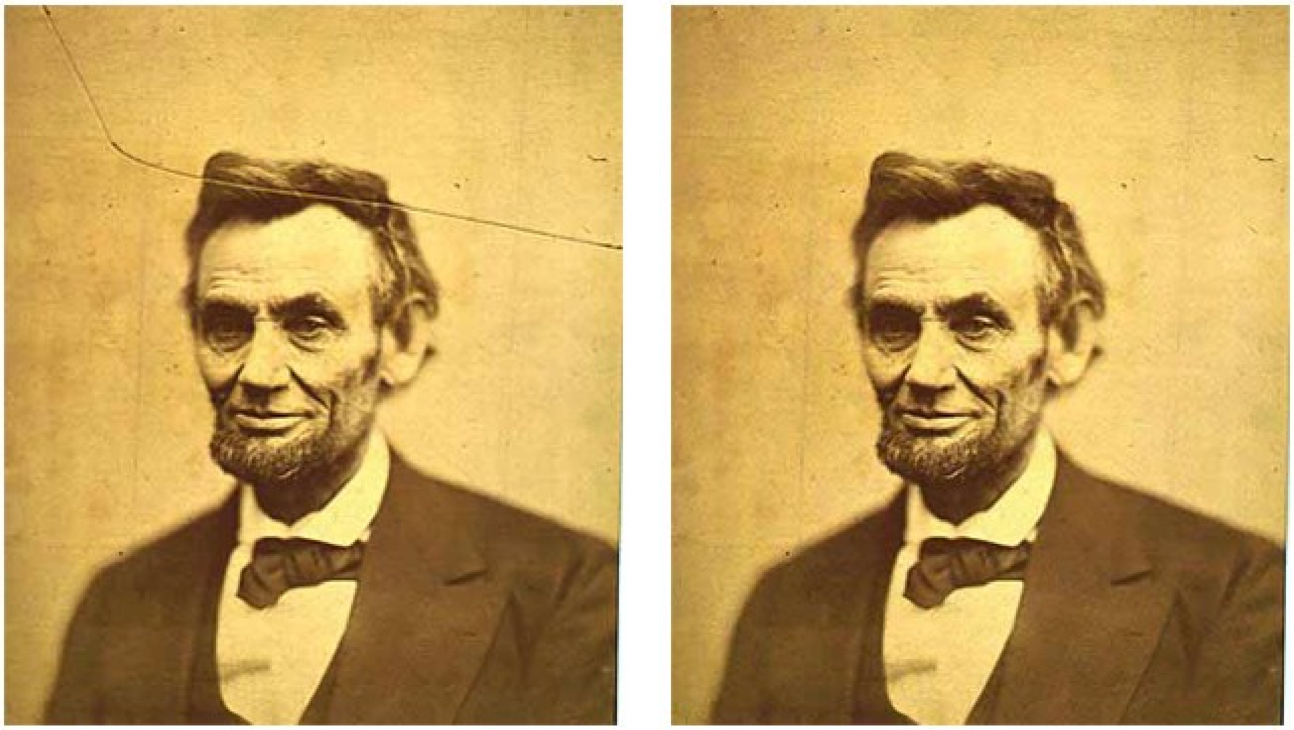
Inpainting Experiments and Research for new Techniques

by Kamaladdin Ahmadzada



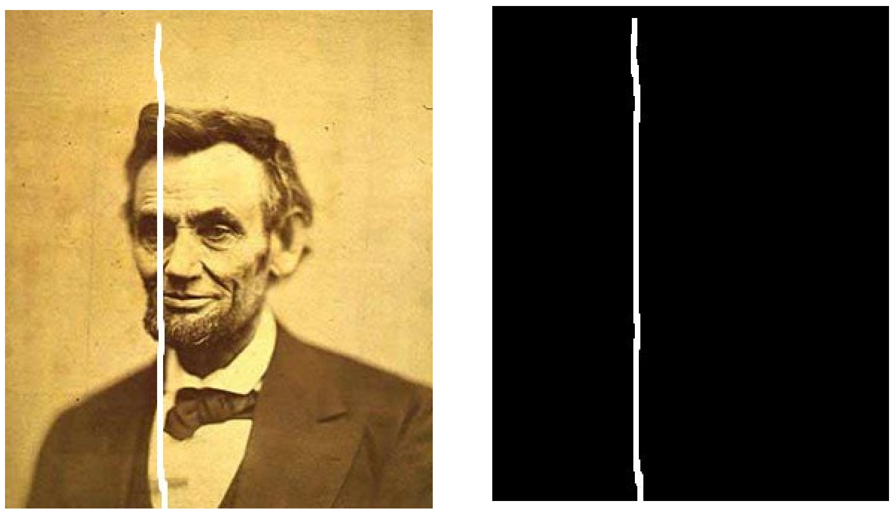
Pic 1. This Picture’s defects eliminated by Matlab inpainting algorithm

