**University of Macau**

**Faculty of Science and Technology**



**Fence Removal From Images**

***by***

**LAM KA CHON No: DB626167**

Final Project Report submitted in partial fulfillment  
of the requirements of the Degree of   
Bachelor of Science in Computer Science

Project Supervisor

Prof. Chan Long

**DECLARATION**

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name : Lam Ka Chon

Student No. : DB626167

Date : 29 May 2020

**ACKNOWLEDGEMENTS**

I would like to thank my supervisor, Prof.Chen Long give us many support and suggest to help us to finish the final year project. He gave a lot of in-depth guidance on the project's conception, framework and theoretical application, making the project successfully completed

**TABLE OF CONTENTS**

**CHAPTER 0. ABSTRACT 8**

**CHAPTER 1. INTRODUCTION 9**

1.1 Motivation 9

1.2 Research context 9

1.3 Research problem 9

1.4 Research contributions 9

1.5 Research purpose and significance 10

**CHARTER 2. PREPARATION BEFOREHAND 11**

2.1 Image photography 11

2.2 Data 11

**CHARTER 3. FENCE PROCESSING 13**

3.1 Image acquisition 13

3.2 The process of identifying and analyzing image areas 14

3.3 Identification and extraction of fences 16

3.3.1 Extraction using Gaussian blur 16

3.3.2 Other extraction methodse 18

3.4 Optimize the raised fence part 18

3.4.1 Extract the fence parts 19

3.4.2 Reduced or increased the fence size 19

3.5 Remove the fence 19

**CHARTER 4. IMAGE RECOVERING 21**

4.1 Exemplar-based inpainting algorithms 21

4.1.1 Computing patch priorities 21

4.1.2 Spread image texture 21

4.1.3 Update confidence values 22

4.2 Repair image to output fenceless image 22

4.2.1 Image downsampling and texture restoration 22

4.2.2 Upsampling of inpainted region 23

4.3 Impact of different repair methods on the output image 24

4.3.1 Harmonic Inpainting 24

4.3.2 Mumford-Shah Inpainting with Ambrosio-Tortorelli approximation 24

**CHAPTER 5. PROGRAM CODE 25**

5.1 Demo.m 25

5.2 component.m 26

5.3 inpainting.m 27

5.4 recovering.m 30

5.5 shrink.m 31

5.6 expand.m 31

**CHAPTER 6. EXPERIMENTS AND RESULTS 32**

6.1 Image operation time 32

6.2 Simple results 32

6.3 Fail results 34

6.4 Result analysis 35

**CHAPTER 7. REKATED WORK 36**

**CHAPTER 8. CONCLUSION 37**

**CHAPTER 9. REFERENCES 38**

**CHAPTER 10. APPENDIX 39**

11.1 Summary of the meetings 39

11.2 Flow chart 40

**LIST OF FIGURES**

Figure 2-1 Image folder 11

Figure 2-2 Object focusing images 11

Figure 2-3 Fence focusing images 12

Figure 3-1 Image acquisition 13

Figure 3-2 Object focusing image 13

Figure 3-3 Fence focusing image 13

Figure 3-4 Relation between the background, the fence, and the image plane 14

Figure 3-5 Relation between the fence and the image plane when the fence are in focus 16

Figure 3-6 Subtract object focusing image from object focusing image after gaussian filter 17

Figure 3-7 Subtract fence focusing image from fence focusing image after gaussian filter 17

Figure 3-8 Extraction fence 17

Figure 3-9 Mask image 18

Figure 3-10 Manually point the color pixel of the fence 18

Figure 3-11 Extraction fence 18

Figure 3-12 Extract the largest component 19

Figure 3-13 Reduced or increased the coverage area 19

Figure 3-14 An image with fence mask 20

Figure4-1 The notation diagram of inpainting domain / target region 21

Figure4-2 Downsampling image 23

Figure4-3 Upsampling image 23

Figure4-4 Restored image 24

Figure4-5 Harmonic Inpainting 24

Figure4-6 Mumford\_Shah Inpainting 25

**LIST OF TABLES**

Table 1-1 Research contributions 9

Table 6-1 Calculation process time 32

Table 6-2 Sample results 33

Table 6-3 Fail results 34

**CHAPTER 0: ABSTRACT**

With the popularity of smartphones and cameras, people often use social software, photography has become a way to record images and share memorable moments. In some cases, such as zoos, parks, and gardens. The scene people want to shoot is blocked by a fence, and people are also reluctant to have fences appear in photos. it is because it would affect aesthetics. Therefore, there is a need for an effective method of removing the fence, which can solve the above problem, and also improve the ability of computer vision.

When there is an obstruction between the camera and the subject, the obstructive field of view will affect the image quality of the subject. For example, when a camera captures an image of a scene through a fence, the fence would interrupt the object.Sometimes we take the photo outside the fence, some parts of the photo will be blocked by the fence, we can not get the whole photo, so we will use two different focal points to capture the image, and then we will get two images. One images fence is clear and the other is the opposite, so we can get all the material in the scene, restored the images, and get a clear image with the fence removed.

**CHAPTER 1: INTRODUCTION**

**1.1 Motivation**

This project is based on the topics selected under graduation requirements of the Cisc4000 Final Year Project. Since we are interested in image processing, and the skills and knowledge required for the topics have been learned in the past four years, we are convinced that these skills and knowledge will help to complete this project, so we selected this be the topic of our project.

**1.2 Research Context**

The research background of this project is based on our understanding of "Image De-fencing Framework with Hybrid Inpainting Algorithm"

(https://www.irisa.fr/vista/Papers/2004\_ip\_criminisi.pdf) this paper, and then our group to experiment, we think that the method proposed in this paper will help us to carry out, At the same time, I understand that there are some parts that can be improved, for example: using two photos can extract more details more effectively. So we follow the steps of the paper, take this as the core, and practice it step by step to establish a better removal method.

**1.3 Research Problem**

In this project, removing the blocked fence from the image and restoring it are important issues. Because the question we are studying is: how should we extract the fence, and whether the fence can be covered after the extraction, whether we need to optimize the image, and how we should solve the image restored method.

**1.4 Research contributions**

This part is about our contribution to various parts of the whole project, We show in the Table1-1.

|  |  |  |
| --- | --- | --- |
|  | LAM KA CHON | LAM PENG IAT |
| Data collection | x | x |
| Image taken | x | x |
| Image filter | x |  |
| Programming | x | x |
| Image reprocessing |  | x |
| Image processing | x |  |
| Inpainting methods | x | x |
| Program test | x | x |
| Debug | x | x |
| Collection of papers | x | x |
| Report writing | x | x |
| Meeting note | x | x |
| Data testing | x |  |
| Result statistics | x |  |

***Table 1-1: Research contributions***

**1.5 Research purpose and significance**

The purpose of the Final year project is using matlab to analyze the removal of fences, combined with current theoretical understanding and comparison of published data, to have a comprehensive understanding of fence removal and image restoration. Through literature review-based models and experimental results, efforts have been made to achieve the following goals:

* Image acquisition
* The process of identifying and analyzing image areas
* Identification and extraction of fences
* Optimize the raised fence part
* Remove the fence
* Repair image to output fenceless image
* Study the impact of different repair methods on the output image

1. Harmonic Inpainting
2. Mumford-Shah Inpainting with Ambrosio-Tortorelli approximation

The result of this work will provide important references for the study of extracting fences and repairing images.

