Computer Vision Challenge 2020

Abdullah Özbay, Ahmet Gulmez,
Baris Sen, Mine Tülü

Chair for Data Processing
Technical University of Munich
Germany

Abstract

The phrase *Computer Vision* is described as a scientific field that aims to give computers the capability of human vision. The main and primary points of the domain are given as image reading, image analysis, and image process within the understanding of the content of digital images. Applications and studies in the field of computer vision have proceeded since 1960 [3]. Computer vision provides a wide range of applications and advantages in various areas such as the 3D reconstruction of scenes, object recognition, navigation, and visual reality. As a multidisciplinary area of study, computer vision uses techniques and methods of disparate engineering and computer science fields. It has been seen over the years that machine learning and artificial intelligence studies comprehend computer vision work. The primary goal of the planned work in *Computer Vision Challenge 2020* is the distinction of foreground and background of an image, thereby unwanted parts of a scene can be surrogated.

Keywords: Image Processing, 3D Recognition, Object Recognition, Machine Vision, Feature Extraction

Contents

List of Figures

1 Introduction

The primary purpose of this year's computer vision challenge is the segmentation of any participant in a video conference from the background recorded by the 3 different camera. Designating this task is that many people have to continue their work at home due to the coronavirus, and video conferences are the only option to hold meetings during this period. The report covers how the data set of different sources are used to determine the foreground and the background and how the participant of the video conference is fractured from the background. The data set includes recordings of surveillance cameras on various portals. These recordings are filmed for each scene by three different cameras. With the help of the defined position(left or right) of the cameras, segmentation will be applied to the image to locate objects in image [6]. The implementation part covers how algorithms, methods, and techniques are used to accomplish the challenge's primary goals. The last part of the report includes the results and the conclusion of the work. The flow of the process of the Computer Vision Challenge is given in Figure 1.

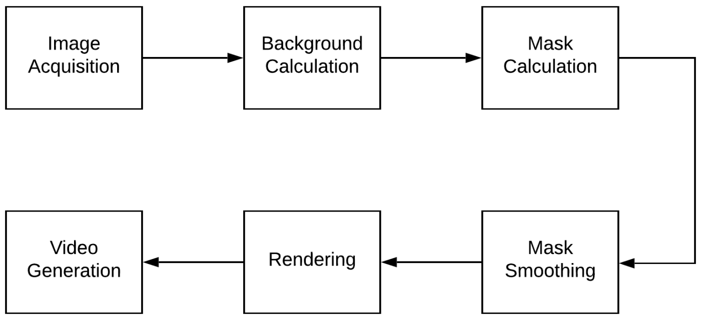


Figure 1: Flowchart of the process

2 Implementation

This assignment's focus is the segmentation between foreground and background and the possibility of replacing undesired parts of the scene. The section 2 outlines all the main parts for getting the desired distinction between background and foreground.

2.1 Data Set

At the beginning of the assignment, many different files were received as sources. The correct functioning of the algorithm for each source was essential in this assignment. Every source consists of sequential video frames that are images in JPEG format.

2.2 Image Reader

A class was created to play two of the three video streams from a scene folder as an endless loop, and its name is *ImageReader*. First, we determined the properties of the object in the image reader class. There are two main functions.

The first function includes a constructor that creates an ImageReader object. The object contains five input parameters in sequence: source, L, R, start, N. The source needs to be a scene folder that contains the frames of the video and is located in the Choke Point folder. The folder Choke Point includes a file format as i.e *P1E_S1* and source choose one out three sub-folders of the *P1E_S1*. The input parameter start is the index of the first frame; if the value for a start is not specified, then it is 1. L and R are the numbers from $1, 2, 3$ that indicate which cameras we should use for background subtraction. The functions assigned all those variables sequentially in the object.

From the given source path (ex. "./ChokePoint/P1E_S1"), the name of the folder (ex. "P1E_S1") was retrieved to use for getting the name of the camera folders. This corresponds to the last five characters of the source path. The names of the camera folders were initialized (ex. "P1E_S1_C1"). All the source paths and the number of images were stored in the attributes.

The object *ir.next* from *ImageReader* class is responsible for returning two tensors (*left* and *right*) and a flag *loop*. These tensors, which are packages of N frames following the current frame, will then be used in *segmentation* function. If there are at least N images following the current frame, then return N+1 images for the left and right cameras. Otherwise, return all the remaining frames. The function includes three variables, left, right, and loop, respectively. The variable "loop" indicates whether the current frame is the last. If not, then sets the loop variable to zero, else to one.

