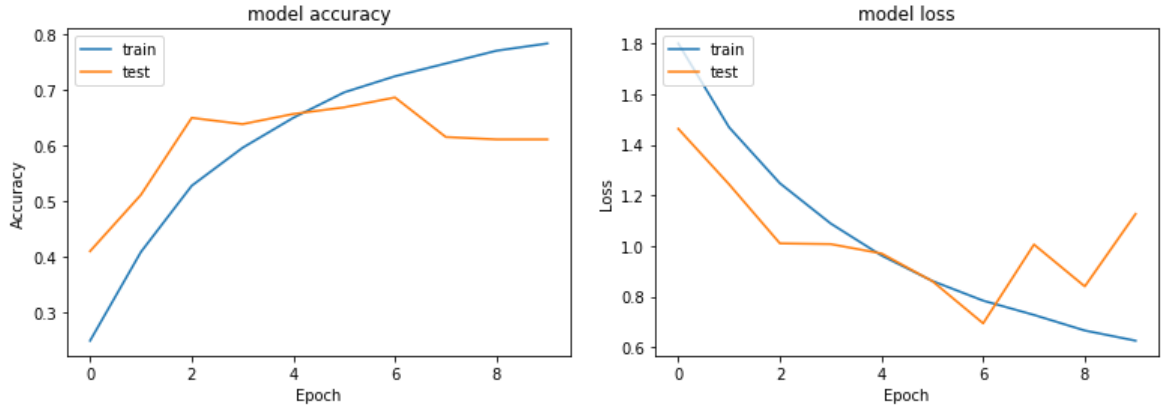
Appendix

A Figures



(a) The accuracy (b) The loss

Figure 1: Loss and accuracy of the network (Gesture Recognition) during the 10 epochs of training on training and validation sets



(a) The accuracy (b) The loss

Figure 2: Loss and accuracy of the network (specialization) during the 10 epochs of training on training and validation sets

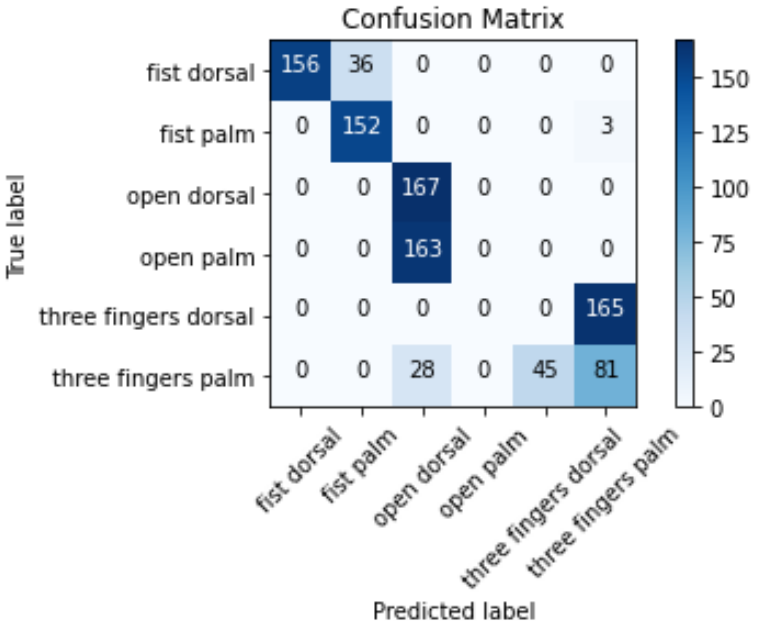


Figure 3: The confusion matrix for the network (specialization) on the test set

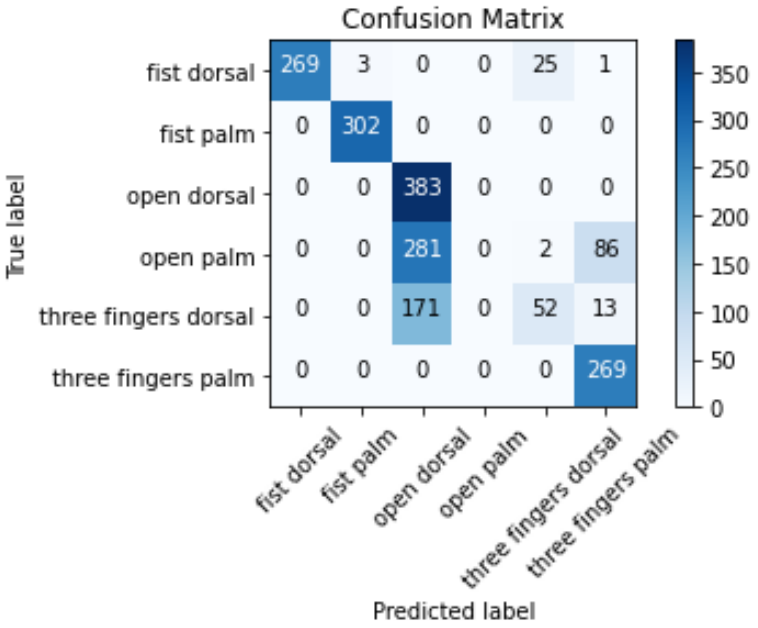


Figure 4: The confusion matrix for the network (specialization) on the validation set

B Project specification

B.1 Project initiation documentation

(a) **What is the project?**

The purpose of this project is to develop a system that takes a video of a user showing different hand gestures as input and shows a virtual agent reacting to these gestures.