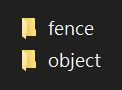
**CHARTER 2. PREPARATION BEFOREHAND**

**2.1 Image photography**

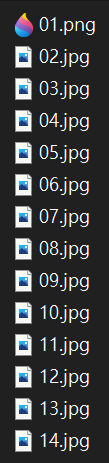
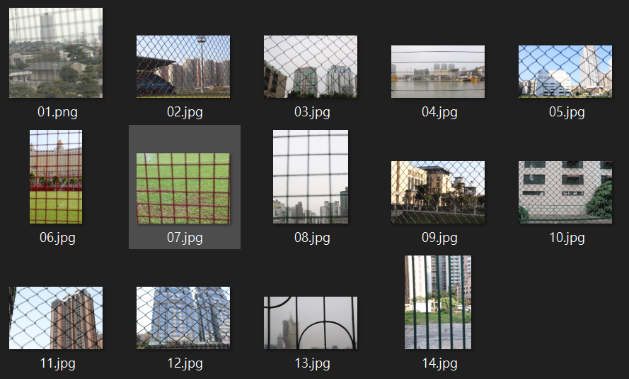
We searched for different types of fences for experimental shooting, and drawn different examples to have more conclusions for the final experimental analysis. Use camera equipment for shooting, including zoom camera and tripod. The zoom camera can change the focal length within a certain range, so as to obtain different wide and narrow fields of view, different size images and different sceneries. The zoom camera can change the shooting range by changing the focal length without changing the shooting distance. Therefore, the background or the object and the fence can be shot separately, from which the images of the clear object and the clear fence can be obtained respectively. Highlight the outline of the fence. The use of a tripod can avoid hand shake and result in blurred imaging. Fix the camera to shoot and fix the shooting point. When the image is extracted from the fence, the points of the two images fences can correspond to the corresponding positions.

**2.2 Data**

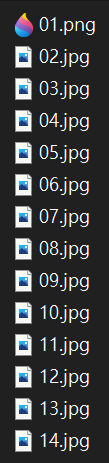
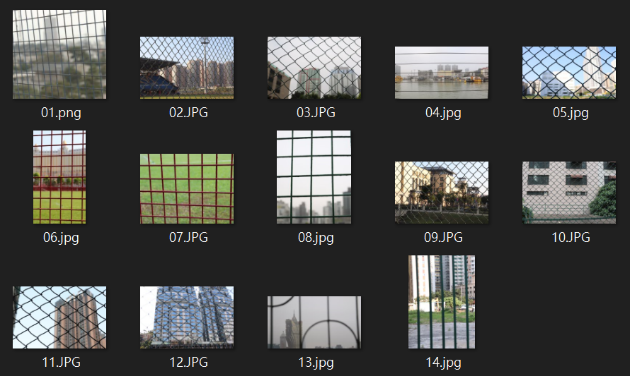
Our "data" file contains material photos, and almost all of the photos are taken by us because images with different focus are needed, and there is not enough material on the Internet for us to use. We divided the captured images into two files and classified them into object focus (see Figure 2-2) and fence focus (see Figure 2-3), and placed them in object folder and fence folder (see Figure 2-1), Which makes it easier to find images, query images, and read images.



***Figure 2-1 Image folder***

******

***Figure 2-2 Object focusing images***

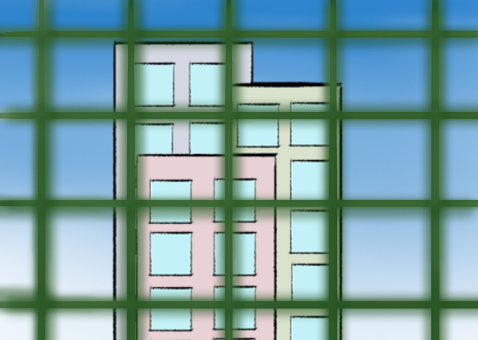
******

***Figure 2-3 Fence focusing images***

**CHAPTER 3: FENCE PROCESSING**

**3.1 Image acquisition**

This part is using two images to the acquisition background area(see Figure 3-1). One of the images is focused on the background(see Figure 3-2), and the other is the image focused on the fence(see Figure 3-3).

***(a)Object focusing image (b)Fence focusing image***

***Figure 3-1 Image acquisition***

******

***Figure 3-2 Object focusing image***

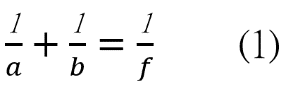
******

***Figure 3-3 Fence focusing image***

**3.2 The process of identifying and analyzing image areas**

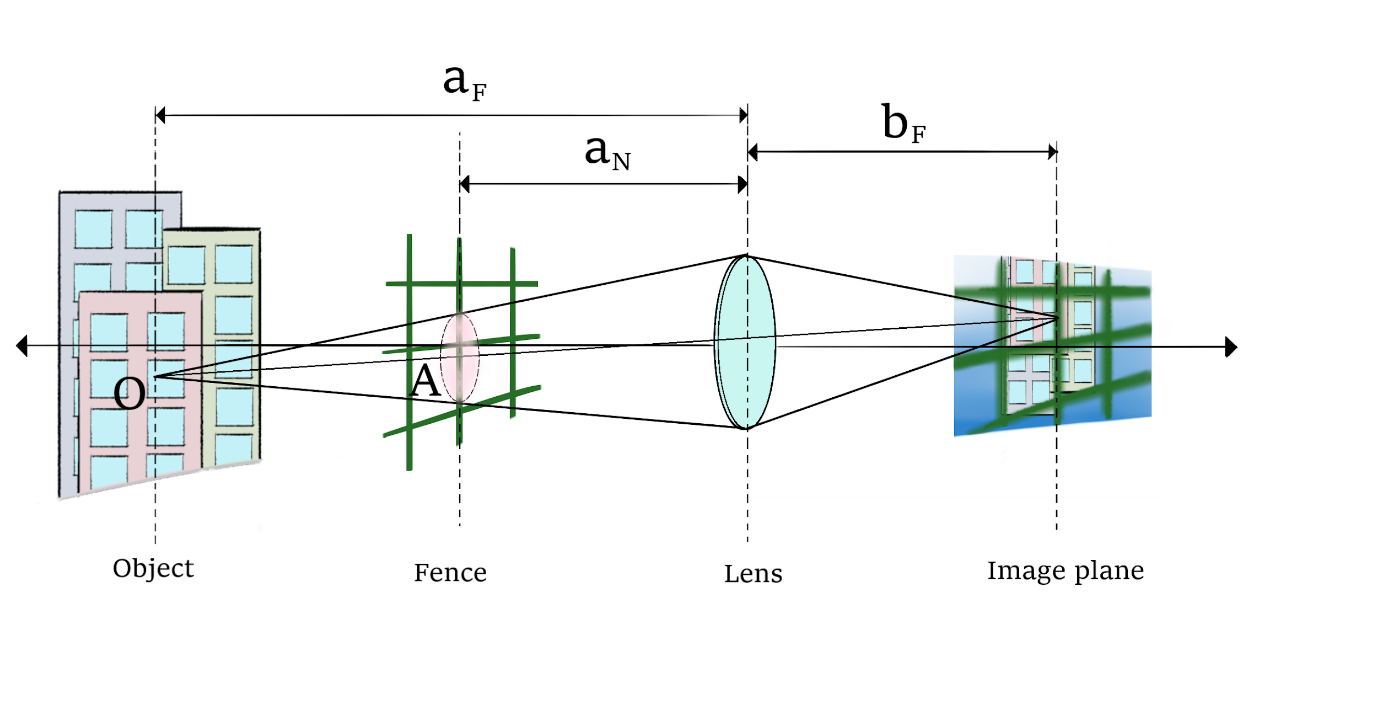
we used is a simple camera model consisting of an image plane and a thin lens. [1] The following is based on the thin lens formula to explain the focusing of geometric optics. [2].