2.3 Segmentation

The task of the segmentation function is to provide an optimum background mask. To achieve the goal, a background model should be obtained. This background estimation will be handled in section 2.3.1.

The calculated background model will then be used in background subtraction as follows:

For each layer R, B, and G: It was checked whether the difference between each pixel of the background image and the current image is greater than a constant threshold. In this case, the pixel was set to one (foreground). The threshold was used for each color channel to observe the difference, and the value was set to one in the matrix when one of the channels exceeded the threshold.

The mask which was created from the above-mentioned background subtraction does not contain the complete foreground in it. Besides, it had some noise that can be caused by the shadows of the people and random local luminance differences. Therefore it was used a smoothing algorithm that has the duty to reduce the noise and have a better foreground detection. The smoothing function was explained in details in section 2.3.3.

2.3.1 Background Modelling

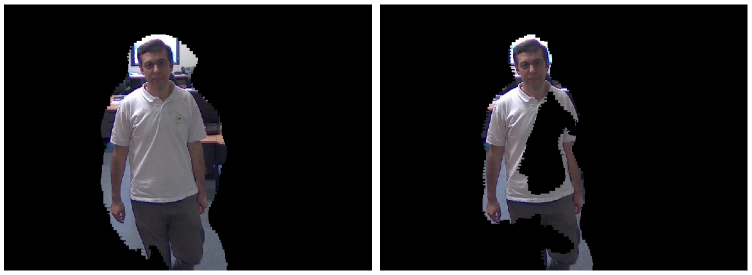
The background model is one of the most important concepts in the project. To have an ideal foreground detection, creating a good estimated background model is crucial. This model was calculated at the beginning of the segmentation algorithm. The model was formed from N number of frames, which are the ten previous and next frames. A filter will afterward process these N frames. This filter makes it possible to gain a model that is dominated by the background. This calculated background will be shown in section 3.1 from different captures. The filter that was used for this estimation is obtained empirically from two filters; mean and median filter.

2.3.2 Mean Filter vs. Median Filter

Median filtering is a nonlinear operation generally used in image processing to reduce noise [1]. The median filtering algorithm has good noise-reducing effects, but its time complexity is not desirable. The algorithm uses the correlation of the image to process the filtering mask features over the image. It can resize the mask according to noise levels of the mask [7]. The median filtering selects many neighbors from the central points. The median is the new value that replaces incorrect points. So that is the refinement process.

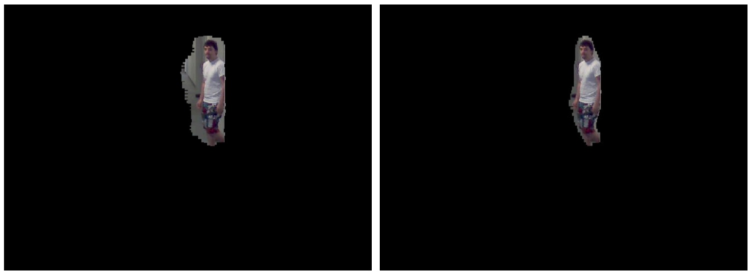
Mean filtering is a linear non-adaptive filtering method, where their intensity values average specific amounts of images. For background modeling, N number of images was operated to calculate the mean out of it [2].

The accuracy of the mean and median filters in our algorithm is shown in the figures below figures 2 and 3. As observed, the accuracy of the filtering types can vary in different scenes. Figure 2 demonstrates the mean filter is more accurate in such scenes. The mean filter has more information about the complete tensors, whereas the median filtering chooses one of N pixels. On the contrary, the median filter is more accurate in edge detection, as evident in figure 3. One of the reasons for that is that the mean filter weights all the pixels in N number of frames equally, which brings it to have a less accurate edge detection. Moreover, the median filter reduces the noise more than the mean filter.



(a) Mean Filter (b) Median Filter

Figure 2: Mean vs Median Filtering



(a) Mean Filter (b) Median Filter

Figure 3: Mean vs Median Filtering

2.3.3 Binary Mask Smoothing Algorithm

The raw mask is formed by subtraction of the background model from the current frame. Nevertheless, this mask had noise and some undetected foreground areas. Therefore a smoothing function was implemented to solve this problem. A sliding window was used to smooth the binary mask, and it was assigned as a matrix with ones and zeros. The sliding window was formed over some part of the data, and this window slides over the data to capture different parts of the data, which is the image frame in this work [4]. The sliding window goes through the image and inspects the determined specifications for each pixel [5].

