Moving Object Removal in Video Using OpenCV and Python

Yueh Lin, Tsou
California State Polytechnic University, Pomona, 3801 West Temple Avenue Pomona, CA 91768
ytsou@cpp.edu

Abstract

Today, removing objects in photos is very common. However, image processing not only includes pictures and photos but also videos. People produce a variety of videos every day, and therefore, video processing becomes very important, whether it pertains to TV shows, movies, or even records of daily life. This project attempts to separate and remove moving objects from video, and is based on Kalman filtering, background subtraction, and image inpainting. This project presents another method to inpaint the frame image during the process. In the paper which I implement [1], they use exemplar-based texture synthesis for image completion [4]. I present another method that uses texture synthesis to repair the image by considering image information from other frames.

Keywords: video processing, object removal, image inpainting, background subtraction, kalman filter, texture synthesis, temporality image.

1. Introduction

Removing undesired objects or people in the video is one attractive research topic in the field of computer vision. And it also a common task in professional video and movie production. For instance, sometimes in the process of filming, there will inevitably be unnecessary interference, like during landscape shooting and someone walks pass in front of the camera, or in costume dramas, and some objects appear that don't belong in the era of the film. Both of these situations require the removing of some moving objects that accidentally appear in the film. In other words, the film crew needs to remove these objects during post-processing. Therefore they need "Moving Object Removal in Video" such method to achieve this.

This project presents the method of using the Kalman Filter and background subtraction to separate foreground and background. The Kalman filter is a well-known algorithm for motion segmentation and it is ideal for a system that is continuously changing. They have the advantage of being light on memory, and also being recursive, so that new measurements can be processed as they arrive. Hence, they are very fast, making them well suited for real-time problems. Background subtraction is used when approaching and detecting moving objects in videos shot from static cameras. Its concept is detecting the moving objects from the difference between the current frame and a reference frame, which is like the "background image", to separate that object from the background. This paper merges these two methods to separate moving object from the background.

Once a moving object is separated and removed from the background, we can use image inpainting or image and the video completion algorithm to repair the background in the removed foreground area. In this paper, they use exemplear-based texture synthesis for image completion [4]. This method fills the pixels of the hole based on their assigned priorities. In this project, I constructed temporality image before repairing the image with texture synthesis. By using this method, the program repair the defective image by separating the background information from the other frames. This is because the background occluded by a person at a given frame might be revealed in another frame as the target moves. The occluded region is visible in other frames. Hence, we can use temporality information to repair the image that might more correctly repair the image than only consider the information on one image.

In this project, I used Dlib library to track the objects which I wanted to remove instead of Kalman Filter. This paper will still describe how to use the Kalman filter as a tracking method, as it is used in the paper [1]. Dlib is a library consisting of machine learning, computer vision, image processing, data mining…This library is used in Robot, Embedded System, mobile phone, even in the large computing construction, and can also run on different operating systems.

My result successfully separates the moving foreground and static background. I created a mask for the object which I want to remove from the video, then output every frame without the object in order to construct the temporality image. My first method to repair the image only used one background information to repair the image hole. This method was faster but not a very good option because the background image will change when the time goes forward. The second method used exemplar-based texture synthesis to repair the hole which is introduce in the paper. The third method used texture synthesis to repair the hole in the temporality image. This method is hard to compete because with the program that I implemented took lots of time to

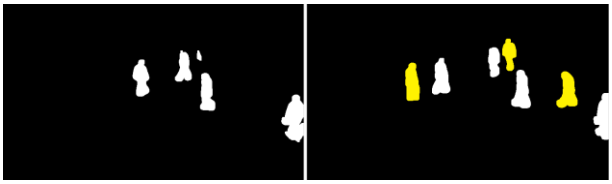
even complete one image. Hence I did the process in regions that only had moving targets, in order to reduce the running time.

2. Problem Description

During the process, there are some problems that produce a result of not such high quality.