The distance from the lens to the object is a, and the distance from the image to the lens is b. The formula is as follows:



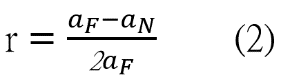
f is the focal length of the lens.

The distance between the lens and the background is , the distance between the image generation and the lens is , and the distance between the fence and the lens is .



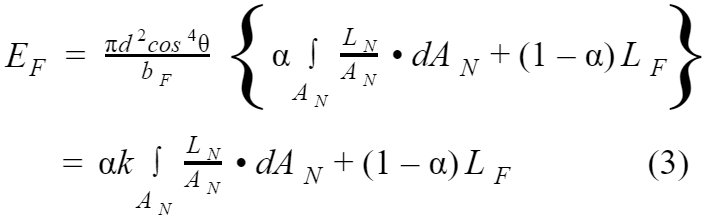
***Figure 3-4 Relation between the background, the fence, and the image plane.***

When we shoot, when the focus is on the object or the background, the part of the fence will become blurred on the photo. From this formula, the blur radius r of the fence part can be calculated:



If it is known fuzzy gray r, it is possible to estimate the impact fence image as the PSF (point spread function).

Because the lens radius can be measured, the parameter d can be obtained, but since the background image and the position of the fence are usually unknown, it is not easy to obtain  and . Therefore, only the gamma coefficient r is estimated from the image.

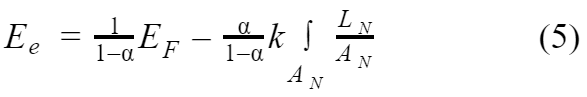


Where angle θ from optical center to image place, and the proportion  to A is α.

If the fence does not exist, equation (3) becomes (4).



This means that (3) in the equation is equal to the irradiance value  that is not interference from fences. Equation (3) can then become (5).

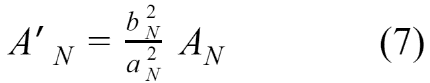


Where 

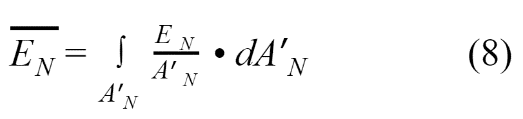
The parameters required for the illuminance value  are ,  and area of rear projection area. In equation (5), we cannot directly obtain the radiation value . We use the relationship between the fence and the image plane when focusing the fence to establish the equation.(see Figure 3-5) The relationship between the irradiance value  and the irradiance value  is expressed as follows:



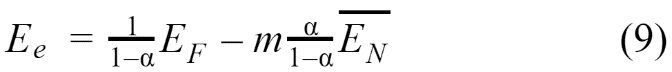
It can be seen from Figure 3-5 that the area  is the surface area , and the following formula is the ratio between  and :

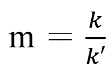


Where  is the distance between the image generation and the lens.

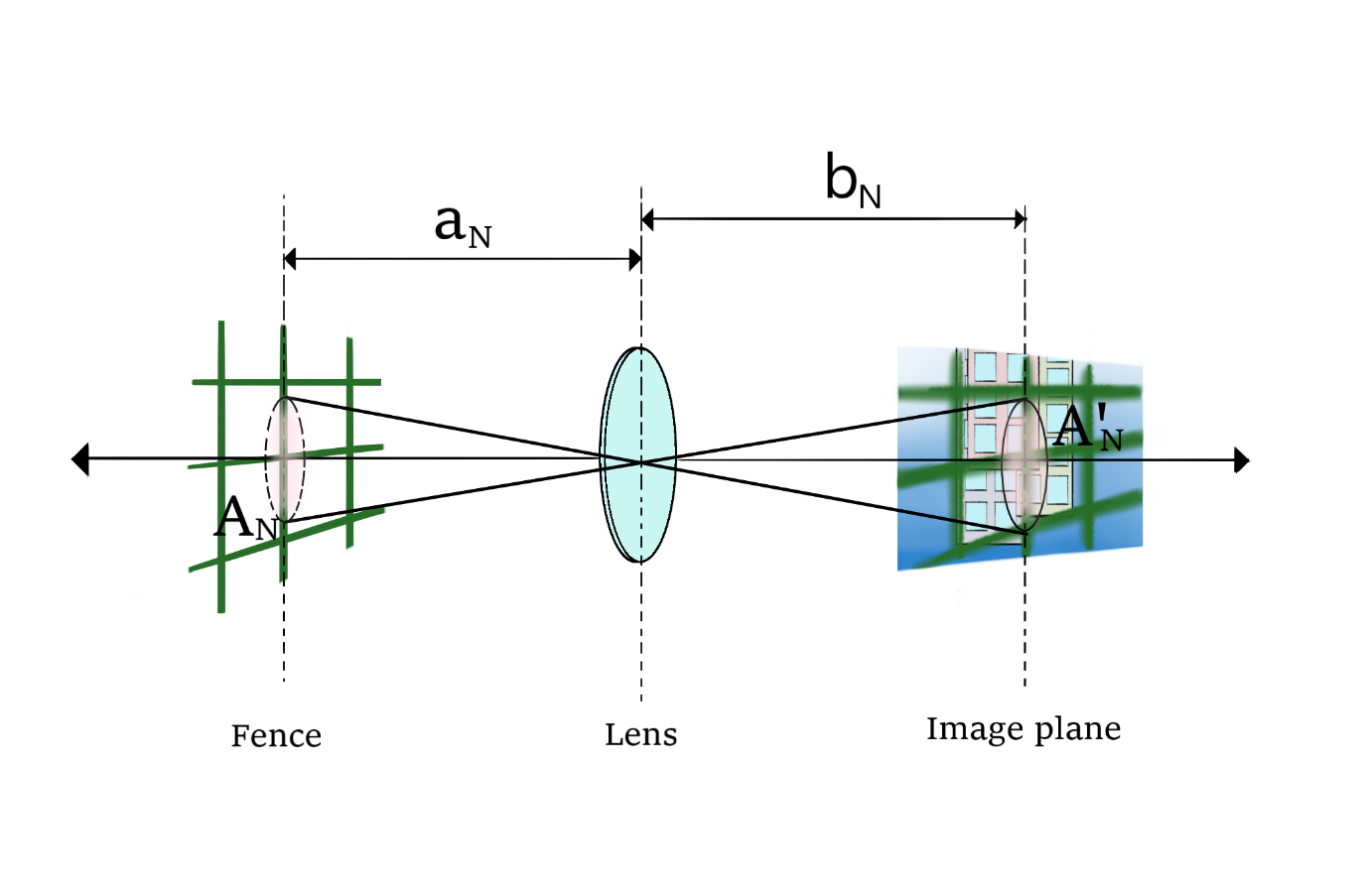


Finally, we can get  by substituting the equation.



 is the correction coefficient, and the average value of the illuminance value in  is .

The relationship between the optical geometry and the captured image is explained from the above formulas and images, which helps in the generation of images and how to obtain the correct image when shooting, and the relationship between the images when extracting the fence later, helps For extraction.



***Figure 3-5 Relation between the fence and the image plane when the fence are in focus***

**3.3 Identification and extraction of fences**

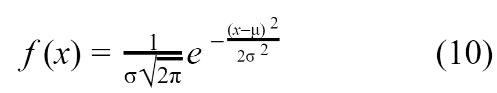
In this part, we will try different methods to extract the fence, in order to find out the convenience for later processing and suitable for our entire programming.