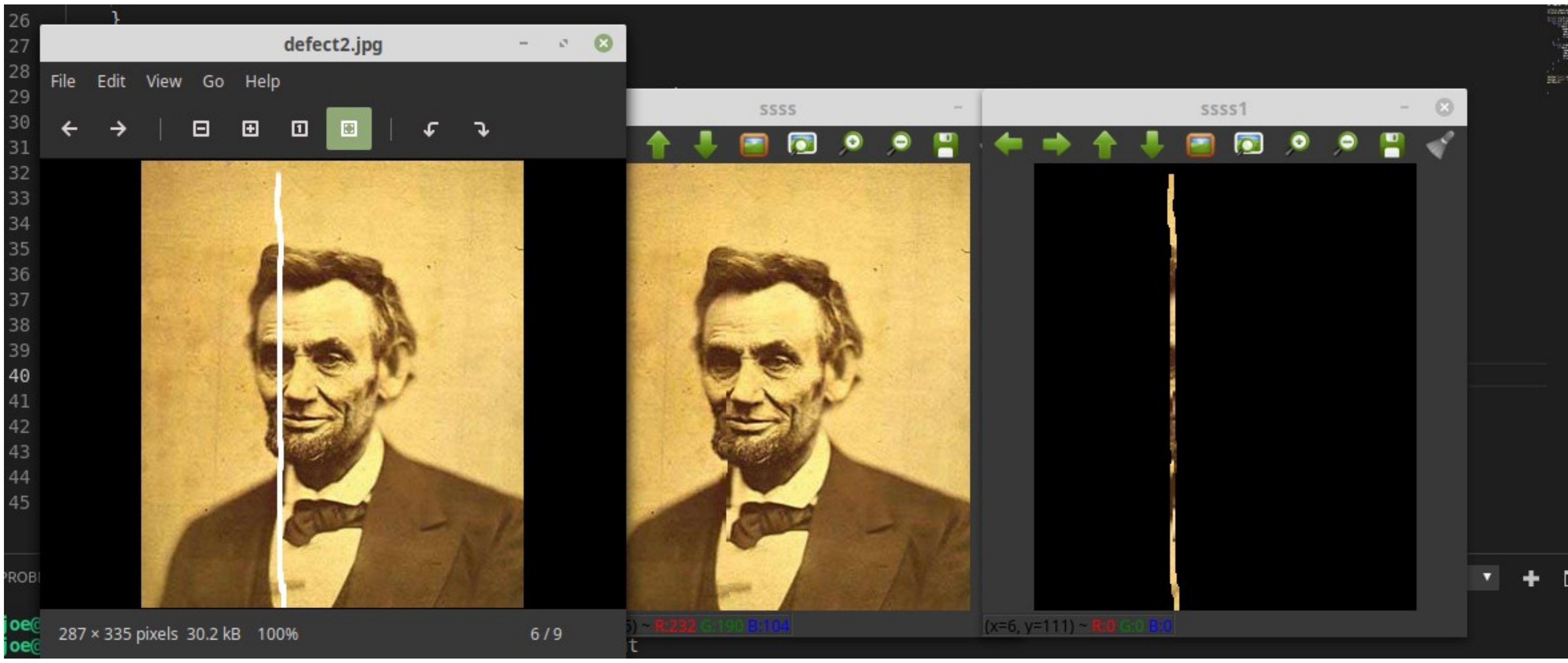
But my purpose is not to use existing algorithms instead creating new kernels or methods for improving and experimenting results.