2.1 Background model problem:

This paper used the first frame to be their background model. However if the moving object is already in the first frame of the video, we have to create a background image which removes all the moving objects, because this image will be used to restore the background in the removed foreground area, and is also used for background subtraction. While doing background subtraction, this project used the background image, and compared it to each frame. If we don't remove the object from the first frame, the result will be like [figure 2], compared with [figure 1], which results in the background image that doesn't reveal any moving objects. In this paper [1] the result it presents is a video with the first frame that doesn't have any moving object in it. This means that they can use the first frame to be the background image and use it to complete the video directly.



[Figure 1] ***[Figure 2]***

Figure 1: Video processing with the background image with the objects removed.

Figure2: Video processing with the background image that already has moving objects in it without being removed. The yellow part is the first frame objects that still appear in the next frame in doing background subtraction.

2.2 Noise with background subtraction:

In the process of implementing this paper [1], after removing the object from the video, the result will have some noise around the edge of the object. In order to improve the accuracy of background subtraction, this project uses erode and inflate function to remove the noise after doing the background subtraction. This method is used to strengthen the image's shape features, like edge and connected area. While doing background subtraction, the background and foreground are separated by setting threshold, setting the background to black, and moving the object into white.

2.3 Object tracking:

This paper [1] used the Kalman filter to track the moving object and also to combine with background subtraction to separate the foreground from the background. In this project, I didn't implement this method to track the object. Hence, in this report, I used the object tracking method with Dlib library to complete the work.

2.4 Program running time:

In the beginning of the project, the running speed of the program was very slow, because it had to process all the frame image pixels at a time. Hence, it took a lot of time to run the algorithm, but most of the calculation was really unnecessary. In order to reduce the program running time and avoid the unnecessary calculation, I converted the image to gray-scale, because the color image and gray image will have the same result if we use the background subtraction algorithm. Also, do the background subtraction on the gray-scale image can reduce the memory used as well as the running time. Then I would use a bounding box to select the object that we want to remove. After selecting the object, the region that we need to process is already inside the bounding box, not the whole frame image because we don't have to change anything about the other objects or the background. This method largely reduces the running time of the program.

2.5 Object overlapping:

If object A, which we want to remove overlaps to another moving object B, in this scenario some part of the object B will missing (figure 3), and the result will be incomplete. To address this issue, there is a research paper [3] "Background Inpainting for Videos with Dynamic Objects and a Free-moving Camera" that solves this kind of problem. They used different masks on the dynamic object, a mask marking the object to be removed, and another mask marking the dynamic objects that are to remain in the scene.

(a) **(b)**

Figure 3: Image (a) and (b) display objects we want to remove that overlap with other objects, which means that some part of the other object will be missing. In Image (a) it's obvious that some of the yellow part is missing.

3. Method for Object Removal

3.1 Create a background image

Remove all the moving objects which appear in the first frame (figure 4), then repair the frame image by using texture synthesis or background information from other frame images, in order to create an image that only has background. This background image can be used to do background subtraction and be the reference to repair the image after removing the target object from the video.

(a) **(b)**

Figure 4: Image (a) is the first frame of the video, and image (b) is the image with no moving object after removed the objects from image (a), it's used to repair the frame after remove the target object and do background subtraction.

3.2 Object tracking with Dlib library

This library has a tool for tracking moving objects in a video stream. We give it a bounding box for an object in the first frame and it attempts to track the object in the box from frame to frame. This tool is an implementation of the method described in the paper: Danelljan, Martin, et al. "Accurate scale estimation for robust visual tracking." Which proceedings of the British Machine Vision Conference in 2014. They use discriminative correlation filters for multidimensional features and a method for joint translation-scale tracking based on learning a 3-dimensional scale space correlation filter.

Figure 5: Track the object that we want to remove with the bounding box, like the right image with the white bounding box. This bounding box also used to let the program know which area is going to be processed.

3.3 Kalman filter

In the paper, they suppose that the system is linear and its initial conditions are given. The notations are defined as follow:

$\hat{x}_{t-1}$: initiate state $\widehat{x'}_t$: predicted value

P_t : covariance matrix H_t: transformation matrix

K: Kalman gain P'_t: prediction error covariance

R_t: measurement error covariance

$\overrightarrow{z_t}$: measurement mean value

Kalman filter has two basic model, system model and measurement model. The system model is for state prediction and the measurement model is for update and correct the prediction by gathering the information.

