Documentation of Computer Vision Challenge SS2020

Group: 35

Author: Jiangnan Huang

Zhiwei Liang

Nan Chen

Ivan Hartono

2020.7.12

Contents

1. Introduction

“Zur Zeit finden immer mehr Video-Calls und Konferenzen in den eigenen vier Wänden statt. Dabei möchte man vielleicht nicht unbedingt das unaufgeräumte Arbeitszimmer, die heimische Kuche oder die Familienfotos an der Wand hinter dem Sofa an Vorgesetzte und Kollegen streamen. Die Computer Vision Challenge beschäftigt sich daher in diesem Jahr mit der Unterscheidung von Vorder- und Hintergrund, sowiemit der Möglichkeit, ungewollte Szenenbestandteile zu ersten.” [4]

The aim of this project is to either recognize foreground and replace background using self-defined virtual background (or black image) or implementing an overlay mode to present background and foreground in two easily distinguishable contrast colors. After processing all images in a predetermined scene folder, a video will be then generated as an output, in which for example the background is replaced by a virtual background to protect privacy. As part of this project, customer friendliness is needed to be considered through the design of a graphic user interface (GUI), so that it is much easier and comfortable to change the necessary parameters in the program to get different expected results.

The program will be split into 6 parts. The first 5 parts realize basic functions and the last one, i.e. GUI, provides a better human-machine-interaction. The following table and block diagram explain functions of each part and how they will be organized.

ImageReader.m	It’s a class to read images from original files folders. There is a next() to read multiple images depending on input further and further.
segmentation.m	It’s a function to generate mask from given images for render.
render.m	It’s a function to apply mask from segmentation to separate foreground and background.
config.m	It’s a configuration file to load necessary parameters in workspace, if there’s no input via GUI.
challenge.m	It’s the main function to process all images and generate a video. It calls other functions and reads variable from workspace.
GUI	It’s the graphic user interface to read or write parameters.

Table 1.1 Function of each part of task

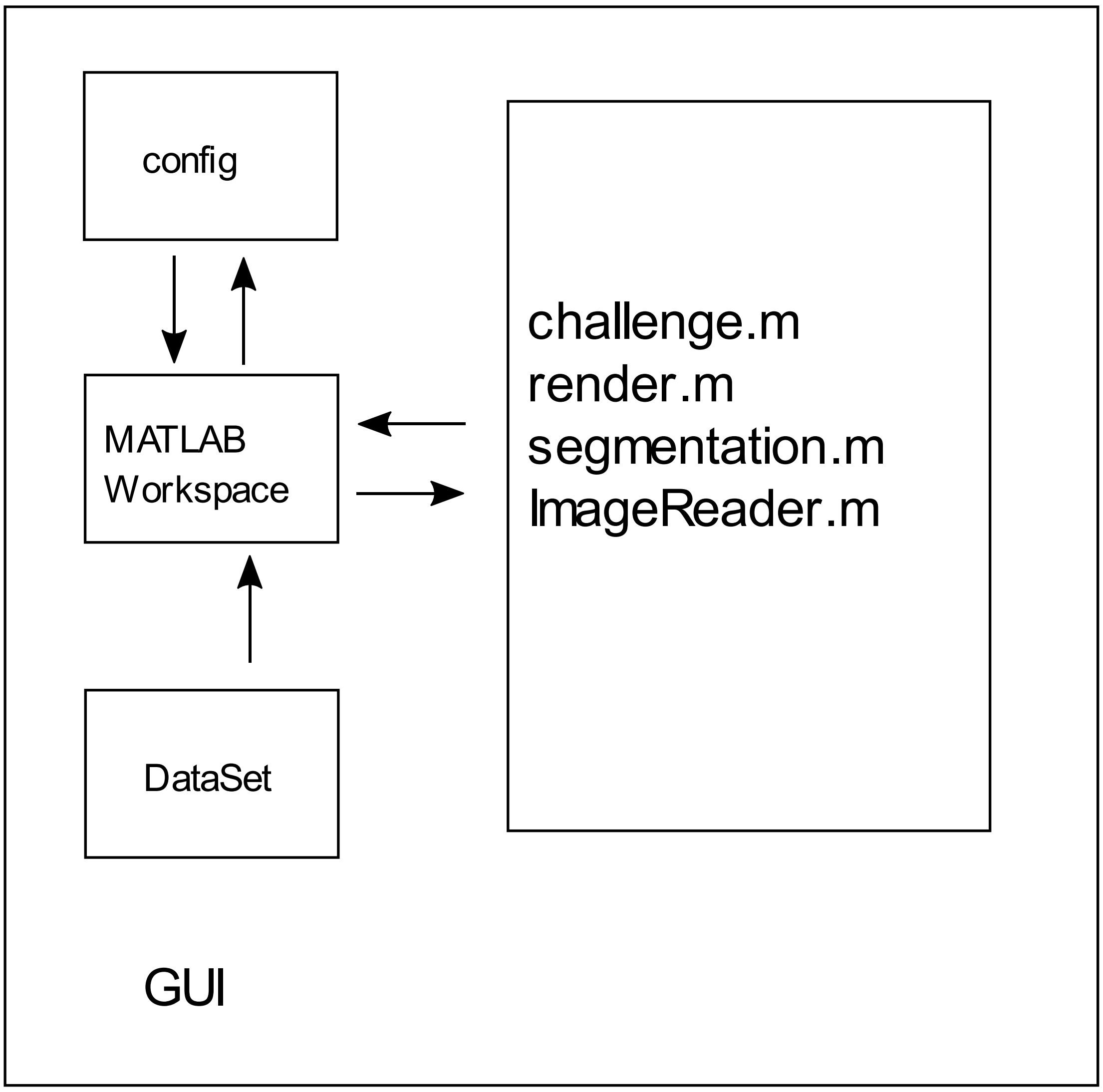


Figure 1.1 Generalized block diagram representing functionality idea.

Using input from DataSet and config on the MATLAB workspace, challenge.m, render.m, segmentation.m, and ImageReader.m will be run as seen in Figure 1.1. To avoid misconception, GUI will not be connected with config.m and challenge.m, but in principle algorithms used inside challenge.m and config can be “refurbished” to fill the needs of GUI, hence functionality-wise, config and challenge are part of GUI.