Test No:1

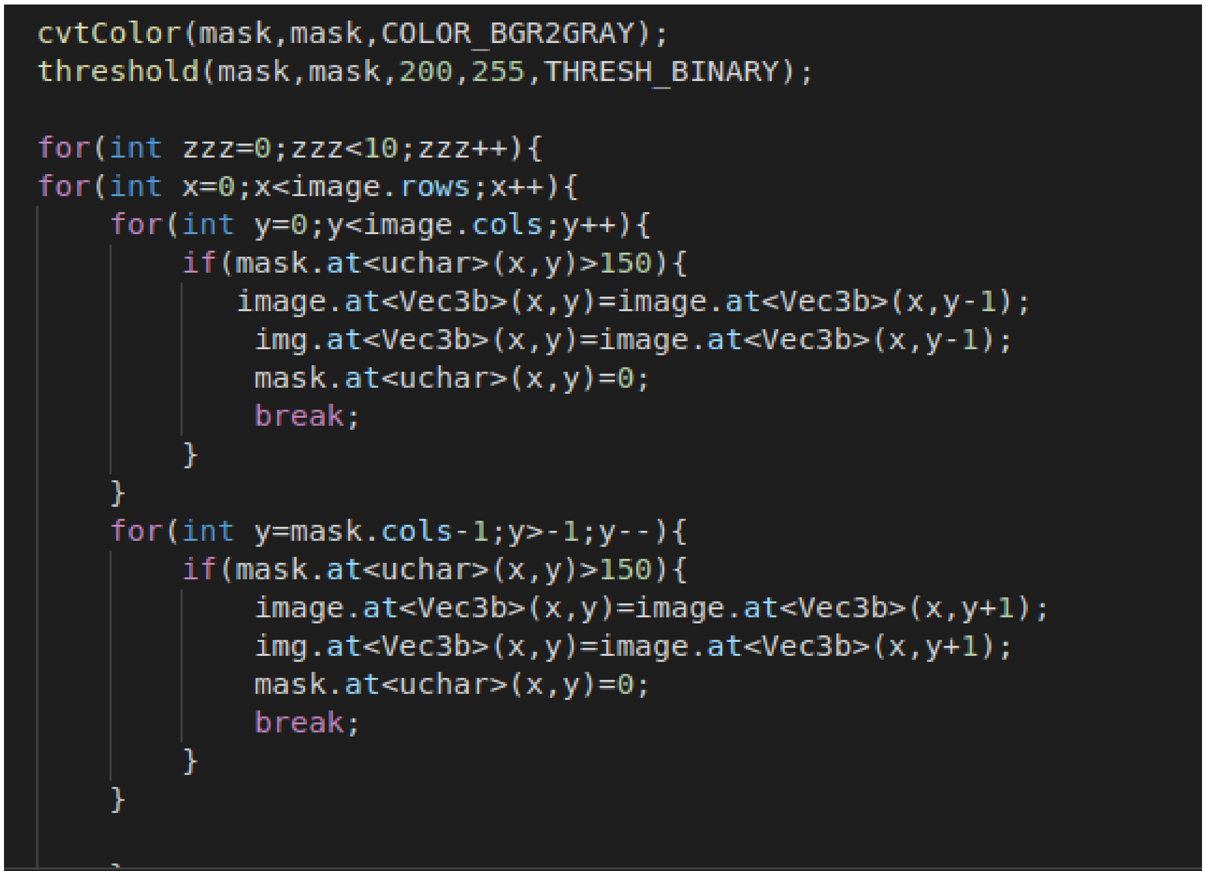
Instead of diving complex defects I chose little bit easy one :

Defects in Black background (Mask) must given by user.



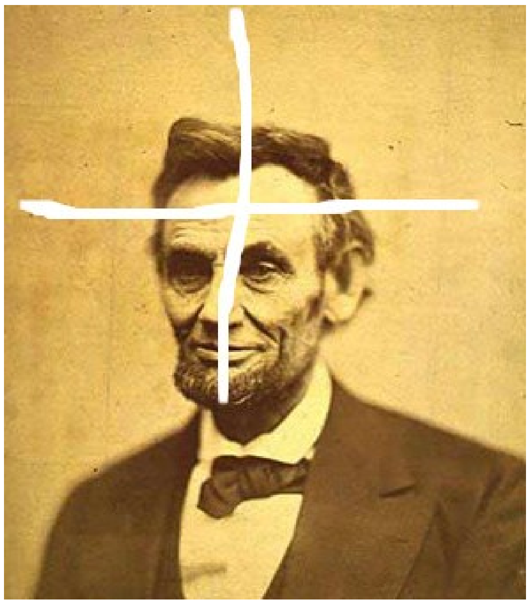

defect2.jpg
File Edit View Go Help
287 × 335 pixels 30.2 kB 100% 6 / 9
ssss
ssss1
R:232 G:190 B:104
(x=6, y=111) ~ R:0 G:0 B:0

