

**Documentation of Computer Vision Challenge SS2020**

*Group: 35*

*Author: Jiangnan Huang*

*Zhiwei Lin*

*Nan Chen*

*Ivan Hartono*

2020.7.12

**Contents**

[1. Introduction 2](#_Toc9428)

[2. ImageReader.m 2](#_Toc29596)

[3. segmentation.m 2](#_Toc29441)

[4. render.m 2](#_Toc1932)

[5. config.m 2](#_Toc20461)

[6. challenge.m 2](#_Toc27185)

[7. GUI 2](#_Toc28893)

[8. Appendix 2](#_Toc3425)

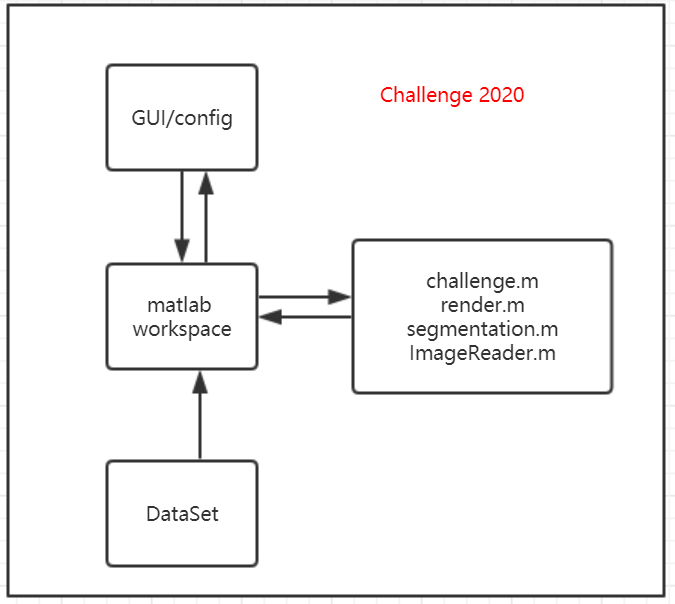
# Introduction

“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 erstzen.”(Computer Vision Challenge, Doktor Claus Diepold, 2020)

The aim of this project is to recognize foreground and replace background with virtual one or all black, and vise verse. Except that we add a overlay mode to present both of them. After processing all images in the folder, a video will be generated as output, in which for example the background is replaced by a virtual background to protect privacy. The customer friendliness is very important and it is also considered in this project. For that a graphic user interface is designed, so that it is much easier and comfortable to change some parameters in the program and to get different expected results.

The program will be split into 6 part. 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. |



# ImageReader.m

In the ImageReader class are two functions defined.

The first function with the same name will be used, when the object ir is initialized. An inputparser 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.

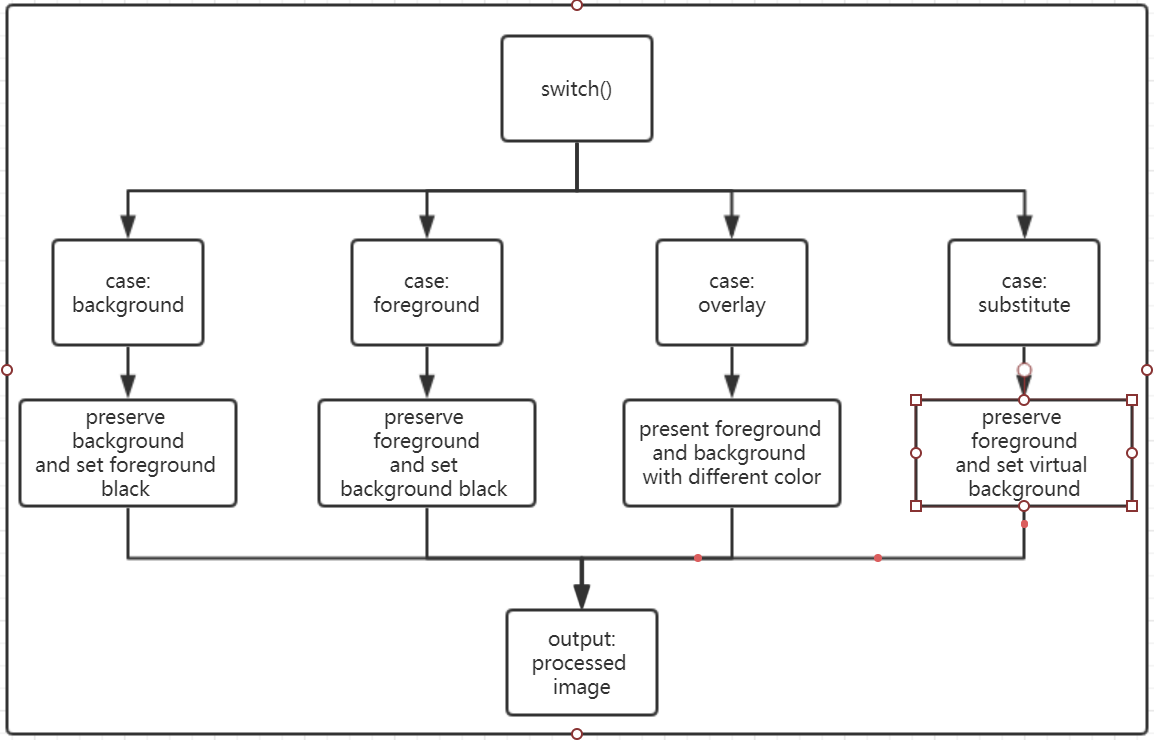
For an efficient implementation I use first dir() to get all information of images in the folder. Then I just go indexing further and further till end. In the end I use if condition to judge if I should go back and reset the “index”. For example, I use allimage=dir() first. This variable “allimage” is a 1xN dimentional struct variable and it saves path and name of every image in this folder. Then I can use a index variable, 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++.

