— Report P&O CW 2019–2020 Task 3 —

Department of Computer Science – KU Leuven October 21, 2019

Team 12	
Toon Sauvillers	60h
Seppe Van Steenbergen	54h
Bert Van den Bosch	52h
Frédéric Blondeel	20h

1 Introduction

(Few sentences.) *Clearly state the problem you tackled. The task's assignment defined the functionality you have to offer, not the problem(s) you solved. You have to phrase the exact problem statement, i.e., state the assumptions you made. On top of the problem statement we expect you to state the relevance of the problem and what solution is chosen. Later in §3 you can go into detail as why this is your preferred solution. Throughout the text we expect you to back up claims with (scientific) references.* ***This section is best written after the remainder of the report is finished.***

We are developing a multiscreen casting framework that supports a heterogeneous set of displays. The very first hurdle is the design of a communication protocol that connects all these devices. Devices are connected over the web via a client-server architecture. Clients connect to the server by loading a webpage in their browser. To establish a two-way communication channel, a connection is maintained through sockets [**?**].

Based on these principles we developed a basic multiscreen casting framework that offers the following functionality:

- The ability to transmit complex data such as images.
- Client-side access to the camera.
- Client identification such that clients can receive specific commands.
- Support of two roles, i.e., master and slave.

2 Design schematics and screenshots

(≈ 1 page. Depending on the need for a design overview.) *Design schematics, deployment diagrams, class diagrams, sequence diagrams, . . . Whatever you think we need to understand your design. You could add a little bit of text here but keep that under half a page. You reference the illustrations here from §3 when needed.*

All communication between clients runs through a nodejs server. We use the Socketio [**?**] library to set up a bidirectional communication channel between client and server. A deployment diagram can be found in Figure 1. Clients can take on a specific master or slave role by opening a specific webpage within the browser, i.e., the slaves will surf to `webaddress/slave`, while a client that wants to be the master loads the webpage found at `webaddress/master`.

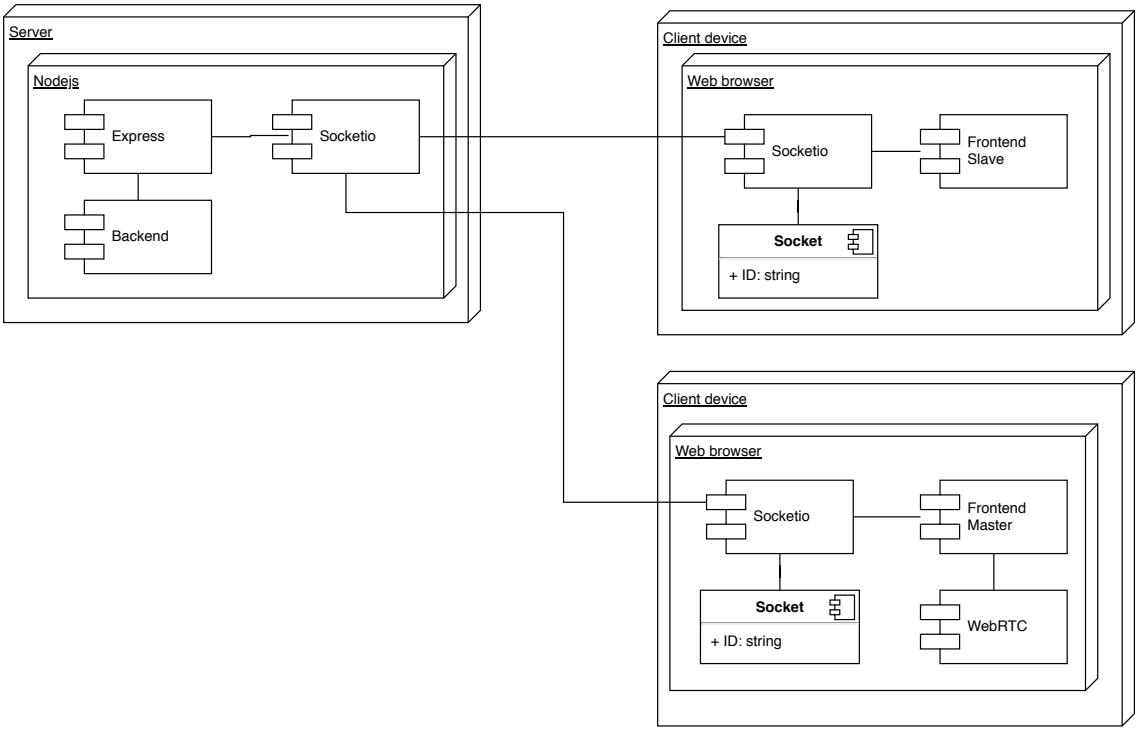

Figure 1: Deployment diagram of the multiscreen casting framework.

We rely on webrtc [?] to collect a client-side video stream. This module offers the necessary tools for serialization, i.e., as a base64 encoded string. Serialization is required for transmission [?]. Server-side the backend code will process the image. In our demo we broadcast the image by sending it to all slaves. Slaves and masters can be reached by the server, as we use the concept of a namespace [?]. A namespace can be used to group clients with a similar role, and broadcast to them.

A socket is assigned a unique identifier by Socketio. This identifier can be used to emit messages from the server to a specific client. A list of connected clients is available through the API of Socketio [?].

3 Algorithms

(≈ 1 page. Depending on necessity) *When you design or use an algorithm you should be able to explain how well it performs and if it satisfies the requirements. Often there will be alternatives or system parameters that need to be chosen. We want to know why you chose a specific set of parameters, or why you chose an approach over another. We supply some example text of last year.*

Correctly identifying players is an important requirement of the game. For this end we adapt a Face Recognition system that builds on the `facerecognizer.js` framework [?]. We show that by carefully selecting hyper parameters we can assure that misidentification will happen in less than 1% of identifications.

