Advancde computer vision

高等電腦視覺

111618018 自動化碩一 吳祐毅

Homework 1

一、 C 語言影像處理(without other library)

1. Image Read/Write

Image 為儲存影像每一個像素數值，其大小為長*寬*(rgb 三個通道)，header 為儲存.bmp 檔案格式，其檔案格式大小為 54bytes。

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    unsigned char *image;
    int xsize = 512;
    int ysize = 512;

    image = (unsigned char *)malloc((size_t)xsize * ysize * 3);
    if (image == NULL)
        return -1;

    char fname_bmp[128];
    FILE *fp;

    unsigned char header[54] = {
    0x42, 0x4d, 0, 0, 0, 0, 0, 0, 0, 0,
    54, 0, 0, 0, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 24, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0
    };
```

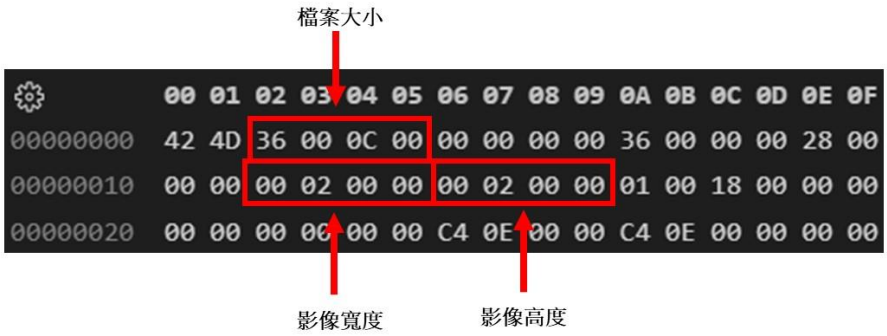
使用 fread 來讀取.bmp 圖像資訊，先讀取影像檔案格式，在讀取影像每一個像素資訊。

```c
    sprintf(fname_bmp, "%s.bmp", "lena");

    if (!(fp = fopen(fname_bmp, "rb")))
        return -1;

    fread(header, sizeof(unsigned char), 54, fp);
    fread(image, sizeof(unsigned char), (size_t)(long)xsize * ysize * 3, fp);
```

file_size 表示檔案大小，其檔案大小為 786,486 個 bytes，而 width 和 height 分別為影像尺寸大小，其大小為 512*512 個像素。在.bmp 的檔案格式裡，第 3~6 的位置為表示影像儲存的大小，第 19~22 為影像寬度，第 23~26 為影像高度，每一項都使用了 4bytes 來儲存，並且每一個 bytes 使用 16 進制的儲存方式，在這裡計算每一個 byte 使用了位元運算的方式來進行儲存，




```
31      long file_size = (long)xsize * (long)ysize * 3 + 54;
32      long width, height;
33
34      header[2] = (unsigned char)(file_size &0x000000ff);
35      header[3] = (file_size >> 8) & 0x000000ff;
36      header[4] = (file_size >> 16) & 0x000000ff;
37      header[5] = (file_size >> 24) & 0x000000ff;
38
39      width = xsize;
40      header[18] = width & 0x000000ff;
41      header[19] = (width >> 8) &0x000000ff;
42      header[20] = (width >> 16) &0x000000ff;
43      header[21] = (width >> 24) &0x000000ff;
44
45      height = ysize;
46      header[22] = height &0x000000ff;
47      header[23] = (height >> 8) &0x000000ff;
48      header[24] = (height >> 16) &0x000000ff;
49      header[25] = (height >> 24) &0x000000ff;
```

最後使用 fwrite 將影像進行儲存，先儲存影像檔案格式，再儲存影像的像素資訊。

```
51      sprintf(fname_bmp, "%s.bmp", "lena_write");
52      if (!(fp = fopen(fname_bmp, "wb")))
53          return -1;
54      fwrite(header, sizeof(unsigned char), 54, fp); //標頭檔先寫入
55      fwrite(image, sizeof(unsigned char), (size_t)(long)xsize * ysize * 3, fp);
56      fclose(fp);
57
58      free(image);
59  }
```

2. Generate a color negative image

透過上一題的程式範本進行修改，由於 rgb 的表示範圍為 0~255，所以將每一個像素的值進行下圖的運算，即可得到負影像。