The input parameters of the function *smoothMask* are mask, window_radius, paint_radius, sampling_radius, filling_threshold, emptying_threshold. The variable mask is where the noise will be removed. The environment of the pixels was used to decide if a pixel should be removed. So, if the parameter "window_radius" is given as "r", the area A (equation (1)) of the mask is examined.

$$A = [x - r : x + r, y - r : y + r] \tag{1}$$

If a pixel is defined as a noise pixel, most likely, the neighboring pixels comprise noise, too. To improve the efficiency of the smoothing function, all pixels recognized as noise pixels were painted in the range of the parameter "paint_radius". Since the pixel area of size S (equation (2)) was corrected, checking all pixels was avoided by looking through only every row and every column of the parameter "sampling_radius".

$$S = [2 \times paint\_radius + 1, 2 \times paint\_radius + 1] \tag{2}$$

This improves the efficiency by O($sampling\_radius^2$). The area is filled, if the rate of the number of 1's in the environment is more than the assigned value of the parameter "filling_threshold" The area is cleaned, if the rate of the amount of 1's in the environment to the number of pixels is less than the value of "emptying_threshold".

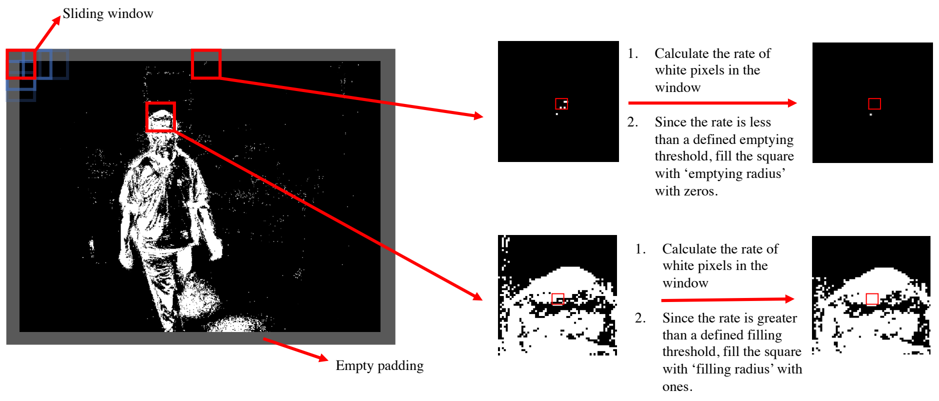

Sliding window
Empty padding
1. Calculate the rate of white pixels in the window
2. Since the rate is less than a defined emptying threshold, fill the square with 'emptying radius' with zeros.
1. Calculate the rate of white pixels in the window
2. Since the rate is greater than a defined filling threshold, fill the square with 'filling radius' with ones.

Figure 4: Smoothing process for removing noise in the mask

To apply the sliding window without edge cases, padding was added to the image. Two additional matrices were initialized. The parameter *smoothed* holds the result, and the parameter *visited* checks if a decision about the pixel "i" and "j" has been done. The next part of the code is the iteration over all the pixels. The section 4 demonstrates the smoothing process on the binary mask of a sample frame.

2.4 Rendering

The function named *result* processes the current image of the left camera frame by rendering the segmentation mask onto the image. The function contained four input parameters: frame, mask, bg, and mode. The parameter mode chooses between the modes: foreground, background, overlay, and substitute. For the foreground modes, the mask was applied to set the background to black without changing the foreground as much as possible, and the negated mask was applied for the background modes. The overlay mode provides an image that distinct the fore- and background by different transparent colors. In the last case named substitute, the background was replaced with a virtual background that was passed in the parameter bg.

2.5 Configuration

The *Configuration* file is to separate computer-specific settings from the calculation. The Configuration was always called before the *challenge.m* and filled the workspace with all necessary variables, settings, and paths.

2.6 Challenge

The main goal in *Challenge script challenge.m* is running the functions that were created before in various scripts and generate a movie of the participant in the source files with the virtual background. The first step was given as closing all the image tools and cleaning the workplace. A timer should be generated to calculate the running time of the code, thence timer is started before generating the movie. After the video is generated, the timer will be stopped.