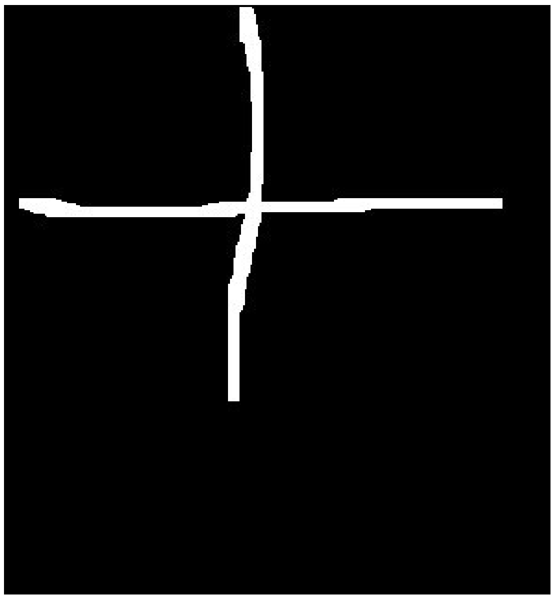
First try I think really impressive : Here is my method which belongs to me : First It analyze mask and trying to find white pixels after that it recover image starting from right and left white pixel and uses previous and next pixel values . But This method not perfect for horizontal cases ! . How to overcome this obstacle ? Well same principle but this time previous ans next row values we have to use because otherwise our algorithm returns wrong pixel values which is not match our image !

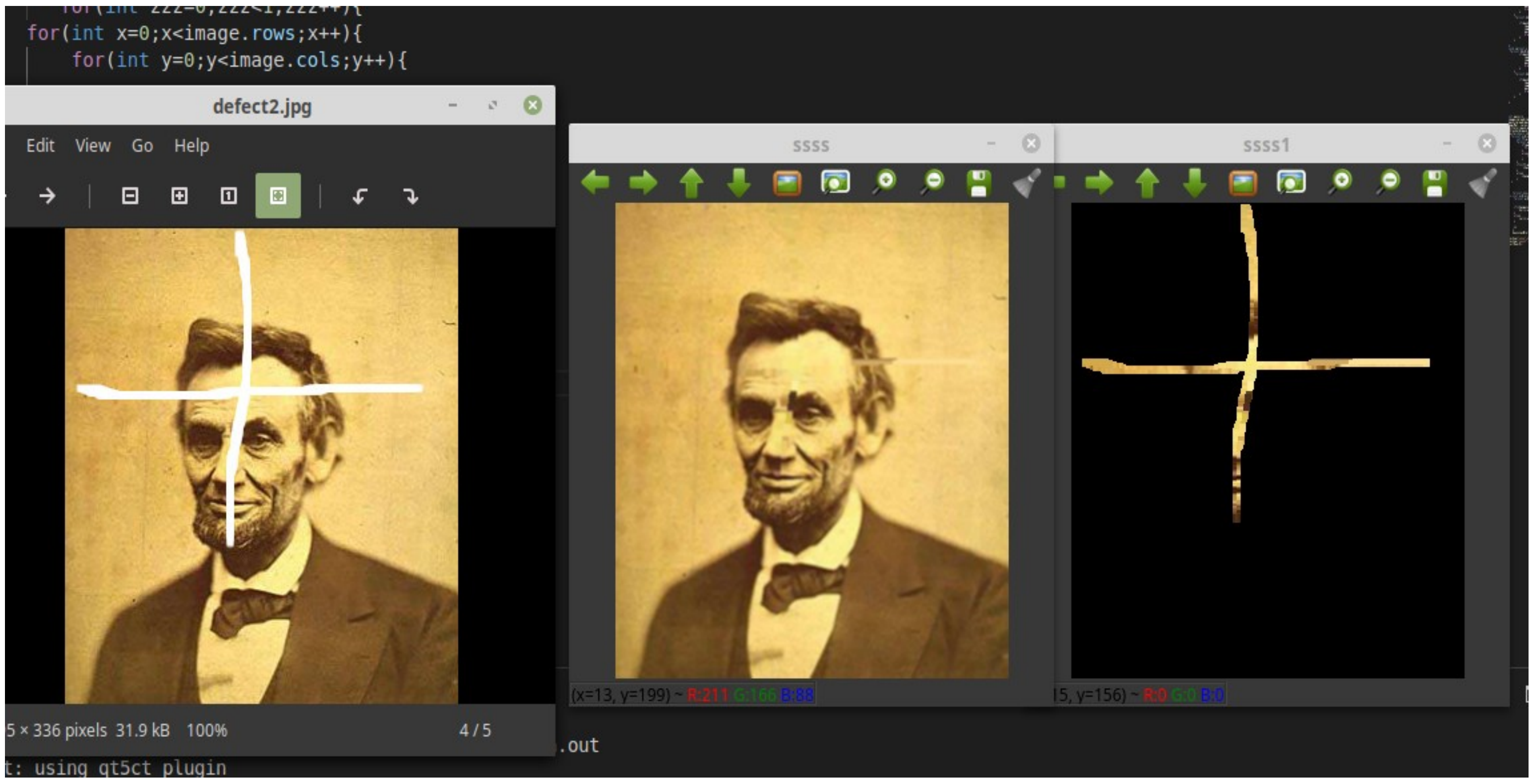


```
cvtColor(mask,mask,COLOR_BGR2GRAY);
threshold(mask,mask,200,255,THRESH_BINARY);

for(int zzz=0;zzz<10;zzz++){
for(int x=0;x<image.rows;x++){
    for(int y=0;y<image.cols;y++){
        if(mask.at<uchar>(x,y)>150){
           image.at<Vec3b>(x,y)=image.at<Vec3b>(x,y-1);
            img.at<Vec3b>(x,y)=image.at<Vec3b>(x,y-1);
            mask.at<uchar>(x,y)=0;
            break;
        }
    }
    for(int y=mask.cols-1;y>-1;y--){
        if(mask.at<uchar>(x,y)>150){
            image.at<Vec3b>(x,y)=image.at<Vec3b>(x,y+1);
            img.at<Vec3b>(x,y)=image.at<Vec3b>(x,y+1);
            mask.at<uchar>(x,y)=0;
            break;
        }
    }
```