In prediction step, kalman filter predicts the value of present state $\widehat{x'}_t$ from previous state $\hat{x}_{t-1}$:

$$\widehat{x'}_t = F_t \cdot \hat{x}_{t-1} \quad (1)$$

Then according (1), the covariance matrix in the prediction step will be:

$$P'_t = F_t \cdot P_{t-1} \cdot F_t^T \quad (2)$$

Consider external influence $B_t u_t$ and external uncertainty Q_t, the prediction equation will be:

$$\hat{x}_t = F_t \cdot \hat{x}_{t-1} + B_t u_t \quad (3)$$

$$P'_t = F_t \cdot P_{t-1} \cdot F_t^T + Q_t \quad (4)$$

In measurement step, maps the state parameters into the measurement domain by use matrix H_k.

$$\vec{\mu}_{expected} = H_t \cdot \hat{x}_t \quad (5)$$

Then refining the value of present state with measurements by combining prediction state and measurement result (figure 6). We will get K (the Kalman coefficients matrix). The coefficients are high if the measurement noise is low, and vice versa. In other words, the measured values which do not fit the actual system behavior get small weights [1]. K is computed as follow:

$$K = H_t \cdot P_t \cdot H_t^T (H_t \cdot P_t \cdot H_t^T + R_k)^{-1} \quad (6)$$

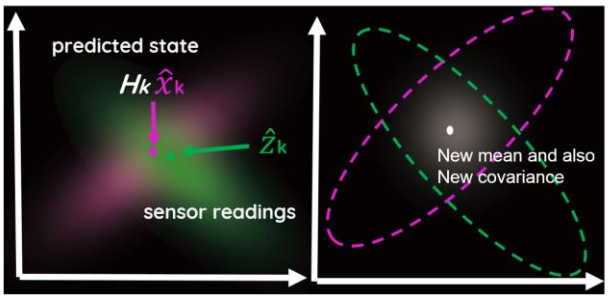


Figure 6: Combining prediction state and measurement result to correct and update the prediction

Complete equations for the update step. $\widehat{x'}$t is our new best estimate, and we can go on and feed it (along with P't) back into another round of predict or update

$$\widehat{x'}\text{t} = \hat{x}\text{t} + K'(\overrightarrow{z_t} - H_t\,\hat{x}\text{t}) \tag{7}$$

$$P'\text{t} = P_t - K'H_tP_t \tag{8}$$

$$K' = P_t \cdot H_t^T (H_t \cdot P_t \cdot H_t^T + R_k)^{-1} \tag{9}$$

Using Kalman Filter to do foreground and background detection. They suppose that $\hat{x}$t (x, y) is the intensity of pixel (x,y) in image sequence t. The prediction and differentiation are presented by $\widehat{x'}$t (x, y) and $\widehat{\dot{x}'}$t (x, y). The matrix F_t has constant elements as used in [9].

$$F_t = \begin{bmatrix} 1 & 0.7 \\ 0 & 0.7 \end{bmatrix} \tag{10}$$

And the measurement matrix is also constant

$$Z = [1 \quad 0] \tag{11}$$

Then they use a matrix m_{t-1}(x, y) to represents pixel (x, y) at frame t as foreground or background.

In this section, Kalman filtering was used to separate foreground from background. But if the foregrounds are moving objects with no-continuous movement, Kalman filter cannot determine foreground from background accurately, especially, when the object does not move [1]. Hence, this paper combined background subtraction to improve foreground separation.

3.4 Background subtraction.

Background subtraction is used for generating a foreground mask (figure 7) by using static cameras. It calculates the foreground mask by performing a subtraction between the current frame and a background model. In the paper, they suppose that there is at least one frame without the given foregrounds in the set of frames. Such frame is applied as the background model.

This method computes differences between each frame F_t and background model (B) then set thresholds to separate the moving object from the background. Two thresholds, they are applied in difference intensity and direction.

