CHALMERS UNIVERSITY OF TECHNOLOGY

RRY025 Image Processing

Project6 Report

Yankun Xu (940630-0237)

Jiazhi Xu (931027-1292)

2017-10-12

1 Introduction

With the advent of the digital age, many valuable archival pictures, ancient calligraphy, old movies, etc. are digitalized and stored in the computer. However, in the preservation process, due to some natural or unnatural factors, the image is often prone to some defects.

In this project, the picture is from *http://fy.chalmers.se/∼romeo/RRY025/mfiles/film1_big.jpg*. The picture is shown in Figure 1. The first column is original picture and the middle column is the picture after color correction with many obvious scratches, and the third column is hopeful picture after scratch removal. To detect the scratch, we compare the difference between frames on different color channels (Red, Green, Blue), and different filters are tested to remove the scratch.

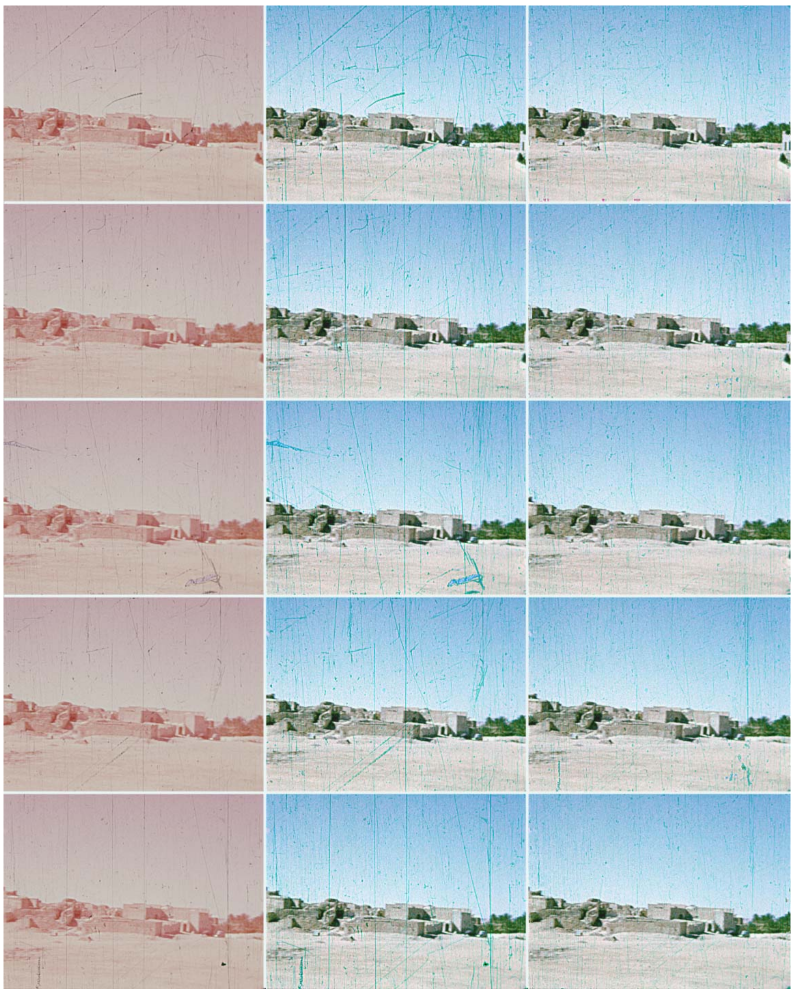


Figure 1: The original 'film1_big.jpg'

2 Detect and remove scratches

The image is loaded and the middle column is cut to begin with, due to there are some white blank between frames, we cut four pixels for all sides of each frame. Three color channels are plotted in Figure 2, we could find that the red channel is more affected by the scratches.

