Matting using a known background and a chroma-key minimization scheme

Aleksei Simonov , Elmar Eisemann , Ruben Wiersma
TU Delft

Abstract

In this paper, we investigate the problem of segmenting the foreground objects from the known background. The target application is a setting where a large display serves as the backdrop for a scene. The method proposed in this paper solves this problem in two steps. We try to separate the foreground and the background with the modified background subtraction base technique, which involves taking the per-pixel difference of the image with and without the foreground. The modification includes blurring the image before taking the difference and finding contours to separate noise and small objects in the background from the foreground. If the background subtraction fails, the chroma-key method is used: we put the solid colour screen behind the foreground object to ease the separation. To minimize the adverse effects of the chroma-key on the lighting of the objects, we propose the method for the background screen minimization using the pinhole camera theory.

1 Introduction

The new technology of compositing actors and background was introduced in the Mandalorian TV show. Instead of only using green screens behind the actors, that series used screens with the same environment added to the movie, see Figure 1. Consequently, light on the actor was a good approximation of what it should be in the film scene. However, sometimes the actors and objects in the scene need to be separated from the background to change backgrounds or composite additional visual effects. As the images on the screens behind the actor are known, it is possible to use the background information to extract the actor from the scene.

Nevertheless, sometimes different difficulties may arise during the separation process. The foreground may have colours identical to those that are in the background. As a result, the separation of the foreground and the background becomes complicated. Therefore, the chroma-keying technique might be used in such situations: placing the solid colour screen behind the actor will ease the foreground extraction.

Some works describe methods to separate foreground and background, such as [4] and [9]. Many video matting techniques exist, such as chroma-keying, difference matting, and rotoscoping [3]. When they are combined, the matting results are improved [4]. Another video matting technique based on motion in video is optical flow [9]. However, none of them explore video and image matting, having information about the background. Neither do they observe methods to minimize the amount of solid colour background screen needed. Using the least amount of that screen behind the actor can improve the overall quality of the video: the light on the objects will be more natural, and shiny objects will have less chance of having the wrong colour on them. That is why the question was raised: "How can we separate a foreground from a projected background, given the background, and how to do it using the minimal amount of solid colour background screen, in case that screen is needed?"

Figure 1: Mandalorian setup. Image by Devin Coldewey [5].

The first problem that arises is foreground-background separation. We analyze the existing works on image and video matting, and with the obtained knowledge, try to improve the baseline technique of the difference matting [4].

If the colours in the foreground are almost identical to the colours in the background, the difference matting will not give the proper results. As a solution to that problem, the chroma-key method may be used: we put a solid colour screen behind the foreground object to ease the foreground extraction [1]. To minimize the effects imposed by the chroma-key screen on the lighting and the shiny objects, the screen area should be minimized and have the closest to

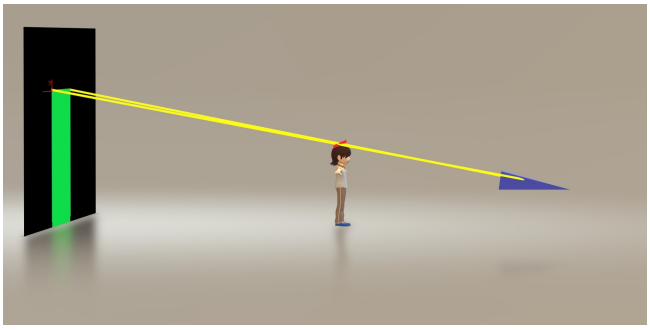

Figure 2: Green screen combined with the background colour screen.

the original colour while still allowing a safe foreground extraction. Figure 2 illustrates the concept: the size of the green screen behind the human is just enough to cover the entire area of the body.

To create a technique for background screen minimization, we analyze the theory of a pinhole camera model. Using the camera model, we find the points on the background screen which capture the camera view and the foreground object. These points correspond to the minimized chroma-key region.

This paragraph gives an outline of the paper. First of all, the paper introduces the main problem, looks at similar problems, and explains methods used to solve the main problem. Further, the author's contribution is explained: what theory researches were done and how they influenced the research questions? The written code is presented and explained. Moreover, the results are presented, and method disadvantages are analyzed. Furthermore, the discussion on the reproducibility and ethical component of the work takes place. Finally, a conclusion is drawn, and open issues are listed and briefly discussed.

