

國立高雄應用科技大學

影像處理暨實習報告



題目：

處理圖片雜訊、顯示任意圖形

班級：四電三丙

學號：1102104305

姓名：羅左欣

指導老師：謝勝治

目錄

零、相關說明

壹、加入圖片雜訊

- (一) 胡椒鹽(salt-and-pepper)雜訊
- (二) 高斯(Gaussian)雜訊
- (三) 加入”胡椒鹽(salt-and-pepper)”和高斯(Gaussian)雜訊

貳、消除圖片雜訊

(一) 空間域濾波

- 1. 平均濾波器 (Mean Filter) - Lecture 5-17
- 2. 中間值濾波器 (Median Filter) - Lecture 5-21
- 3. 適應性濾波器 (Adaptive Filter) - Lecture 5-25 ~ 5-27

(二) 頻域濾波

- 4. 平滑化濾波器 (Gaussian Filter) - Lecture 4-38 ~ 4-39
- 5. 銳化濾波器 (Laplacian Filter) - Lecture 4-48 ~ 4-51

(三) 其他

- 6. 傅立葉轉換 (Fourier Transform) - Lecture 4-2

參、心得

零、相關說明

◎ 使用工具：OpenCV - 3.1.0

◎ 作業系統：Lubuntu 16.04

◎ 程式語言：C++

◎ 資料來源：

1. OpenCV.org - <http://opencv.org/>

2. Add Salt&Pepper Noise - https://github.com/timlrentse/Add_Salt-Pepper_Noise

3. 阿州的程式教學 - <http://monkeycoding.com/?p=864>

◎ 本報告中所有範例程式以及圖片，皆存放在以下位置：

https://github.com/shouzo/OpenCV_Programming_pages/tree/master/Homework/

◎ 程式結構：

下述之濾波器程式皆由以下結構所組成，分別為”(一) 前置程式碼”、”(二) 主程式碼”以及”(三) 後置程式碼”，說明如下：

(一) 前置程式碼(標頭檔、引入使用空間...)

```
#include <iostream>
#include <vector>
#include <cstring>
#include <stdlib.h>
#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/features2d/features2d.hpp"

using namespace std;
using namespace cv;

/* Global Variables 全域變數區 */
int DELAY_CAPTION = 3000; // 在這裡可以調整投影片顯示的間隔時間
int DELAY_BLUR = 500;
int MAX_KERNEL_LENGTH = 100; // 調整處理影像的時間
int xxx_slider = 0; // 設定 xxx 濾波器滑動條的初始值

Mat src; Mat dst;
const char* file_name;
char window_name[] = "xxx"; // 設定視窗名稱

/* Function headers 副程式宣告區 */
int noise( void ); // 處理雜訊圖片
int display_caption( const char* caption ); // 投影片隔頁(顯示文字)
int display_dst( int delay ); // 顯示圖片視窗

int trackbar( void ); // 顯示調整條控制視窗
void on_trackbar( int, void* ); // 顯示附有調整條之副程式
```

(二) 主程式碼 (主要功能區) 在各個段落中皆有註明。

(三) 後置程式碼(視窗關閉、投影片放映功能)

```
/* function noise 處理雜訊圖片之附程式 */
int noise( void )
{
    src = imread( file_name, 1 );           // 載入來源圖片
    if( display_caption( "Original Image (Before)" ) != 0 ) { return 0; }
    dst = src.clone();
    if( display_dst( DELAY_CAPTION ) != 0 ) { return 0; }

    // Applying xxx blur 使用 xxx 濾波器
    if( display_caption( "xxx Blur (After)" ) != 0 ) { return 0; }
    for ( int i = 1; i < MAX_KERNEL_LENGTH; i = i + 2 )
    { xxxBlur ( ... );                       // xxx 濾波器函式
      if( display_dst( DELAY_BLUR ) != 0 ) { return 0; } }
}

/* function display_caption 投影片隔頁(顯示文字) */
int display_caption( const char* caption )
{
    dst = Mat::zeros( src.size(), src.type() );
    putText( dst, caption,
             Point( src.cols/4, src.rows/2),
             FONT_HERSHEY_COMPLEX, 1, Scalar(255, 255, 255) );
    imshow( window_name, dst );
    int c = waitKey( DELAY_CAPTION );
    if( c >= 0 ) { return -1; }
    return 0;
}

/* function display_dst 顯示圖片視窗 */
int display_dst( int delay )
{
    imshow( window_name, dst );
    int c = waitKey ( delay );
    if( c >= 0 ) { return -1; }
    return 0;
}
```

```

/* 顯示調整條控制視窗 */
int trackbar( void )
{
    src = imread( file_name, 1 ); // 載入來源圖片
    dst = src.clone();
    createTrackbar( "xxx blur", window_name, &xxx_slider,
MAX_KERNEL_LENGTH, on_trackbar );
    on_trackbar( xxx_slider, 0 );
    return 0;
}

/* 顯示附有調整條之副程式 */
void on_trackbar( int, void* )
{
    xxxBlur( ... ); // xxx 濾波器函式
    imshow( window_name, dst );

    // 設定經濾波過後的新圖片名稱
    char new_name[200] = "(after)";
    strcat( new_name, file_name );
    imwrite( new_name, dst ); // 產生新圖片
}