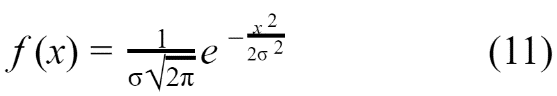
**3.3.1 Extraction using Gaussian blur**

Use the Focus method to identify the fence, and use Gaussian Blur to process the image[3]. This is a data smoothing technology. Each pixel takes the average value of the surrounding pixels. Numerically, this help to smoothes the image and produces a blur effect. In Gaussian blur, the larger the average value range used, the greater the blur effect. The larger the blur radius, the more blurred the image.

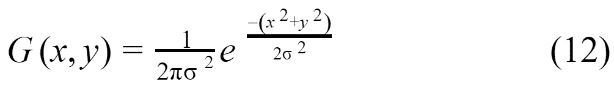
We need to be calculated with a two-dimensional normal distribution density function, In Gaussian function[4]

The one-dimensional Gaussian function formula is as follows:





The two-dimensional Gaussian function[5] formula is as follows:



Use this function to calculate the weight of each point, and then multiply the weight of the area by the pixels of the area to calculate the value of Gaussian blur.

Gaussian Blur was performed on the two images and then compared with the original image. The clear position of the original image and the image after Gaussian Blur was applied, the difference between their points will be larger. The blurred position of the original image and the image after Gaussian Blur, the difference between their points will be less.

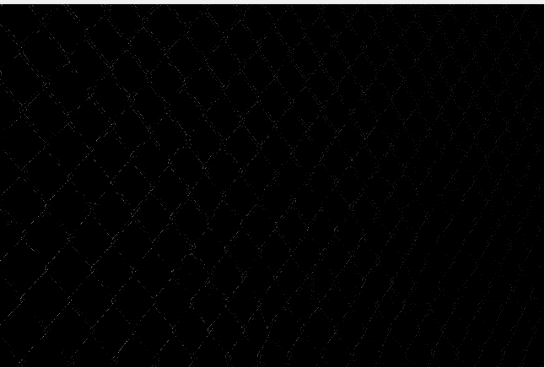
(see Figure 3-6)(see Figure 3-7)

Subtract the Gaussian Blur image from its original image, subtract from pixel to pixel, and then a value will appear. The larger the value is the clear position, and the smaller the value is the blurred position so that the clear position of the pixel will be on the fence and the subject.(see Figure 3-8)

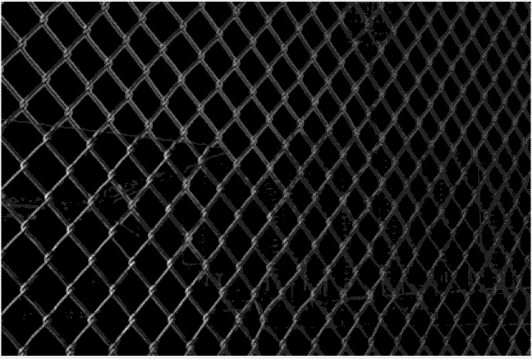
The mask of extraction fence image.(see Figure 3-9)



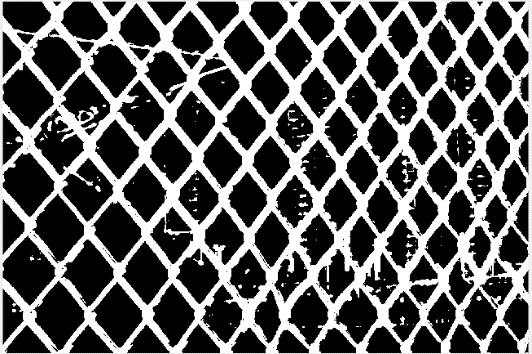
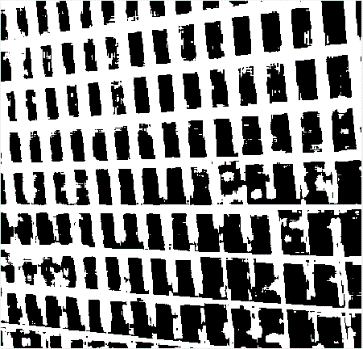
***Figure 3-6 Subtract object focusing image from object focusing image after gaussian filter***



***Figure 3-7 Subtract fence focusing image from fence focusing image after gaussian filter***

******

***Figure 3-8 Extraction fence***

******

***Figure 3-9 Mask image***

**3.3.2 Other extraction methods**

The user can manually enter the pixels of the fence (see Figure 3-10). The blue dots in the figure are the pixels of the points where the user manually enters the fence. The more points you select manually, the more the pixel color of the corresponding fence will be extracted, which can extract the entire fence and the effect will be better. We can model the fence through the points entered by the user. In order to minimize selection errors and increase data samples for better accuracy modeling, any manually selected pixels and their neighboring pixels are regarded as samples of the fence, so the manually selected fence pixels can be effectively Large.(see Figure 3-11).



***Figure 3-10 Manually point the color pixel of the fence***



***Figure 3-11 Extraction fence***

**3.4 Optimize the raised fence part**

The mask image we obtained after extracting the fence from the above steps can be seen from Figure 3-9. The non-fence part is also extracted. The mask image does not completely cover the fence part of the object image, so adjustment is required and optimize the mask image.

**3.4.2 Extract the fence parts**

In order to accurately detect the fence, it usually appears as non-fences in the form of small dots. Many small dots on the images that are not on the fence are also selected. In order to eliminate errors, we have to extract, eliminate small points that are not on the fence, and take the largest continuous part and remove the discontinuous part.

By arranging the connected components in descending order, then select the largest part as part of the fence.



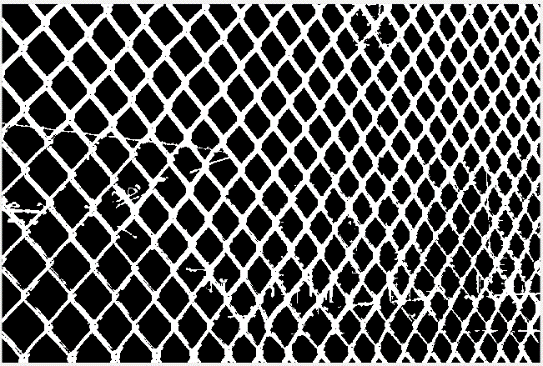
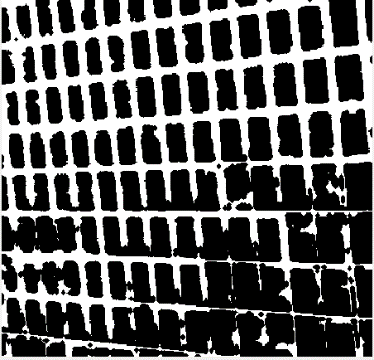
***Figure 3-12 Extract the largest component***

**3.4.3 Reduced or increased the fence size**

As the fences selected are too many or insufficient, the fence size needs to be made finer. Using the morphological operator[6], let the mark image correspond to the defocused fence.

Reduced or increased the coverage area expectations of the fences, while retaining the local information fence near.In this way, when inpainting, the effect will be better because there are fewer patching positions.