```
56      for(int i = 0; i < (long)xsize * (long)ysize * 3; i++){
57          image[i] = 255 - image[i];
58      }
```

結果圖

3. Split and reassemble into new image

Image_rot_90、Image_rot_180、Image_rot_270 分別儲存以原始影像並旋轉 90 度、180 度、270 度的影像，透過前一次旋轉後的影像為基礎並旋轉了 3 次才得到旋轉 270 度的影像，而旋轉的原理為:

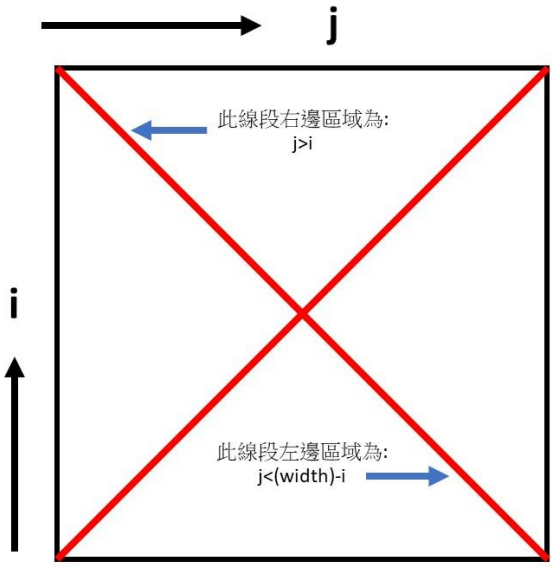
旋轉後影像的二維座標(I, j) = 原始影像的二維座標(j, height – I - 1)

```
71      //轉 90 度
72      for(int i = 0; i < height; i++){
73          for(int j = 0; j < width; j++){
74              for(int k = 0; k < 3; k++){
75                  image_rot_90[(i * width + j) * 3 + k] = image[(j * width + height - i - 1) * 3 + k];
76              }
77          }
78      }
79      //轉180度
80      for(int i = 0; i < height; i++){
81          for(int j = 0; j < width; j++){
82              for(int k = 0; k < 3; k++){
83                  image_rot_180[(i * width + j) * 3 + k] = image_rot_90[(j * width + height - i - 1) * 3 + k];
84              }
85          }
86      }
87
88      //轉270度
89      for(int i = 0; i < height; i++){
90          for(int j = 0; j < width; j++){
91              for(int k = 0; k < 3; k++){
92                  image_rot_270[(i * width + j) * 3 + k] = image_rot_180[(j * width + height - i - 1) * 3 + k];
93              }
94          }
95      }
```

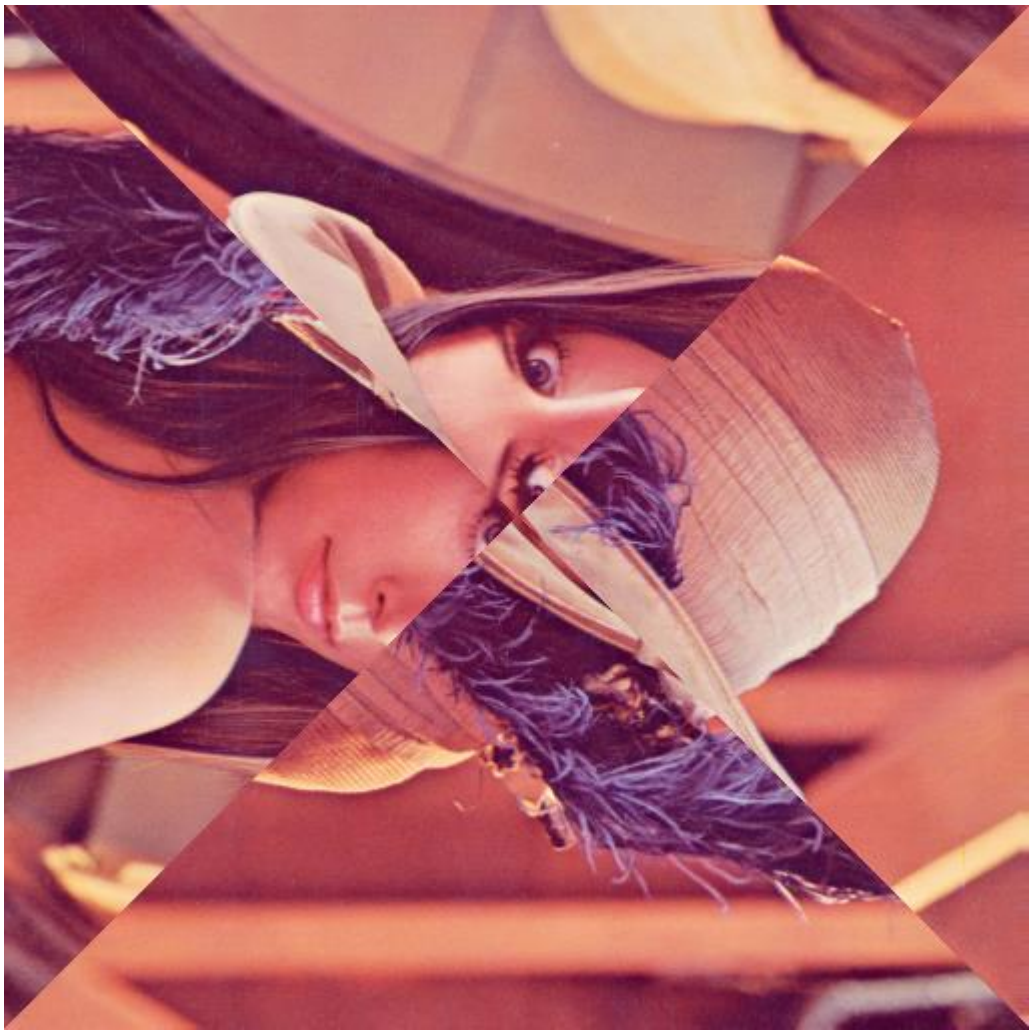
在切割的部分，我們以旋轉 90 度的影像為基準進行切割，並且透過迴圈的方式瀏覽每一個像素的二為座標，每一個像素的二為座標若符合以下情況，則需要將該像素改為旋轉 270 度的影像。

下方區域: j < I && j > (width) – i

上方區域: j > I && j < (width) – i