If $|I_t - I_B| < I_{low}$, $I_t = 0$
If $|I_t - I_B| > I_{high}$, $I_t = 255$

If $\cos\theta < T_{color}$, $I_t = 0$
If $\cos\theta > T_{color}$, $I_t = 255$

The direction of color is defined as:

$$\cos\theta = \frac{|C_t \cdot C_B|}{\|C_t\| \cdot \|C_B\|} \tag{12}$$

Figure 7: The left figure is the image which separated background and foreground from the right image. The background is black and the moving object is set to be white.

3.5 Noise removal

The mask for the moving object will appear some noise (figure 8 (a)), these noises will cause bad result and the program has to spend more time to repair these regions. In order to reduce the unnecessary process and running time, I erode the mask image to remove the noise then inflate the image back to the original size.

First, erodes the source image by using the specified structuring element that determines the shape of a pixel neighborhood over which the minimum is taken:

$$\mathtt{dst}(x,y) = \min_{(x',y'):\,\mathtt{element}(x',y')\neq 0} \mathtt{src}(x+x', y+y')$$

For two set I and H, I is our target object and H is structuring element, like a filter move over all the pixels in I. Suppose we want to erode I by H, it will be define as:

$$I \ominus H \tag{13}$$

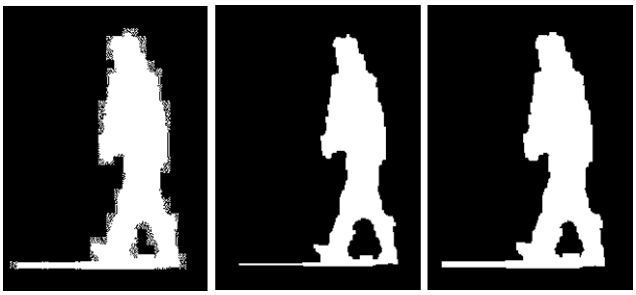
If H is smaller than I, our target will been erode, if H is larger than I, our target will been removed from the image. Hence, we have to decide a size of H to remove the noise (figure 8 (b)).

Then, dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken:

$$\mathtt{dst}(x,y) = \max_{(x',y'):\,\mathtt{element}(x',y')\neq 0} \mathtt{src}(x+x', y+y')$$

Inflate I' which has been eroded in the previous function back into original size by H (figure 8 (c)), it will be define as:

$$I' \oplus H \tag{14}$$



(a) (b) (c)

Figure 8: Image (a) is the result of separate moving foreground and background, we can notice that there has some noise around the target. Image (b) is the result of eroding the image, compare with image (a) all the noses are removed. Image (c) is the result of inflating image (b) which return the target region back into the original size.

3.6 Remove the selected object.

Remove the selected object from the video with the mask which creates in the previous steps. Compare each pixel p(x, y) with the mask image, if the mask pixel value m_p(R, G, B) = (255, 255, 255) then remove the same position pixel on the image, set f_p(R, G, B) = (255, 255, 255). In this project, I output every frame images without the target object to be the data set for construct temporality images.

3.7 Temporality image construction

The purpose of constructing temporality images is to create correlation between the same background positions in different frames. Because the occluded region is visible in other frames, we can repair the defect images by using information from the other frame images. Used temporality image for image inpainting might more correctly then repair the image only consider the information from one image.

To create temporality images, we pick i^{th} column from each frame to construct the images (figure 9)

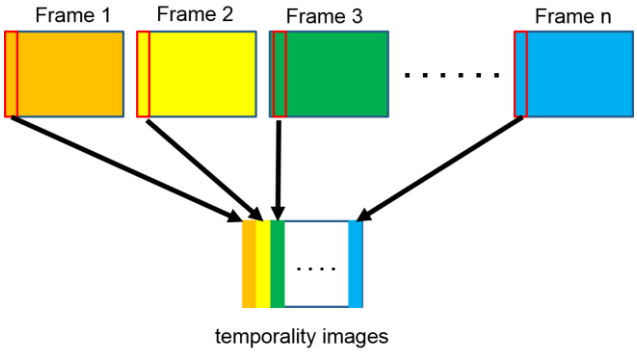


Figure 9. In this figure, we select and combine the first column from every frame images to be our first temporality image. The change is very noticable in this column, as time moves forward.