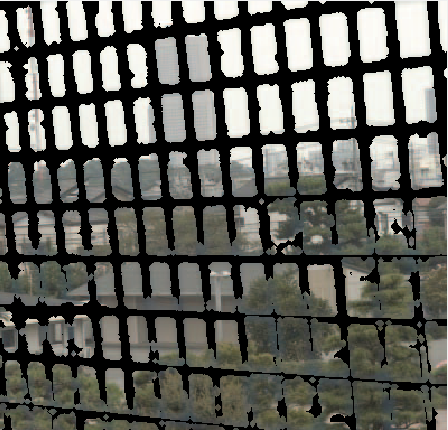
After covering, there is also a part of the fence protruding, which can increase the value of the number of picked components on the picked-up connected components. Conversely, it increases the number of pickup components.(see Figure 3-13)



***Figure 3-13 Reduced or increased the coverage area***

**3.5 Remove the fence**

This part is to combine the mask image optimized before and the object focusing image to remove the position of the fence in the original image. The process of using Gaussian Blur to process the image so that the clear pixels will be focused on the fence and the process of removing the fence is outlined. Use the original image to combine the selected part of the fence mask with the image, and compare whether the mask position of the merged image corresponds to and covers the position of the fence on the original image.(see Figure 3-14)

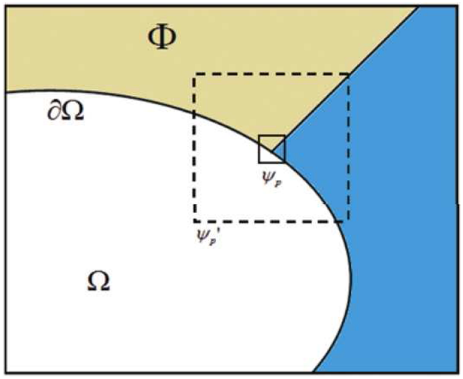


***Figure 3-14 An image with fence mask***

**CHAPTER 4: IMAGE RECOVERING**

**4.1 Exemplar-based inpainting algorithms**

Exemplar-based algorithms[7] is the image restoration technology. In the image restoration, we gradually restore the fence in the image. We can first remove the unnecessary fence parts in the original image, then use the repair method to fill in, fill the geometry and texture of the image to the corresponding target area.

****

***Figure 4-1 The notation diagram of inpainting domain / target region***

In the Figure 4-1, the Φ is the part original image, the Ω is the part to be repaired and the  Border between the Φ and the Ω. For each point p on the boundary , We use the center p to set a square patch.  is used to fill,  is use the center p to set a larger square patch [8].

The area to be filled, the boundary develops inward with the development of the algorithm. The source area Φ as a source to provide filling materials.

**4.1.1 Computing patch priorities**

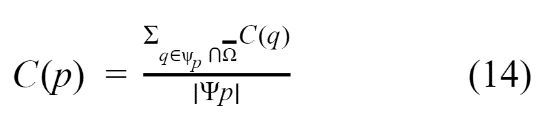
The target area is synthesized in the concentric layer from the outside to the inside. This task is accomplished by the best priority filling algorithm, which is entirely dependent on the priority value distribution each patch at the filling front end. Pixel patches with high confidence and continuous on the edge will be calculated first.

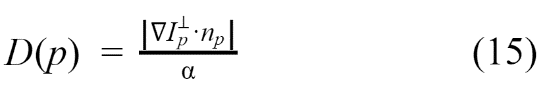
The  centred have some point p ∈ δΩ, its computing the priority of each patch P(p):



C(p) is the values of confidence term:

D(p) is the values of data term:

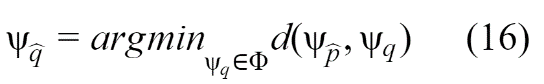




 is the complementary set of the part to be repaired Ω,  is the area of patch, α is normalization constant,  is a orthogonal vector,  is an isophote vector. D(p) is propagate the geometry to the part that needs to be repaired, and C(p) is the correlation between  and the surrounding pixels in the original image. If having more pixels around pixel p of the original image, C(p) get a higher value. The priority is calculated for each boundary patch, and a different patch is provide every pixel on the border that needs to be repaired.

**4.1.2 Spread image texture**

Instead, we spread the image texture by directly sampling the source area. We select the highest priority color patch  in the original image, Search for the most similar color patch in the original imageΦ to fill the color patch, and measure the similarity between the two patches.[8] Formally:

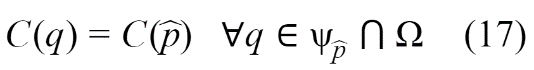


The distance  between two generic patches  and .

After finding the sample  in the original image, the pixel in  are filled for the value of each pixel p′, p′ ∈  ∩ Ω.

**4.1.3 Updating the confidence value**

The confidence C(p) needs to be updated. When the defined area in patch  is filled with new pixel values, following formula:



Repeat the above steps until the area Ω to be repaired is completely filled.

**4.2 Repair image to output fenceless image**

When restoring a fenced area, the fence can along with whole image and can cover a large part of image. This is a pyramid interpolation algorithm[9], and we used the example-based patching capabilities[10] to restore the image.

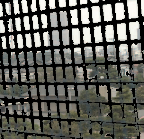
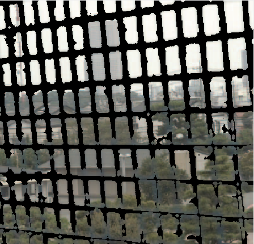
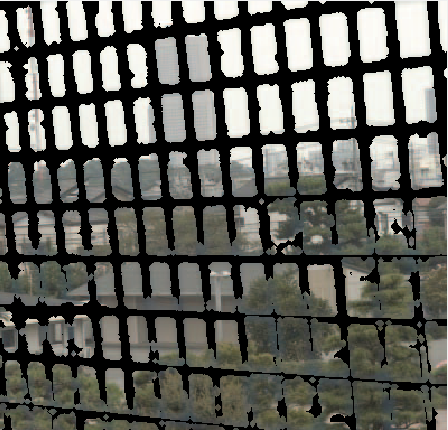
The repair algorithm is performed in two steps[11]

1. Continuously down-sampling the image to reduce the size of the image makes the need for inpainting less and faster.
2. Use a patch-based patch to patch fence holes at a lower resolution. Second, upsampling the patched area and copying it into the corresponding full-resolution image.

**4.2.1 Image downsampling**

In this part, the image is downsampling, the image I (m, n) and the part of the fence that needs to be filled Ω (m, n) are continuously downsampling to generate the image set {，，...，}，{，， ...，}, where Ik,  is the result of the downsampling factor 2 of k. By downsampling the maximum absolute difference between the fitted high-resolution images, the image scale is reduced.

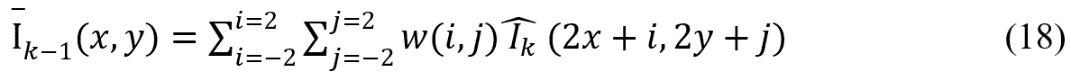
After downsampling, we have reduced the image scale, and the fence part  has also been reduced. It can therefore be recovered more easily.

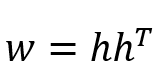
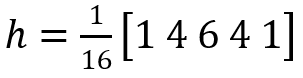


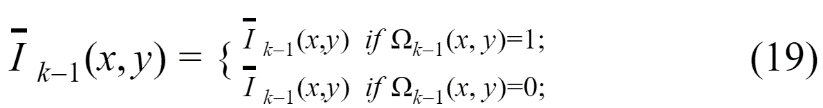
***Figure 4-2 Downsampling image***

**4.2.2 Image upsampling**

This part is to restore the original size of the downsampled image that has been repaired by inpainting and the part of the fence filled.  represents an image restored with a resolution of level k, that is. Any image  recovered at k levels can be up-sampled using Gaussian filtering defined in the following way:



Where and  are Gaussian weighting functions, and  is an upsampling of. Use upsampling to restore the pixels of the fence to the sampling level k.