```
    //切割
    image_split = image_rot_90; //先存旋轉90度影像
    for(int i = 0; i < height; i++){
        for(int j = 0; j < width; j++){
            for(int k = 0; k < 3; k++){
                if((j >= i && j <= (width - i)) || ((j <= i && j >= (width - i)))){
                    image_split[(i * width + j) * 3 + k] = image_rot_270[(i * width + j) * 3 + k]; //再存270度影像
                }
            }
        }
    }
```



結果圖

＊ Bonus1: Resize the image one-half size

ysize_one_half 為一半大小的高度。

```
int main() {
    unsigned char *image, *image_one_half;
    int xsize = 512;
    int ysize = 512;
    int ysize_one_half = 288;
```

需要更改.bmp 檔案格式，在檔案大小儲存方面需要更改為 512 * 288 * 3 + 54 = 442422 bytes，而高度也需要改為 288 個像素。

```
36      long file_size = (long)xsize * (long)ysize_one_half * 3 + 54;
37      long width, height;
38
39      header[2] = (unsigned char)(file_size &0x000000ff);
40      header[3] = (file_size >> 8) & 0x000000ff;
41      header[4] = (file_size >> 16) & 0x000000ff;
42      header[5] = (file_size >> 24) & 0x000000ff;
43
44      width = xsize;
45      header[18] = width & 0x000000ff;
46      header[19] = (width >> 8) &0x000000ff;
47      header[20] = (width >> 16) &0x000000ff;
48      header[21] = (width >> 24) &0x000000ff;
49
50      height = ysize_one_half;
51      header[22] = height &0x000000ff;
52      header[23] = (height >> 8) &0x000000ff;
53      header[24] = (height >> 16) &0x000000ff;
54      header[25] = (height >> 24) &0x000000ff;
```

由於影像讀取順序是由左下到右上，所以在儲存原始大小的一半影像時，需要從原始高度的一半以上開始儲存。

```
61      for(int i = 0; i < (long)xsize * (long)ysize_one_half * 3; i++){ //影像由左下到右上
62          image_one_half[i] = image[((long)xsize * (long)(ysize - ysize_one_half) * 3) + i];
63      }
```

結果圖

＊ Bonus2: Resize the image double size

需要先更改檔案儲存大小為 1024 * 1024 * 3 + 54 = 3,145,782 bytes

