視覺感測技術應用實務

影像平滑 邊緣偵測 二維卷積 影像銳利化

第二組

組員:李宗晏 608470042(100%)

目錄

- 實現方法
- 結果展示
- 獨特設計之處
- 組員分工表

實現方法--使用類別方法的類型

- 本次程式BaseIP、ConvIP以及Project3Interface的成員函式都使用Bounded methods
  - 原因:透過self參數來直接使用別的函數計算結果後所儲存的變數，相對於Static methods需要回傳才能使用

實現方法--程式(cv2IP.py)

- 匯入的套件

```
import cv2
import numpy as np
import enum
from matplotlib import pyplot as plt
import math
```

實現方法--程式(cv2IP.py)

- ConvIP類別的建構式以及建立enumclass

```
class ConvIP(BaseIP):

    def __init__(self):
        super().__init__()
        self.__InitRobertsKernel()
        self.__InitPrewittKernel()
        self.__InitKirschKernel()

    class SmoothType(enum.IntEnum):
        BLUR = 1
        BOX = 2
        GAUSSIAN = 3
        MEDIAN = 4
        BILATERAL = 5

    class EdgeType(enum.IntEnum):
        SOBEL = 1
        CANNY = 2
        SCHARR = 3
        LAPLACE = 4
        COLOR_SOBEL = 5

    class SharpType(enum.IntEnum):
        LAPLACE_TYPE1 = 1
        LAPLACE_TYPE2 = 2
        SECOND_ORDER_LOG = 3
        UNSHARP_MASK = 4
```

實現方法--程式(cv2IP.py)

- 影像平滑

```
def Smooth2D(self, SrcImg, ksize = 5, SmType = SmoothType.BLUR,
             d = 9, sigma = 75):
    if SmType == self.SmoothType.BLUR:
        OutImg = cv2.blur(SrcImg, (ksize, ksize))
    elif SmType == self.SmoothType.BOX:
        OutImg = cv2.boxFilter(SrcImg, -1, (ksize, ksize))
    elif SmType == self.SmoothType.GAUSSIAN:
        OutImg = cv2.GaussianBlur(SrcImg, (ksize, ksize), 0)
    elif SmType == self.SmoothType.MEDIAN:
        OutImg = cv2.medianBlur(SrcImg, ksize)
    elif SmType == self.SmoothType.BILATERAL:
        OutImg = cv2.bilateralFilter(SrcImg, d, sigma, sigma)
    return OutImg
```

實現方法--程式(cv2IP.py)

- 邊緣偵測(SOBEL、CANNY以及SCHARR)

```
def EdgeDetect(self, SrcImg, EdType = EdgeType.SOBEL, ksize = 3,
               lowThreshold = 80, highThreshold = 150):
    if EdType == self.EdgeType.SOBEL:
        GrayImg = self.ImBGR2Gray(SrcImg)
        Gradient_X_64F = cv2.Sobel(GrayImg, cv2.CV_64F, 1, 0, ksize = ksize)
        Gradient_Y_64F = cv2.Sobel(GrayImg, cv2.CV_64F, 0, 1, ksize = ksize)
        Gradient_X_8U = cv2.convertScaleAbs(Gradient_X_64F)
        Gradient_Y_8U = cv2.convertScaleAbs(Gradient_Y_64F)
        OutImg = cv2.convertScaleAbs(Gradient_X_8U * 0.5 + Gradient_Y_8U * 0.5)
    elif EdType == self.EdgeType.CANNY:
        GrayImg = self.ImBGR2Gray(SrcImg)
        BlurredImg = cv2.GaussianBlur(GrayImg, (ksize, ksize), 0)
        OutImg = cv2.Canny(BlurredImg, lowThreshold, highThreshold)
    elif EdType == self.EdgeType.SCHARR:
        GrayImg = self.ImBGR2Gray(SrcImg)
        Gradient_X_64F = cv2.Scharr(GrayImg, cv2.CV_64F, 1, 0)
        Gradient_Y_64F = cv2.Scharr(GrayImg, cv2.CV_64F, 0, 1)
        Gradient_X_8U = cv2.convertScaleAbs(Gradient_X_64F)
        Gradient_Y_8U = cv2.convertScaleAbs(Gradient_Y_64F)
        OutImg = cv2.convertScaleAbs(Gradient_X_8U * 0.5 + Gradient_Y_8U * 0.5)
```

實現方法--程式(cv2IP.py)