2. ImageReader.m

In the ImageReader class two functions are defined.

The first function with the same name will be used, when the object “ir” is initialized. An input parser is used for different input variables. Here we must at least pass 3 variables, i.e. “src”, the full path of original image files, “L”, index of image folder for camera left and “R”, index of image folder for camera right. We can also give another 2 parameters. The parameter “start” indicates the index of first image to be processed. The parameter “N” indicates number of following images.

The second function next() reads multiple images according to “N” and packs them together in form of AxBx3*(N+1) as output. Then It will read next unread images further till end. For example, if N=1, it will read the current and next image from folder for camera left and right separately, and pack them into output variables “left” and “right”. In the end, if the left images are not enough to satisfy “N”, all of rest images will be packed together regardless of “N”, and by next call the function will read images from beginning. It is important to note that in this project N is to be strictly defined as N=4 as an increase or decrease in N may affect the overall quality and accuracy of the detection process.

For an efficient implementation dir() will be used to get all information of images in the folder. Then indexing will be accomplished further and further until the end of the process. In the end an if condition is implemented to judge whether the process should go back and reset the “index”. For example, allimage=dir(), in which the variable “allimage” is a 1xN dimentional struct variable and it saves path and name of every image in the chosen folder. Then an index variable can be used, for example, “index”, to go through all images from start point. The start point is “start” from before. Every time the next() is called, the “index” increases about “N”. Because all variables here are temporal and will be deleted after calling, the “index” will be introduced as “persistent”. This is a data type in Matlab as “static” in C++. [6]

3. segmentation.m

The segmentation part is mainly based on the “3 frames difference method”, which uses 3 consecutive frames of a scene to detect the contour of the moving object (foreground), to be precise, the contour of that in 2nd frame. In this case, one camera is enough to finish the task of detection. [7]

The biggest advantage of this method is that the contour of the moving object can be easily found, in case we don’t have the image of the stationary background. That means the contour can be detected in a changing scene. However, this method has several disadvantages. First, the detected contour consisting of discrete pixels is hard to be converted into a closed region exactly which could be easier to generate the mask later. Second, the foreground contour won’t be detected precisely if the moving part has a large area with the same or similar color. In view of these points, it might be very hard to compute a perfect mask for every image with this method. What we should do is to approach this as perfect as possible.

Every time this segmentation function is called, 5 consecutive frames are needed, meaning $N \equiv 4$. And we pick the 1st, 3rd and 5th frames for the “3 frames difference method”. The reason why we use 5 frames instead of 3 is that the moving speed of the people in this task is not fast enough to draw the contour exactly. So using 5 frames here is a small trick.

As mentioned above, we must approach the perfect mask of the middle frame. In this part, we try to use morphology knowledge to process the raw contour detected by “3 frames difference method” and then get the final mask. The following functions of morphology are called in this segmentation function: imdilate(), imerode(), imopen(), imfill(), bwareaopen() and medfilt2(). The main idea here is as follows: First, we try to remove the noise of the background and get a complete contour of the foreground as perfect as possible. Then we fill the black regions inside the contour to get a binary mask. Finally, we process the contour of the mask with enrode() to approach the perfect mask and use median filter to get a smooth contour.

The whole process can be roughly shown in the block diagram below:

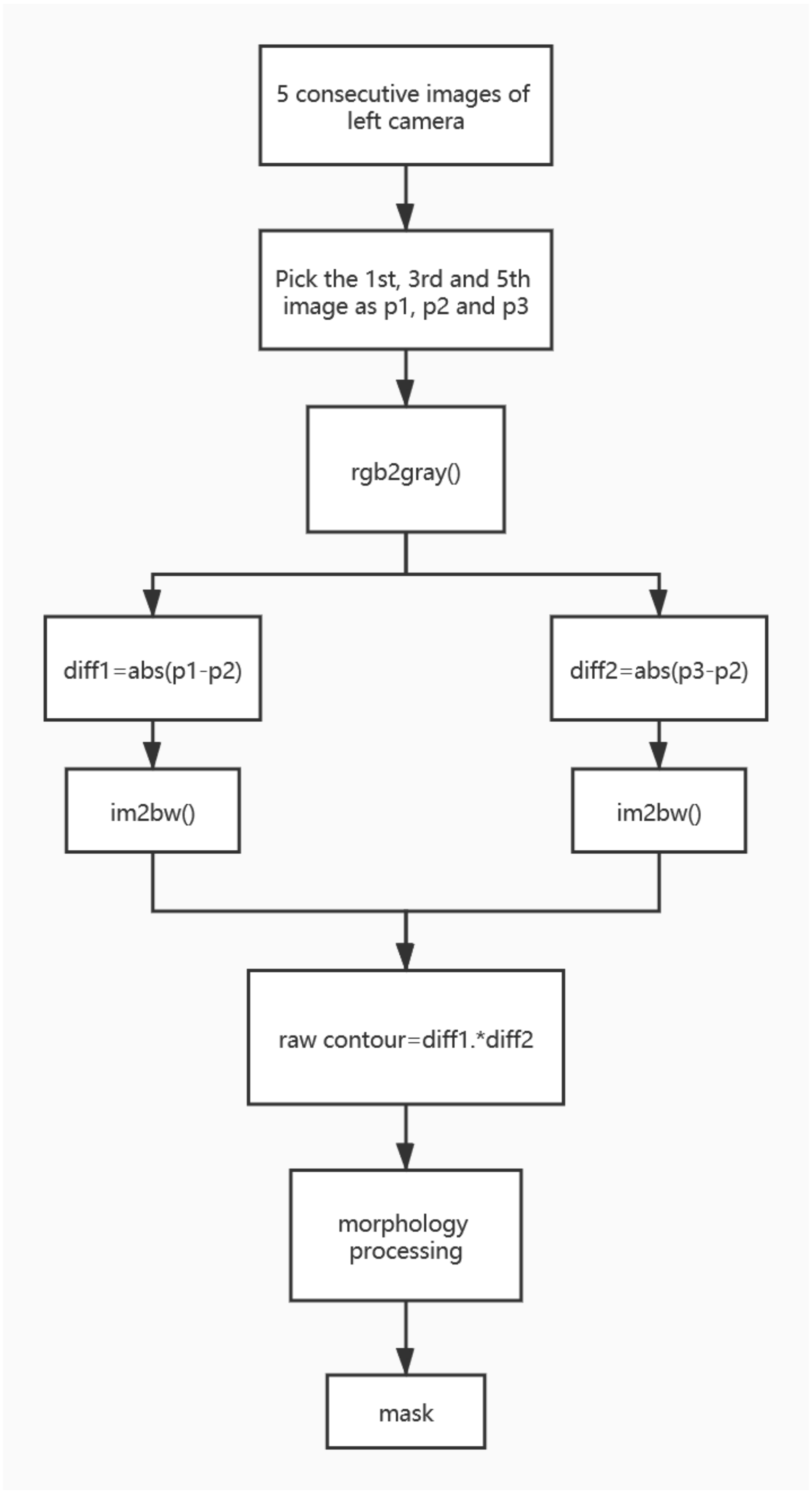


Figure 3.1 Block diagram representing “3 frame difference method”

4. render.m

The render.m applies mask from segmentation.m on image to separate foreground and background. With switch...case... the image will be processed depending on given mode and then given as an output. Figure 4.1 shows the generalized structure.

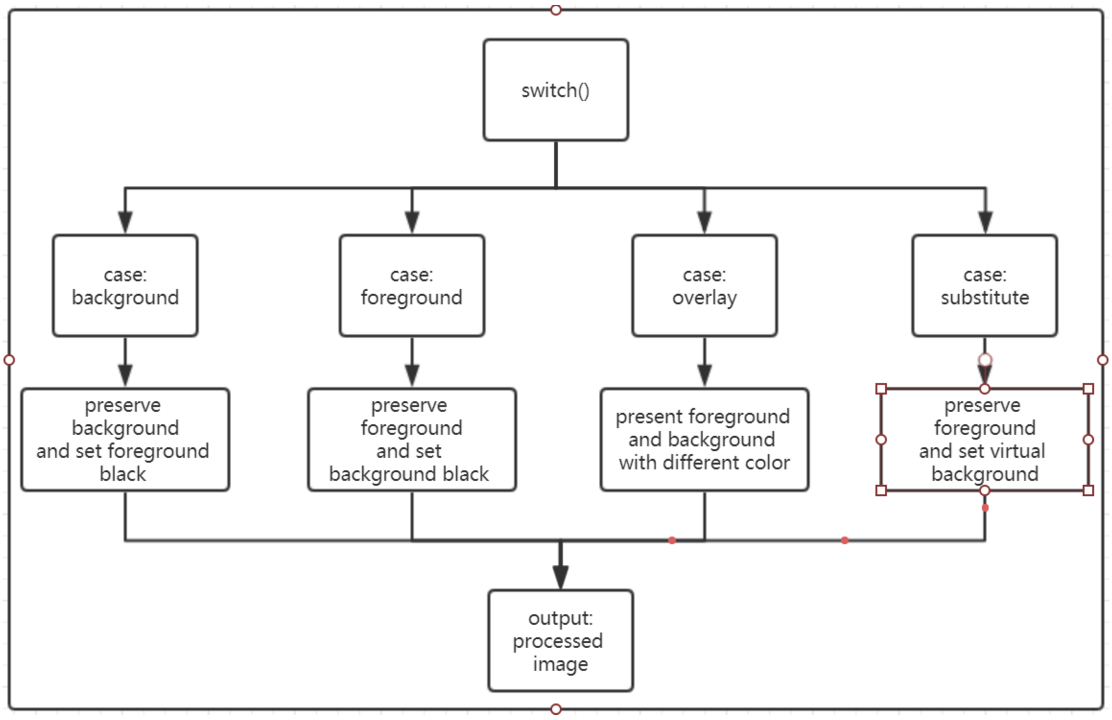

switch()
case: background
case: foreground
case: overlay
case: substitute
preserve background and set foreground black
preserve foreground and set background black
present foreground and background with different color
preserve foreground and set virtual background
output: processed image

Figure 4.1 Generalized block diagram for different rendering mode

5. config.m

The config.m initializes parameters with default values as substitute for GUI during challenge.m processes, as both config.m and challenge.m must not be connected to the start_gui.m. These parameters will be used for image processing later. The following table lists all parameters introduced by config.m. In this part an OS recognition is also added because the path can be different for different system, e.g. Windows and MacOS. The MATLAB predefined function ismac(), ispc() and isunix() are used. If the OS is something else, the program will report an error message.

parameter	meaning
bg	background image for mode substitute
bgpath	full path of this background image
dest	full path of output video
group_number	Group number
i	Index for “movie”, initialized as 1
ir	a object of class ImageReader
L	Index of camera left
loop	Flag loop for challenge.m
mail	E-Mails of group members
members	Names of group members
render_mode	Mode(background, foreground, overlay and substitute)
movie	Cell to store processed images
movie_flag	flag for movie, initialized as 0
N	Number of following images
R	Index of camera right
src	Full path of image folder
start	Index of first image to be processed
store	Flag for saving images as video

Table 5.1 config.m parameter list

6. challenge.m

The challenge.m is the main function, which calls others to process the images, to record running time and to export the video. The following block diagram shows how it works. The “DataSet” contains images to be used. In a loop the next() function will be called again and again for reading images from “DataSet”. After that the read images will be used to generate mask and the mask will be applied on the image again. All of this will be executed in loop until the all the images are processed. These rendered images will first be saved in cell “movie” and transformed into an AVI-Video in the end.