(b) **What are the specific objectives?**

- Develop a Landmark Extraction system, which automatically finds the location of all landmarks in a given frame.
- Develop a Gesture Recognition system which takes a set of landmark locations as input and gives the detected hand gesture as output.
- Develop a Responsive Behaviour system, which takes the detected gesture as input and shows a virtual agent reacting to this gesture.
- Develop an End-to-end learning to the Gesture Recognition system that is achieved from the raw input of images rather than extracting the landmarks.

(c) **What are the requirements?**

We are required to create a system which from videos of users showing different hand gestures make a virtual agent react to these gestures. This can be split into three subsystems: Landmark Extraction, Gesture Recognition and Responsive Behaviour. The output of each subsystem will be the input for the next one. We are also required to use a specific provided dataset which contain a .csv file with landmarks and corresponding videos.

(d) **What (if any) are major issues you can foresee?**

We could run into to some issues due to everything being done remotely. This could lead to issues with communication, mismatching hardware for the program and version control. The scope of this project is a bit large and we have really less time like three weeks than it must be. Moreover, we may not detect the knuckles when extracting the landmarks of the fist because the hand is closed. Also, we may get low accuracy not because of the poor performance of the model, but because of the noise in the data. It may happen since some colleagues did not pay much attention into the annotating their videos in Assignment 1.

(e) **What are the deliverables?**

We will do a presentation showing the executing system on May 25th. Also a final report which documents our work on the project should be submitted no later than May 27th. We can also include a demo video of gesture recognition of one person by putting the gesture text to the left corner of the video.

B.2 Work breakdown structure

1. The tasks that need to be completed for developing Landmark Extraction system are:
   (a) Detecting all knuckles that a hand contains from the images.
   (b) Extract all landmarks in a given frame.
   (c) Map the landmarks as a data set.
   (d) Prepare data set to sending "Gesture Recognition" system.
2. The tasks that need to be completed for developing Gesture Recognition system are:
   (a) Load the data into TensorFlow.
   (b) Train a machine learning model (which is simple Neural Networks).
   (c) Evaluate machine learning model by looking at train/test accuracy and creating the model's confusion matrix.
   (d) Test the effect of data on the model.

3. The tasks that need to be completed for developing Responsive Behaviour system are:
   (a) Define idle states.
   (b) Make animations for states.
   (c) Write script with responses dependent on animations.
   (d) Test states with given gestures.
4. The tasks that need to be completed for End-to-end learning of Gesture Recognition are:
   (a) Load frames into the environment.
   (b) Pre-process images such that they become black-white.
   (c) Forming a circle using some radius threshold by leveraging the root landmark of the hand.
   (d) Ending up just with the objects representing fingers.
   (e) Through the object detection techniques, the number of fingers can be counted so that it will help us to detect the gesture.
   (f) Train and test a CNN model by shuffling and splitting the data
   (g) Tune our model with different hyper-parameters such as train/test split, the number of hidden layers, the number of hidden nodes, batch size, etc.
   (h) Evaluate machine learning model by looking at train/test accuracy and creating the model's confusion matrix.
   (i) Repeat from step (f) to (h) and record the accuracy of the specific model in this step.
   (j) Choose a model which gave the best performance.
   (k) Compare the performance of this model with the performance of the model built in the foundational step.

B.3 Timeline

Week	**Task**	**Assigned person**
18	Gesture Recognition (a)	Can
18	Gesture Recognition (b)	Egemen
18	Gesture Recognition (c)	Aliaksandra
18	Gesture Recognition (d)	Viktor
19	Landmark Extraction (a)	Aliaksandra
19	Landmark Extraction (b)	Egemen
19	Landmark Extraction (c)	Viktor
19	Landmark Extraction (d)	Can
20	Responsive Behaviour (a)	Aliaksandra
20	Responsive Behaviour (b)	Egemen
20	Responsive Behaviour (c)	Viktor
20	Responsive Behaviour (d)	Can
21	Gesture Recognition Extension (a) - (b)	Aliaksandra
21	Gesture Recognition Extension (c) - (d)	Viktor
21	Gesture Recognition Extension (e) - (i)	Egemen
21	Gesture Recognition Extension (j) - (k)	Can