```

◎ 程式的編譯與執行(已在 Lubuntu 安裝好 OpenCV):

(一) 準備相關資料及檔案



(二) 設定”CMakeLists.txt”

`cmake_minimum_required(VERSION 2.8)`

`project(專案名稱)`

`find_package(OpenCV REQUIRED)`

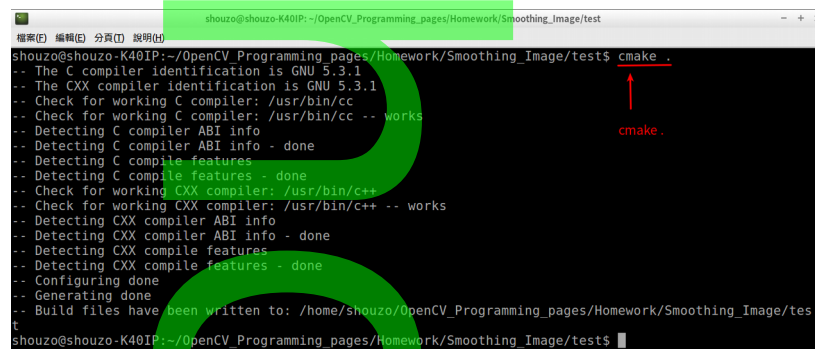
`include_directories(${OpenCV_INCLUDE_DIRS})`

`add_executable(專案名稱 專案名稱.cpp)`

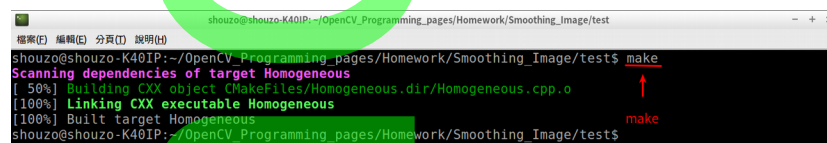
`target_link_libraries(專案名稱 ${OpenCV_LIBS})`

(三) 編譯與執行(在終端機中輸入以下指令)

1. `cmake .`



2. `make`



3. `./<欲執行之程式名稱>`

壹、加入圖片雜訊

(一) 胡椒鹽(salt-and-pepper)雜訊 (脈衝雜訊)

胡椒鹽雜訊是因為影像訊號受到突如其來的強烈干擾而產生，呈現方式是整個影像任意散布黑色或白色（或兩者皆有）的像素。

$$\text{Impulse noise:}$$
$$p(z) = \begin{cases} p_a & \text{for } z = a \\ p_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$

◎ 公式來源：<http://www.slideshare.net/AlaaAhmed13/noise-filtering> (P.17)

根據上述公式，其參數分別為： p_a 、 p_b 。

(二) 高斯(Gaussian)雜訊 (常態分配 or 高斯分配)

高斯雜訊(Gaussian noise)是由訊號中隨機的擾動而造成。

定義：若是隨機變數 X 的機率密度函數為

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in R, \quad \mu \in R, \quad \sigma > 0$$

則稱隨機變數 X 為常態分配，記為 $X \sim N(\mu, \sigma^2)$

◎ 公式來源：<https://www.cyut.edu.tw/~yfahuang/Chapter%20seven.pdf> (P.2)