3.1 Description of the face recognition system

We illustrate the face recognition pipeline in Figure 2. This pipeline is fully offered by the `facerecognizer.js` framework [?]. However, since we have only one enrollment sample,

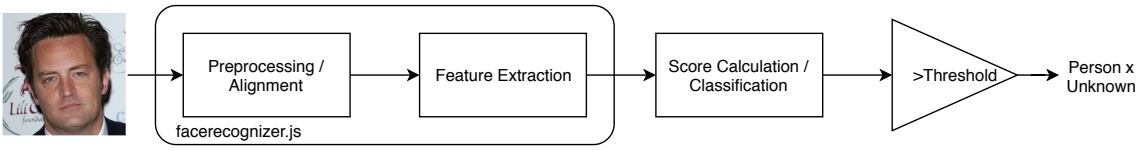

Figure 2: The face recognition pipeline. Face image from lfw data set [?].

we will only use the core technology of the framework, i.e., DeepFace [?], the feature extractor method. Deepface represents the face image as a 128 dimensional vector with values in the interval $[-1, 1]$. In a second step a similarity score is calculated between the enrollment samples of all known identities and the test sample. For this end, we evaluate the impact of two similarity metrics on the performance: `profnuyenssimilarity` [?] and `profjacobsdistance` [?] in §3.3. In a last step the highest score is compared to a threshold. If the highest score is bigger then the threshold we return the identity connected to this score, else we conclude that the test subject is unknown. The two mentioned metrics can both be formulated by

$$S(a,b) = \sqrt{\int_0^1 \int_0^1 \|\mathcal{P}_\nu(a,t_1,t_2) - \mathcal{P}_\nu(b,t_1,t_2)\|_{\mathcal{F}}^2 \, \mathrm{d}t_1 \mathrm{d}t_2},$$

where $\mathcal{P}$ is a polynomial interpolation operator.

3.2 Data set

Representative data is needed to test the performance of a machine learning system. We evaluated our system in a first phase with a standard data set, i.e., labeled face in the wild [?]. This data set contains 10340 identities with approximately 5 images per identity.

3.3 Experiments

We discuss the influence the performance of our identification system in function of two similarity functions: `profnuyenssimilarity` [?] and `profjacobsdistance` [?]. The performance is measured in `adalbertoscore` [?]. In order to achieve the requirement of a misidentification probability of 1%, `adalbertoscore` should be less then -4.

We calculate the feature vector of all the faces in lfw. Then, we split the data set in a enrollment and test set. We make sure that only for enrollment only one sample is available for each identity. We repeat this split procedure 10 times and report the average score. For every split we evaluate the performance in function of the threshold. Figure 3 shows `adalbertoscore` for both similarity functions w.r.t. the threshold. We chose the `profnuyenssimilarity` as this clearly satisfies the requirement of a `adalbertoscore` of -10 at a threshold of 0.5. We note however that the `profnuyenssimilarity` needs a resolution of at least 5MP which is currently not a problem, but might be a concern in the later development such that we keep the `profjacobsdistance` implementation as an alternative.

In a second experiment we vary the number of classes. We fix the threshold to 0.5. Before make the enrollment and test splits we randomly select a specific number of identities. Afterwards we apply the same procedure as in the first experiment. The results are shown in Figure 4. From this figure we conclude that, in general the identification performance is better with a lower number of players. On top of that, when the number of players is less then 200, both similarity functions perform similar.

In a third experiment we investigated the battery consumption...

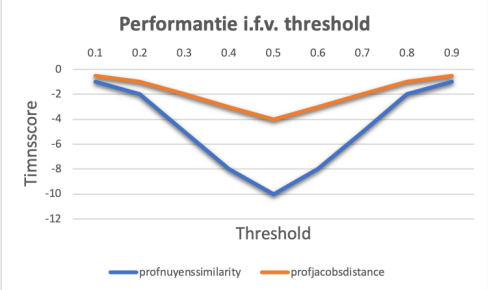

Figure 3: The `adalbertoscore` for both the `profnuyenssimilarity` and `profjacobsdistance` w.r.t. the threshold. The number of classes is 10350.

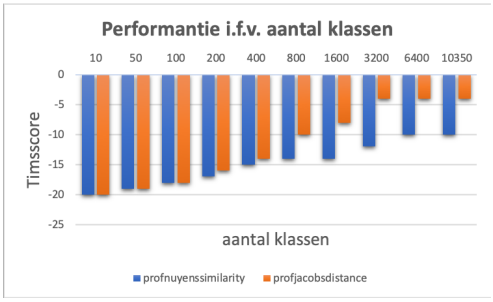

Figure 4: The performance w.r.t. the number of classes for both the `profnuyenssimilarity` and `profjacobsdistance`. The threshold is 0.5.

4 Conclusion and prospects

(≈ 1/4 page.) *This section contains your evaluation of the development. How well did you succeed to complete this task? What can be improved?*

We designed a basic framework for screen casting. We offer all the functionality demanded in task 2. We achieved our goal by relying on basic socket.io functionality, as well as built-in features of front facing html5 technology such as webrtc. We did not discuss in detail the sending of commands to change background colors etc. We offer all functionality for this problem, the core challenge here was socket identification.

In the current state of our project multiple masters can coexist. We do not see this as a problem at the moment. From a security perspective this can be problematic depending on the actual usage scenario of the final application.

A User licenses

For every software package and all images/illustrations/pictures used we want to see that you have looked up its user license and comply with it. Please provide a list here. This may also be a good location to write about what software license you are thinking for your finished project.