— 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 Introductie

Het herkennen van schermen, deze identificeren, lokaliseren uit een foto. Dit verslag behandeld deze uitdagingen. Het zoeken van schermen begint bij een foto. Deze foto bevat een scherm met een bepaald uitzicht, er is gekozen voor een groen-blauwe rand waarop gefilterd kan worden. Vervolgens zoekt het algoritme in de gefilterde foto naar aparte schermen en geeft aan hen een id. Met behulp van het kleurenverschil en bepaalde hoeken wordt de oriëntatie bepaald.

Het identificeren van het scherm gebeurd op dit moment nog apart met een kleurenbarcode. Deze barcode bevat 5 verschillende kleuren waardoor er 120 verschillende schermen geïdentificeerd kunnen worden. In de volgende weken worden deze twee, lokaliseren en identificeren, bij elkaar gezet.
Dit verslag behandeld de keuzes die gemaakt zijn alsook een uitleg bij de gebruikte algoritmen en hun tijdscomplexiteit. De mogelijke beperkingen worden met deze kennis geduid.

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`.

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

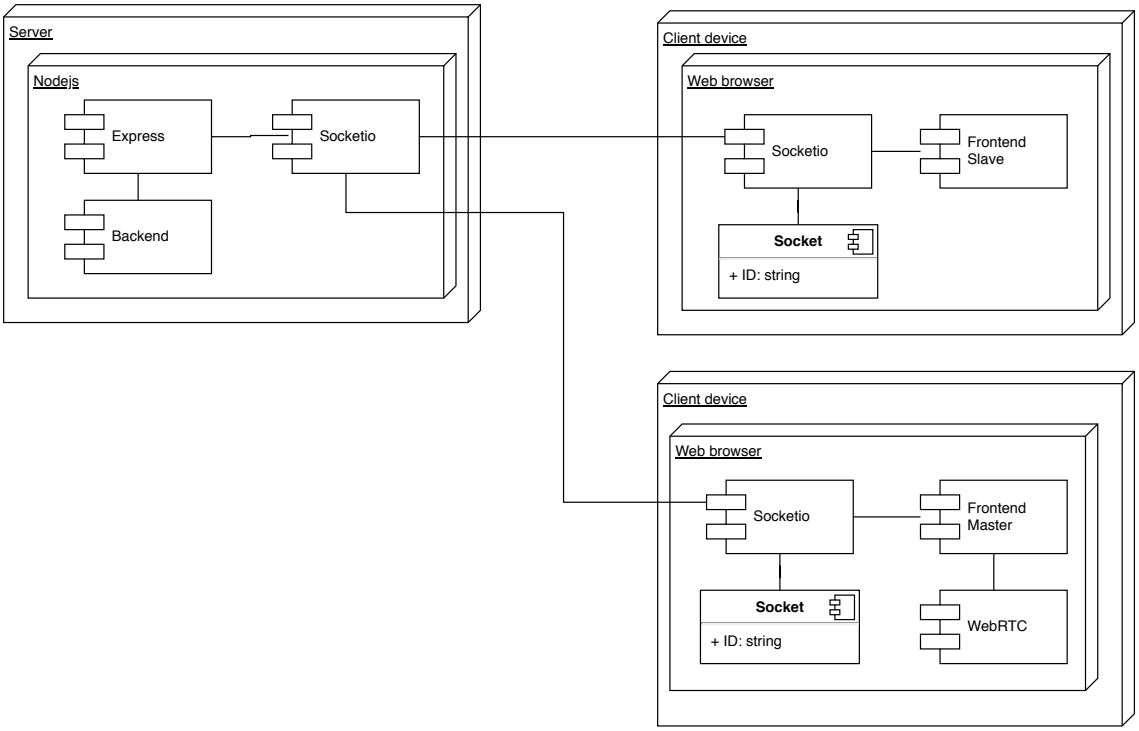

Figure 1: Deployment diagram of the multiscreen casting framework.

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.*

3.1 Scherm identificatie met barcodes

3.1.1 Algemene uitleg

De identificatie van schermen gebeurt aan de hand van een kleuren barcode. Het slave-scherm zal binnen zijn rand(gebruikt voor edge detection) een herhalend patroon van 5 kleuren en wit tonen. Aan de hand van de volgorde kan een unieke code per scherm gelezen worden en zo is elk scherm duidelijk gedefinieerd. De 5 kleuren zijn speciaal gekozen om zo ver mogelijk uit elkaar te liggen in het HSL spectrum om fouten lezingen te vermijden. Ze dragen elk een cijfer van 1 tot en met 5 en worden maar één keer gebruikt in de barcode. Dit zorgt ervoor dat we in theorie een totaal van 5! (= 120) verschillende schermen op één moment kunnen detecteren. Het herhalend patroon geeft ons de mogelijkheid om fouten te vermijden. Na het vinden van het scherm zelf (zie sectie 3.2) zal de barcode gelezen worden. Indien een reflectie of een overlap plaatsvind kan een stuk van het scherm bedekt zijn. De scanner() zal het midden van het scherm lezen in rechte lijn (volgens de orientatie), aangezien een stuk bedekt is zal deze blijven gaan tot hij 5 kleuren en wit tegenkomt. Zo

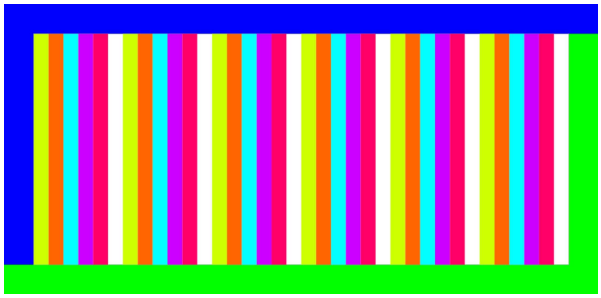

Figure 2: Voorbeeld van een barcode met 6 herhalingen.

niet, zal de lijst met codes leeggemaakt worden en verder gaan. Het herhalend patroon is dus essentieel aan het correct inlezen.

3.1.2 Data set

Om het scanner() algorithme te testen hebben we een aantal foto's genomen vanop verschillende afstanden, met verschillende computers (kleuren kunnen afwijken van scherm tot scherm) en met verschillende aantal barcodes dat op het scherm past. Aangezien een herhalend patroon gebruikt wordt kan men kiezen hoeveel er per scherm staan, dit kan van 3 tot 15 gaan (meer heeft niet veel zin want het onderschijden van kleuren wordt lastig).

3.1.3 Experiments

3.2 Image

3.2.1 Algemene uitleg

3.2.2 Data set

3.2.3 Experiments

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.