2 Related works and methodology

Video matting is a process of separating a foreground of an image or a video from its background, and it is widely used in cinematography and image processing. There are many existing techniques for video matting; however, they have some limitations and disadvantages. Some well-known techniques are chroma-keying [1], difference matting [4], rotoscoping [3], and optical flow [9].

Chroma-key technique requires a solid colour background behind the actor. This technique is widely used in the real world because of its efficiency and accessibility: it is easier to extract the foreground when you know the exact colour of the background. Nevertheless, the chroma-key method introduces the problem of possible incorrect lighting on the foreground objects.

Rotoscoping requires the user to draw an editable curve around the foreground element, so it will be possible to identify and extract it. The limitation of this method is the need for human help in every case. Moreover, sometimes the curve should be adjusted during the video to be more accurate.

The optical flow technique is base on movements inside the video. It tracks the object's motion between two or more consecutive frames. As a consequence, in the absence of movement, it may be very ineffective. Moreover, the technique is not focused only on foreground movements but on all movements in the scene. This will cause problems if there are many motions of small objects in the background.

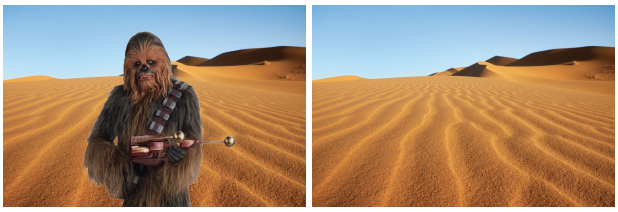

Figure 3: Left image: the image with the foreground and the background. Right image: background of the image.

Difference matting: requires estimating the background and using that info to separate the foreground and the background. The limitation of this method is that background estimation techniques are not perfect in many cases so that the overall result will suffer.

This paper focuses on the difference matting method. What if the background is given? In that case, the limitation of the need to estimate the background will disappear. Figure 3 shows the input that is required in this case. Given both the original image and the image with the background only, it is possible to use the difference matting technique while overcoming its main limitation.

The technique to deal with such a problem is called background subtraction. The most basic formula for background subtraction is

$$x_t(s) = 1 \text{ if } d(I_{s,t}, B_s) > \tau \tag{1}$$

where τ is a threshold, x_t is the motion label field at time t, d is the distance between $I_{s,t}$, the colour of pixel s at time t, and B_s, the background model colour of pixel s [7]. This baseline technique works for simple images, like in Figure 3, but fails for more complex or real-life scenes. In real-life scenes, the light can have small changes in every frame, and objects can appear in the background. Therefore, the colours will be slightly different, not letting the baseline technique provide accurate results. There is plenty of models, which build upon the baseline method, which all have some advantages and disadvantages.

The temporal median filter takes n last frames and uses their median value as the background model. This technique concentrates on recognizing the motion and the shadows in a video [6].

A mixture of Gaussians concentrates on modeling each pixel as a mixture of Gaussians [12]. It is assumed that the background has plenteous temporal changes. This method is perfect for outdoor tracking.

In the setup of the project, where the background screen is used to have good lighting on objects and a correct shining, there is no need to introduce background changes. Moreover, the shadows are not needed in background subtraction for filming, and motion does not always happen. This paper

concentrates on solving the background subtraction problem given the approximately static background.

3 Methodology and contribution

Given the image's background, there are techniques for foreground extraction, but some challenges may arise. First, this chapter introduces the technique for separating foreground and background. Further, it gives a solution in case the simple technique does not give accurate results.

3.1 Foreground extraction technique

In filming, the separation of the foreground and the background is one of the main problems. Suppose the background of the image or the video is given. In that case, the separation may be done with a per-pixel difference of two images: the first image contains just the background, and the second image contains both the actor and the background. The algorithm for the presented method is implemented in Python with the help of the OpenCV library [2].

The first step of image processing is the blurring phase. Before taking the per-pixel difference, the images are blurred using Gaussian blur [8] to compensate for possible small changes introduced by light or small background movements. The pixel values will be closer to the neighborhood average, and small changes will not affect the scene. In Python, Gaussian blur can be done using the OpenCV method `GaussianBlur`.