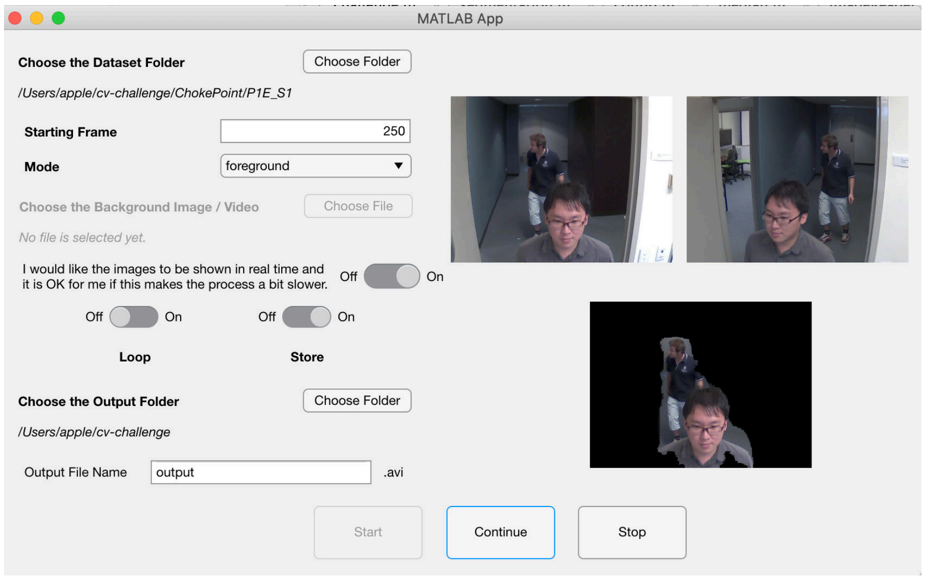
To generate the video *config* script was called to initialize the required parameters. The variable called *loop* was assigned to zero unless the last frame is reached. Another variable named *frameCounter* was written as a counter to keep track of the processed frame index. After the required parameters were defined in the script, the section for writing the movie to the disk starts.

After the video has been opened, a while loop was written to process all the frames. In the while loop next image tensors were assigned as variables named *left*, *right* and *loop*. The next steps are calculating the mask using the segmentation function and disposing of the image in the middle of the tensors. The function with the defined inputs *currentImageL*, *mask*, *bg*, and *render_mode* rendered the mask into the current image. Frames were written in the video, and for each new iteration, the number of frame counter was incremented. Additional while loop was also implemented to wait if the flow is paused from GUI. When all the frames were written in the video, the writer was closed and the elapsed time for processing all images was given as the output.

2.7 Graphical User Interface

This assignment aims to distinguish the foreground and background and replace unwanted parts of the scene. Therefore, the whole process has specific settings; an interface must be defined for users to adjust these settings. A graphical user interface (GUI) is a system of interactive visual components for computer software. Figure 5 shows the developed interface for the assignment. A GUI displays objects that convey information and represent actions that can be taken by the user. The purposes chose starting point, rendering mode, or a file path for a virtual background when the user interacts with them. The steps of GUI for this assignment are given below.

- **Step 1.** Choose a source folder. Folder's name shows the portal and the scene. The program chooses the camera.
- **Step 2.** Select an appropriate frame number as a starting frame.
- **Step 3.** Choose a mode from four different selections *background, foreground, overlay, and substitute*. If the mode chosen as *substitute*, the background of the image will be a video or image.
- **Step 4.** If the mode chosen as *substitute*, select a background image or video. File formats can be PNG, JPG, MP4, and AVI.
- **Step 5.** Select *on*, if the user wants the images to be shown in real-time. It causes extra time for the process. Select *off*, if the user wants images to be uploaded exclusively.
- **Step 6.** For the loop setting, select *on* if the user wants the program starts again, when it is completed. Select *off* for the opposite.
- **Step 7.** Store setting allows the choice if the user wants to upload the video or not. Select *on* to upload, select *off* to not upload.
- **Step 8.** Choose a source path to upload the generated video.
- **Step 9.** Assign a name for the generated video. Users can decide the title.
- **Step 10.** Click *start* to run the code. If the user clicks *pause*, the program stops. The *continue* selection restarts the program, where it left off. If the user choose *stop* option, the program breaks and user can change the GUI parameters, before the program starts from the beginning.


MATLAB App

Choose the Dataset Folder
/Users/apple/cv-challenge/ChokePoint/P1E_S1

Starting Frame: 250
Mode: foreground

Choose the Background Image / Video
No file is selected yet.