Test 2: I think first test was successful that is why I started second test more complex one : This case we have vertical and horizontal defects !
 Same principle but we have to specify in which row algorithm has to do vertical recovery or vice versa. To do that it calculates white region percentage for every row and if row’s white pixel percentage is too much then it applies vertical recovery method .
.



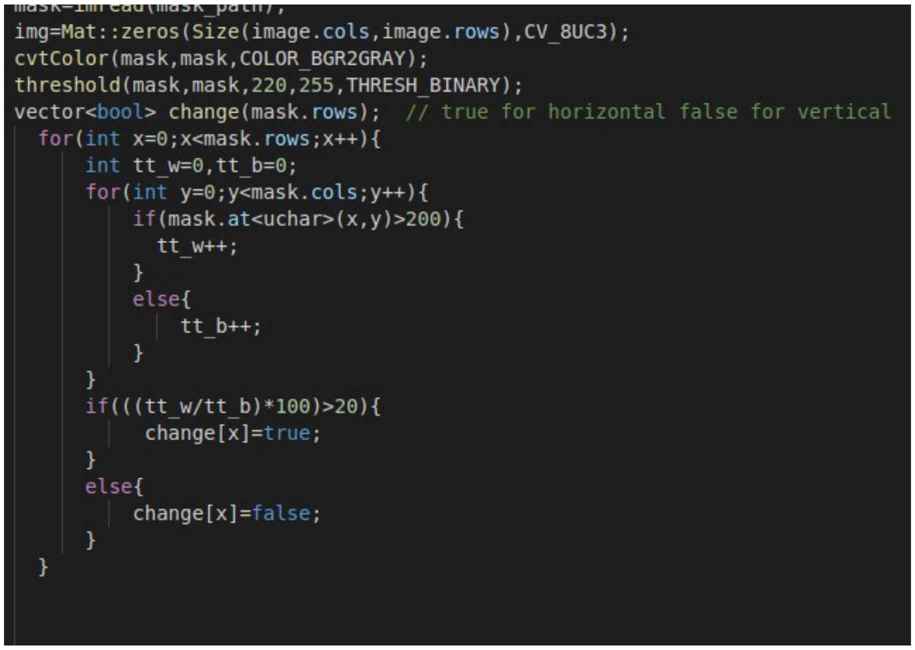




```
for(int x=0;x<image.rows;x++){
    for(int y=0;y<image.cols;y++){
```