This process is repeated until  is calculated, where the fenced area is fully restored.



***Figure 4-3 Upsampling image***



***Figure 4-4 Restored image***

**4.3 Impact of different repair methods on the output image**

In this part, we tried other inpaint methods, Harmonic Inpainting and Mumford-Shah Inpainting with Ambrosio-Tortorelli approximation, the results are as follows:

**4.3.1 Harmonic Inpainting**

Harmonic repair function[12] can smoothly finish the surface and repair.

This method is used as the boundary data for solving the harmonic function. Here is the function to be repaired, and is the boundary of the repair area.(see Figure 4-14)



***Figure 4-5 Harmonic Inpainting***

**4.3.2 Mumford\_Shah Inpainting**

The Mumford\_Shah Inpainting[13], this function by a piecewise smooth function, and it is used to establish the function of used to divide the image into subregions.

The image can reduce the distance between the model and the input image due to the piecewise smoothing function, reduce the smoothness of the subregion and the length of the boundary of the subregion. Through the minimize function, you can calculate the best image segmentation effect (see Figure 4-15).



***Figure 4-6 Mumford\_Shah Inpainting***

**CHAPTER 5: PROGRAM CODE**

**5.1 Demo.m**

close all;

clear all;

object\_image = imread('object\_02.JPG');

fence\_image = imread('fence\_02.JPG');

%Setup

fill\_image = fence\_image;

set1 = 5 ;

set2 = 50 ;

%The value of connected is 4 or 8

%Indicating whether to search the area by 4 connected or 8 connected

connected = 8;

%Threshold of connected components

th = 1;

%Structural element objects for operations such as expansion corrosion and opening and closing operations

se = strel('disk',2);

%Gaussian blur

%Returns a rotationally symmetric Gaussian low-pass filter with dimensions of standard deviation

h1 = fspecial('gaussian',set1,set1\*2);

h2 = fspecial('gaussian',set2,set2\*2);

o\_img\_gray = rgb2gray(object\_image);

f\_img\_gray = rgb2gray(fence\_image);

o\_img\_blur = imfilter(o\_img\_gray , h1, 'symmetric');

f\_img\_blur = imfilter(f\_img\_gray , h1, 'symmetric');

% Identification fences

o\_img\_sub = imsubtract(o\_img\_gray,o\_img\_blur);

f\_img\_sub = imsubtract(f\_img\_gray,f\_img\_blur);

% Extraction fences

f\_img\_sub\_blur = imfilter(f\_img\_sub , h2, 'symmetric');

o\_img\_sub\_D = double(o\_img\_sub);

f\_img\_sub\_blur\_D = double(f\_img\_sub\_blur);

[h,w] = size(o\_img\_sub\_D);

[r,c] = size(f\_img\_sub\_blur\_D);

sub\_img = zeros();

%Into a mask image

for i=1:r

for j=1:c

if f\_img\_sub\_blur\_D(i, j) > 0.1

sub\_img(i, j) = 1;

else

sub\_img(i, j) = 0;

end

end

end

% Optimize the raised fence part

% Extract the fence parts

[sub\_img\_C] = component(sub\_img, connected, th);

% Reduced or increased the fence size

sub\_img\_thin = imerode(sub\_img\_C,se);

%sub\_img\_coarse = imdilate(sub\_img\_C,se);

img\_dil = ones();

img\_imd = imdilate(sub\_img\_thin, img\_dil);

[m n] = size(img\_imd);

for i=1:m

for j=1:n

if img\_imd(i,j)

fill\_image(i,j,1) = 0;

fill\_image(i,j,2) = 0;

fill\_image(i,j,3) = 0;

end

end

end

%Inpainted and restored image

mask = sub\_img\_thin;

mask = im2bw(mask, 0.5);

recovered\_image = recovering(ori\_img, mask, fill\_img, mask);

**5.2 component.m**

function[ M, counts, s\_counts ] = component(mask, connected ,th)

% Extract the connected components of fence according to th

[m,n] = size(mask);

[L,num] = bwlabel(mask,connected);

counts(1:num) = double(1);

display(sprintf(' Number of connected components found: %d', num));

for i= 1:m

for j = 1:n

if(L(i,j) > 0 )

counts(1,L(i,j)) = double(counts(1,L(i,j))+1);

end

end

end

%sorts counts in descending order.

s\_counts = sort(counts , 'descend');

counter = 1;

for i = 1:th

x = s\_counts(i);

location = find(counts == x);

[p q] = size(location);

if(p>1)

for j = 1:p

locations(counter , 1) = location(p,1);

counter = counter + 1;

end

else

locations(counter , 1 ) = location(p,1);

counter = counter + 1;

end

if(counter > th)

break;

end

end

M(m,n) = zeros;

for i = 1:th

z = locations(i , 1);

for j = 1:m

for k =1:n

if( L(j,k) == z)

M(j,k) = 1;

end

end

end

end

end

**5.3 inpainting.m**

function [inpainted\_img,fill\_img] = inpainting(original\_image,fill\_image,fill\_color)

%Exemplar-based inpainting

%Refer by http://www.di.unito.it/~farid/Research/defencing.html

[img,fill\_img,fill\_region] = loadimgs(original\_image,fill\_image,fill\_color);

img = double(img);

ori\_img = img;

ind\_img = img2ind(img);

mn = [size(img,1) size(img,2)];

source\_region = ~fill\_region;

% Initialize the values of isophote vector

[Ix(:,:,3) ,Iy(:,:,3)] = gradient(img(:,:,3));

[Ix(:,:,2) ,Iy(:,:,2)] = gradient(img(:,:,2));

[Ix(:,:,1) ,Iy(:,:,1)] = gradient(img(:,:,1));

Ix = sum(Ix,3)/(3\*255);

Iy = sum(Iy,3)/(3\*255);

% 90 degree rotation

temp = Ix; Ix = -Iy;

Iy = temp;

%Initialize confidence and data terms

C is the values ​​of confidence level

D is values of data item

C = double(source\_region);

D = repmat(-.1,mn);

iteration = 1;

% Visualization stuff

if nargout==6

ori\_img(1,1,:) = fill\_color;

iteration = 2;

end

% Loop until the entire filled area is covered

while any(fill\_region(:))

% Find the outline of the filled area

fill\_region\_D = double(fill\_region);

edge = find(conv2(fill\_region\_D,[1,1,1;1,-8,1;1,1,1],'same')>0);

[Jx,Jy] = gradient(double(~fill\_region));

J = [Jx(edge(:)) Jy(edge(:))];

J = normr(J);

% handle NaN and Inf

J(~isfinite(J))=0;

% Calculate confidence along the fill front

for K=edge'

max\_pri\_patch = getpatch(mn,K);

L = max\_pri\_patch(~(fill\_region(max\_pri\_patch)));

C(K) = sum(C(L))/numel(max\_pri\_patch);

end

D(edge) = abs(Ix(edge).\*J(:,1)+Iy(edge).\*J(:,2)) + 0.001;

priorities = C(edge).\* D(edge);

% Find patch with maximum priority, max\_pri\_patch

[unused,ndx] = max(priorities(:));

M = edge(ndx(1));

[max\_pri\_patch,rows,cols] = getpatch(mn,M);

fillin = fill\_region(max\_pri\_patch);

% Find exemplar that minimizes error, min\_error

min\_error = bestexemplar(img,img(rows,cols,:),fillin',source\_region);