Further, we calculate the image difference and the first version of the mask. The per-pixel difference of the two images is calculated and compared to the threshold. If the difference is more significant than the threshold, the colour at that pixel is evaluated to white, otherwise to black.

Moreover, small regions that should be removed may still be in the mask. Therefore, the contour finding technique is used: it detects the contours of all the elements remaining in the mask. The border detection algorithm [13] is used to detect the contours. OpenCV function `findContours` helps to find those contours. If the regions' area is smaller than the threshold area, the region is removed from the mask. Assuming the foreground element is big enough, this technique will only leave the foreground object in the mask.

Furthermore, the same contour detecting technique is applied, but on the new mask where all the pixel values are inverted pixels of the original mask. This step helps detect tiny regions inside the foreground part of the image that were wrongly removed. If the contour area is less than the small threshold(currently 0.2% of the image area), the area remains on the new mask.

Finally, the masks are combined. This way, the mask for the image is created. After that, the original image is compared to the mask, and only the pixels, where the mask is white, are taken.

The method may fail if the difference between the foreground and the background in the original image is less than the threshold. For example, Figure 4 shows the result of applying the algorithm to a picture with a white bird on a snowy background. As can be seen, there are too many gaps inside the output image. Therefore, another method to solve such exceptional cases is required.

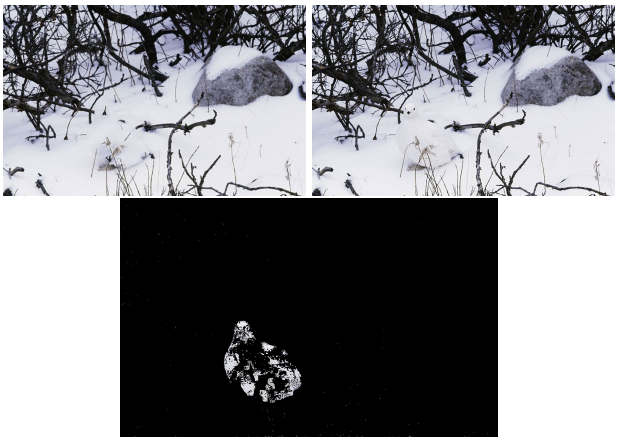

Figure 4: Left image: the image with the background. Right image image: the image with both the foreground and the background. Bottom image: image with the difference of two images.

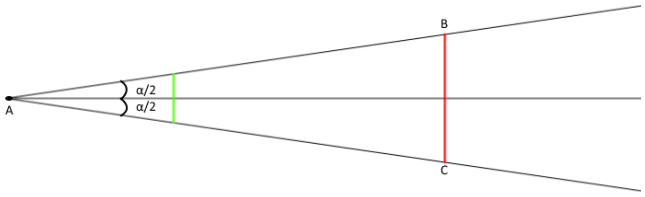

Figure 5: Pinhole camera simple image.

3.2 Background changes

To achieve acceptable results in the scenario where the background has the same colour as the foreground, the background colour should be slightly changed. If the snow behind the bird in Figure 4 were closer to gray, the white bird would be perfectly visible on that background. Therefore, the chroma-keying method using a grayish background is required. Nevertheless, the screen area should be minimized to lower the effect of the wrong colour on the lighting of the foreground object.

Pinhole camera theory

To minimize the required background area, the pinhole camera model is used. Consider the pinhole to be the camera center. The pinhole camera model has two clipping planes: the near clipping plane and the far clipping plane. The objects that are closer to the camera center than the near clipping plane and the objects that are further than the far clipping plane are not visible in the camera. We assume in this paper that the far clipping plane is the projected background screen and the near clipping plane is the image plane. Furthermore, every real camera has its angle of view, which corresponds to the visible area in the camera [11].

Figure 5 shows the example of the pinhole camera model.

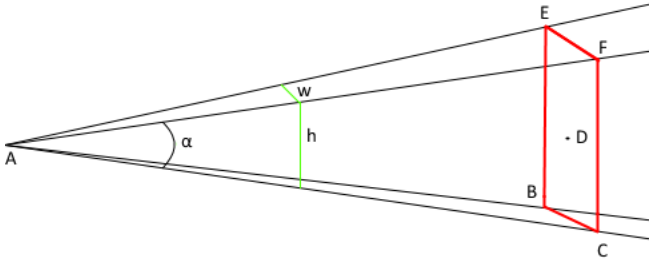

Figure 6: 3D version of the pinhole camera.

