Computer Vision Challenge

Yosra Bahri / 03666016
Hoang Dang/ 03676899
Julia Kanzler /03669983
Madeleine Kaufmann / 03670625
Yamen Mohisn/ 03680775

July 2020

1 Introduction

The scope of this project deals with differentiating the background and foreground of images and videos and eventually substituting one of them. The generated image or video could show one of three options: the same foreground but different background, same background but different foreground, same foreground and background but with different colors. In the following, a deeper explanation is given about this project, its most challenging parts, and its results.

2 Datasets

To determine if the project is successful, our program needs to identify different shapes in the foreground and differentiate them from the background. Which is why we were provided with a dataset called "Chokepoint". The "Chokepoint" dataset represents a collection of images and videos obtained by three surveillance cameras placed above several portals to collect image sequences of pedestrians walking through the portals. The obtained images are quite varied in terms of shapes, illumination conditions, and sharpness as they show different portals meaning different backgrounds, different persons walking through, different angles and two sequences even show a crowded scenario. The aforementioned makes this dataset appropriate for several research areas, most importantly, person identification field.

3 Image Reader

The first step of the project consists in reading the images or videos that will be alternated. In this particular case, the ImageReader class should be able to play two sequences out of the three registered by the cameras mentioned above in an endless loop. To do so, two functions are created: ImageReader and Next. Table 1 lists the properties of the class ImageReader as well as a brief description.

Propreties	**Description**
L	Left camera
R	right camera
srcL	Path to directory with left pictures
srcR	Path to directory with left pictures
start	Number of starting frame for calling next() function; increased in every call of next()
N	number of following frames after start frame to return in next() function
L_jpgFiles	All pictures of the selected left camera
R_jpgFiles	All pictures of the selected right camera
Left	Last left tensor of next() function
Right	Last right tensor of next() function
Number_of_jpgs	Number of frames in L_jpgFiles and R_jpgFiles
Loop	Takes 0 if there are at least N following frames, 1 if there are less than N following frames left

Table 1: Properties of class ImageReader

3.1 ImageReader

This function creates an ImageReader object.
ir = ImageReader(src, varargin)

	Parameters	**Description**
Inputs	src	path to the directory of the selected scene
	varargin	L:left camera, valid values 1,2 ; error message in case of invalid value
		R:right camera, valid values 2, 3; error message in case of invalid value
		Start: frame to begin with; in case of invalid value start = 0
		N : number of frames to load (current frame + N following frames); in case of invalid value N = 1
Output	ir	ImageReader object

Table 2: Parameters of ImageReader() function

3.2 Next

This function returns tensors of size N+1 of left and right images and the loop value.

[left, right, loop] = next(ir)

	Parameters	**Description**
Input	ir	Output of ImageReader
Outputs	left	tensor of N+1 pictures of the left camera, starting at the current start frame
	right	tensor of N+1 pictures of the right camera, starting at the current start frame
	loop	0, if there are at least N following frames; 1, if there are less than N following frames left

Table 3: Parameters of next() function

4 Segmentation

The purpose of this function is to develop a binary mask that would detect the foreground of every image. To get the best possible results, three different algorithms were used to determine whether a pixel belongs to the background or the foreground. The classification is then done by majority vote. In the following the three Algorithms used for the segmentation process are presented:

4.1 Segmentation with mean approximation

The first algorithm uses the mean of the frames to approximate the background. The binary mask is then created through background subtraction, meaning that every pixel in the current frame is subtracted from the corresponding pixel in the "mean frame" and the result is compared to a threshold. If the value is close to zero, the pixel is classified as belonging to the background, if the value is greater then the predefined threshold(in our case 8) the pixel in question is classified as foreground. Finally, a Gauss filter is used to eliminate the noise, hence smooth the binary mask.

4.2 Segmentation with Gaussian average

The second algorithm uses a Gauss function to approximate the intensity of each pixel. The Gauss function is updated with every new pixel, and subsequently, the values that vary from the average computed by the Gauss function are classified as foreground, if not it's classified as background. The Gauss filter is also used here to smooth the binary mask.

4.3 Segmentation with morphological gradient

The morphological gradient is defined as the difference between the dilation and erosion of an image. The idea behind this algorithm is to assign a value to each pixel that indicates the contrast intensity of the neighboring pixels and thus reconstruct the gradient image using several MATLAB functions such as "imb2bw" to convert images into binary images and "bwareafilt" for the extraction of the binary mask.