```
37      long file_size = (long)xsize_double * (long)ysize_double * 3 + 54;
38      long width, height;
39
40      header[2] = (unsigned char)(file_size &0x000000ff);
41      header[3] = (file_size >> 8) & 0x000000ff;
42      header[4] = (file_size >> 16) & 0x000000ff;
43      header[5] = (file_size >> 24) & 0x000000ff;
44
45      width = xsize_double;
46      header[18] = width & 0x000000ff;
47      header[19] = (width >> 8) &0x000000ff;
48      header[20] = (width >> 16) &0x000000ff;
49      header[21] = (width >> 24) &0x000000ff;
50
51      height = ysize_double;
52      header[22] = height &0x000000ff;
53      header[23] = (height >> 8) &0x000000ff;
54      header[24] = (height >> 16) &0x000000ff;
55      header[25] = (height >> 24) &0x000000ff;
```

放大後的每一個像素數值需要從原影像對應的位置進行複製，其位置對應關係為:

放大後的二維座標 / 2 = 原影像的二維座標

對應到原始二為座標後需要考慮該座標與周圍鄰近的 4 個座標像素數值進行加權加總後得到:

number = (右上 + 右下 + 左上 + 左下) / 4

```
float number = 0;
for(int i = 0; i < height; i++){ //影像由左下到右上
    for(int j = 0; j < width; j++){
        for(int k = 0; k < 3; k++){
            number = 0;
            int a_1 = i / 2 - 0.5; //上下左右座標
            int a_2 = i / 2 + 0.5;
            int b_1 = j / 2 - 0.5;
            int b_2 = j / 2 + 0.5;
            number = (image[(a_1 * xsize + b_1) * 3 + k] + image[(a_1 * xsize + b_2) * 3 + k]
            + image[(a_2 * xsize + b_1) * 3 + k] + image[(a_2 * xsize + b_2) * 3 + k]) / 4;
            image_double[(i * width + j) * 3 + k] = number;
        }
    }
}
```

二、 OpenCV 影像處理

1. Image Read/Write

Imread 為寫入影像，imwrite 為寫出影像

```
10      //ans1
11      Mat img = imread("lena.jpg");
12      imwrite("lena_out.jpg", img);
13      imshow("show", img);
```

2. Generate a color negative image

bgr 為儲存每一個像素的 rgb 三個值，其變數形式為 Vec3b

```
15      //ans2
16      Mat img_negative;
17      img.copyTo(img_negative); //複製原始影像
18      for (int i = 0; i < img.rows; i++) {
19          for (int j = 0; j < img.cols; j++) {
20              Vec3b bgr = img_negative.at<Vec3b>(i, j); //獲取rgb的三個值
21              img_negative.at<Vec3b>(i, j)[0] = 255 - bgr[0];
22              img_negative.at<Vec3b>(i, j)[1] = 255 - bgr[1];
23              img_negative.at<Vec3b>(i, j)[2] = 255 - bgr[2];
24          }
25      }
26      imwrite("lena_negative.jpg", img_negative);
27      imshow("img_negative", img_negative);
```

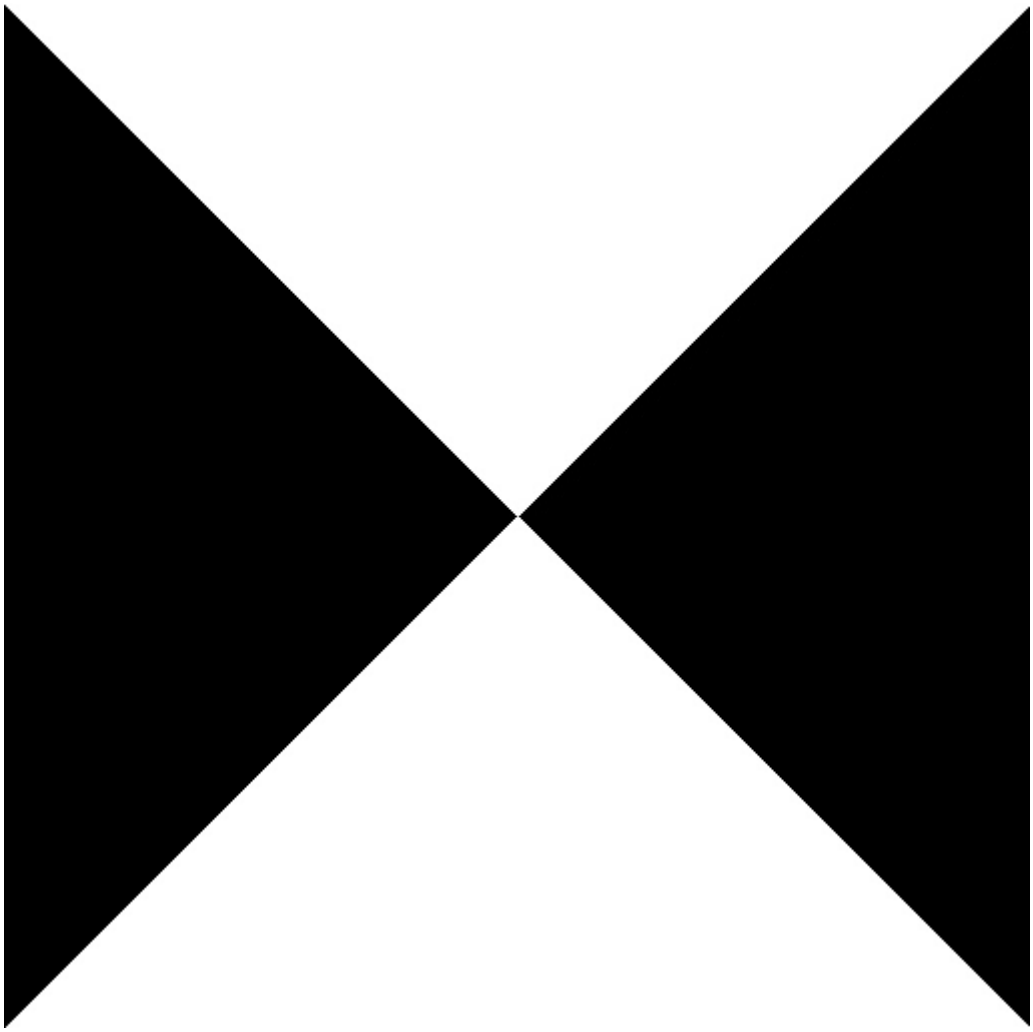
3. Split and reassemble into new image

Point2f center 為取得影像旋轉中心點座標，getRotationMatrix()函式為取得旋轉矩陣，warpAffine()函式為對影像進行旋轉處理。