I would like the images to be shown in real time and it is OK for me if this makes the process a bit slower. Off / On

Loop: Off / On
Store: Off / On

Choose the Output Folder
/Users/apple/cv-challenge

Output File Name: output .avi

Figure 5: GUI

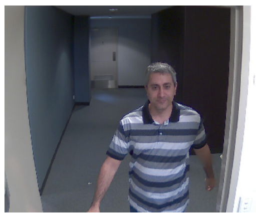
3 Results

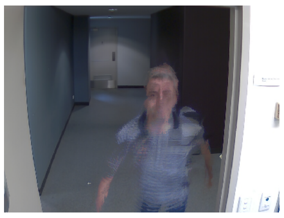
3.1 Effects of the Different Algorithms

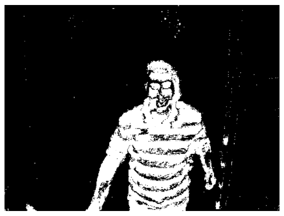
In this part, the results of the algorithm that was handled in section 2 will be demonstrated through some captures from different scenes. With the help of chosen captures, certain characteristics of the algorithm can be examined. Each below figure is separated into five images; therefore, the algorithm is demonstrated in four steps:

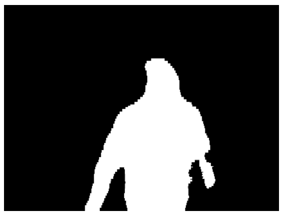
- The background that is calculated by median filtering (b)
- The initial mask provided by the subtraction of the current frame and the filtered background (c)
- The smoothed mask that is calculated afterward which is explained in the part 2.3.3 (d)
- The filled mask that is accomplished by *imfill()* function from MATLAB (e)

In Figure 6, the performance of the smoothed mask is distinguishable. The smoothed achievement depends on a successful initial mask, which relates to a successful background calculation. If the input mask has some small holes or gaps, the smoothing mask completes the task on its own.


(a) Original Image


(b) Calculated Background


(c) Initial Mask


(d) Smoothed Mask

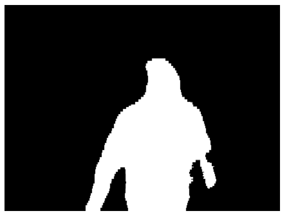
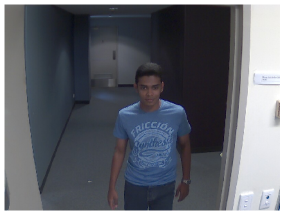
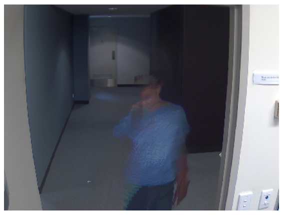
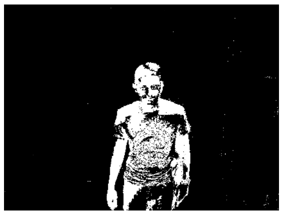

(e) Filled Mask

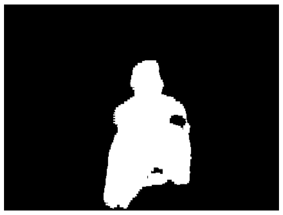
Figure 6: Scene P1E S1

In the figures 7 and 8 shows the performance of the filling functionality in the algorithm. The filling can only be helpful for undetected areas, which are rounded by detected areas. As can be seen in the figures, the undetected areas of the foreground at the bottom are still not recognized after the imfill() function.


(a) Original Image


(b) Calculated Background


(c) Initial Mask


(d) Smoothed Mask

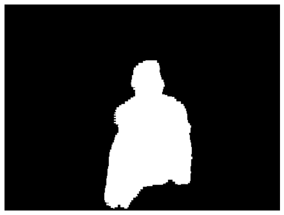
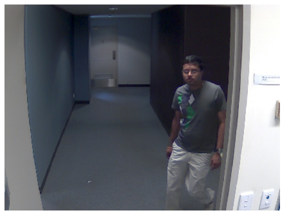
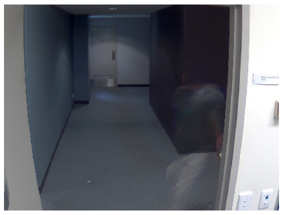
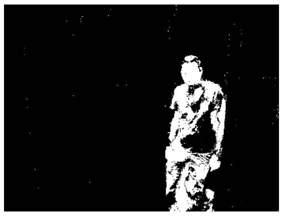
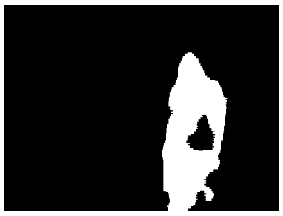