4.4 Filtering with majority vote

This final function uses the three above mentioned functions for the segmentation, and through a majority vote, classifies each pixel accordingly to the foreground or background.

5 Render

This part displays a few of the possible uses of the mask developed in the segmentation. Four possible modes are defined inside of this function, where the user can either display the foreground of the image without background using the mode "foreground" or display the background without the foreground using "background". The third possible mode is called "overlay", and it consists of showing differentiating the background and the foreground using two distinct transparent colors. This mode was implemented using the MATLAB function "imfuse" that creates a composite image of two other images. Last but not least, the "substitute" mode that simply swaps the background of the given frame with another background from another image. For the last mode an additional input needs to be provided which is the replacement-background image.

6 Configuration

After the rendering modes, this step consists of defining the necessary variables and configurations for the program. In our case, we define our group information, the virtual background image, and the object ir of ImageReader.

7 Challenge

This step implements all the previous functions in a loop and applies them on all frames until the selected scene folder is exhausted. The resulting images are all stored in an AVI video file and the time needed to perform this action is calculated by a timer.

8 Graphic User Interface

The last step of the project is concerned with developing a user-friendly graphic interface (or the so-called GUI). The GUI is created based on the steps that the users need to follow to get the desired output. The GUI is started with the Start GUI command. The import button is used to get the path of the