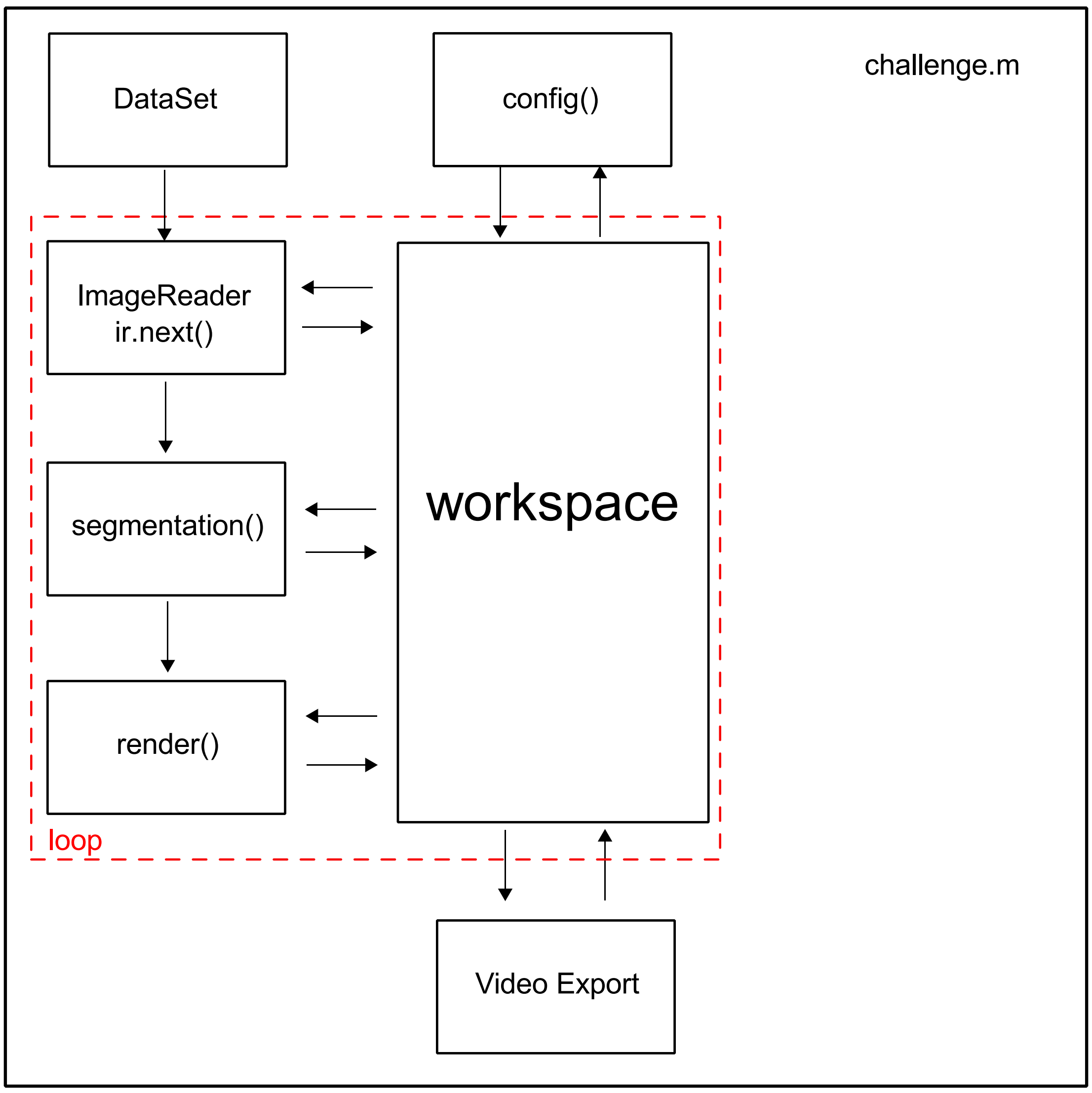


Figure 6.1 Subsystem connection diagram representing challenge.m

7. GUI

Graphical user interface (GUI) is to be defined as a form of user interface allowing users to interact with electronic devices through in this specific application graphical icons as primary notation with text navigation and text input as secondary notation. [1]

In this project GUI will be used not only as some kind of "advanced" reconfigurable config.m file for parameter processing initialization inside loop as seen in Figure 7.1, but also including algorithms to call necessary functions to process image (ImageReader; ir.next(); segmentation(); render()), convert processed image into video to be exported and displayed on the GUI as the final part of the process.