- 邊緣偵測(LAPLACE以及COLOR_SOBEL)

```
elif EdType == self.EdgeType.LAPLACE:
    GrayImg = self.ImBGR2Gray(SrcImg)
    OutImg_64F = cv2.Laplacian(GrayImg, cv2.CV_64F, ksize = ksize)
    OutImg = cv2.convertScaleAbs(OutImg_64F)
elif EdType == self.EdgeType.COLOR_SOBEL:
    Gradient_X_64F = cv2.Sobel(SrcImg, cv2.CV_64F, 1, 0, ksize = ksize)
    Gradient_Y_64F = cv2.Sobel(SrcImg, cv2.CV_64F, 0, 1, ksize = ksize)
    Gradient_X_8U = cv2.convertScaleAbs(Gradient_X_64F)
    Gradient_Y_8U = cv2.convertScaleAbs(Gradient_Y_64F)
    OutImg = cv2.convertScaleAbs(Gradient_X_8U * 0.5 + Gradient_Y_8U * 0.5)
return OutImg
```

實現方法--程式(cv2IP.py)

- 二維卷積(Roberts以及Prewitt)的遮罩

```
def __InitRobertsKernel(self):
    self.__RobertsKernel = np.zeros((2, 2, 2), dtype=np.int)
    #----------------Gx---------------#
    self.__RobertsKernel[0,0,0] =  1
    self.__RobertsKernel[1,1,0] = -1
    #----------------Gy---------------#
    self.__RobertsKernel[1,0,1] =  1
    self.__RobertsKernel[0,1,1] = -1

def __InitPrewittKernel(self):
    self.__PrewittKernel = np.zeros((3, 3, 2), dtype=np.int)
    #----------------x----------------#
    self.__PrewittKernel[0,0,0] = -1
    self.__PrewittKernel[2,0,0] =  1
    self.__PrewittKernel[0,1,0] = -1
    self.__PrewittKernel[2,1,0] =  1
    self.__PrewittKernel[0,2,0] = -1
    self.__PrewittKernel[2,2,0] =  1
    #----------------y----------------#
    self.__PrewittKernel[0,0,1] = -1
    self.__PrewittKernel[1,0,1] = -1
    self.__PrewittKernel[2,0,1] = -1
    self.__PrewittKernel[0,2,1] =  1
    self.__PrewittKernel[1,2,1] =  1
    self.__PrewittKernel[2,2,1] =  1
```

實現方法--程式(cv2IP.py)

- 二維卷積(Kirsch)的遮罩

```
def __InitKirschKernel(self):
    self.__KirschKernel = np.full((3, 3, 8), -3, dtype=np.int)
    #----------------E----------------#
    self.__KirschKernel[2,0,0] = 5
    self.__KirschKernel[1,1,0] = 0
    self.__KirschKernel[2,1,0] = 5
    self.__KirschKernel[2,2,0] = 5
    #----------------NE---------------#
    self.__KirschKernel[1,0,1] = 5
    self.__KirschKernel[2,0,1] = 5
    self.__KirschKernel[1,1,1] = 0
    self.__KirschKernel[2,1,1] = 5
    #----------------N----------------#
    self.__KirschKernel[0,0,2] = 5
    self.__KirschKernel[1,0,2] = 5
    self.__KirschKernel[2,0,2] = 5
    self.__KirschKernel[1,1,2] = 0
    #----------------NW---------------#
    self.__KirschKernel[0,0,3] = 5
    self.__KirschKernel[1,0,3] = 5
    self.__KirschKernel[0,1,3] = 5
    self.__KirschKernel[1,1,3] = 0
```

```
#----------------W----------------#
self.__KirschKernel[0,0,4] = 5
self.__KirschKernel[0,1,4] = 5
self.__KirschKernel[1,1,4] = 0
self.__KirschKernel[0,2,4] = 5
#----------------SW---------------#
self.__KirschKernel[0,1,5] = 5
self.__KirschKernel[1,1,5] = 0
self.__KirschKernel[0,2,5] = 5
self.__KirschKernel[1,2,5] = 5
#----------------S----------------#
self.__KirschKernel[1,1,6] = 0
self.__KirschKernel[0,2,6] = 5
self.__KirschKernel[1,2,6] = 5
self.__KirschKernel[2,2,6] = 5
#----------------SE---------------#
self.__KirschKernel[1,1,7] = 0
self.__KirschKernel[2,1,7] = 5
self.__KirschKernel[1,2,7] = 5
self.__KirschKernel[2,2,7] = 5
```

實現方法--程式(cv2IP.py)