The angle of view of the camera is α, so half-angle is $\frac{\alpha}{2}$. The green line is the near clipping plane, and the red line is the far clipping plane. Given that the camera center is point A, the point on the far clipping plane is point D, and given the angle of view is α, it is possible to calculate points B and C of the intersection of camera rays with the background screen. If point A coordinates are $\begin{vmatrix} x \\ y \end{vmatrix}$ and point D coordinates are $\begin{vmatrix} d \\ e \end{vmatrix}$, points B and C can be found using the equations

$$B = \begin{vmatrix} d \\ y + (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

$$C = \begin{vmatrix} d \\ y - (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

where x-d is the horizontal distance between point A and the far clipping plane.

In 3D geometry, the virtual camera has four outgoing rays, and one more parameter is necessary to calculate the intersection of those rays with the far clipping plane. The aspect ratio of the camera is the proportion of its width to its height [10]. This ratio is also specific per camera.

In Figure 6, the aspect ratio is the relation $\frac{w}{h}$, where w is the width, and h is the height. Point D is the point on the far clipping plane. Taking aspect ratio into account, assuming the coordinates of point A are $\begin{vmatrix} x \\ y \\ z \end{vmatrix}$ and the coordinates of point D are $\begin{vmatrix} d \\ e \\ f \end{vmatrix}$, points C, B, E, F can be found using the equations

$$C = \begin{vmatrix} d \\ y + (x-d) \cdot \tan\frac{\alpha}{2} \cdot \frac{w}{h} \\ z - (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

$$B = \begin{vmatrix} d \\ y - (x-d) \cdot \tan\frac{\alpha}{2} \cdot \frac{w}{h} \\ z - (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

Figure 7: The bounding box.

$$E = \begin{vmatrix} d \\ y - (x-d) \cdot \tan\frac{\alpha}{2} \cdot \frac{w}{h} \\ z + (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

$$F = \begin{vmatrix} d \\ y + (x-d) \cdot \tan\frac{\alpha}{2} \cdot \frac{w}{h} \\ z + (x-d) \cdot \tan\frac{\alpha}{2} \end{vmatrix}$$

Obtaining these four points on the far clipping plain gives the area visible from the camera. This is the area that the chroma-key screen should cover.

Minimization based on human

In some cases, the amount of chroma-key area in the background can be minimized even more if the bounding box with the foreground is given. Figure 7 shows the example where the bounding box coordinates with the needed foreground are given. The given image is the view that is visible from the camera, so it is on the image plane, which we assume is equivalent to the near clipping plane.

From the image, the info about its width and height can be gathered, and these two values give the aspect ratio value. The coordinates of the projected area on the background screen are obtained using the bounding box coordinates. In Figure 8, the blue rectangle is the image plane, and the dashed blue rectangle is the bounding box. The green rectangle is the full view from the camera of the background screen. The red rectangle is calculated and filled in with the required colour, using background screen coordinates, coordinates of the camera, and the bounding box.

To obtain the required colour, the image's dominant colour with both the background and the foreground is calculated. Further, it is increased or decreased by some fixed amount. That guarantees that the foreground will not be of the same colour as the background.

Minimization scheme implementation

```
def minimization(camera, plane, angle,
    image, bounding_box):

    aspect_ratio = image.width/image.height;
    points = find_points(camera,plane,angle,
    aspect_ratio)
```

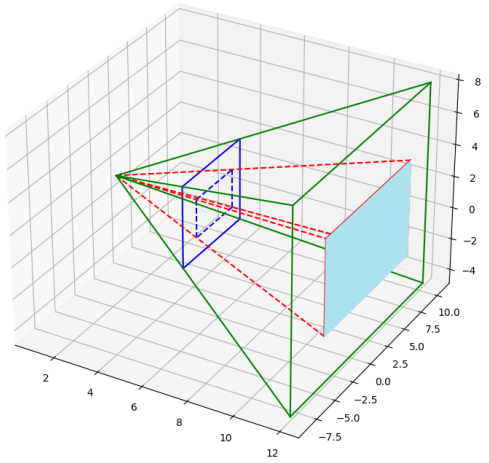

Figure 8: Required chroma-key area.

```
colour = dominant colour of image
factor = 15
for every pixel:
if pixelValue + 2 * factor > 255:
colour = colour - factor
else:
colour = colour + factor

box = convertToImagePlane(bounding_box,
image_plane, width, height)
chroma_key = chromaKeyPoints(camera, plane,
box)
```

Listing 1: Find points ob background screen

The code that creates the background chroma-keying screen is implemented in Python. Listing 1 shows the main parts of the code. The method `minimization` is the primary method of the program that calls all the other methods. As an input, it takes the coordinates of the camera center, the point on the background plane, the vertical angle of view of the camera, and the given image with both the foreground and the background.