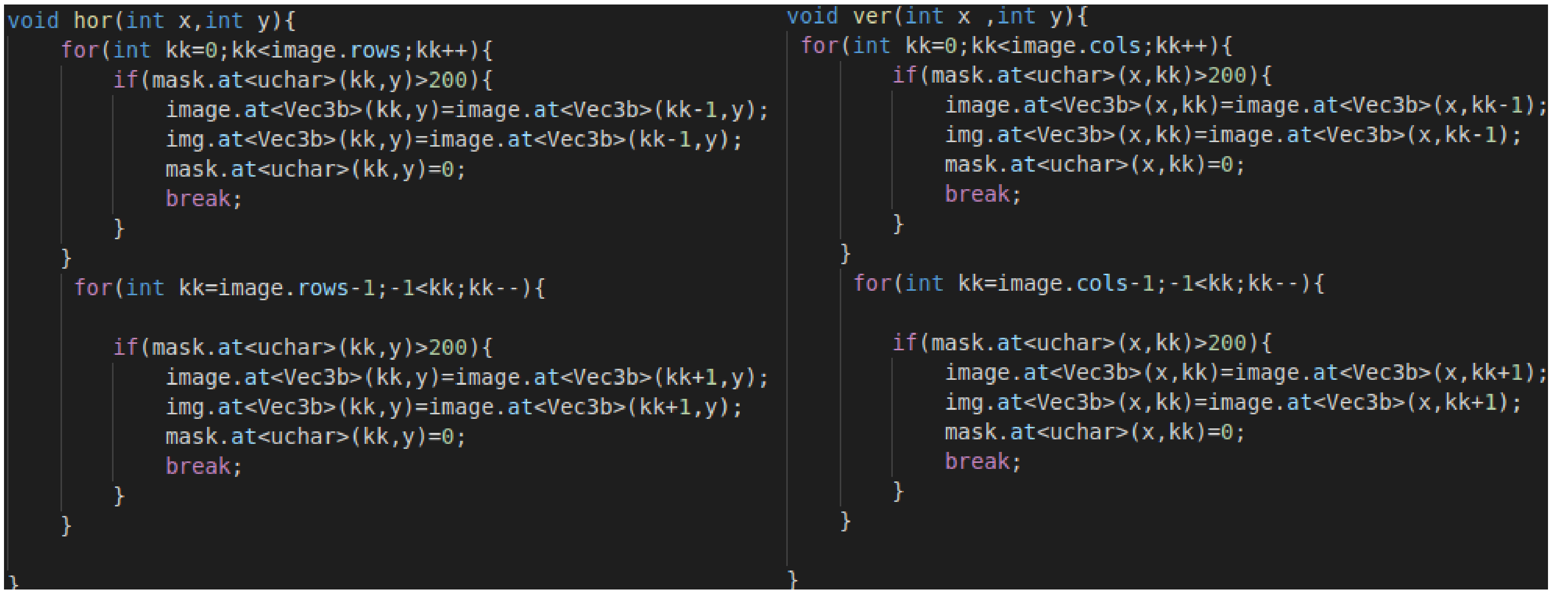
```
5 × 336 pixels 31.9 kB 100%    4 / 5
(x=13, y=199) ~ R:211 G:166 B:88
15, y=156) ~ R:0 G:0 B:0
.out
t: using qt5ct plugin
```

here is Algorithm codes:
In picture below you can see code that calculates pixel percentages for every row . And it stored in change vector for further use.



```
mask=imread(mask_path);
img=Mat::zeros(Size(image.cols,image.rows),CV_8UC3);
cvtColor(mask,mask,COLOR_BGR2GRAY);
threshold(mask,mask,220,255,THRESH_BINARY);
vector<bool> change(mask.rows);  // true for horizontal false for vertical
  for(int x=0;x<mask.rows;x++){
      int tt_w=0,tt_b=0;
      for(int y=0;y<mask.cols;y++){
          if(mask.at<uchar>(x,y)>200){
            tt_w++;
          }
          else{
              tt_b++;
          }
      }
      if(((tt_w/tt_b)*100)>20){
           change[x]=true;
      }
      else{
          change[x]=false;
      }
  }
```