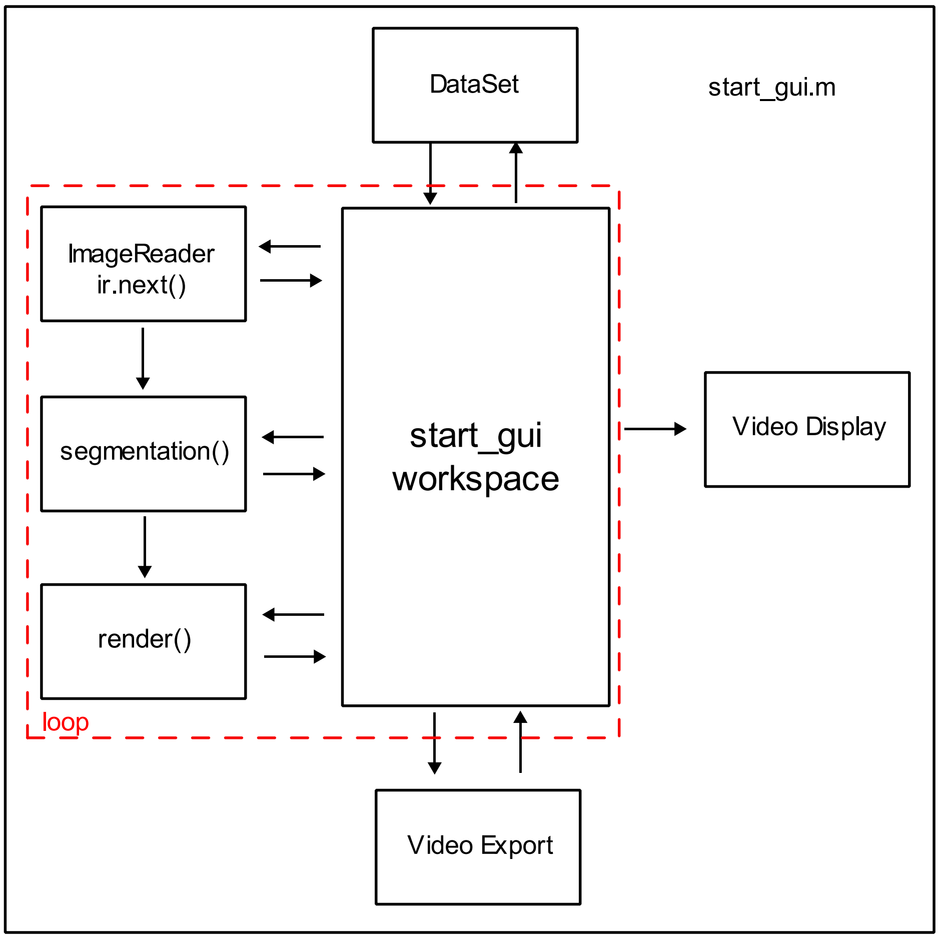
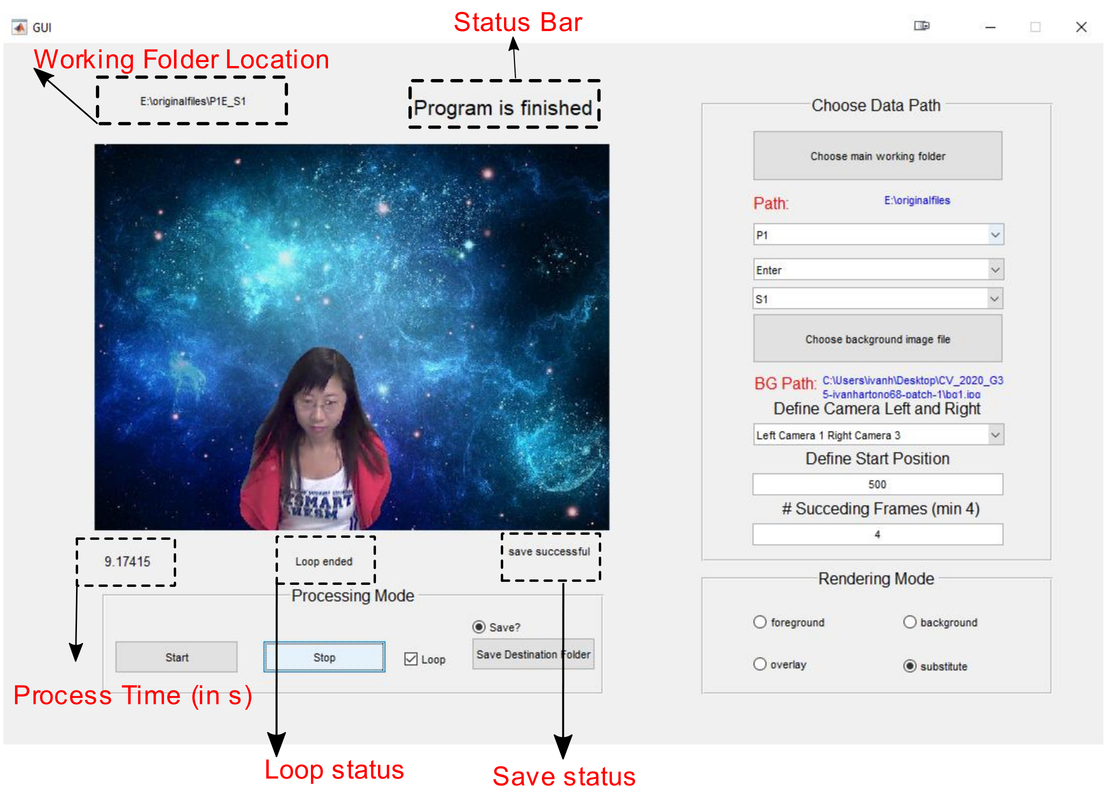


Figure 7.1 Subsystem connection diagram representing working process of GUI

Despite an overall design similarity to challenge.m, there are some crucial differences important to be noted to use the GUI as intended. This subchapter will then start from the type of input intended for "DataSet" compilation, including folder structure of the image to be processed, continued with brief explanation of status bar on the GUI, initiation procedures, short descriptions of each control input, and concluded with a short analysis on the overall functionality process.

- GUI Overview

  GUI of the project is designed using GUIDE function of MATLAB [2], as it is still widely used nowadays, easy-to-use interfaces and will still be executable in MATLAB even after its removal in future release. Upon starting start_gui, GUI will look as seen in Figure 7.2.



E:\originalfiles\P1E_S1

Program is finished

9.17415

Loop ended

save successful

Path: E:\originalfiles

P1

Enter

S1

BG Path: C:\Users\ivanh\Desktop\CV_2020_G35-ivanhartono68-patch-1\bg1.jpg

Define Camera Left and Right: Left Camera 1 Right Camera 3

Define Start Position: 500

Succeding Frames (min 4): 4

[✓ Save?]

[✓ Loop]

○ foreground ○ background ○ overlay [✓ substitute]

Figure 7.2 Different status bar around video output giving necessary information

Status Bar (output_val): defined as output_val, status bar displays state of the processing, in which “Starting Program” is written upon pressing Start button, numbers representing number of iteration done, “Saving Program” defining start of video export process, “Showing Video” defining start of video display, and “Program is finished” signaling end of all processing in app.

Loop status: defining the start and end of loop as represented as red dotted lines in Figure 7.1.

- Initiation Procedure: Input
    - Working Folder

First part of GUI to be discussed in detail is the choose data path button group as seen in Figure 7.3. This button group consists of a button to choose main working folder, "Path" signifying chosen main working folder, input parameter possibility for portal, entrance method, and scene, button to choose background image file location with corresponding "BG Path", definition of left and right camera, starting position of processing, and number of succeeding frames.