# segmentation.m

Liang write here!

# 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 outputted. The figure below shows the coarse structure.



# config.m

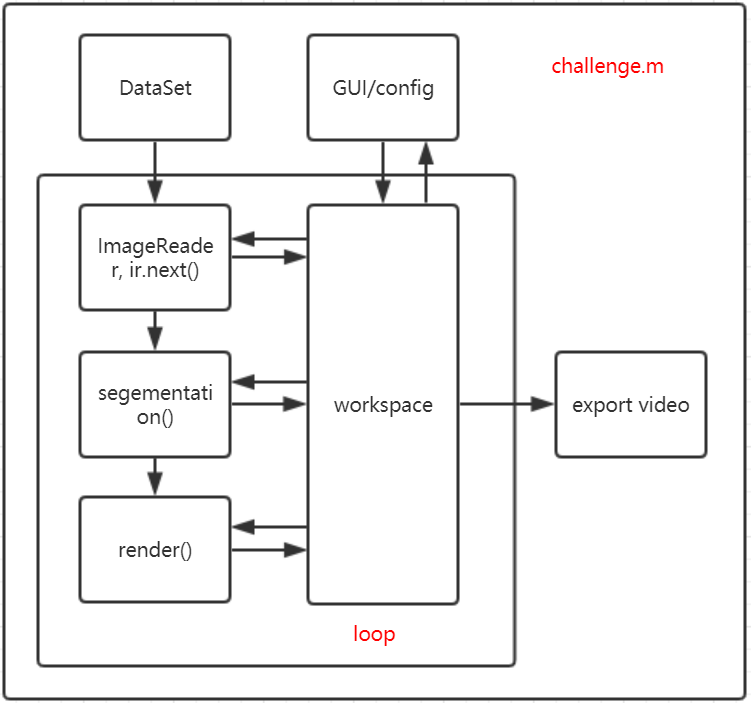
The config.m initializes parameters with default values, if there are no essential parameters passed via GUI. These parameters will be used for image processing later. The following table lists all parameters introduced by config.m. In this part a operation system recognition is also added because the path can be different. The Matlab predefined function ismac(), ispc() and isunix() are used. If the system 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 |
| dst | 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 |
| 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 |

# 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. This part will work with GUI together via the mutual base workspace of Matlab. The following block diagram shows how it exactly works. The “DataSet” is the images to be used. In such 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 apply on image again. All of this will be executed in loop until the all of the images are already processed. These rendered images will first be saved in cell “movie” and transformed into a AVI-Video in the end.

The GUI runs independently from others. That means it can read and write parameters and variables from or into workspace simultaneously. So only if the GUI doesn’t pass enough parameters and variables, or the program is going to be reset, the config.m will be used instead, in order to load default values.



# GUI

Chen and Ivan write here!

# Appendix

## 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](http://arma.sourceforge.net/chokepoint/" \l "download).

## 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 Lin |
| render.m | Jiangnan Huang and Zhiwei Lin |
| config.m | Jiangnan Huang |
| challenge.m | Jiangnan Huang |
| GUI | Nan Chen and Ivan Hartono |
| Documentation and readme | all |

## rendered images

Add best processed images here! See requirement in pdf docu part.