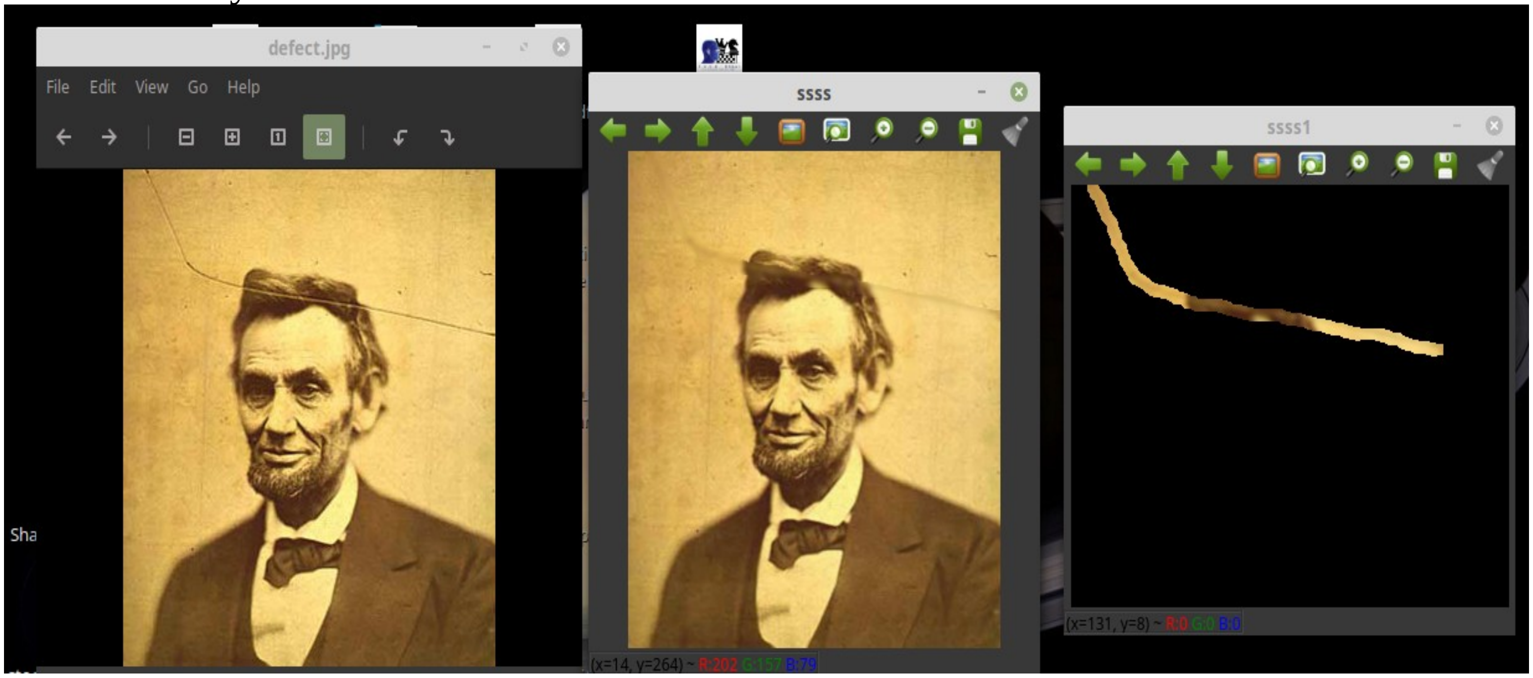
For Vertical and Horizontal rescuer : they approach image by different direction and use previous and next pixel values for recovery.