Moreover, suppose the user wants to minimize the screen according to the human position from the camera view. In that case, it can give the bounding box coordinates, which contain the foreground object. If these coordinates are not given, the method will minimize the background area according to the view from the camera.

The first block calculates the coordinates within which the area visible in the camera is encountered. Aspect ratio is calculated from the images' width and height, and the method `findPoints` obtains the coordinates of the points B, C, E, F from Figure 6 using the respective formulas.

The next block determines the colour that the background needs to have. First, the colour that is most frequent in the image is determined. Next, the margin is added to red, green, and blue channels, so the colour becomes slightly different from the original.

The final block relates to the method for determining the chroma-key area if the bounding box is given. The user should give the left-bottom and top-right points of the bounding box in terms of the size of the image. Method `convertToImagePlane` converts image coordinates to image plane coordinates, using formula

$$P = \begin{vmatrix} x_{plane} \\ y_{planeLeft} + w_{plane} \cdot \dfrac{y_{point}}{w_{box}} \\ z_{planeBottom} + h_{plane} \cdot \dfrac{z_{point}}{h_{box}} \end{vmatrix}$$

where x_{plane} - x coordinate of image plane, $y_{planeLeft}$ and $z_{planeBottom}$ - y and z coordinates of the bottom-left point of the image plane, w_{plane} and h_{plane} - width and height of the image plane, w_{box} and h_{box} - width and height the of bounding box, and y_{point} and y_{point} are y and z coordinates of either top-right or bottom-left point, depending on which point you are calculating. The points correspond to the vertices of the inner blue rectangle in Figure 8.

Method `chromaKeyPoints` finds the projection of coordinates, found in the previous method, on the screen plane. It uses the equations for calculating 3D coordinates from the previous section but using a different angle. The tangent of that angle is calculated with $tan = \dfrac{cameraCoordinate - boxCoordinate}{focalLength}$ in both vertical and horizontal directions.

Moreover, the code contains some drawing rays and plains methods, but they do not influence the results and are created just for visualization.

All these methods were combined to create a proper matting technique. Even though the difference matting method is enough in most cases, there are cases where chroma-key screen behind the actor is still necessary to extract the actor. The following section shows the results that are obtained with this combined technique.

4 Setup and results

This section presents the results of the matting technique that is discussed in the paper. Two types of images are tested: virtual images are created by combining on computer some random background photo with random foreground object; real-life images are created by a real camera with a screen as a background.

4.1 Virtual image creation

The virtual images are created using the simple background image and the foreground object from some other photo. The foreground element is created using a manual selection. Figure 9 shows the result of background subtraction using the created technique. In this example, the difference method works correctly, and there is no need for changing the background.

Returning to the example from Figure 4, it needs the background change. In Figure 10, you can see that the background behind the bird was changed from white to gray. After that, we put the bird back to the image and used the difference matting method to obtain the bird. The result became much better than it was before applying the new background.

Figure 9: Upper-left image: the image with the background. Upper-right image: the image with both the foreground and the background. Bottom-left image: mask of the image. Bottom-right image: image difference.

Figure 10: Upper-left image: the image with the background. Upper-right image: the image with both the foreground and the background. Bottom-left image: mask of the image. Bottom-right image: image difference.

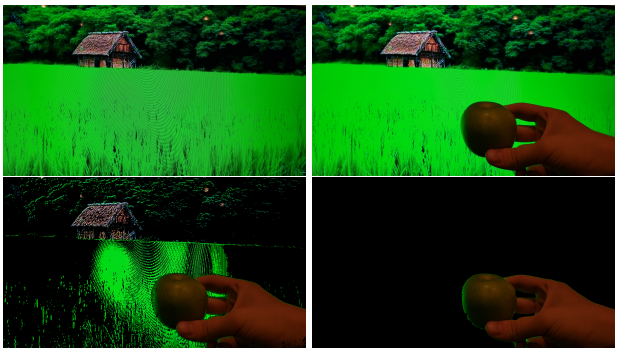

Figure 11: Upper-left image: the image with the background. Upper-right image: the image with both the foreground and the background. Bottom-left image: image difference with baseline approach. Bottom-right: image difference with created technique.