(e) Filled Mask

Figure 7: Scene P1E S1


(a) Original Image


(b) Calculated Background


(c) Initial Mask


(d) Smoothed Mask

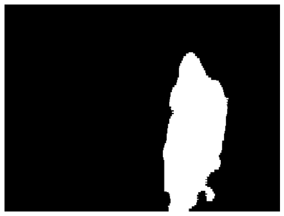
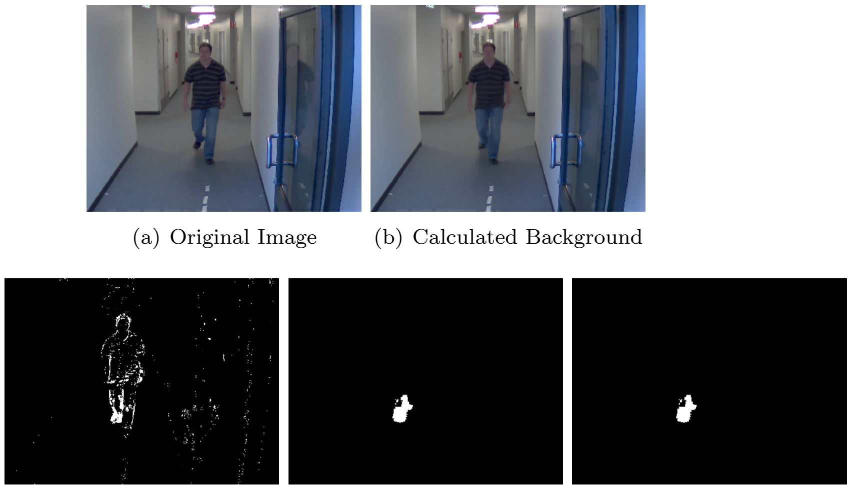

(e) Filled Mask

Figure 8: Scene P1E S2

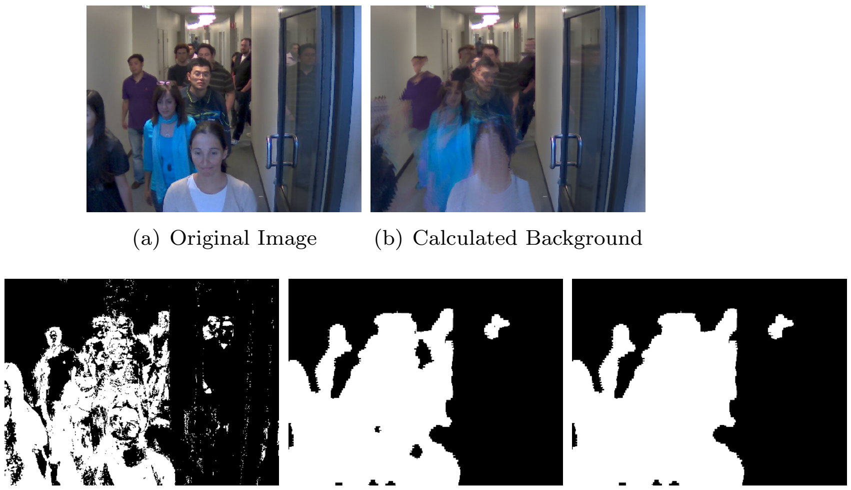
The shortcomings of the algorithm is more obvious in the figure 9. If the input mask has a noticeable amount of undetected areas then the smoothing and filling algorithm can be ineffective.


(a) Original Image (b) Calculated Background

(c) Initial Mask (d) Smoothed Mask (e) Filled Mask

Figure 9: Scene P2L S4

The figure 10 below shows the reaction of the algorithm to a greater foreground with many people. The algorithm is even more effective if many objects are close together as it is demonstrated here. Besides this, the smoothing algorithm is advantageous for false detected areas in the input mask.


(a) Original Image (b) Calculated Background

(c) Initial Mask (d) Smoothed Mask (e) Filled Mask

Figure 10: Scene P2L S5

3.2 Several Captures from the Data Set

In the following, several captures will be demonstrated in *foreground* mode with at least one person in the scene.