- 回傳二維卷積的遮罩

```
def GetRobertsKernel(self):
    return (self.__RobertsKernel[:,:,0], self.__RobertsKernel[:,:,1])

def GetPrewittKernel(self):
    return (self.__PrewittKernel[:,:,0], self.__PrewittKernel[:,:,1])

def GetKirschKernel(self):
    return (self.__KirschKernel[:,:,0], self.__KirschKernel[:,:,1],
            self.__KirschKernel[:,:,2], self.__KirschKernel[:,:,3],
            self.__KirschKernel[:,:,4], self.__KirschKernel[:,:,5],
            self.__KirschKernel[:,:,6], self.__KirschKernel[:,:,7])
```

實現方法--程式(cv2IP.py)

- 二維卷積函式

```
def Conv2D(self, SrcImg, Kernel):
    DstImg = cv2.filter2D(SrcImg, -1, Kernel)
    return DstImg
```

實現方法--程式(cv2IP.py)

- 影像銳利化(LAPLACE_TYPE1以及LAPLACE_TYPE2)

```
def ImSharpening(self, SrcImg, SpType = SharpType.UNSHARP_MASK, SmType = SmoothType.BILATERAL,
                 ksize = 5, d = 9, sigma = 75, Landa = 1.0):
    if SpType == self.SharpType.LAPLACE_TYPE1:
        Original = np.zeros((3, 3), dtype=np.int)
        Original[1,1] =  1
        Filtered = np.zeros((3, 3), dtype=np.int)
        Filtered[1,0] = -1
        Filtered[0,1] = -1
        Filtered[1,1] =  4
        Filtered[2,1] = -1
        Filtered[1,2] = -1
        Resulting = Original + Landa * Filtered
        DstImg = self.Conv2D(SrcImg, Resulting)
    elif SpType == self.SharpType.LAPLACE_TYPE2:
        Original = np.zeros((3, 3), dtype=np.int)
        Original[1,1] = 1
        Filtered = np.full((3, 3), -1, dtype=np.int)
        Filtered[1,1] = 8
        Resulting = Original + Landa * Filtered
        DstImg = self.Conv2D(SrcImg, Resulting)
```

實現方法--程式(cv2IP.py)

- 影像銳利化

  (SECOND_ORDER_LOG以及UNSHARP_MASK)

```
elif SpType == self.SharpType.SECOND_ORDER_LOG:
    Original = np.zeros((5, 5), dtype=np.int)
    Original[2,2] = 1
    Filtered = np.zeros((5, 5), dtype=np.int)
    Filtered[2,0] = -1
    Filtered[1,1] = -1
    Filtered[2,1] = -2
    Filtered[3,1] = -1
    Filtered[0,2] = -1
    Filtered[1,2] = -2
    Filtered[2,2] = 16
    Filtered[3,2] = -2
    Filtered[4,2] = -1
    Filtered[1,3] = -1
    Filtered[2,3] = -2
    Filtered[3,3] = -1
    Filtered[2,4] = -1
    Resulting = Original + Landa * Filtered
    DstImg = self.Conv2D(SrcImg, Resulting)
elif SpType == self.SharpType.UNSHARP_MASK:
    Coarse = self.Smooth2D(SrcImg, ksize, SmType, d, sigma)
    Fine = SrcImg * 1.0 - Coarse * 1.0
    DstImg = cv2.convertScaleAbs(SrcImg + Landa * Fine)
return DstImg
```

實現方法--程式(Project3.py)

- 匯入的套件

```
import cv2
import numpy as np
import cv2IP
import Interface
```

- 主函式

```
if __name__ == '__main__':
    Interface.Project3Interface()
```

實現方法--程式(Project3.py)

- 影像平滑的範例程式

```
def Example_ImSmooth(SmType):
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/InputIm_1_FixdPoint.bmp")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    OutImg = IP.Smooth2D(SrcBGR, 5, SmType)
    IP.ImShow("Smoothed Color Image - 5", OutImg)

    OutImg = IP.Smooth2D(SrcBGR, 15, SmType)
    IP.ImShow("Smoothed Color Image - 15", OutImg)
    cv2.waitKey(0)
    del IP
```

實現方法--程式(Project3.py)

- 邊緣偵測的範例程式

```
def Example_ImEdge(EdType):
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/foreGroundAsset.png")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    OutImg = IP.EdgeDetect(SrcBGR, EdType)

    if EdType == IP.EdgeType.SOBEL:
        IP.ImShow("Sobel Edge Image", OutImg)
    elif EdType == IP.EdgeType.CANNY:
        IP.ImShow("Canny Edge Image", OutImg)
    elif EdType == IP.EdgeType.SCHARR:
        IP.ImShow("SCHARR Edge Image", OutImg)
    elif EdType == IP.EdgeType.LAPLACE:
        IP.ImShow("LAPLACE Edge Image", OutImg)
    elif EdType == IP.EdgeType.COLOR_SOBEL:
        IP.ImShow("COLOR SOBEL Edge Image", OutImg)
    cv2.waitKey(0)
    del IP
```