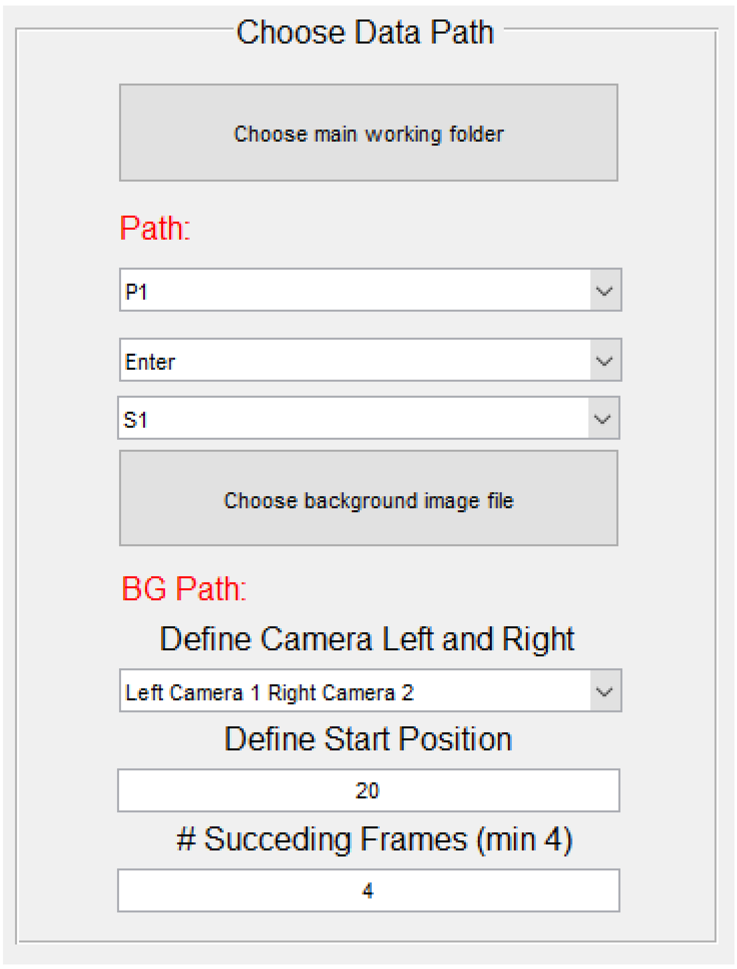

Choose Data Path
Choose main working folder
Path:
P1
Enter
S1
Choose background image file
BG Path:
Define Camera Left and Right
Left Camera 1 Right Camera 2
Define Start Position
20
# Succeding Frames (min 4)
4

Figure 7.3 Choose data path button group

To understand the choose main working folder, path, entrance method, and scene relation to build the needed src for further processing, Figure 7.4 can be seen.

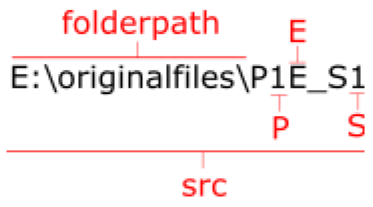


Figure 7.4 folderpath, P, E and S parameters combination are to be defined as src

Similar to processes before, a ChokePoint Dataset [3] of different portals P, cameras C, scenes S, and entering state E_L is saved under certain folder structure as seen in Figure 7.5.

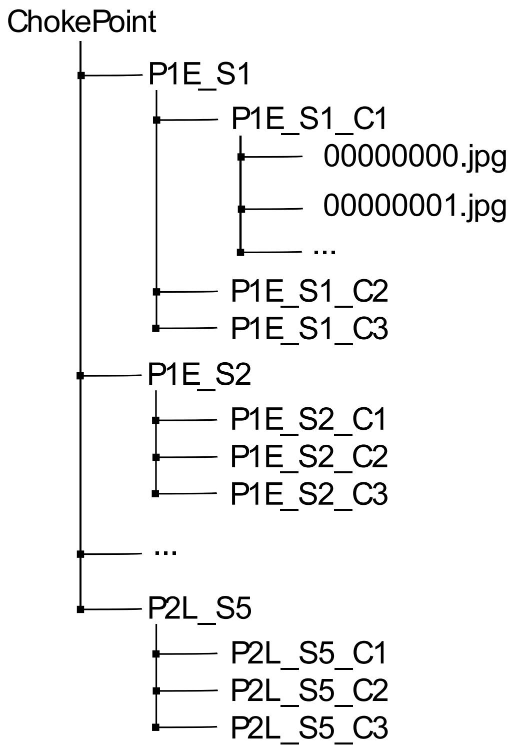


Figure 7.5 Folder structure of image representation redefined from Computer Vision Challenge 2020 for clarity [4]

Taking into consideration with issue in folder duplication in Portal 2 dataset, and other extraction problem as posted by other students in [4], it is highly suggested to adapt the folder for processing following the structure in Figure 7.2, in which P2E_S1_C1.2; P2E_S2_C1.2; P2E_S3_C1.2; P2E_S4_C1.2; P2L_S1_C1.2; P2L_S2_C1.2; P2L_S3_C1.2; P2L_S4_C1.2; … are to be **deleted** and P2E_S1_C1.1; P2E_S2_C1.1; P2E_S3_C1.1; P2E_S4_C1.1;… are to be **renamed** accordingly (P2E_S1_C1;…).

- With this in mind, folderpath is the path only until the ChokePoint main folder ("~/ChokePoint") in Figure 7.5, **not** e.g. "~/ChokePoint/P1E_S1".
- The next input parameter to be considered is left and right camera, and there are in total only 3 possibilities of its pairing, which are

  (L, R) = {(1,2); (1,3); (2,3)}.
- Because of the increasing error in result by an increase in number of succeeding frames N, and an increase in computational load and other problems by too much reduction of N, in this project N is to be strictly defined as equal to 4.

- Initiation Procedure: Rendering Mode

As seen in Figure 7.5, this part of GUI is needed to determine which mode is to be used in the rendering process, see chapter 4 for more detail on rendering process.

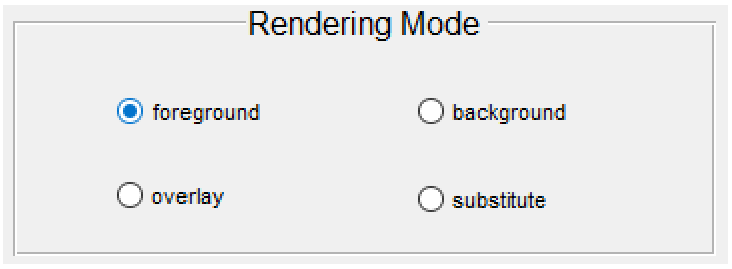


Figure 7.5 Rendering mode for render()

- Initiation Procedure: Processing Mode

This part of GUI deals with storing files, starting process, ending process, and defining whether non-terminating loop should exist or not for the process as seen in Figure 7.6.