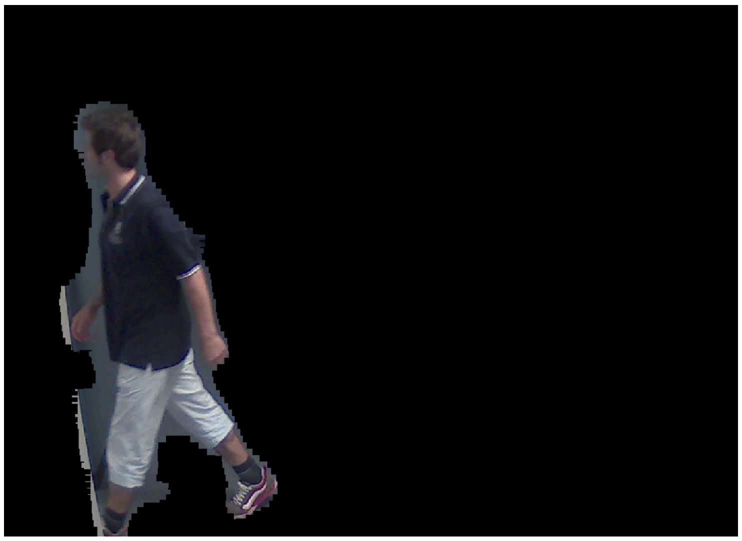


Figure 11: P1E_S1

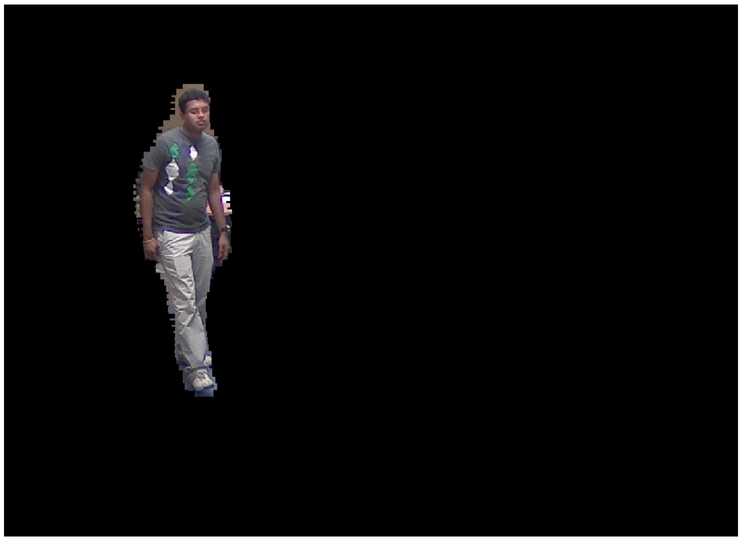


Figure 12: P1L_S3

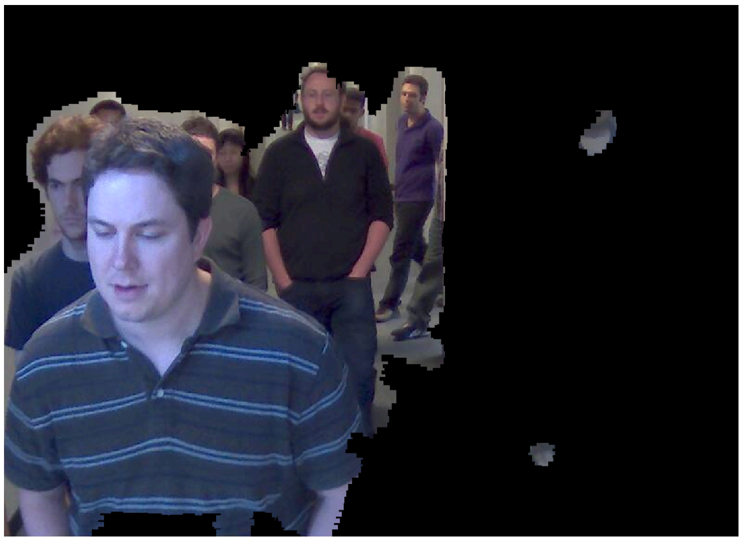


Figure 13: P2L_S5

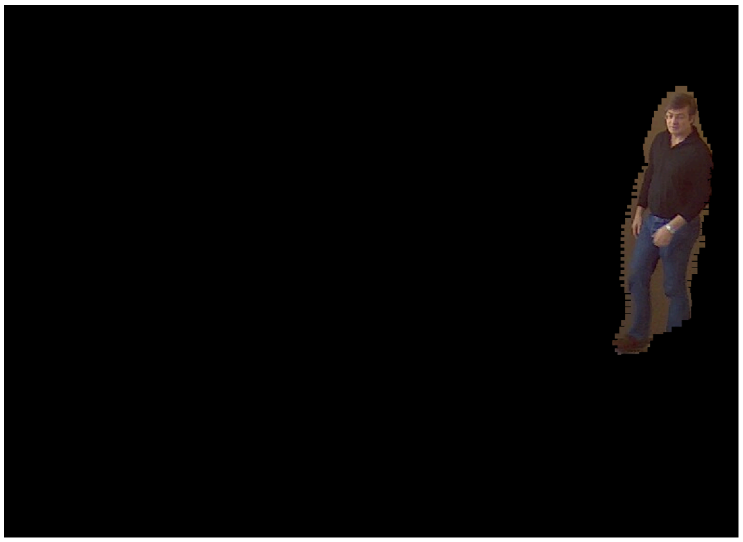


Figure 14: P2E_S4

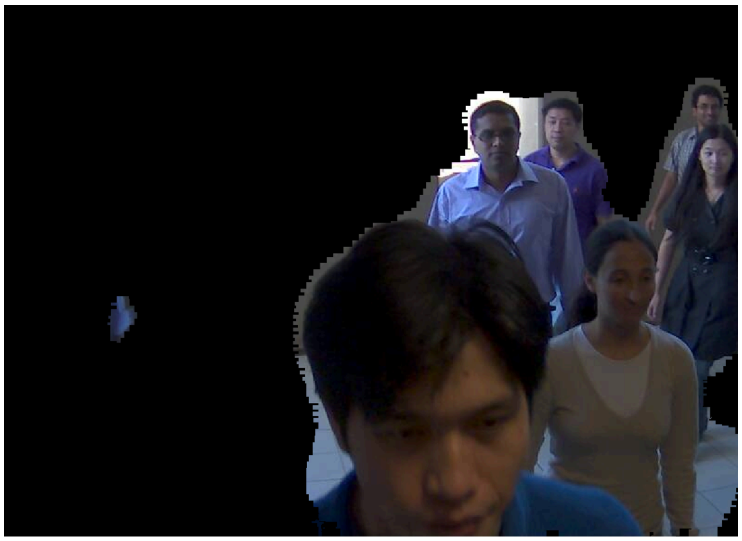


Figure 15: P2E_S5

4 Conclusion

The main conclusion that can be drawn is that the object in a video can be extracted with background calculation and application of a smooth algorithm. With the help of the recordings from three different angles, frames were chosen for background subtraction. The numbers were assigned that indicates which cameras should be used for background subtraction. The best result for the background mask was achieved with a median filter overall (N+1) frames, as it is demonstrated in the section 2.3.2. A threshold was given for each color channel to observe the difference between each pixel of the background image and the current image. The calculated mask had shown noise, thence a smoothing function by dint of a sliding window was written, which includes several hyper-parameters that were adjusted empirically. This function was performed twice to achieve the best results. A background video was also read and played. With the given algorithms and methods, the result was generated in a video, which shows the objects in the image and the background as a given background video or image.

References

[1] 2-d median filtering. https://www.mathworks.com/help/images/ref/medfilt2.html, 2020.

[2] G. Gajanand. Algorithm for image processing using improved median filter and comparison of mean, median and improved median filter. *International Journal of Soft Computing and Engineering (IJSCE)*, 1:304–311, 2011.

[3] A. B. Margaret. Mind as machine: A history of cognitive science. *Clarendon Press*, 1712:16–20, 2008.

[4] J. Moore. Sliding window algorithms. https://levelup.gitconnected.com/an-introduction-to-sliding-window-algorithms-5533c4fe1cc7, 2020.

[5] V. V. Sergeyev N. I. Glumov, E. I. Kolomiyetz. Detection of objects on the image using a sliding window mode. *Optics and Laser Technology*, 27:241–249, 1995.

[6] K. P. Sankar R. P. Nikhil. A review on image segmentation techniques. *Pattern Recognition*, 26:1277–1294, 2003.

[7] H. Cheng Z. Youlian. An improved median filtering algorithm for image noise reduction. *Physics Procedia*, 25:609–616, 2012.