In this project, in order to reduce the running time, I construct the temporality image for the region only include the moving object. First, I found the leftmost, rightmost, top and bottom point of the object in the video (Figure 10). Then use these point information to restrict the process region (Figure 10). Because only the pixels which in this region will be occluded by the moving object and only this region need to be repaired, hence the program doesn't need to process the image outside this region.

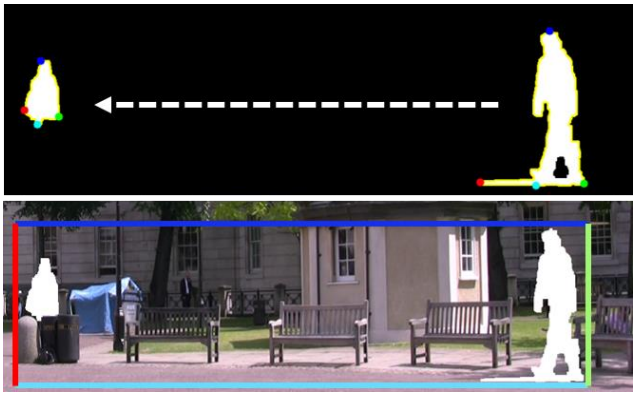


Figure10

3.8 Temporality image inpainting

In the paper [1], they use exemplar-based texture synthesis for image completion [4]. This method fills the pixels of hole based on their priorities. The priority is defined for each boundary pixel of hole as below:

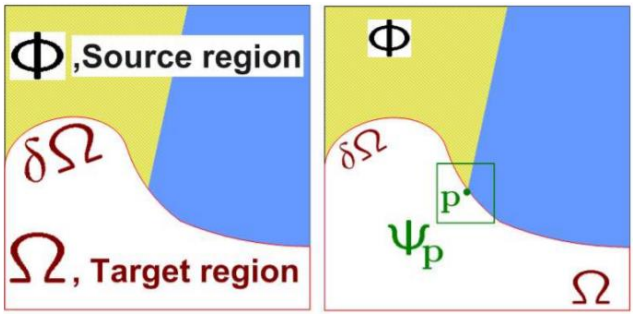
$$P(p) = C(p)\,D(p) \tag{15}$$

In this equation, C(p) is confidence term, it calculates the proportion of the source region which in Ψp. The high proportion means the less region to repair, hence the window Ψp with less target region will have higher repair priority. D(p) is data term, a function of the strength of isophotes (direction and intensity) hitting the front δΩ at each iteration. and they are defined as follows:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (\mathcal{I}-\Omega)} C(q)}{|\Psi_p|} \tag{16}$$

$$D(p) = \frac{\left|\nabla I_p^{\perp} \cdot n_p\right|}{\alpha} \tag{17}$$

The process shows in (figure 11) and (figure 12).

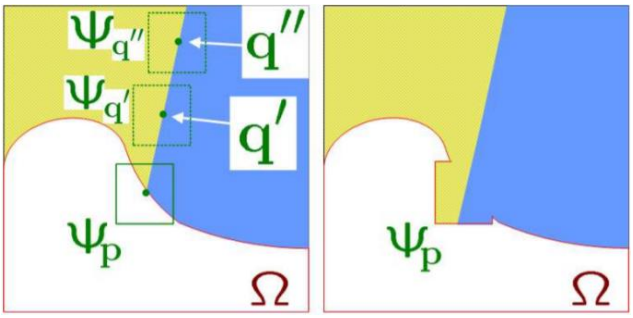


(a) (b)

Figure 11. (a) Image with the target region Ω which is the hole on the image, its contour δΩ, and the source region Φ which is our image region that doesn't be removed. (b) The area where we want to synthesis are delimited by the patch Ψp which centred on the point p on the contour δΩ.