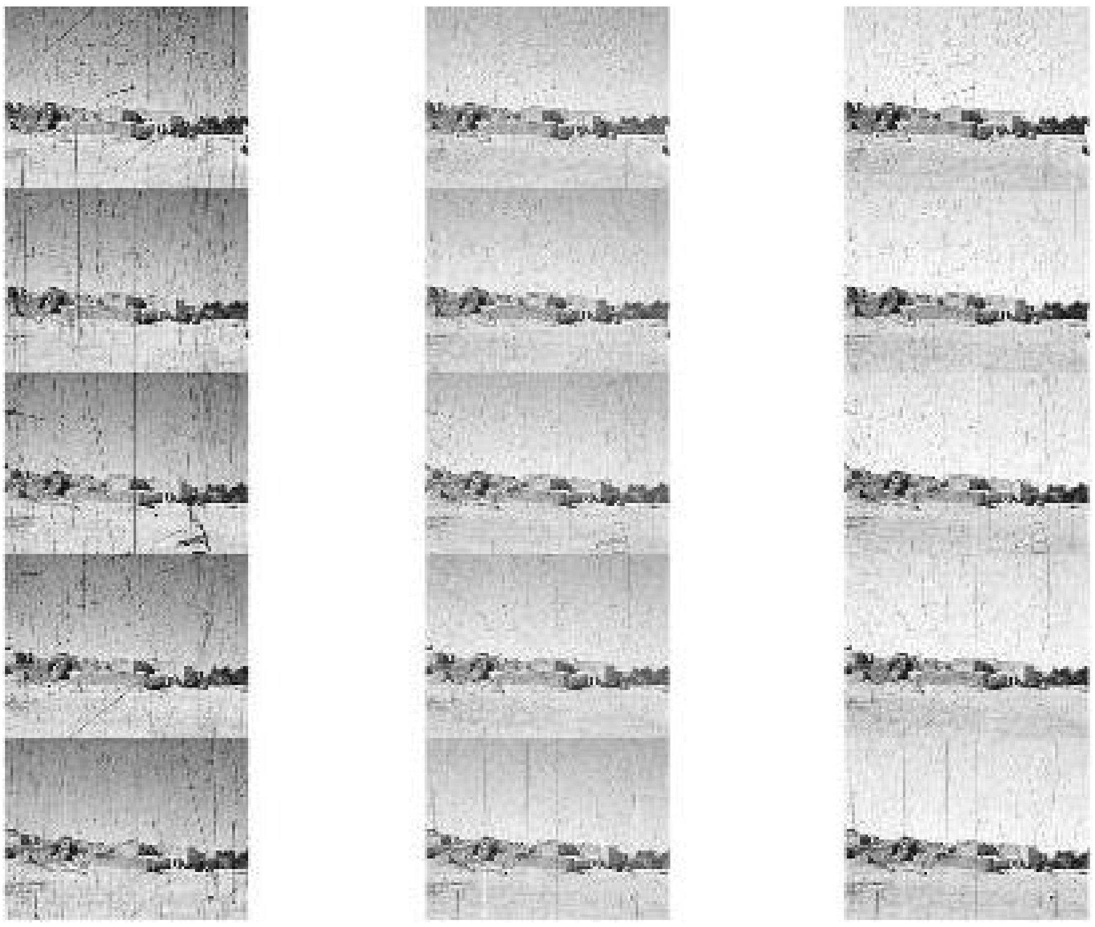
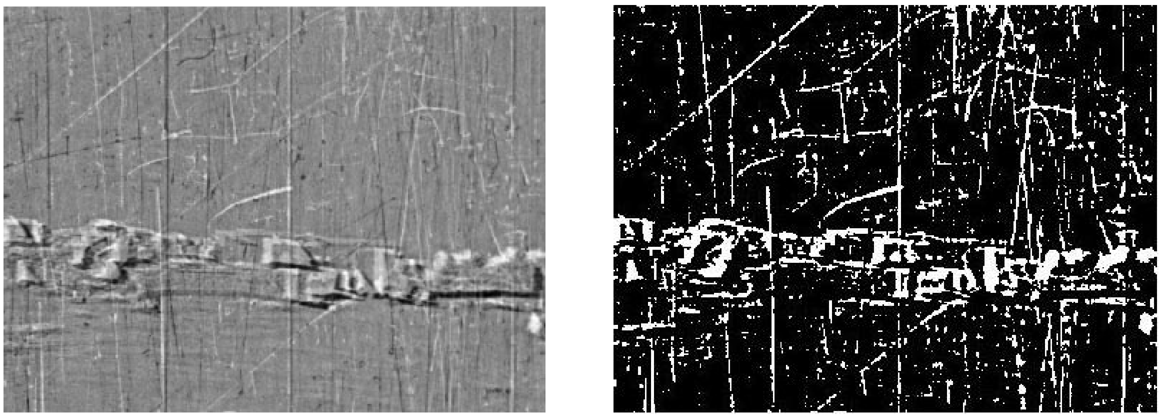


Figure 2: The middle column on three channels(R,G,B)

We are not very sure about the reason why red channel is affected by scratches more, Linus and Renaud from other groups suggest that it should be caused by color correction, original frame looks more red meaning original frame has more red component and scratches exist in all channel, so red channel looks being affected more after color correction. But there is still a question, color correction just express the property of red channel obviously, it doesn't explain why red channel is affected more! Actually, we has searched some materials about it, although we are not sure, we think that in spectrum red light has lowest frequency and highest wavelength among three channels, and these scratches are high frequency part in the picture, intuitively channel with lower frequency would be affected more by noise(scratches). We also search some theory about principle of camera capture RGB image, red photons has lowest energy because energy is proportion to frequency, camera use Bayer filter to capture RGB three channels photons, each channel should plot similar energy, so red channel will capture more photons, thus for the scratches, red channel would be affected more.