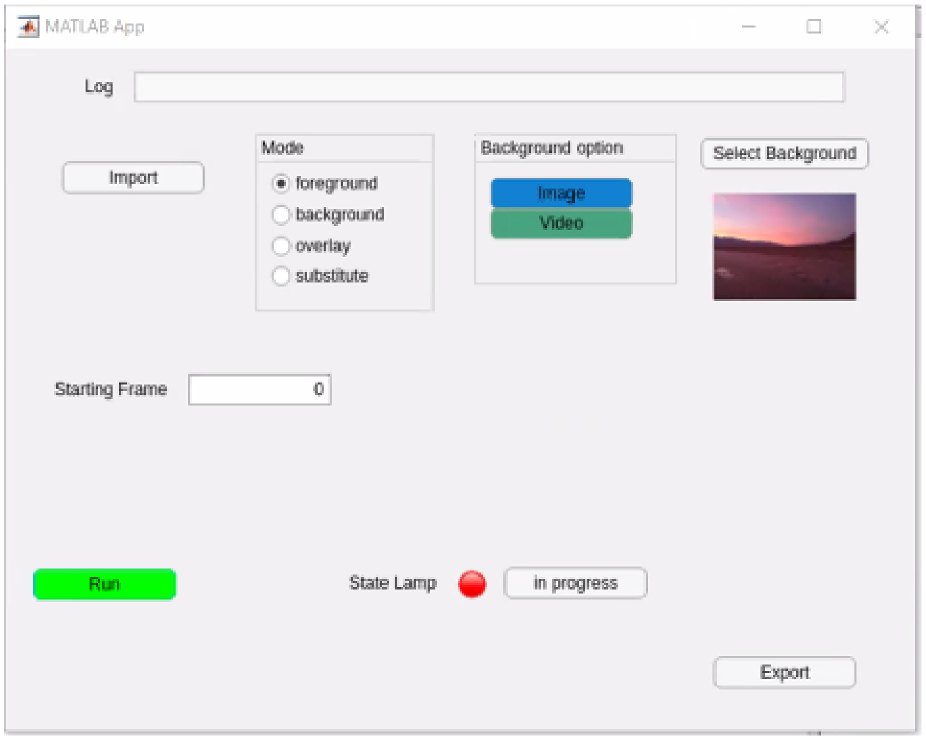

Figure 1: Graphic User Interface

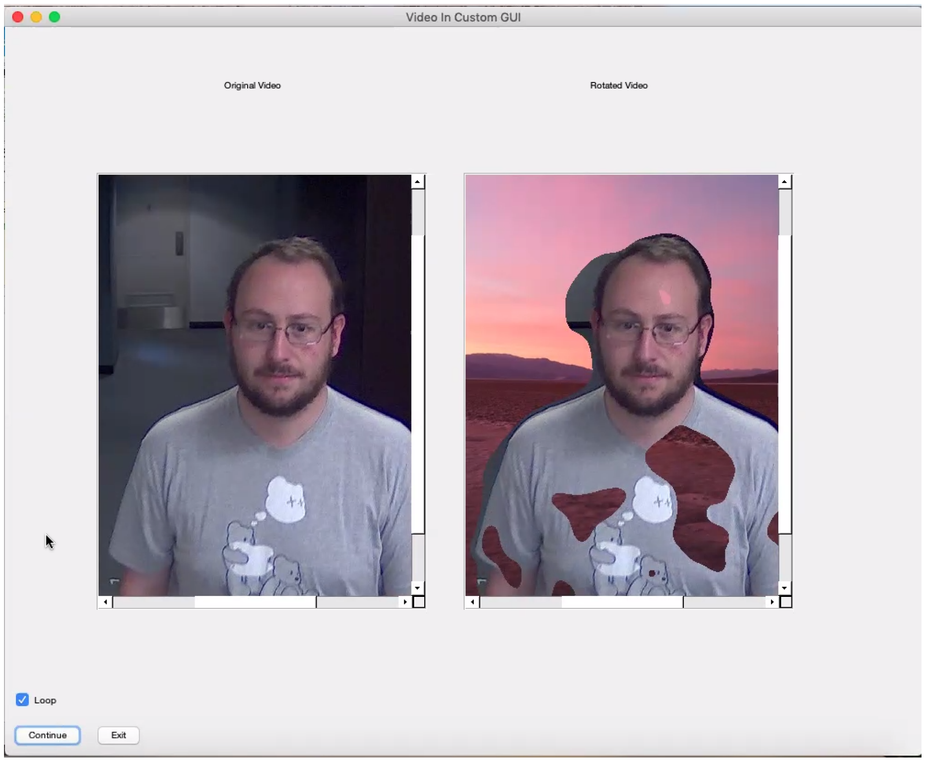

Figure 2: Graphic User Interface 2

scene folder, where the video files are stored. The user is then able to choose the segmentation mode. Afterwards, the user can select an image or a video as the background for the rendered video. After executing the Run button, the segmentation of the video file that was imported is then carried out and then saved in the output folder specified under ”stereoinputstream.avi” and ”outputstream.avi” using the Export button.

Once the output video is ready, a new window appears [Mat] where two videos are displayed simultaneously: the original video is displayed on the left and the rendered video is displayed on the right as seen in figure 2 below. If the box Loop is checked, the program displays the video in an endless loop. The buttons play and stop as the names indicate are used to play the video or stop it. The button exit allows exiting the window. In table**??**, a list of variables and their descriptions are available.

The Mode Radio Button enables the user to choose the rendering mode of the segmentation.

Variables	Type	Description
Start_gui	Command	Starts the program after writing in command window
Log	Edit Field	Gives information, e.g. if wrong start value was chosen
Import Button	Push Button	Importing a video for foreground segmentation
Mode	Radio Button	Select the Mode for rendering
Background type	Toggle button	Select either image or video for the replacement background
Select background	Push Button	Image or video that will replace the background of the original frame
Starting Frame	Edit Field	Select the first frame the segmentation has to start with
Run	Push Button	Runs the segmentation and rendering process
State Lamp	State Lamp	Green, if rendering is done; red during and before rendering
State Button	Push Button	”In progress” during rendering; ”Not ready” if input is missing (background or video frames); ”Ready”, if ready for rendering
Export	Push Button	Exporting the rendered video to a path of your choice
Image	Image	Display the image selected for the background, if video (background type) selected display the first frame of the video.
Loop	Check Box	Display the resulting video in an endless loop
Play	Push Button	Play video
Stop	Push Button	Pause video

Table 4: Variables in GUI and their description

9 Results

In the following, six rendered images of the ”Chokepoint” dataset are displayed.