實現方法--程式(Project3.py)

- 二維卷積(Roberts)的範例程式

```
def Example_ImConv2D_Roberts():
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/foreGroundAsset.png")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    SrcGray = IP.ImBGR2Gray(SrcBGR)
    Kernels = IP.GetRobertsKernel()
    Grad_Planes = []
    for i in range(0, len(Kernels)):
        Grad_Planes.append(IP.Conv2D(SrcGray, Kernels[i]))
        Grad_Planes[i] = cv2.convertScaleAbs(Grad_Planes[i])

    GradImg = cv2.convertScaleAbs(Grad_Planes[0] * 0.5 + Grad_Planes[1] * 0.5)
    IP.ImShow("Roberts Image", GradImg)
    cv2.waitKey(0)
    del IP
```

實現方法--程式(Project3.py)

- 二維卷積(Prewitt)的範例程式

```
def Example_ImConv2D_Prewitt():
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/foreGroundAsset.png")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    SrcGray = IP.ImBGR2Gray(SrcBGR)
    Kernels = IP.GetPrewittKernel()
    Grad_Planes = []
    for i in range(0, len(Kernels)):
        Grad_Planes.append(IP.Conv2D(SrcGray, Kernels[i]))
        Grad_Planes[i] = cv2.convertScaleAbs(Grad_Planes[i])

    GradImg = cv2.convertScaleAbs(Grad_Planes[0] * 0.5 + Grad_Planes[1] * 0.5)
    IP.ImShow("Prewitt Image", GradImg)
    cv2.waitKey(0)
    del IP
```

實現方法--程式(Project3.py)

- 二維卷積(Kirsch)的範例程式

```
def Example_ImConv2D_Kirsch():
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/foreGroundAsset.png")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    SrcGray = IP.ImBGR2Gray(SrcBGR)
    Kernels = IP.GetKirschKernel()
    Grad_Planes = []
    for i in range(0, len(Kernels)):
        Grad_Planes.append(IP.Conv2D(SrcGray, Kernels[i]))
        Grad_Planes[i] = cv2.convertScaleAbs(Grad_Planes[i])

    GradImg = cv2.max(Grad_Planes[0], Grad_Planes[1])
    for i in range(2, len(Kernels)):
        GradImg = cv2.max(GradImg, Grad_Planes[i])
    IP.ImShow("Kirsch Image", GradImg)
    cv2.waitKey(0)
    del IP
```

實現方法--程式(Project3.py)

- 影像銳利化的範例程式

```
def Example_ImSharpening(SpType, SmType):
    IP = cv2IP.ConvIP()
    # source image
    SrcImg = IP.ImRead("img/src/foreGroundAsset.png")
    IP.ImShow("Original Image", SrcImg)
    if SrcImg.shape[2] == 4:
        SrcBGR = np.array(SrcImg[:,:,:3])
    else:
        SrcBGR = np.array(SrcImg)

    DstImg = IP.ImSharpening(SrcBGR, SpType, SmType)
    IP.ImShow("Sharpening Image", DstImg)
    cv2.waitKey(0)
    del IP
```

結果展示

- 影像平滑(BLUR)

來源圖

結果圖
(ksize=5)

結果圖
(ksize=15)

結果展示

- 影像平滑(BOX)

來源圖

結果圖
(ksize=5)

結果圖
(ksize=15)

結果展示

- 影像平滑(GAUSSIAN)

來源圖

結果圖
(ksize=5)