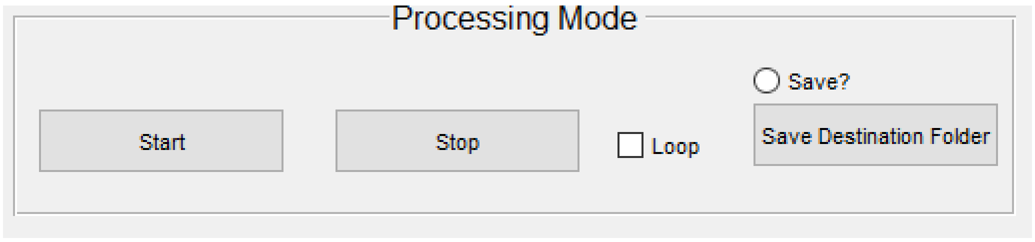


Figure 7.6 Processing mode including save, loop checkbox, start and stop

  - Save button
    In order to store files in this GUI, 2 process is needed to be done. One is to enable the checkbox for save to determine whether saving is done or not. Second part of the process is to define where to save the file. By pressing the “Save Destination Folder”, an explorer window will be opened, in which we need to decide where to save the video. It is very important to note that whether the video composed from frame is saved or not, the video will always be displayed on the GUI at the end of the process.
  - Stop button
    This button works only to stop the non-terminating loop when Loop checkbox is activated.
  - Start button
    This button is to be used to start all operation. It is important to note that necessary parameters are needed to be chosen first before pressing this button. Some inputs are necessary to be given, e.g. destination folder, as every computer has differing destination folders, and optional input with default value provided if no input is given to the parameter at GUI.

Button Name	Status	Default value for GUI
Choose main working folder	Required	-
P1	Optional	P1
Enter	Optional	Enter
S1	Optional	S1
Choose background image file	Required	-
Define Camera Left and Right	Optional	Left Camera 1 Right Camera 2
Define Start Position	Optional	20
Rendering Mode	Optional	foreground
Save?	Optional	no
Save Destination Folder	Optional (is Required if Save? Is checked)	-
Loop	Optional	no
N	Automatic Input	4

Table 7.1 overview of GUI’s input determining which is optional and required.

For exact initiation procedure, which part to start etc. please look at Readme.txt

8. Appendix

1. file preservation

All codes and needed files will be preserved on Github as Project CV_2020_G35.

https://github.com/kooste2018/CV_2020_G35

The original image files can be downloaded from

http://arma.sourceforge.net/chokepoint/#download.

2. contributions

Thanks for all team members for their active communications and code contributions during this pandemic time. Here is a table as reference for grading if needed.

Partial function	Contributed by
ImageReader.m	Jiangnan Huang
segmentation.m	Zhiwei Liang
render.m	Jiangnan Huang and Zhiwei Liang
config.m	Jiangnan Huang
challenge.m	Jiangnan Huang
GUI	Nan Chen and Ivan Hartono
Documentation and readme	all

3. rendered images

The following figures are the processed images with subtracted background of different scenes.