根據上述公式，其參數分別為：標準差(Sigma)、平均值(Mean)。

(三) 加入”胡椒鹽(salt-and-pepper)”和高斯(Gaussian)雜訊

◎ 參考資料: https://github.com/timlentse/Add_Salt-Pepper_Noise

[範例程式碼] Noise.cpp

```
#include <iostream>
#include<opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>

using namespace std;
using namespace cv;

// "胡椒鹽雜訊"副程式
void Add_salt_pepper_Noise(Mat &srcArr, float pa, float pb )
{
    RNG rng; // rand number generate
    int amount1 = srcArr.rows*srcArr.cols*pa;
    int amount2 = srcArr.rows*srcArr.cols*pb;

    for(int counter=0; counter<amount1; ++counter)
    {
        srcArr.at<uchar>(rng.uniform( 0,srcArr.rows), rng.uniform(0, srcArr.cols)) =
0;

    }
    for (int counter=0; counter<amount2; ++counter)
    {
        srcArr.at<uchar>(rng.uniform(0,srcArr.rows), rng.uniform(0,srcArr.cols)) =
255;
    }
}

// "增加高斯雜訊"副程式
void Add_gaussian_Noise(Mat &srcArr,double mean,double sigma)
{
    Mat NoiseArr = srcArr.clone();
    RNG rng;
    rng.fill(NoiseArr, RNG::NORMAL, mean,sigma); // 產生高斯雜訊矩陣

    // 矩陣"srcArr + NoiseArr -> srcArr": 向 srcArr 增加高斯雜訊
    add(srcArr, NoiseArr, srcArr);
}
```

```

// 主程式
int main(int argc, char *argv[])
{
    Mat srcArr;
    if (argc<=1)
    {   srcArr = imread("shouzo.jpg"); }
    else if (argc>=2)
    {   srcArr = imread(argv[1]); }

    cvtColor(srcArr,srcArr, CV_RGB2GRAY,1); // 將原圖先灰階處理
    imshow("The original Image", srcArr);
    imwrite("The original Image.jpg", srcArr);

    Mat srcArr1 = srcArr.clone();
    Mat srcArr2 = srcArr.clone();

    float sigma, mean, pa, pb;
    cout<<"請分別輸入'胡椒鹽雜訊'的處理參數 pa 和 pb: ";
    cin>>pa>>pb;      // 輸入"胡椒鹽雜訊"的處理參數

    cout<<"請分別輸入'高斯雜訊'的處理參數 sigma 和 mean: ";
    cin>>sigma>>mean;  // 輸入"高斯雜訊"的處理參數

    // 使用"胡椒鹽雜訊"進行影像處理
    Add_salt_pepper_Noise(srcArr1, pa, pb);
    imshow("Add salt and pepper noise to image ", srcArr1);
    imwrite("salt&pepper noise image.jpg", srcArr1);

    // 輸入"高斯雜訊"進行影像處理
    Add_gaussian_Noise(srcArr2, mean, sigma);
    imshow("Add gaussian noise to image", srcArr2);
    imwrite("gaussian noise image.jpg", srcArr2);

    waitKey(0);
    return 0;
}

```

< 執行結果 >

在終端機裡設定雜訊參數：

(1) 胡椒鹽雜訊： $pa = 0.1$ 、 $pb = 0.05$

(2) 高斯雜訊： $\sigma = 20$ 、 $\text{mean} = 100$

```
shouzo@shouzo-K40IP: ~/OpenCV_Programming_pages/Homework/Noise
檔案(E) 編輯(E) 分頁(T) 說明(H)
shouzo@shouzo-K40IP:~/OpenCV_Programming_pages/Homework/Noise$ ./Noise
請分別輸入 '胡椒鹽雜訊' 的處理參數 pa 和 pb: 0.1 0.05
請分別輸入 '高斯雜訊' 的處理參數 sigma 和 mean: 20 100
█
```



加入高斯雜訊
檔名: "gaussian-noise-image.jpg"



加入椒鹽雜訊
檔名: "salt-and-pepper-noise-image.jpg"

在接下來的每一個消除雜訊的操作中，都會以上方的這兩張圖作為濾波效果比較。

貳、消除圖片雜訊

(一) 空間域濾波

1. 平均濾波器 (Mean Filter) - Lecture 5-17

[範例程式碼 (主程式)] Homogeneous.cpp © 註: Homogeneous(均勻)

```
/* function main 主程式入口 */
int main( void )
{
    // 處理"高斯雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "gaussian-noise-image.jpg";    // 設定要處理之圖片名
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." ); // 顯示隔頁文字(調整條)
    trackbar();                                  // 顯示調整條控制視窗
    waitKey(0);                                 // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );              // 銷毀視窗

    // 處理"胡椒鹽雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "salt-and-pepper-noise-image.jpg"; // 設定圖片名稱
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." ); // 顯示隔頁文字(調整條)
    trackbar();                                  // 顯示調整條控制視窗
    waitKey(0);                                 // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );              // 銷毀視窗

    waitKey(0);
    return 0;
}
```