After decide the priority of the repair order, search the patch Ψq in source region for repair the target region. The difference between two generic patches Ψp and Ψq is defined as the sum of squared differences. The most similar patch Ψq is defined as:

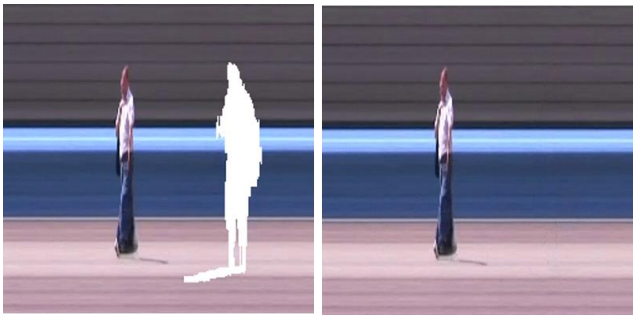
$$\Psi_q = \arg\ min_{\Psi_q \in \Phi}\, d(\Psi_p, \Psi_q) \tag{18}$$



(a) (b)

Figure 12. These images show how to find the best patch Ψq to repair the target image depend on the equation (18)

In this project, I use texture synthesis to repair temporality image. With the program that I used successfully repair the image with the hole (Figure 13)



(a) (b)

Figure 13. (a) A temporality image which was created in the previous steps, the white region is the hole where the object has been removed. (b) After repairing the image by using texture synthesis the white region has been repaired with the temporality pixel information around it.

3.9 Video reconstruction

The last step of this project is reconstructing the frame images by using temporality images which have been repaired by texture synthesis. We return every column in the temporality image back into each frame to reconstruct the frame images then output these frames as a video sequence (figure 14).

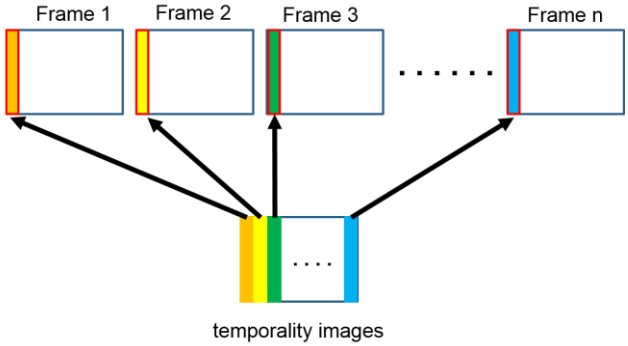


Figure 14: Return every column in the temporality image back into each frame to reconstruct the frame images.

4. Result

4.1 The result by following the concept of the research paper is as follow:

(a) The concept of the research paper: Step1. Use Kalman filter and background subtraction to separate foreground and background. Step2. Remove the object from the video. Step 3. Repair the image by using texture synthesis.

(b) My method in this project: Step 1. Create a background image (figure 4). Step 2. Use a bounding box to select the object. This step not only for the object selection but also for reduce the running time and avoid the unnecessary process (figure 5). Step 3. Use Dlib library and background subtraction to separate foreground and background (figure 15 (a)). Step 4. Remove the selected object. Step 5. Inpaint the frame image (figure 15 (b)) by using (A) only background information (B) apply texture synthesis to repair each frame images (C) construct temporality image then repair by texture synthesis.

(a) (b)

Figure 15. **Background Subtraction**: Image (a) which is the result of separating the background and foreground. Image (b) is the result that removed the undesired object from the video then repair the frame image by using only background information.

4.1.1 Compare two different methods for image repair-
(1) Image is repaired by only one background information
(2) Image is repaired by using texture synthesis

(a) (b)

Figure 16. The red rectangle is the repair region. Image (a) is repaired by the background image and Image (b) is repaired by using texture synthesis. Both methods work well in this video because the background in this video doesn't change a lot.

4.2 The third method for image inpainting.

In this project, the third method that I apply for image inpainting is to construct temporality images then use texture synthesis to repair the frame images. During the project, I found that the background which is occluded by the moving object will appear in other frame images. So we can use this temporality information to repair the images not only consider the information on one image. By using temporality images we also can know the information about how background changes in the video, hence we also can use this information to adjust our result.

First construct temporality image for each frame columns (figure 17) then create a mask for each temporality image to be the input of texture synthesis (figure 18).