For each frame, scratches among three channels are located in the same position, scratches in red layer are more obvious though. Firstly, we should investigate the scratches, we could compare each neighboring frames in each channel, the scratches can be regraded as random distributions in each frame, meanwhile the landscape in the frame is nearly same, intuitively we could make difference between neighboring frames, let's take difference between 1st and 2nd frame in red channel(if you want to get scratches in 1st frame, you should take 2nd frame minus 1st frame), which is shown as Figure 3a. We cannot see anything in this shitty difference. How about taking binary of the difference? Look at Figure 3b, it looks better, in this binary image, almost thin vertical or horizontal white areas are scratches, it is what we want to remove, but meanwhile there are also many larger pieces in the middle area, that is slight movement among the frame sequences which is not we want to pick up to remove.

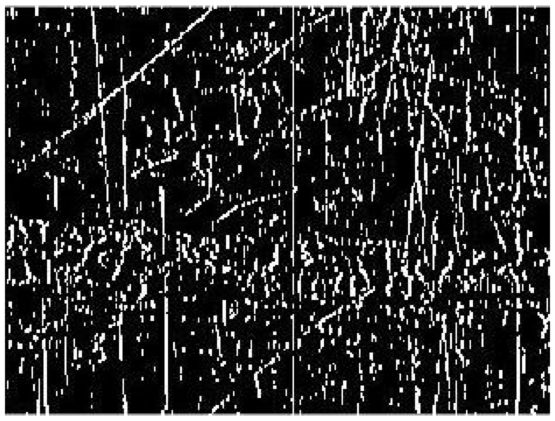


(a) Difference between 1st and 2rd frame **(b)** Binarized difference with level 0.1

Figure 3: Difference and binarized difference between 1st and 2rd frame in red channel

The different shape between scratches and landscape motion inspires us that these scratches could be considered as edges, then we remind that the method of edge detection of Coke in studio exercise maybe useful for this task. We build up sbv = [1 0 -1; 2 0 -2; 1 0 -1] filter to detect vertical edges(scratches) and sbh = [1 2 1; 0 0 0; -1 -2 -1] filter to detect horizontal edges(scratches). The binarized result for Figure 3b after filtering is shown in Figure 4. Compared to Figure 3b, Figure 4a and 4b almost keep the vertical and horizontal scratches and larger pieces in landscape area becomes smaller thin edges, but we still cannot pick up all white pixel in Figure 4 as scratches we want to remove. However, we still can find some differences between real scratches and landscape area, white pixels of real vertical or horizontal scratches are almost connected and pixels in landscape area are separated and scattered.

We could use this property to detect the scratches, fortunately there is an awesome command in Matlab called *bwlabel*, which is to measure the connective area of a logical matrix, and binarized image is logical! There are so many connective areas in Figure 4a and 4b, and we could set a threshold of the length of these areas to detect scratches. If threshold is too small, we will also pick up many pixels in the landscape area as scratches, finally we set 20 as threshold after many trials. By the way, in the use of *bwlabel* command, we set 4-connective parameter in case of choosing pixels in landscape area.


(a) Figure 3a filtered by sbv

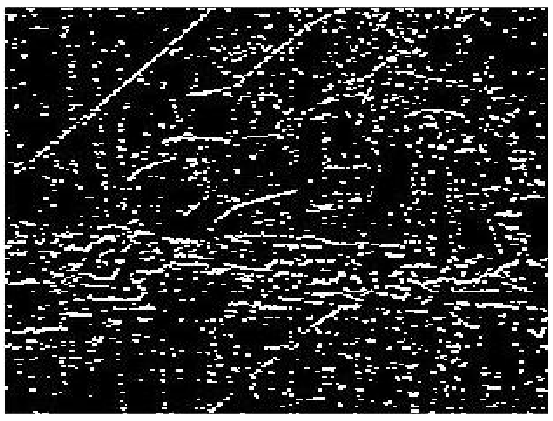

(b) Figure 3a filtered by sbh

Figure 4: Figure 3a after sbv and sbh filter

The next step is to replace these pixel values by filter thereby removing scratches. We have tried many filters, such as average filter, median filter, maximum filter, Gaussian filter, finally we pick up median filter with size $(2n+1) \times (2n+1)$ for better performance. It is important to explain the meaning of n, we embed filtering algorithm into the function *median_filter* , there is a parameter n, if you set $n = 1, 2, ...n$, that means you want to use median filter with size $3 \times 3, 5 \times 5, ...(2n+1) \times (2n+1)$. Furthermore, we use a little trick in filter, we don't use filter with full size $(2n+1) \times (2n+1)$, actually when we filter vertical scratches, we take median values only in horizontal row with size of $1 \times (2n+1)$ to replace each pixel. In a similar way, we replace pixel of horizontal scratches by taking convolution with median filter with size of $(2n+1) \times 1$.