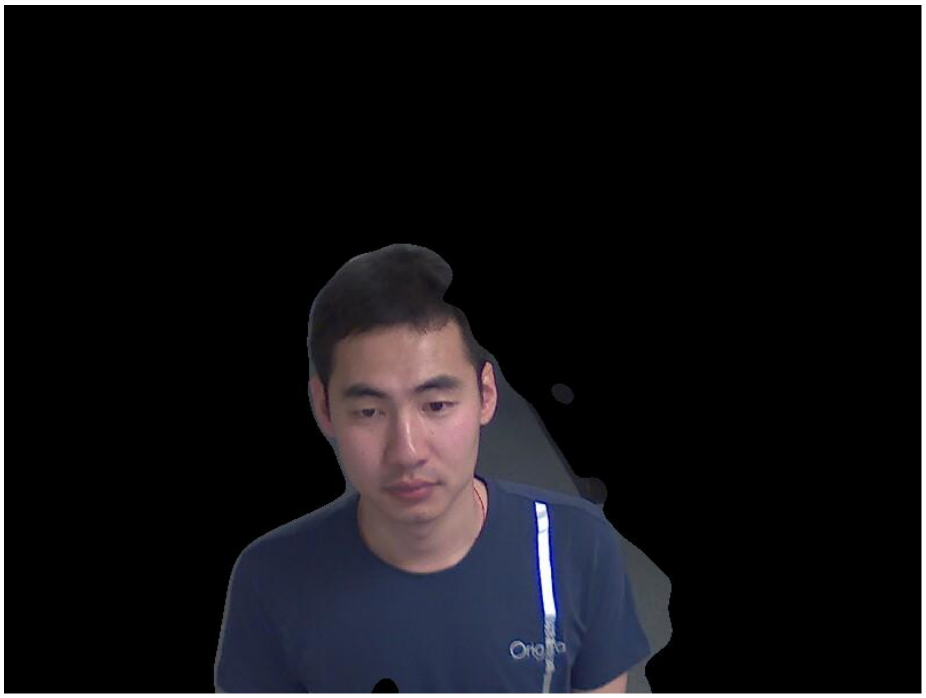

Figure 3: P1E S1

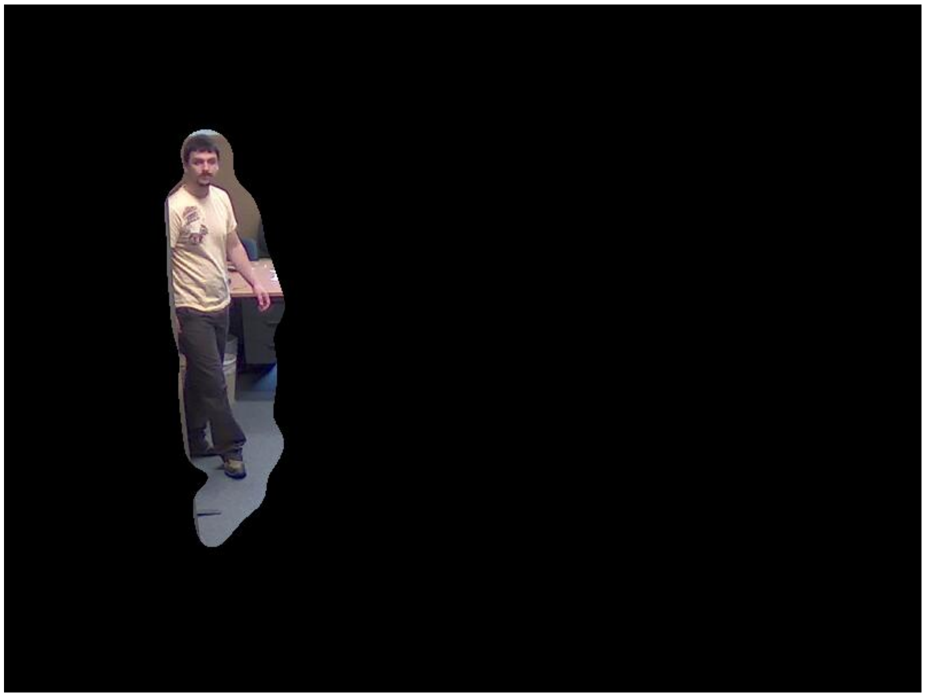

Figure 4: P1L S3

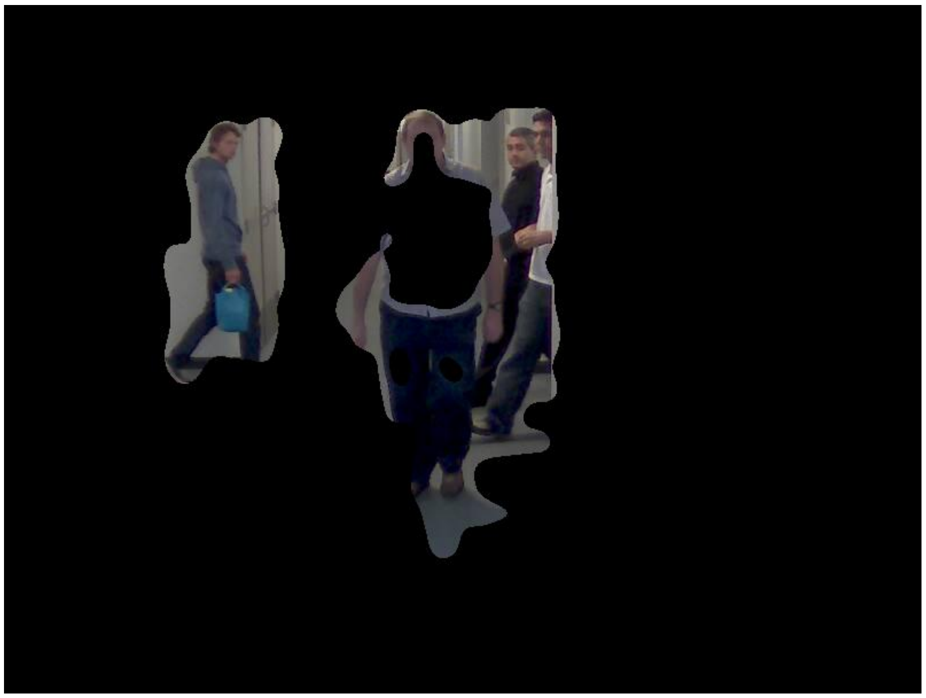

Figure 5: P2L S4

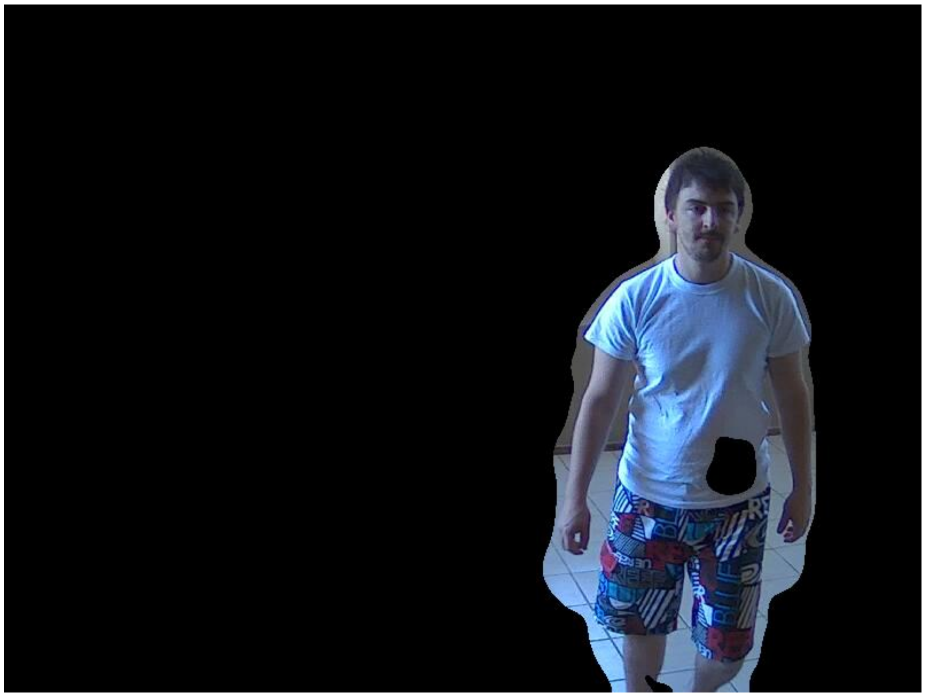

Figure 6: P2E S2

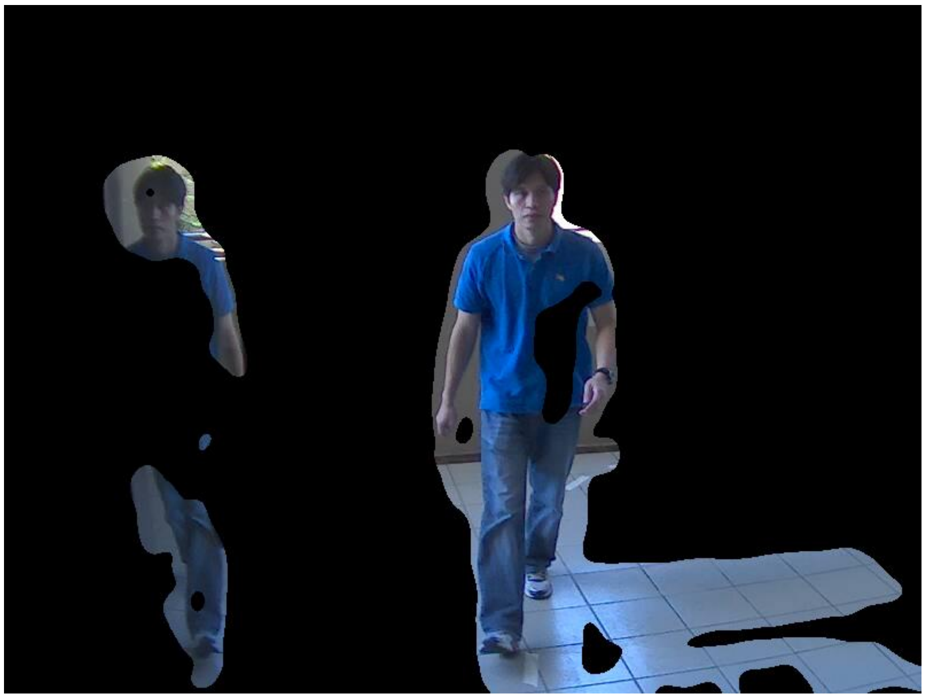
Figure 7: P2E S4

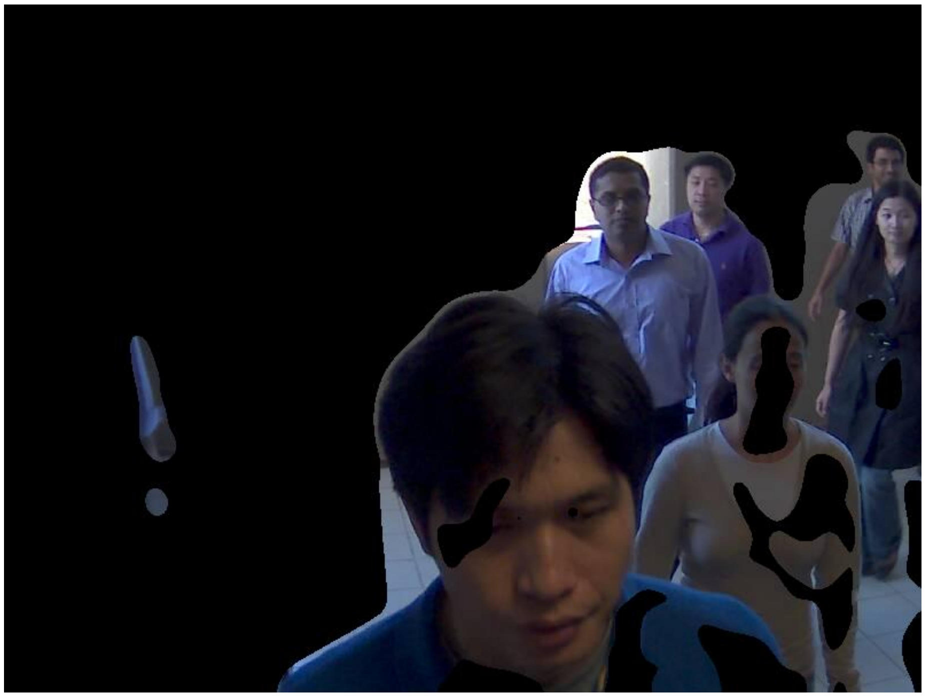
Figure 8: P2E S5

10 Discussion

All in all, the program fulfills the given requirements accordingly. However, the segmentation function can be improved to achieve better recognition of the foreground shapes.
Another option would be to run the segmentation and rendering in real-time, even though this could mean that the processing power of the used device would be affected. Consequently, the resulting run time would be influenced either positively or negatively, since it is dependent on the processing power. Another part of the program to improve is the video player in the GUI so that the original video and the rendered video would be played more smoothly.

References

[Mat] Mathworks. Video Display in a Custom User Interface - MATLAB amp; Simulink - MathWorks Deutschland. ”Accessed: 2020-07-09”. URL: http://cgi.cse.unsw.edu.au/ kevine/thesisguide.html.