```
29      //ans3
30      Mat img_rotate_90;
31      Size img_rotate_90_sz(img.cols, img.rows);
32      Point2f center_90(static_cast<float>(img.cols / 2.), static_cast<float>(img.rows / 2.));
33      Mat rot_mat_90 = getRotationMatrix2D(center_90, 90, 1.0);
34      warpAffine(img, img_rotate_90, rot_mat_90, img_rotate_90_sz, cv::INTER_LINEAR, cv::BORDER_REPLICATE);
35      //imshow("img_rotate_90", img_rotate_90);
36
37      Mat img_rotate_270;
38      Size img_rotate_270_sz(img.cols, img.rows);
39      Point2f center_270(static_cast<float>(img.cols / 2.), static_cast<float>(img.rows / 2.));
40      Mat rot_mat_270 = getRotationMatrix2D(center_270, 270, 1.0);
41      warpAffine(img, img_rotate_270, rot_mat_270, img_rotate_270_sz, cv::INTER_LINEAR, cv::BORDER_REPLICATE);
42      //imshow("img_rotate_270", img_rotate_270);
```

先選擇 roi 的區域其繪製的外圍座標 pts_up、pts_down，在使用 drawContours 函式將 roi 的區域繪製出來。

```
44      Mat roi = Mat::zeros(img.size(), CV_8U);
45      vector<vector<Point>> contour_up, contour_down;
46      vector<Point> pts_up, pts_down;
47      pts_up.push_back(Point(0, 0));
48      pts_up.push_back(Point(256, 256));
49      pts_up.push_back(Point(512, 0));
50      contour_up.push_back(pts_up);
51
52      pts_down.push_back(Point(0, 512));
53      pts_down.push_back(Point(256, 256));
54      pts_down.push_back(Point(512, 512));
55      contour_down.push_back(pts_down);
56