During the time of filtering, there is a problem that edges of the frame is not able to be filtered. To solve this, we build up a function *expand_img* to add the flipped part around the edge for symmetric expand with size n (according to filter size). The median filter is used for vertical and horizontal directions. Another problem comes from the sbv and sbh filter, during which there will be both negative and positive values shows up. So a binarization is used here to get rid of the negative value, then we use the pixel at the place that is -1 pixel difference in vertical or horizontal direction to overcome this. We use *median_filter* to filter each frame channel by channel, then combine them together. The final result is shown in Figure 5. The performance is satisfied, most of thinner scratches are removed, but there are still some large ones in third frame.

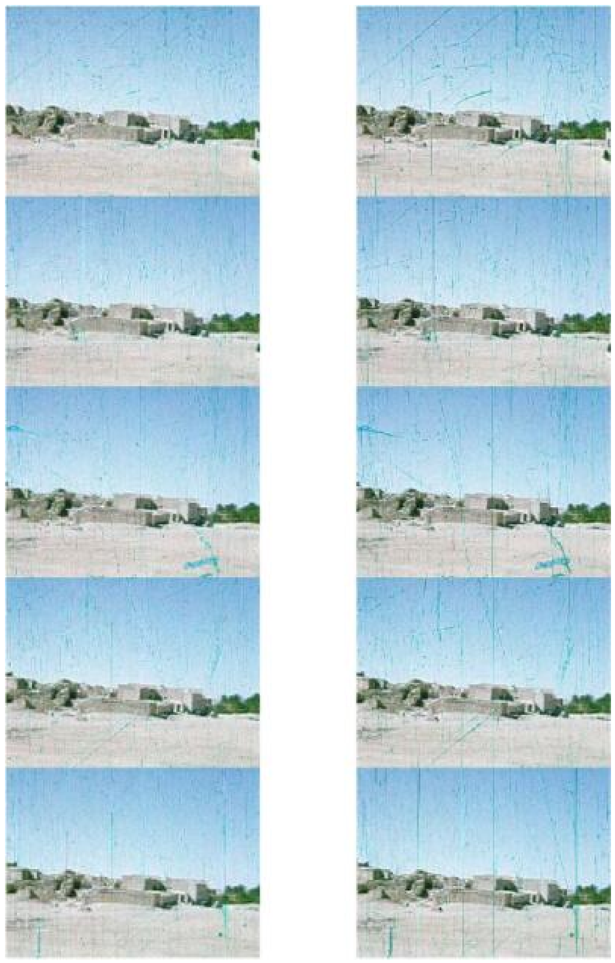


Figure 5: The result for frame processed(left) and original(right)

3 Remove the large scratches

To remove the large scratches, we start with the result we just get in Figure 5, the third frame is what we need to focus on, its RGB channels are plotted in Figure 6.

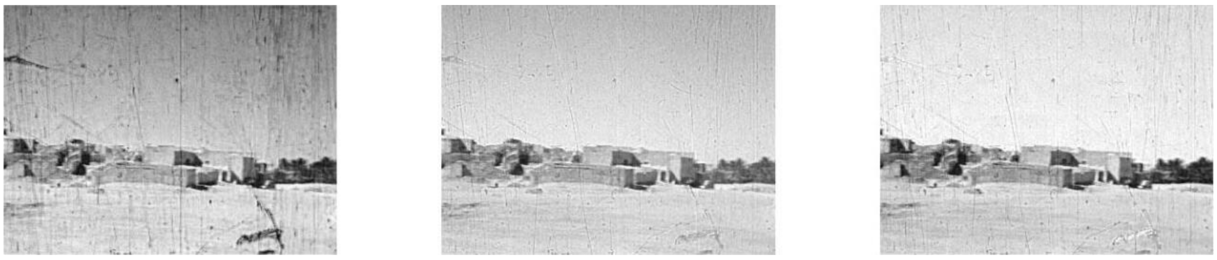


Figure 6: The R,G,B channel for the third frame after median filter

In R and G channels, the large patterns are more darker rather in the B channel, it means pixels of scratches are low value in R and G channel and high value in B channel. The position of the big scratches is same in all three channels and they are almost horizontal, so we can just use the red channels for scratches detection. We still use the same way to do a difference between frames and then binarize the measured gradients result to get rid of the negative part, the result is shown in Figure 7a.