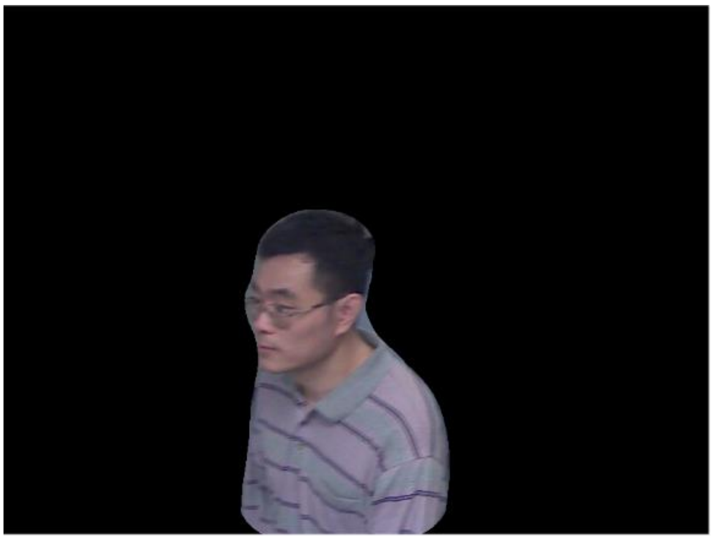


Figure 1 P1E_S1

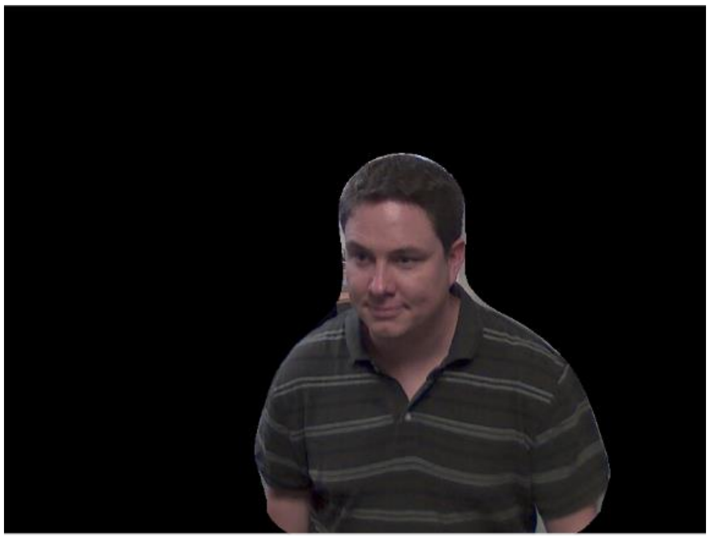


Figure 2 P1L_S3

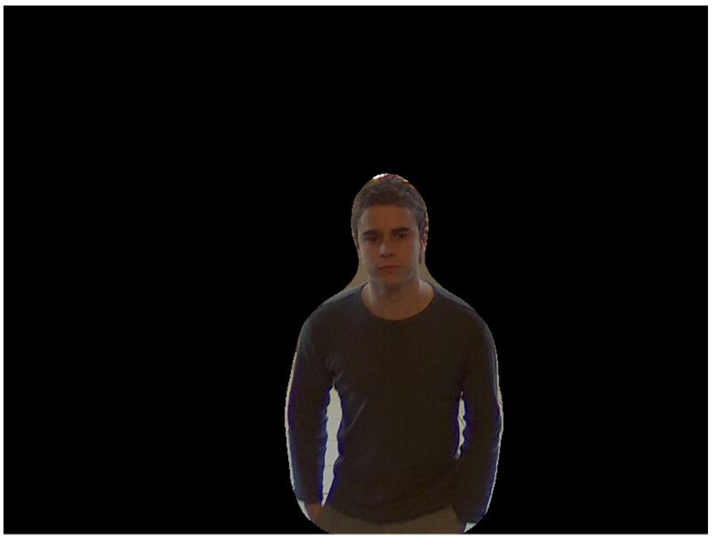


Figure 3 P2E_S2

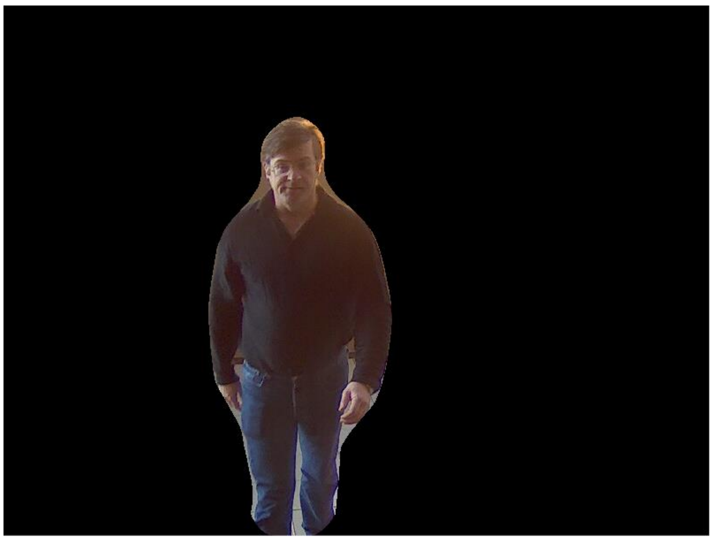


Figure 4 P2E_S4

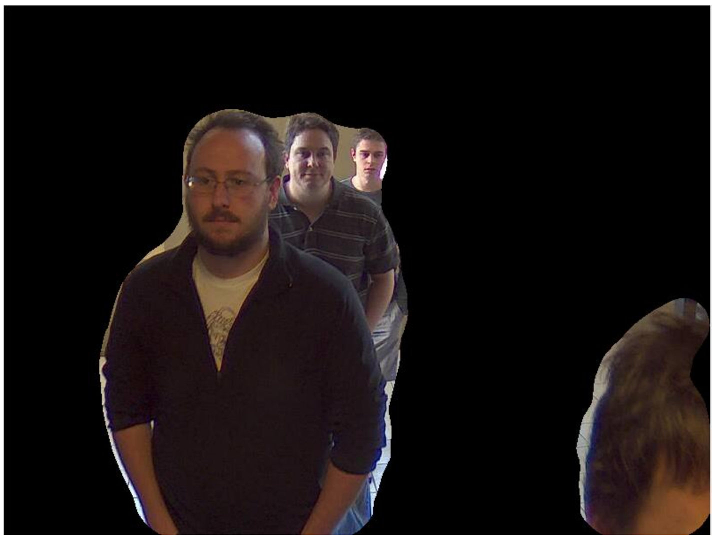


Figure 5 P2E_S5

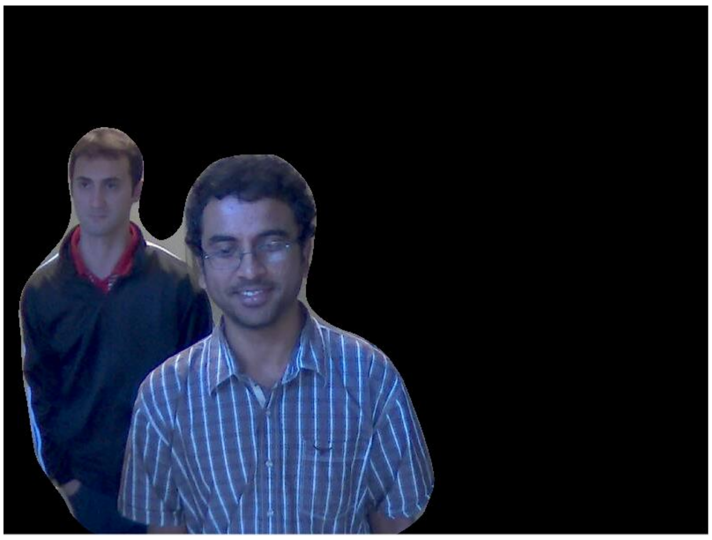


Figure 6 P2L_S5

9. References

[1] Levy, Steven. "Graphical User Interface (GUI)". Britannica.com. Retrieved 2019-06-12.

[2] https://de.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html

[3] Y. Wong, S. Chen, S. Mau, C. Sanderson, and B. C. Lovell, \Patch-based probabilistic image quality assessment for face selection and improved video-based face recognition," in IEEE Biometrics Workshop, Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 81{88, IEEE, June 2011).

[4] Diepold, Klaus; Röhrl, Stefan. "Computer Vision Challenge", Chair of Data Processing, Technische Universität München, June 2020

[5] "checking extracted folder scenes", https://www.moodle.tum.de/mod/forum/discuss.php?d=199125

[6] https://de.mathworks.com/help/matlab/ref/persistent.html

[7] Guo-Wu Yuan, Jian Gong, Mei-Ni Deng, Hao Zhou and Dan Xu, 2014. A Moving Objects Detection Algorithm Based on Three-Frame Difference and Sparse Optical Flow. Information Technology Journal, 13: 1863-1867.