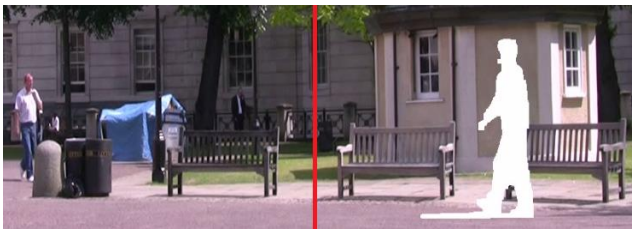
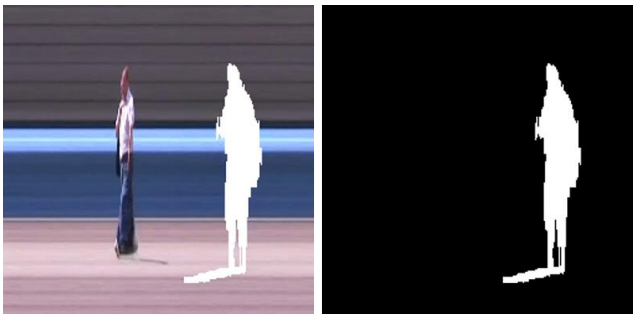


Figure 17. Construct temporality image for the red column, the result shows in *Figure 18*.



(a) (b)

Figure 18. Image (a) is a temporality image and image (b) is the mask for image (a) which will be used to do the texture synthesis.

4.2.1 The result of texture synthesis

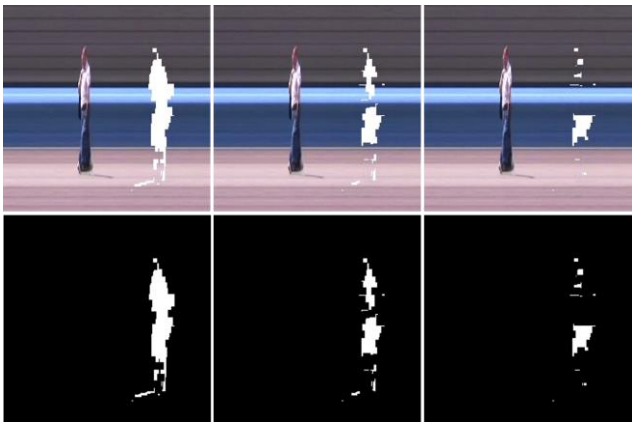


Figure 19. **Temporality Image and Texture Synthesis**: These images show how the program repair the temporality image. The repair order is from left to right, the program calculates the priority for repair the image depends on the equation P(p) = C(p) * D(p), P is the priority, C is confidence term and D is data term. In the first row of the images, there is a clear line in the middle of the image, hence the target region (repair region) which on the line will have higher priority to be repaired like the above show.

4.3 Reconstruct the video

The last step of this project is reconstructing the frame images by using temporality images which have been repaired by texture synthesis. In this project, I use a video only has 10 seconds, and separate all the step in different programs. However the running time of the repairing process still very long, the program has to take almost three hours to repair one image and there have total 862 temporality images, It has to take about 108 days to complete the work. Hence, it is hard to complete the project, especially if the video is very long or with high resolution.

4.4 Compare the difference between the research paper and my result.

First of all, in the research paper, they suppose that there has a frame image without moving object, they use this image to be their background model. However, we cannot guarantee that every video will have such frame image. Hence, I add a step to create a background image in the process.

Second, there have many different methods to separate the moving foreground from the background. For static background video, background subtraction is good enough to separate foreground and background, hence I only apply background subtraction and used Dlib library for object tracking. I didn't implement Kalman filter in this project because after I understood this method it still hard for me to implement.

Third, for image inpainting, I applied three different methods to repair the frame image after removing the object. Method (a) only depend on one background image: this method is faster than other two methods because it doesn't have much calculation during the process. But it only can be used with the video which background color and brightness doesn't change a lot. Method (b) apply texture synthesis to repair every frame image is the method which is used in this research paper. This method only considers the pixel information from only one frame, hence I introduce the third method (c) construct temporality image for texture synthesis, this method considers the pixel information from every frame and select the best patch to repair the hole. But Method (c) have to spend a lot of time to complete the result video.