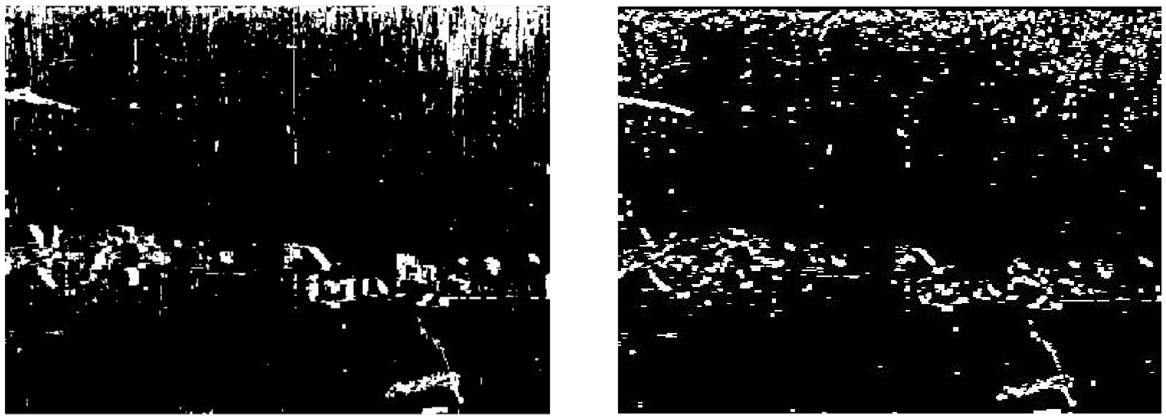

(a) The difference between 2nd and 3rd frame **(b)** The difference filtered by 5×3 sbh filter

Figure 7: Difference and filtered difference in red channel

We could see from Figure 7a that although there are many thin scratches and landscape movement, the large scratches are more thicker and horizontal, it inspires us that it maybe useful to use a larger sbh filter, like 5×5, to detect thicker horizontal edges(scratches). After taking convolution by larger sbh (we use a trick again, we set filter size as 5×3 sbh = [1 2 1; 0 0 0; 0 0 0; 0 0 0; -1 -2 -1] instead of 5×5 to achieve less connective pixels in areas other than large scratches), we get Figure 7b after binarizing. Result is not bad because we not only keep property of thick and horizontal for large scratches, and also separate big pieces of landscape and thin scratches both into small pieces.

Just like in section 2, considering the size of the scratches, we choose the threshold 200 for the connected pixels. If we look the large scratches carefully, which is shown in Figure 8a, the color of these area is more blue, it is reasonable because same pixels in blue channel have higher value, which is mentioned in the last paragraph. Therefore we decide to use different filters for different color channels to get better results. After many trials, We choose maximum filter and mean filter for R and G channel, median filter is used for B channel. However, the previous method is not enough for the scratches of this size. So we came up an idea to do both negative and positive parts.So we let sbh = -sbh, repeat again for the other part. The processed result is shown in Figure 8b.

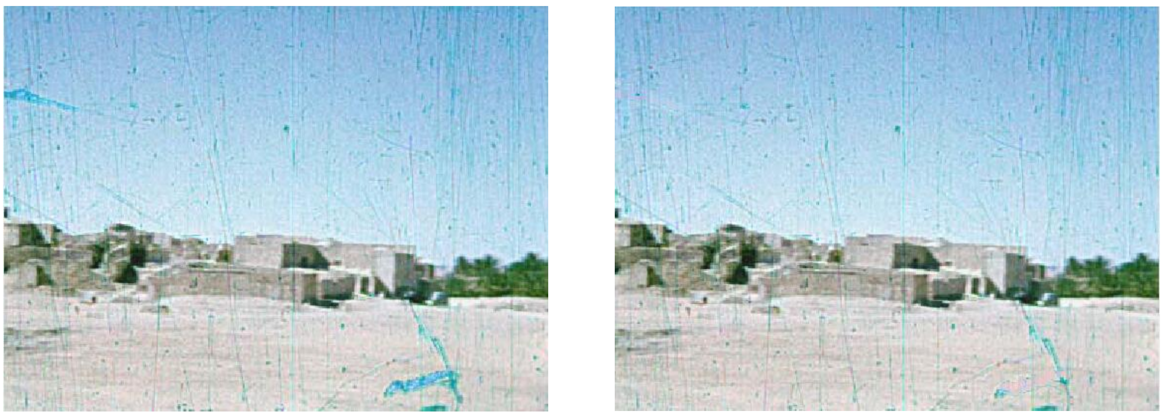

(a) Before large scratch removal processing **(b)** After lagrge scratch removal processing

Figure 8: Third frame before and after large scratch removal processing

Then we use this function *filter_large_scratch* to process all frame, the final performance is shown as Figure 9. Actually, function *filter_large_scratch* only works in third frame actually, because there are no large scratches in other frames.