[圖片處理效果] 使用“平均濾波器(Mean Filter)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

2. 中間值濾波器 (Median Filter) - Lecture 5-21

[範例程式碼 (主程式)] Median.cpp

```
/* function main 主程式入口 */
int main( void )
{
    // 處理"高斯雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "gaussian-noise-image.jpg";// 設定要處理之圖片名稱
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." );// 顯示隔頁文字(調整條)
    trackbar();                                  // 顯示調整條控制視窗
    waitKey(0);                                  // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );               // 銷毀視窗

    // 處理"胡椒鹽雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "salt-and-pepper-noise-image.jpg"; // 設定圖片名稱
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." );// 顯示隔頁文字(調整條)
    trackbar();                                  // 顯示調整條控制視窗
    waitKey(0);                                  // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );               // 銷毀視窗

    waitKey(0);
    return 0;
}
```


[圖片處理效果] 使用“中間值濾波器(Median Filter)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

3. 適應性雙邊濾波器 (Adaptive Bilateral Filter)

- Lecture 5-25 ~ 5-27

[範例程式碼 (主程式)] [Bilateral.cpp](#)

```
/* function main 主程式入口 */
int main( void )
{
    // 處理"高斯雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "gaussian-noise-image.jpg";// 設定要處理之圖片名稱
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." ); // 顯示隔頁文字(調整
條)

    trackbar();                                // 顯示調整條控制視窗
    waitKey(0);                                // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );             // 銷毀視窗

    // 處理"胡椒鹽雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "salt-and-pepper-noise-image.jpg"; // 設定圖片名稱
    noise();                                     // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." );// 顯示隔頁文字(調整條)
    trackbar();                                // 顯示調整條控制視窗
    waitKey(0);                                // 等待鍵盤按下任一按鍵
    destroyWindow( window_name );             // 銷毀視窗

    waitKey(0);
    return 0;
}
```


[圖片處理效果] 使用“適應性雙邊濾波器(Adaptive Bilateral Filter)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

(二) 頻域濾波

4. 平滑化濾波器 (Gaussian Filter) - Lecture 4-38 ~ 4-39

[範例程式碼 (主程式)] Gaussian.cpp

```
/* function main 主程式入口 */
int main( void )
{
    // 處理"高斯雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "gaussian-noise-image.jpg";// 設定要處理之圖片名稱
    noise(); // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." );// 顯示隔頁文字(調整條)
    trackbar(); // 顯示調整條控制視窗
    waitKey(0); // 等待鍵盤按下任一按鍵
    destroyWindow( window_name ); // 銷毀視窗

    // 處理"胡椒鹽雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "salt-and-pepper-noise-image.jpg"; // 設定圖片名稱
    noise(); // 將圖片去除雜訊(投影片)
    display_caption( "Move the trackbar..." );// 顯示隔頁文字(調整條)
    trackbar(); // 顯示調整條控制視窗
    waitKey(0); // 等待鍵盤按下任一按鍵
    destroyWindow( window_name ); // 銷毀視窗

    waitKey(0);
    return 0;
}
```

[圖片處理效果] 使用”平滑化濾波器 (Gaussian Filter)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