% Update the filled area

fillin = logical(fillin);

fill\_region(max\_pri\_patch(fillin)) = false;

% Propagate confidence & isophote values

C(max\_pri\_patch(fillin)) = C(M);

Ix(max\_pri\_patch(fillin)) = Ix(min\_error(fillin));

Iy(max\_pri\_patch(fillin)) = Iy(min\_error(fillin));

% Copy image data from min\_error to max\_pri\_patch

ind\_img(max\_pri\_patch(fillin)) = ind\_img(min\_error(fillin));

img(rows,cols,:) = ind2img(ind\_img(rows,cols),ori\_img);

if nargout==6

ind2\_img = ind\_img;

ind2\_img(logical(fill\_region)) = 1;

end

iteration = iteration+1;

end

inpainted\_img=img;

%Call C/C ++ and Fortran MEX function from MATLAB

function low\_error = bestexemplar(img,Ip,fillin,source\_region)

m=size(Ip,1);

mm=size(img,1);

n=size(Ip,2);

nn=size(img,2);

best = bestexemplarhelper(mm,nn,m,n,img,Ip,fillin,source\_region);

low\_error = sub2ndx(best(1):best(2),(best(3):best(4))',mm);

% Return color patch centered on the pixel index

function [max\_pri\_patch,rows,cols] = getpatch(mn,pixel)

w=4;

pixel=pixel-1;

y=floor(pixel/mn(1))+1;

pixel=rem(pixel,mn(1));

x=floor(pixel)+1;

rows = max(x-w,1):min(x+w,mn(1));

cols = (max(y-w,1):min(y+w,mn(2)));

max\_pri\_patch = sub2ndx(rows,cols,mn(1));

% Index style conversion to Matlab

function N = sub2ndx(rows,cols,total\_rows)

X = rows(ones(length(cols),1),:);

Y = cols(:,ones(1,length(rows)));

N = X+(Y-1)\*total\_rows;

% Convert to colormap

function img\_ind2 = ind2img(ind,img)

for i=3:-1:1, temp=img(:,:,i);

img\_ind2(:,:,i)=temp(ind);

end

%Use the image itself as a color map to convert RGB images to index images

function ind\_img = img2ind(img)

s=size(img);

ind\_img=reshape(1:s(1)\*s(2),s(1),s(2));

% Find the pixels and areas to be filled

function [img,fill\_img,fill\_region] = loadimgs(img,fill\_img,fill\_color)

fill\_region = fill\_img(:,:,1)==fill\_color(1) & ...

fill\_img(:,:,2)==fill\_color(2) & fill\_img(:,:,3)==fill\_color(3);

**5.4 recovering.m**

function [ recovered\_img ] = recovering(original\_image, original\_mask, fill\_image, mask)

[height,width] = size(mask);

level = floor(log2(min(height, width)));

pyramid\_fill\_img = cell(1, level);

pyramid\_mask = cell(1, level);

flag = 1; %Check for the presence of marked region

count = 0; %Counts the levels

pyramid\_fill\_img{count+1} = fill\_image;

pyramid\_mask{count+1} = mask;

count = count+1;

% Downsampling

% Recursive Identification Methods (RIM)

for i=1:level

if(flag==1)

[RIM] = shrink(fill\_image);

[mask] = shrink(mask);

% Check for black

ind = find(mask == 1);

if ~size(ind)

flag = 0;

elseif size(mask, 1) <= 300

flag = 0;

[RIM] = inpainting(RIM, RIM, [0 0 0]);

RIM = uint8(RIM);

end

count = count+1;

pyramid\_fill\_img{count} = RIM;

pyramid\_mask{count} = mask;

fill\_image = RIM;

clear rim;

else

break;

end

end

%Matlab’s built-in cellfun function enabled ‘isempty’ processing functions.

pyramid\_fill\_img = pyramid\_fill\_img(~cellfun('isempty', pyramid\_fill\_img));

pyramid\_mask = pyramid\_mask(~cellfun('isempty', pyramid\_mask));

% Upsampling

for i=count:-1:2

img = pyramid\_fill\_img{i};

exp = expand(img);

mask = pyramid\_mask{i-1};

recovered\_img = pyramid\_fill\_img{i-1};

[x,y] = find(mask==1);

for j=1:size(x)

if mask(x(j), y(j))

recovered\_img(x(j), y(j),1) = exp(x(j), y(j),1);

recovered\_img(x(j), y(j),2) = exp(x(j), y(j),2);

recovered\_img(x(j), y(j),3) = exp(x(j), y(j),3);

end

end

pyramid\_fill\_img{i-1} = recovered\_img;

if i ~= 2

clear recovered\_img im exp mask

end

end

end

**5.5 shrink.m**

function [ output\_image ] = shrink( image )

% Maximum absolute difference between high-resolution images fitted by downsampling.

[M, N, ~]=size(image);

output\_image=image(1:2:M, 1:2:N,:);

end

**5.6 expand.m**

function img = expand(image)

%Upsampling

%Pixels restore the image accordingly at the downsampling level.

%gaussian weighting functions

gaussian\_weighting = [1 4 6 4 1]/16;

img\_size = size(rgb2gray(image));

y = zeros(2\*img\_size);

y(1:2:2\*img\_size(1),1:2:2\*img\_size(2)) = image(:,:,1);

img(:,:,1) = 4\*conv2(gaussian\_weighting,gaussian\_weighting,y,'same');

y = zeros(2\*img\_size);

y(1:2:2\*img\_size(1),1:2:2\*img\_size(2)) = image(:,:,2);

img(:,:,2) = 4\*conv2(gaussian\_weighting,gaussian\_weighting,y,'same');

y = zeros(2\*img\_size);

y(1:2:2\*img\_size(1),1:2:2\*img\_size(2)) = image(:,:,3);

img(:,:,3) = 4\*conv2(gaussian\_weighting,gaussian\_weighting,y,'same');

img = uint8(img);

end

##### CHAPTER 6: EXPERIMENTS AND RESULTS

**6.1 Image operation time**

We report the results of nine experiments in Table 5-1, and summarize the information in Table 5-1. We have the description about the image, and the size of image, we calculate separately the time of fence removal technique, respectively the time of of extraction fence, the time of optimize fence, the time of inpainting.

We show the calculation process time in the Table 5-1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Demo** | **Description** | **Image size** | **Time of extraction fence** | **Time of optimize fence** | **Time of inpainting** | **Total process time** |
| 1 | House | 384x400 | 0.37s | 0.06s | 25.40s | 25.83s |
| 2 | Playground | 6000x4000 | 33.34s | 1.73s | 151.92s | 186.99s |
| 3 | Building | 6000x4000 | 31.39s | 3.84s | 91.91s | 127.14s |
| 4 | Lakeside | 6000x3375 | 22.13s | 3.70s | 28.49s | 54.32s |
| 5 | Building | 6000x3375 | 21.71s | 23.30s | 63.29s | 108.3s |
| 6 | Casino | 2250x4000 | 12.28s | 1.05s | 18.03s | 31.36s |
| 7 | Grass | 4500x3375 | 17.65s | 5.73s | 46.44s | 69.82s |
| 8 | Building | 2697x3371 | 10.50s | 3.90s | 13.31s | 27.71s |
| 9 | College | 6000x4000 | 33.51s | 11.92s | 129.38s | 174.81s |

***Table 5-1: Calculation process time***

**6.2 Sample results**

In this chapter, the proposed fence removal technique will be tested on several images. We conducted experiments outside the scene and photographed 14 experimental results. We used different focal points to shoot. We took objects focusing images and fence focusing images, and we took different sizes and types of fences to test the effect of the inpainting algorithm on fence. We show the sample results in the table 5-2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Object focusing image** | **Fence focusing image** | **Mask image** | **Restored image** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Table 5-2: Sample results***

**6.3 Fail results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Object focusing image** | **Fence focusing image** | **Mask image** | **Restored image** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Table 6-3: Fail results***

**6.4 Result analysis**

The above is the result of our experiment. Table 6-2 is a relatively good repair image. In the photo of extracting the fence, some extra parts have been extracted, it can be seen that the entire fence has been removed. In the image restoration, the result of the image fence repair in each example is the same as the background object color, which indicates that the algorithm has successfully solved the problem. Table 6-3 is the failed image. In this figure of extracting the fence, we can see that the image distance between the background and the fence is short, and the image with the background color and the fence color almost the same will make the fence difficult It will be extracted and at the same time it will make redundant parts appear during the repair.