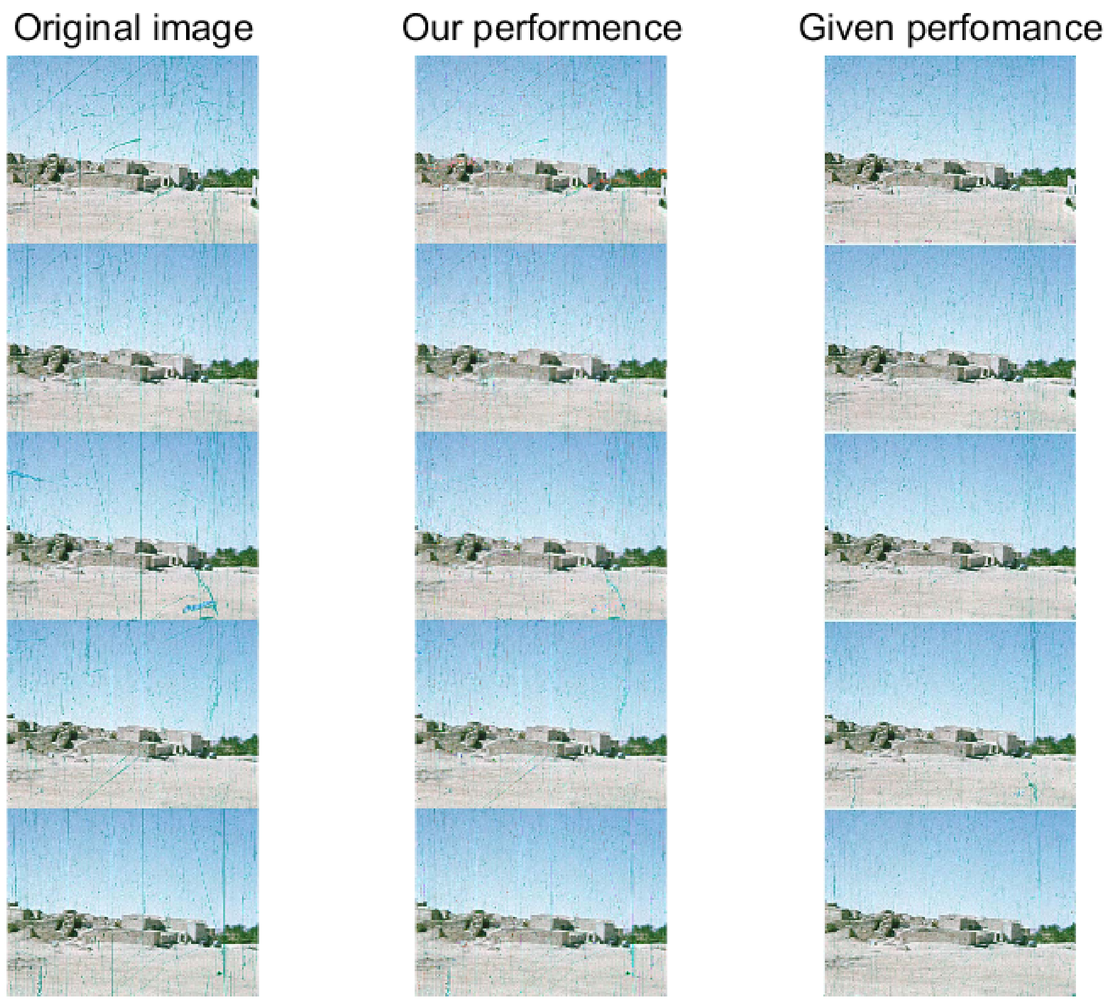


Figure 9: Original frame, our result and the given result

4 Conclusion and future work

The result is shown in Figure 9, honestly we are satisfied about our performance in general even if there are still many improvements we could do. Firstly, in third frame, the large scratches are not removed totally as give result shown, there are many vertical components retained. Secondly, still in third frame, there are a small dark area on the edge of bottom right are not removed, we think about it for a while, but it is hard to be detected. Thirdly, we don't have good performance in the last frame as in the given result, especially the scratches in the bottom left area are still there.

It is glad that we have better performance in forth frame compared to given result, and we keep almost details in the landscape area, at least we couldn't find differences in that area by eyes. Actually, at the beginning of the project, we have searched many articles about how to remove scratches in digital image, there are so many awesome and smart method to make it, we don't think those methods are very suitable for this small project in a short period. However, if we have half a year or more times, we may take a try by that smart methods. So far, we think we would come up a smart adaptive filter whose size could be changed according to the size of scrathes in the procedure of detecting .