5. 銳化濾波器 (Laplacian) - Lecture 4-48 ~ 4-51

先進行高斯濾波後，再進行 Laplace Operate。

[範例程式碼 (主要程式碼)] Laplace.cpp

```
/* Global Variables 全域變數區 */
int DELAY_CAPTION = 3000; // 在這裡可以調整投影片顯示的間隔時間
int DELAY_BLUR = 3000;
int MAX_KERNEL_LENGTH = 15; // 設定處理影像的最大核心值
int laplace_slider = 0;      // 設定 Laplace operate 滑動條的初始值

int scale = 1;
int delta = 0;
int ddepth = CV_16S;

Mat src, src_gray, dst;
const char* file_name;           // 設定圖片名稱
const char* window_name = "Laplace Demo"; // 設定視窗名稱

/* function main 主程式入口 */
int main( void )
{
    // 處理"高斯雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "gaussian-noise-image.jpg";// 設定要處理之圖片名稱
    noise(); // 將圖片去除雜訊(投影片)
    display_caption( "Laplace Operate..." ); // 顯示隔頁文字(調整條)
    trackbar(); // 顯示調整條控制視窗
    waitKey(0); // 等待鍵盤按下任一按鍵
    destroyWindow( window_name ); // 銷毀視窗

    // 處理"椒鹽雜訊"之圖片
    namedWindow( window_name, WINDOW_AUTOSIZE );// 新增視窗
    file_name = "salt&pepper-noise-image.jpg";//設定圖片名稱
    noise(); // 將圖片去除雜訊(投影片)
    display_caption( "Laplace Operate..." ); // 顯示隔頁文字(調整條)
    trackbar(); // 顯示調整條控制視窗
    waitKey(0); // 等待鍵盤按下任一按鍵
    destroyWindow( window_name ); // 銷毀視窗

    waitKey(0);
    return 0;
}
```


[圖片處理效果] 使用“銳化濾波器(Laplacian Filter)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

(三) 其他

6. 傅立葉轉換 (Fourier Transform) - Lecture 4-2

[範例程式碼 (主要程式碼)] Fourier.cpp

```
int Fourier( void )
{
    I = imread( file_name, CV_LOAD_IMAGE_GRAYSCALE );

    Mat padded;          //expand input image to optimal size
    int m = getOptimalDFTSize( I.rows );
    int n = getOptimalDFTSize( I.cols ); // on the border add zero values
    copyMakeBorder(I, padded, 0, m - I.rows, 0, n - I.cols,
    BORDER_CONSTANT, Scalar::all(0));

    Mat planes[] = {Mat_<float>(padded), Mat::zeros(padded.size(),
    CV_32F)};
    Mat complexI;
    merge(planes, 2, complexI); // Add to the expanded another
    plane with zeros

    dft(complexI, complexI); // this way the result may fit in the
    source matrix

    // compute the magnitude and switch to logarithmic scale
    // => log(1 + sqrt(Re(DFT(I))^2 + Im(DFT(I))^2))
    split(complexI, planes); // planes[0] = Re(DFT(I), planes[1] =
    Im(DFT(I))
    magnitude(planes[0], planes[1], planes[0]); // planes[0] = magnitude
    Mat magI = planes[0];

    magI += Scalar::all(1); // switch to logarithmic scale
    log(magI, magI);

    // crop the spectrum, if it has an odd number of rows or columns
    magI = magI(Rect(0, 0, magI.cols & -2, magI.rows & -2));

    // rearrange the quadrants of Fourier image so that the origin is at
    the image center
    int cx = magI.cols/2;
    int cy = magI.rows/2;

    Mat q0(magI, Rect(0, 0, cx, cy)); // Top-Left - Create a ROI per
    quadrant
```

```

Mat q1(magI, Rect(cx, 0, cx, cy)); // Top-Right
Mat q2(magI, Rect(0, cy, cx, cy)); // Bottom-Left
Mat q3(magI, Rect(cx, cy, cx, cy)); // Bottom-Right

Mat tmp; // swap quadrants (Top-Left with Bottom-Right)
q0.copyTo(tmp);
q3.copyTo(q0);
tmp.copyTo(q3);

q1.copyTo(tmp); // swap quadrant (Top-Right with Bottom-Left)
q2.copyTo(q1);
tmp.copyTo(q2);

normalize(magI, magI, 0, 1, NORM_MINMAX); // Transform the
matrix with float values into a viewable image form (float between values
0 and 1).
dst = magI.clone();

if( display_dst( DELAY_CAPTION ) != 0 )
{ return 0; }
}

```

[圖片處理效果] 使用“傅立葉轉換(Fourier Transform)”

	原始圖檔	經處理過後之圖檔
高斯雜訊		
椒鹽雜訊		

參、心得

自從這學期學了這門“影像處理暨實習”之後，我對於業界上常常使用的“影像處理技術”以及“OpenCV”有了初步的了解。雖然在“影像處理”和“OpenCV”程式碼的理解上花了相當的時間，但我也從中獲得了實用的知識和程式設計技巧。真心感謝老師開“影像處理暨實習”這門課程，除了讓我獲得相當寶貴的資料外，也讓我習得“資料探勘”必備的“影像處理”技巧！