Figure 12: Upper-left image: the image with the background. Upper-right image: the image with both the foreground and the background. Bottom image: image difference with created technique.

4.2 Real-life setup

For the real-life image, the following setup has been used: Samsung Galaxy S9 camera with angle = 77°, camera aspect ratio = 16/9, camera coordinates = (0, 0, 0.89), screen plane point = (1.02, 0, 0.89). All the coordinates were measured with tape and are in meters. Figure 11 shows the result of applying the technique on the foreground object with the background screen. As can be seen, the baseline solution did not give any acceptable results, deleting only a small amount of pixels. The improved technique performs much better, giving only the foreground object.

A small change to setup was made to check the second part of the pipeline: the new camera coordinates are (0, 0, 1.1), and the new screen point is (0.52, 0, 1.1). As a foreground, we have chosen the notepad with an image of the same colour as the background, see Figure 12.

As can be seen, the method cannot correctly detect the foreground object because the colours in the foreground are almost identical to the colours in the background. Therefore, we pass on to the background changing technique.

For the image of size 1008 * 567, we take a bounding box with lower-left point = (302, 0) and upper-right point = (1008, 380). Using formulas for calculating intersections of camera

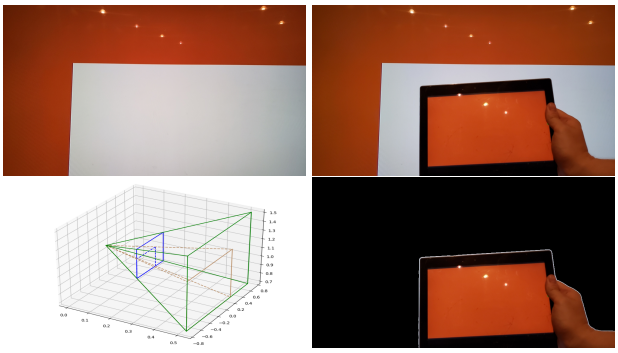

Figure 13: Upper-left image: the new background. Upper-right image: the new background with the object. Bottom-left image: plot of the area required. Bottom-right: result with chroma-key.

rays with the screen plane, we obtain the points on the screen plane: lower-left point = (0.52, 0.29, 0.686) and upper-right point = (0.52, 0.73, 1.24). Further, we cover the region enclosed by those points with a chroma-key of another colour. Figure 13 shows the new background together with the final matting result. Compared to the image-difference method, the result is much better: the foreground object is entirely in the image, and the hand does not have any artifacts.

Overall, 20 images were analyzed with the given method. We considered the following measure to quantify the results: we manually cut the foreground from all the images. After that, we calculated the number of pixels that differ between that ground truth and the result, calculated by the algorithm. As the measure, we took the percentage of that difference to the total number of pixels in the image. Moreover, we calculated that percentage for the baseline technique.

Out of those 20, 14 gave almost perfect results with the average percentage difference equal to 0.73%, while the baseline method gave the difference of 14.4%. Three images had some extra pixels or were missing some significant number of pixels; the average percentage difference was 2.03% using the developed technique and 27.5% using the baseline method. The last three images failed to detect the foreground object or remove a sufficient amount of background pixels. Their percentage difference was 10.09%. All of them had similar colours in the foreground and the background. Nevertheless, these 3 cases were fixed with the changed background, resulting in a 1.24% difference.

4.3 Limitations

Even though the created technique shows promising results and improves upon the baseline method, it has some disadvantages discussed in this subsection.

At the moment, the algorithm is highly dependent on a suitable threshold value. One of the steps of the algorithm is to pick the pixels where the difference between the original image and the background image is more than the threshold. This threshold highly influences the results. For example, in Figure 10, the threshold value is equal to 1, whereas, for the image in Figure 11, the threshold is equal to 32. There is no universally good threshold for all the images, and the algorithm to determine the best for any single example was not implemented. The possibility to change the threshold is passed to the user, with the default value set to 12.