5 Appendix A - MATLAB Codes

5.1 Main programe

```
clear; close; clc; close all

%% preprocess data
img_oringin = im2double(imread('film1_big.jpg'));
img_middle = img_oringin(:,size(img_oringin,2)/3 : 2*size(img_oringin,2)/3,:);
img_right = img_oringin(:,2*size(img_oringin,2)/3 : end,:);
img_left = img_oringin(:,1:size(img_oringin,2)/3,:);
img_middle1 = img_middle(3:size(img_middle)/5-3,4:end-4,:);
img_middle2 = img_middle(size(img_middle)/5+4:2*size(img_middle)/5-2,4:end-4,:);
img_middle3 = img_middle(2*size(img_middle)/5+4:3*size(img_middle)/5-2,4:end-4,:);
img_middle4 = img_middle(3*size(img_middle)/5+4:4*size(img_middle)/5-2,4:end-4,:);
img_middle5 = img_middle(4*size(img_middle)/5+4:end-2,4:end-4,:);
img_middle_cut= cat(1,img_middle1,img_middle2,img_middle3,img_middle4,img_middle5);
imgR = img_middle_cut(:,:,1);
imgG = img_middle_cut(:,:,2);
imgB = img_middle_cut(:,:,3);
figure;imshow(img_middle_cut,[])

%% get time squence img in each layer
imgR_in_time = img_in_time( imgR );
imgG_in_time = img_in_time( imgG );
imgB_in_time = img_in_time( imgB );

%% median filtering
imgR1 = median_filter(imgR_in_time(:,:,1),imgR_in_time(:,:,2),3);
imgG1 = median_filter(imgG_in_time(:,:,1),imgG_in_time(:,:,2),3);
imgB1 = median_filter(imgB_in_time(:,:,1),imgB_in_time(:,:,2),3);

imgR2 = median_filter(imgR_in_time(:,:,2),imgR_in_time(:,:,1),3);
imgG2 = median_filter(imgG_in_time(:,:,2),imgG_in_time(:,:,1),3);
imgB2 = median_filter(imgB_in_time(:,:,2),imgB_in_time(:,:,1),3);

imgR3 = median_filter(imgR_in_time(:,:,3),imgR_in_time(:,:,2),3);
imgG3 = median_filter(imgG_in_time(:,:,3),imgG_in_time(:,:,2),3);
imgB3 = median_filter(imgB_in_time(:,:,3),imgB_in_time(:,:,2),3);
```

```
imgR4 = median_filter(imgR_in_time(:,:,4),imgR_in_time(:,:,5),3);
imgG4 = median_filter(imgG_in_time(:,:,4),imgG_in_time(:,:,5),3);
imgB4 = median_filter(imgB_in_time(:,:,4),imgB_in_time(:,:,5),3);

imgR5 = median_filter(imgR_in_time(:,:,5),imgR_in_time(:,:,4),3);
imgG5 = median_filter(imgG_in_time(:,:,5),imgG_in_time(:,:,4),3);
imgB5 = median_filter(imgB_in_time(:,:,5),imgB_in_time(:,:,4),3);

img1 = cat(3,imgR1,imgG1,imgB1);
img2 = cat(3,imgR2,imgG2,imgB2);
img3 = cat(3,imgR3,imgG3,imgB3);
img4 = cat(3,imgR4,imgG4,imgB4);
img5 = cat(3,imgR5,imgG5,imgB5);

img = cat(1,img1,img2,img3,img4,img5);
figure;
imshow(img,[])

%% filter large scratch
img1_new = filter_large_scratch(img1,img2,5);
img2_new = filter_large_scratch(img2,img3,5);
img3_new = filter_large_scratch(img3,img2,5);
img4_new = filter_large_scratch(img4,img5,5);
img5_new = filter_large_scratch(img5,img4,5);

img_middle_new = cat(1,img1_new,img2_new,img3_new,img4_new,img5_new);
figure; imshow(img3_new,[])

%% plot
figure;
subplot(1,3,1);imshow(img_middle_cut,[])
title('Original image');set(gca,'fontsize', 15);
subplot(1,3,2);imshow(img_middle_new,[])
title('Our performence');set(gca,'fontsize', 15);
subplot(1,3,3);imshow(img_right,[])
title('Given perfomance');set(gca,'fontsize', 15);
```