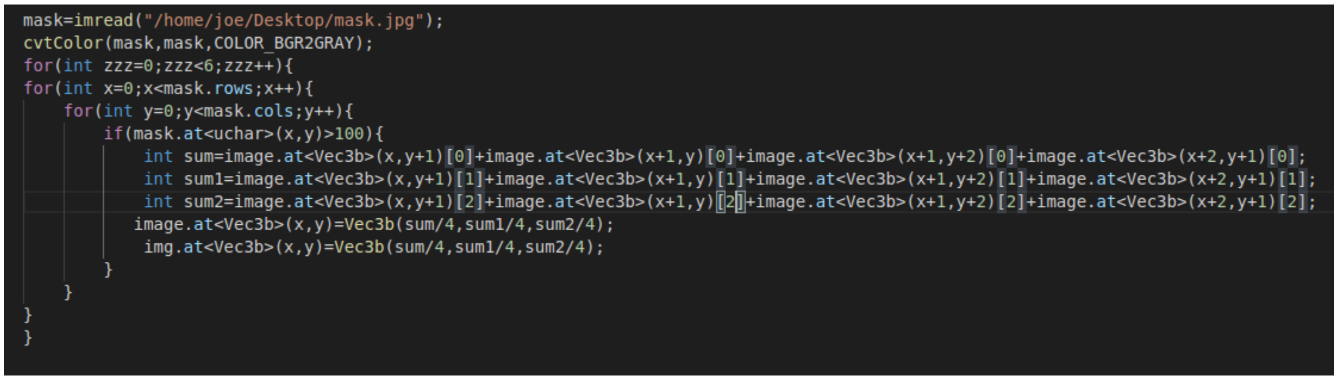
```
void hor(int x,int y){
    for(int kk=0;kk<image.rows;kk++){
        if(mask.at<uchar>(kk,y)>200){
            image.at<Vec3b>(kk,y)=image.at<Vec3b>(kk-1,y);
            img.at<Vec3b>(kk,y)=image.at<Vec3b>(kk-1,y);
            mask.at<uchar>(kk,y)=0;
            break;
        }
    }
     for(int kk=image.rows-1;-1<kk;kk--){

        if(mask.at<uchar>(kk,y)>200){
            image.at<Vec3b>(kk,y)=image.at<Vec3b>(kk+1,y);
            img.at<Vec3b>(kk,y)=image.at<Vec3b>(kk+1,y);
            mask.at<uchar>(kk,y)=0;
            break;
        }
    }


}
```

```
void ver(int x ,int y){
 for(int kk=0;kk<image.cols;kk++){
        if(mask.at<uchar>(x,kk)>200){
            image.at<Vec3b>(x,kk)=image.at<Vec3b>(x,kk-1);
            img.at<Vec3b>(x,kk)=image.at<Vec3b>(x,kk-1);
            mask.at<uchar>(x,kk)=0;
            break;
        }
    }
     for(int kk=image.cols-1;-1<kk;kk--){

        if(mask.at<uchar>(x,kk)>200){
            image.at<Vec3b>(x,kk)=image.at<Vec3b>(x,kk+1);
            img.at<Vec3b>(x,kk)=image.at<Vec3b>(x,kk+1);
            mask.at<uchar>(x,kk)=0;
            break;
        }
    }


}
```

Test 3 : In this case I used arithmetic sum of next and previous pixels for recovery.



Because of arithmetic mean in picture above result looks little bit blurred .


```
mask=imread("/home/joe/Desktop/mask.jpg");
cvtColor(mask,mask,COLOR_BGR2GRAY);
for(int zzz=0;zzz<6;zzz++){
for(int x=0;x<mask.rows;x++){
    for(int y=0;y<mask.cols;y++){
        if(mask.at<uchar>(x,y)>100){
            int sum=image.at<Vec3b>(x,y+1)[0]+image.at<Vec3b>(x+1,y)[0]+image.at<Vec3b>(x+1,y+2)[0]+image.at<Vec3b>(x+2,y+1)[0];
            int sum1=image.at<Vec3b>(x,y+1)[1]+image.at<Vec3b>(x+1,y)[1]+image.at<Vec3b>(x+1,y+2)[1]+image.at<Vec3b>(x+2,y+1)[1];
            int sum2=image.at<Vec3b>(x,y+1)[2]+image.at<Vec3b>(x+1,y)[2]+image.at<Vec3b>(x+1,y+2)[2]+image.at<Vec3b>(x+2,y+1)[2];
           image.at<Vec3b>(x,y)=Vec3b(sum/4,sum1/4,sum2/4);
            img.at<Vec3b>(x,y)=Vec3b(sum/4,sum1/4,sum2/4);
        }
    }
}
}
```

Conclusion:

Off Course all of these algorithms so primitive but in order to achieve best recovery advanced techniques must be used .Examples for other techniques : Inpainting by Segmentation , search based inpainting , hybrid inpainting are some of them .With new techniques such as DNN or CNN based applications makes perfect job. Nvidia ‘s new algorithm recover image even some of the parts are erased .