There is a border around the foreground object at almost every result image, especially for virtually created images. The algorithm detects these borders to be of different colours in original and background images. However, in the film-making setup, it is assumed that the background used has the same colours as the background in the final version of the film. Therefore, having the one-pixel borderline should not be visible when the extracted image is inserted into the final film version.

Orthogonality assumption is another limitation. The camera plane and screen plane should be orthogonal to the same line for the minimization scheme to work. This is the strong assumption, which does not let to use the method for the situations where the camera does not look straight on the screen plane.

5 Responsible Research

For the results to be meaning full, they should be reproducible by anyone interested in the work. To reproduce the part with virtually created images, you need to combine two photos into one, which can be done in Photoshop. Further, the real-life setup can be easily reproduced. For that, you only need a camera with the know FOV and aspect ratio and a background screen. Putting the camera in some position and measuring the distance to the screen is trivial. The camera should remain static, so the foreground and the background images do not differ too much. You can make the shot for the background image with that info and then make the image with the foreground object in it. If the background solid colour screen is required, you will have to find something of the same colour as the one determined by the program.

The code to observe the algorithms is available on Gitlab: https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-60/rp-group-60-alexsimonov. We created all the results using this code. To obtain the best results for different images, you might want to change the threshold parameter cause the algorithm is highly dependent on it. Increasing that parameter will decrease the amount of non-black pixels in the result images. Decreasing the threshold will return some parts to the image.

The ethical component should not be forgotten. Therefore, possible malicious use of the application is considered. The proposed technique can be used to create fake images or fake videos. For example, the malicious user can find the video containing some person, extract the person from the video, and use this extracted part to create a fake video. This process will undoubtedly harm the one present in the video.

6 Conclusions and Future Work

Matting the scene is essential in image processing and finding a different approach for situations can improve the resulting quality. This paper presented the research on two different ways of matting: matting with the known background and minimizing the required background solid colour screen if matting with the known background fails.

Researching difference matting technique led to the background subtraction base technique, with some existing solutions. Further, using different image processing techniques, such as blurring and contour finding, helped improve baseline technique results. Even though the method worked in most cases, some images caused it to fail. Nevertheless, with the knowledge from pinhole camera theory, the method for minimizing the background solid colour screen was created. After applying this method to the failed cases, all of those cases succeeded. Therefore, this combined technique solves the central question of the report.

Nevertheless, there is always a place for improvement. Further research can combine existing background subtraction methods with some parts of the solution proposed in this paper, leading to further improvements of results. Moreover, a more human-friendly environment for using the method may be created. It was out of the scope of this research but can still improve the usability of the method. Finally, some techniques for threshold value optimization or dominant colour findings might be added to the current solution.

References

[1] Hiroki Agata, Atsushi Yamashita, and Toru Kaneko. Region extraction using chroma key with a checker pattern background. *Faculty of Engineering, Shizuoka University, Japan*, 2006.

[2] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Benjamin Bratt. *Rotoscoping*. Taylor & Francis, 2012.

[4] Yung-Yu Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski. Video matting of complex scenes. *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002.

[5] Devin Coldewey. How 'the mandalorian' and ilm invisibly reinvented film and tv production, Feb 2020.

[6] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[7] Miss Helly M Desai and Mr. Vaibhav Gandhi. A survey: Background subtraction techniques. *International Journal of Scientific Engineering Research*, 5(12), 12 2014.

[8] Estevão S. Gedraite and Murielle Hadad. Investigation on the effect of a gaussian blur in image filtering and segmentation. In *Proceedings ELMAR-2011*, pages 393–396, 2011.

[9] Srivatsa Prativadibhayankaram, Huynh Van Luong, Ha Le, and André Kaup. Compressive online video background–foreground separation using multiple prior information and optical flow. *Journal of Imaging*, 4:90, 07 2018.

[10] Jean-Colas Prunier. Learn computer graphics from scratch. https://www.scratchapixel.com/, 2009. [Online].

[11] R. Sawdon Smith, A. Fox, and M. Langford. *Langford's Basic Photography: The Guide for Serious Photographers*, volume 9. Routledge, 2010.

[12] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, 1999.

[13] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.*, 30:32–46, 1985.