5.2 Function used

```
function img_t = img_in_time( img )
% separate whole frame into 5 frames in time sequence
t = 5;
row = size(img,1)/t;

img_1 = img(1:row,:);
img_2 = img(1+row:row*2,:);
img_3 = img(1+row*2:row*3,:);
img_4 = img(1+row*3:row*4,:);
img_5 = img(1+row*4:row*5,:);

img_t = cat(3, img_1, img_2, img_3, img_4, img_5);
end


function output_img = expand_img( input_img , n)

input_img1 = cat(1, flipud(input_img(1:n,:)),input_img, ...
    flipud(input_img(size(input_img,1)-n+1:end,:)) );

output_img = cat(2, fliplr(input_img1(:,1:n)), input_img1, ...
    fliplr(input_img1(:,size(input_img1,2)-n+1:end)));

end


function img = median_filter( img1, img2, n)

% img1 is image what we want to filter
% img2 is the image between the img1 in the time sequence
% n is integer and >=1 ,  filter size = 2*n+1 * 2*n+1 , e.g. n=1 size = 3*3

diff = img2 - img1;
diff_bin = imbinarize(diff,0.1);
% we set 0.1 level to keep more features of difference

sbh=[1 2 1; 0 0 0; -1 -2 -1];
sbv=[1 0 -1; 2 0 -2; 1 0 -1];
```

```
diff_v = filter2(sbv,diff_bin);
diff_v_bin = imbinarize(diff_v);

diff_h = filter2(sbh,diff_bin);
diff_h_bin = imbinarize(diff_h,0.5);

% figure;
% subplot(1,2,1); imshow(diff_v_bin)
% subplot(1,2,2); imshow(diff_h_bin)

connect_4_v = bwlabel(diff_v_bin ,4);
num_connect = max(max(connect_4_v));

img = expand_img(img1,n);
for i = 1:num_connect
    if length(find(connect_4_v==i)) > 20
        [row,col] = find(connect_4_v==i);
        for j = 1:length(row)
            img(row(j)+n,col(j)+n-1) = median(img(row(j)+n,col(j) : col(j)+2*n-1));
        end
    end
end

connect_4_h = bwlabel(diff_h_bin,4);
num_connect = max(max(connect_4_h));

for i = 1:num_connect
    if length(find(connect_4_h==i)) > 20
        [row,col] = find(connect_4_h==i);
        for j = 1:length(row)
            img(row(j)+n-1,col(j)+n) = median(img(row(j) : row(j)+2*n-1, col(j)+n));
        end
    end
end

img = img(1+n:size(img1,1)+n, 1+n:size(img1,2)+n);

end
```

```
function img_new = filter_large_scratch( img_colour1,img_colour2, n )

% img_colour1 and img_colour2 are both 3-channel

img1R = img_colour1(:,:,1);
img1G = img_colour1(:,:,2);
img1B = img_colour1(:,:,3);

img2R = img_colour2(:,:,1);
img2G = img_colour2(:,:,2);
img2B = img_colour2(:,:,3);

img1R_bin = imbinarize(img1R,0.5);
img1G_bin = imbinarize(img1G,0.5);
img1B_bin = imbinarize(img1B,0.5);

img2R_bin = imbinarize(img2R,0.4);
img2G_bin = imbinarize(img2G,0.5);
img2B_bin = imbinarize(img2B,0.5);

diff = imbinarize(img2R_bin - img1R_bin) ; % we get rid of negatice values
% figure;
% imshow(diff,[])

sbh=-[-1 -1 -2 -1 -1;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0;1 1 2 1 1]; % we set 5*5 firstly
sbh = sbh(:,2:end-1);  % we think 5*3 has better performance
diff_h = filter2(sbh,diff);
diff_h_bin = imbinarize(diff_h); % should be plotted
% figure;
% imshow(diff_h_bin,[])

threshold = 200;
connect_4 = bwlabel(diff_h_bin,4);
num_connect = max(max(connect_4));

imgR_new = expand_img(img1R,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
            imgR_new(row(j)+n-2,col(j)+n) = max(max(imgR_new(row(j) : row(j)+2*n-1 , c
        end
    end
end
imgR_new = imgR_new(1+n:size(img1R,1)+n, 1+n:size(img1R,2)+n);

imgG_new = expand_img(img1G,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
            imgG_new(row(j)+n-2,col(j)+n) = mean(mean(imgG_new(row(j) : row(j)+2*n-1
        end
    end
end
imgG_new = imgG_new(1+n:size(img1G,1)+n, 1+n:size(img1G,2)+n);

imgB_new = expand_img(img1B,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
            imgB_new(row(j)+n-2,col(j)+n) = median(median(imgB_new(row(j) : row(j)+2*r
        end
    end
end
imgB_new = imgB_new(1+n:size(img1B,1)+n, 1+n:size(img1B,2)+n);

sbh = -sbh;
diff_h = (filter2(sbh,diff));
diff_h_bin = imbinarize(diff_h);
connect_4 = bwlabel(diff_h_bin,4);
num_connect = max(max(connect_4));

imgR_new = expand_img(imgR_new ,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
```

```
            imgR_new(row(j)+n+2,col(j)+n) = max(max(imgR_new(row(j) : row(j)+2*n-1 , c
        end
    end
end
imgR_new = imgR_new(1+n:size(img1R,1)+n, 1+n:size(img1R,2)+n);

imgG_new = expand_img(imgG_new,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
            imgG_new(row(j)+n+2,col(j)+n) = mean(mean(imgG_new(row(j) : row(j)+2*n-1
        end
    end
end
imgG_new = imgG_new(1+n:size(img1G,1)+n, 1+n:size(img1G,2)+n);

imgB_new = expand_img(imgB_new,n);
for i = 1:num_connect
    if length(find(connect_4==i)) > threshold
        [row,col] = find(connect_4==i);
        for j = 1:length(row)
            imgB_new(row(j)+n+2,col(j)+n) = median(median(imgB_new(row(j) : row(j)+2*r
        end
    end
end
imgB_new = imgB_new(1+n:size(img1B,1)+n, 1+n:size(img1B,2)+n);

img_new = cat(3, imgR_new, imgG_new, imgB_new);
end
```