4.5 Data usage and time spending

In the first and second image inpainting methods, they didn't create much image data and didn't have serious time spending problem. For the third image inpainting method, I separate the process into three different programs (1) generate images after removing the object (2) construct temporality image (3) texture synthesis. I used a video which has 251 frames, so in the first step, it will create 251 images without the moving object. Then the size of each frame is 1440*1056, hence in the second step will create 1440 temporality image. But I do some video processing as I show in "figure 10" to reduce the amount of the temporality images to 863. On the other hand, the running time to create a temporality image is about 30 second per image, so it has to take about 7 hours to complete the step. In the last step for texture synthesis, it has to take about 3 hours to repair an image and the total time spending will be 2589 hours. This step has to take too much time and the hardware of my laptop is not that strong to support this kind of calculation, I just do several frame images to show the result.

5. Conclusions and future work

5.1 Conclusions

First of all, in this project, I took a lot of time to understand a well-known method – Kalman Filter. It's very useful to understand this method because it can be used in many different fields and it also has many different kinds of application. During this project, I also know how to use background subtraction to separate the moving object and create a good mask for the target by setting the color boundary and analysis the histogram.

Second, I learn three kinds of object tracking method during the process, the Kalman filter, Camshift, and Dlib library. Dlib library is easier to apply than Kalman filter and

its result is better than Camshift. Camshift is a method that tracks the object base on the color histogram model, so it doesn't work very well for tracking the people in the video or the video with complex frame images.

Finally, I understood how to use texture synthesis for image inpainting, then apply this method to repair different kinds of image like frame images and temporality images. And compare the difference repair result between texture synthesis and repair images only use background model. They both work well in the video which background doesn't change a lot, but if the color or brightness change in the video background the method which only uses background model will not produce a result of such high quality.

5.2 Future work

(1) Free moving camera

Remove the moving object from the video which has moving background. The works in this project were based on the static camera. Then we can try to remove the moving object with scene-parallel-moving or free moving camera. In paper [3] they provide a method for removing the moving object with the free moving camera.

(2) Inconsistent Results

Some method will produce inconsistent results, like the result shows in figure 20, in the red region. We can correct this problem in the future.

Figure 20

(3) Moving objects overlap and image repair

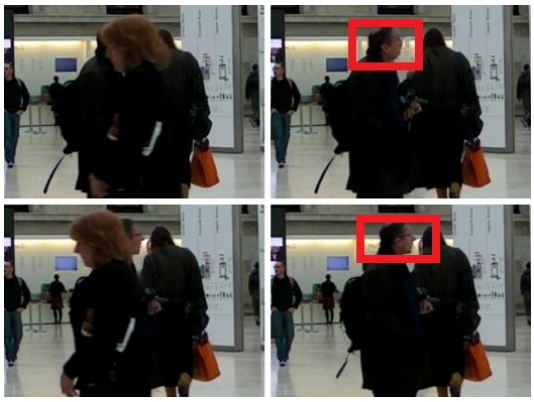


Figure 21: This result is from research paper [2]. They want to remove the foremost person from this video, both the dynamic scene elements and the background behind it need to be restored. The right-hand-side of each frame pair shows the inpainted result. But the result still missing some parts of the person behind the target object.

References

[1] S. Kamel, H. Ebrahimnezhad, A. Ebrahimi, "Moving Object Removal in Video Sequence and Background Restoration using Kalman Filter", 2008

[2] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, C. Theobalt, "How Not to Be Seen - Object Removal from Videos of Crowded Scenes", Eurographics, 2012.

[3] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, C. Theobalt, "Background Inpainting for Videos with Dynamic Objects and a Free-moving Camera", ECCV, 2012.

[4] A. Criminisi, P. Perez, and K. Toyama, "Object Removal by ExemplarBased Inpainting," Proc. IEEE Conf. Computer Vision and pattern Recognition, vol.2, pp. 721-728, Jun. 2003.

Link for the Code and Data:
https://drive.google.com/drive/folders/1r_fLDqiBuamrgyDBt9m9NJChwbNNZFGo?usp=sharing