##### CHAPTER 7: RELATED WORK

In this project, many of us completed it together, data collection, image taken, programming, inpainting methods, program test, debug, collection of papers, Report writing, meeting note.

In image taken, we take photos everywhere in Macau, and find a fenced place to shoot. In programming, I'm mainly responsible for image processing, I have studied through the theory I have learned in the past. Using Gaussian Blur to process the image, I extraction the fence, and then optimize the image. By extracting the largest parts, I remove the non- fence parts In order to filter out the part of the fence, and then process the part of the fence, in order to make the mask image cover the original fence position, the fence mask image should be thickened or reduced, because the focus point is different, the fence image There is an extension, so in the experiment, it is necessary to bold to cover the fence.

In inpainting methods, I followed exemplar-based image inpainting to inpaint our image, because it effect is better. In order to perform inpainting more quickly, I downsample and upsample the image, first reduce An image with fence mask to the maximum downsampling level to reduce the image size below a predetermined threshold, then perform inpaint, and then inpaint the image is restored to the original resolution according to the Gaussian weighting function. After completing the overall function, I will test different demos and debugs, and some values will have to be changed according to the different image, so that we can better go to the Extract fence and restore the image later.

In report writing, I am mainly responsible for finding theories to explain the methods we use, and I typeset and write reports.

##### CHAPTER 8: CONCLUSION

In this project, We the two focus method to identify the fence, then apply Gaussian blur to the image, we have try other extraction methods, that the method requires the user to manually point on the images, instead of automatically extracting the fence, and the more points you need to point, that make the better the effect, so this method is not better than our methods. Then let the defocused fence is eliminated from the original image. We use the exemplar-based inpainting and pixel interpolation-based to inpaint the image, it restores the fence regions efﬁciently, other methods are smoothing function composition and harmonic repair to smooth the surface, the result is more blurred. In order to inpaint the image faster, we have performed downsampling and upsampling, first reducing the image proportionally, then inpainting, and then expand it, the whole method is fast and the effect is regions efﬁciently.

##### CHAPTER 9: REFERENCE

**9.1 Camera model**

[1] Weng, J., Cohen, P., & Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattern Analysis & Machine Intelligence, (10), 965-980.

**9.2** **Geometric optics formula**

[2] Willson, R. G., & Shafer, S. A. (1994). What is the center of the image?. JOSA A, 11(11), 2946-2955.

**9.3 Gaussian blur**

[3] Kong, H., Akakin, H. C., & Sarma, S. E. (2013). A generalized Laplacian of Gaussian filter for blob detection and its applications. IEEE transactions on cybernetics, 43(6), 1719-1733.

**9.4 Gaussian function**

[4] Guo, H. (2011). A simple algorithm for fitting a Gaussian function [DSP tips and tricks]. IEEE Signal Processing Magazine, 28(5), 134-137.

**9.5 Two-dimensional Gaussian function**

[5] Haddad, R. A., & Akansu, A. N. (1991). A class of fast Gaussian binomial filters for speech and image processing. IEEE Transactions on Signal Processing, 39(3), 723-727.

**9.6 Morphological operator**

[6] Haralick, R. M., Sternberg, S. R., & Zhuang, X. (1987). Image analysis using mathematical morphology. IEEE transactions on pattern analysis and machine intelligence, (4), 532-550.

**9.7 Exemplar-based image inpainting**

[7] Criminisi, A., Pérez, P., & Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on image processing, 13(9), 1200-1212.

**9.8 Filling order is non-parametric texture synthesis**

[8] Efros, A. A., & Leung, T. K. (1999, September). Texture synthesis by non-parametric sampling. In Proceedings of the seventh IEEE international conference on computer vision (Vol. 2, pp. 1033-1038). IEEE.

**9.9 Pyramid interpolation**

[9] Farid, M. S., Khan, H., & Mahmood, A. (2010, October). Image inpainting based on pyramids. In IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS (pp. 711-715). IEEE.

**9.10 Patching capabilities**

[10]Criminisi, A., Perez, P., Toyama, K.: Region ﬁlling and object removal by exemplar-based image inpainting. IEEE Trans. Image Process. 13(9), 1200–1212 (2004)

**9.11 Upsampling and downsampling**

[11] Gopinath, R. A., & Burrus, C. S. (1994). On upsampling, downsampling, and rational sampling rate filter banks. IEEE Transactions on Signal Processing, 42(4), 812-824.

**9.12 Harmonic Inpainting**

[12] Chui, C. K. (2009). An MRA approach to surface completion and image inpainting. Applied and Computational Harmonic Analysis, 26(2), 270-276.

**9.13 Mumford-Shah Inpainting**

[13] Ambrosio, L., Fusco, N., & Hutchinson, J. E. (2003). Higher integrability of the gradient and dimension of the singular set for minimisers of the Mumford–Shah functional. Calculus of Variations and Partial Differential Equations, 16(2), 187-215.

##### CHAPTER 10: APPENDIX

**10.1 Summary of the meetings**

**Meeting 1:**

How to identify the fence, explain the gaussian filter to reprocess the image, It means the image can be focus on the background or fence.

**Meeting 2:**

After gaussian filter reprocess images, how to identify the fence area and find the mask.

Explain the code about the apple image combination with orange image.

To identify the fence

1. Focus
2. Manule
3. Learning

**Meeting 3:**

Discussed and explained the 2016 paper, provide improved methods for our image processing and provide the suggest for inpaint image.

**Meeting 4:**

Post-processing the fence's mask image, we find the connected components to remove the noise in the fence's mask image. We eventually reduced the coverage area expectations of the fences, while retaining the local information fence near.

**Meeting 5:**

We want to use the Gaussian pyramid and Laplacian pyramid to do the inpaint, the content in the Gaussian pyramid will lose information during the process of Gaussian blur and downsampling, we hope to be able to preserve the information lost in the process of building the Gaussian pyramid in some way.

**Meeting 6:**

We try to use the Total Variation inpainting to do the inpaint, but this inpainting method cannot bring us regain the details of an image, it can only repair large areas and few details.

**10.2 Flow chart**

1. Data collection

2. Image taken

3. Programming

3.1. Image filter

3.2. Image processing

3.2.1. Gaussian Blur

3.2.2. Extract parts of fence

3.2.3. Optimize the raised fence part

3.3. Inpainting

3.3.1. Image downsampling and texture restoration

3.3.2. Inpainting

3.3.3. Upsampling of inpainted region

3.4. Inpainting other methods

4. Program test

5. Debug

6. Demo testing

7. Result statistics