57      drawContours(roi, contour_up, 0, Scalar::all(255), -1);
58      drawContours(roi, contour_down, 0, Scalar::all(255), -1);
59      imwrite("roi.jpg", roi);
```

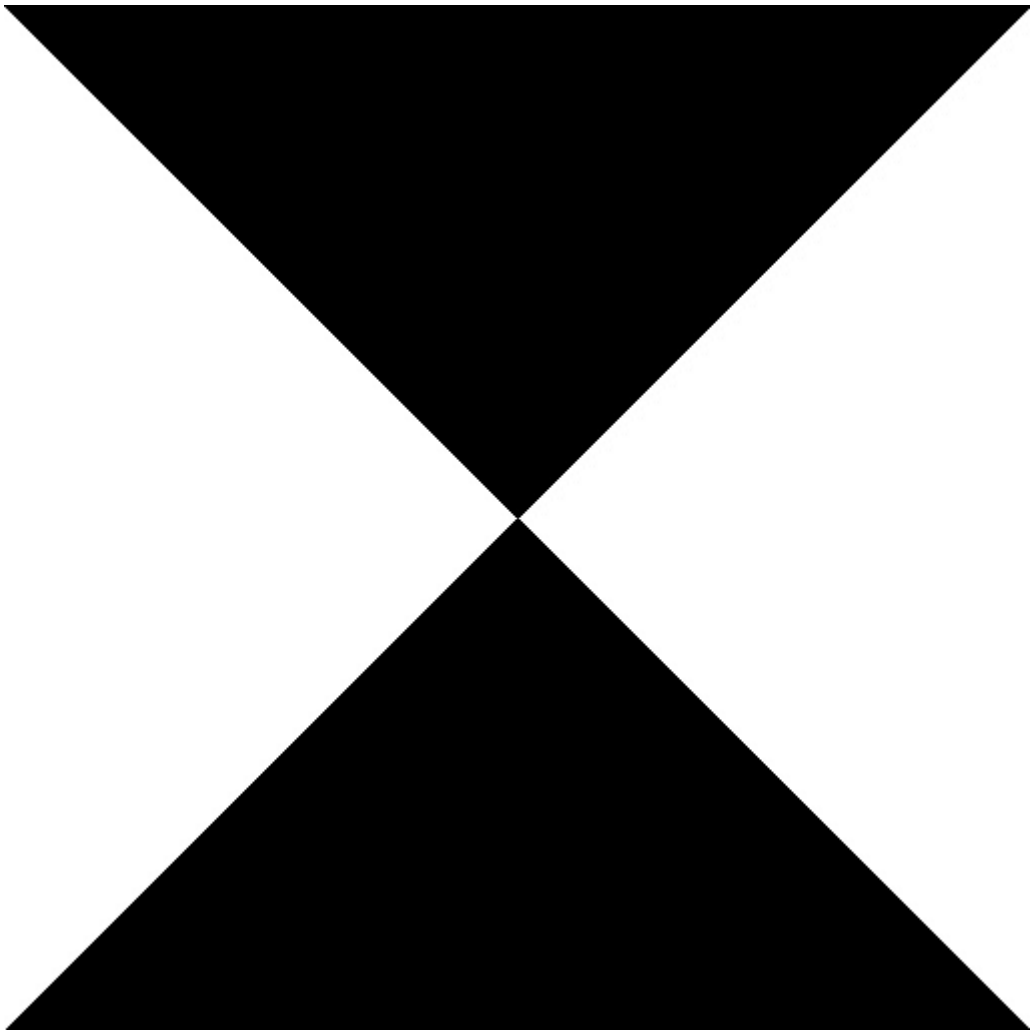


接著使用 toCopy 函示將旋轉 90 的影像複製並和 roi 的影像進行位元運算後得到如下圖:

```
60      Mat img_split;
61      img_rotate_90.copyTo(img_split, roi);
62      imwrite("lena_split_1.jpg", img_split);
```

用 roi2 得到 roi 的負影像，並且用旋轉 270 度的影像和 roi2 進行位元運算後得到最後的結果。

```
63      Mat roi2;
64      roi.copyTo(roi2); //複製原始影像
65
66      for (int i = 0; i < roi2.rows; i++) {
67          for (int j = 0; j < roi2.cols; j++) {
68              int pv = roi2.at<uchar>(i, j);//得到像素值
69              roi2.at<uchar>(i, j) = 255 - pv;
70          }
71      }
72      imwrite("roi2.jpg", roi2);
73      img_rotate_270.copyTo(img_split, roi2);
74      imwrite("lena_split.jpg", img_split);
75      imshow("img_split", img_split);
```



roi2

結果圖

＊ Bonus

第一使用 resize 函式調整影像大小，將影像放大兩倍。第二取得一半影像的方式為使用 Rect 函式選取原始影像的範圍在另外儲存為新的影像。

```
77      //bonus
78      Mat double_image;
79      resize(img, double_image, Size(img.cols * 2, img.rows * 2), 0, 0, INTER_LINEAR);
80      imwrite("double_image.jpg", double_image);
81      imshow("double_image", double_image);
82      Mat one_half_size_image = img(Rect(0, 0, img.cols, img.rows / 2));
83      imshow("one_half_size_image", one_half_size_image);
84      imwrite("one_half_size_image.jpg", one_half_size_image);
```