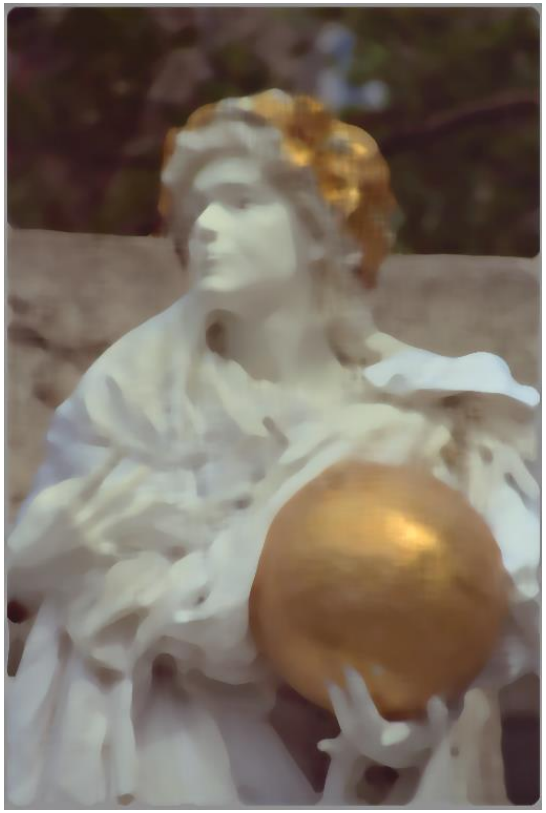
結果圖
(ksize=15)

結果展示

- 影像平滑(MEDIAN)

來源圖

結果圖
(ksize=5)



結果圖
(ksize=15)

結果展示

- 影像平滑(BILATERAL)

來源圖

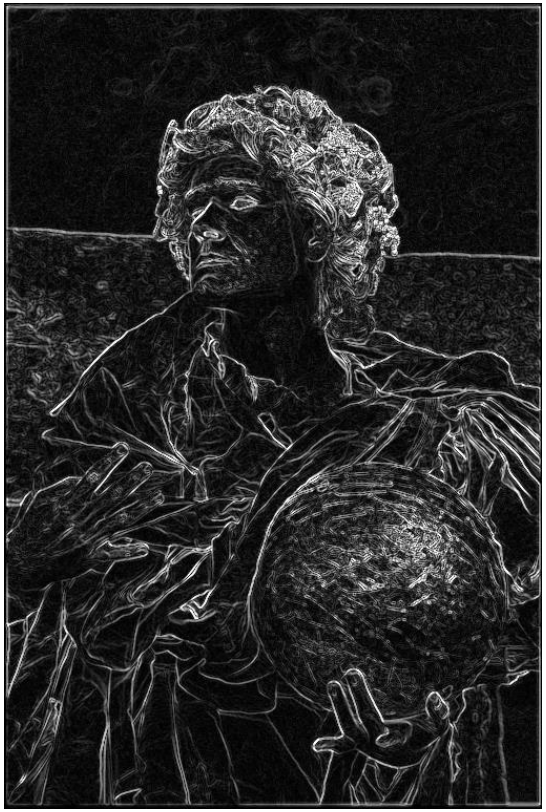
結果圖
(d=9,sigma=75)

結果圖
(d=9,sigma=150)

結果展示

- 邊緣偵測(SOBEL)

來源圖



結果圖
(ksize=3)

結果展示

- 邊緣偵測(CANNY)

來源圖

結果圖(ksize=3, lowThreshold=80,highThreshold=150)

結果展示

- 邊緣偵測(SCHARR)

來源圖

結果圖

結果展示

- 邊緣偵測(LAPLACE)

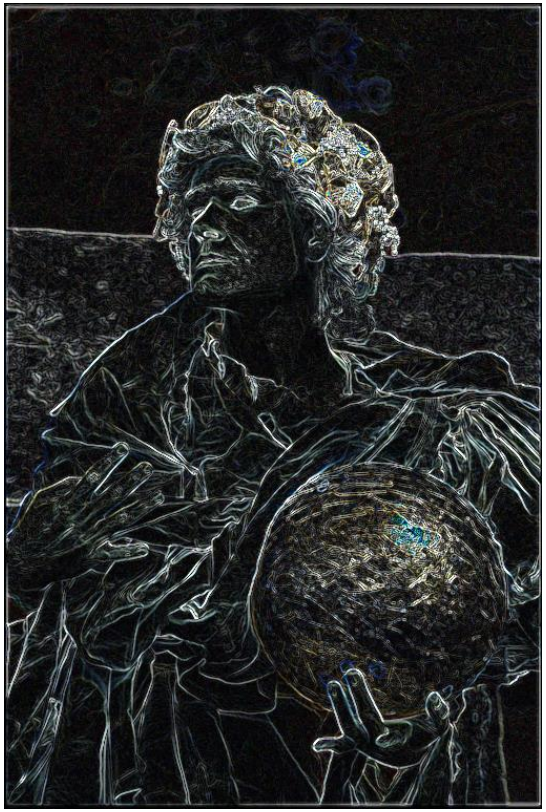
來源圖

結果圖
(ksize=3)

結果展示

- 邊緣偵測(COLOR_SOBEL)

來源圖

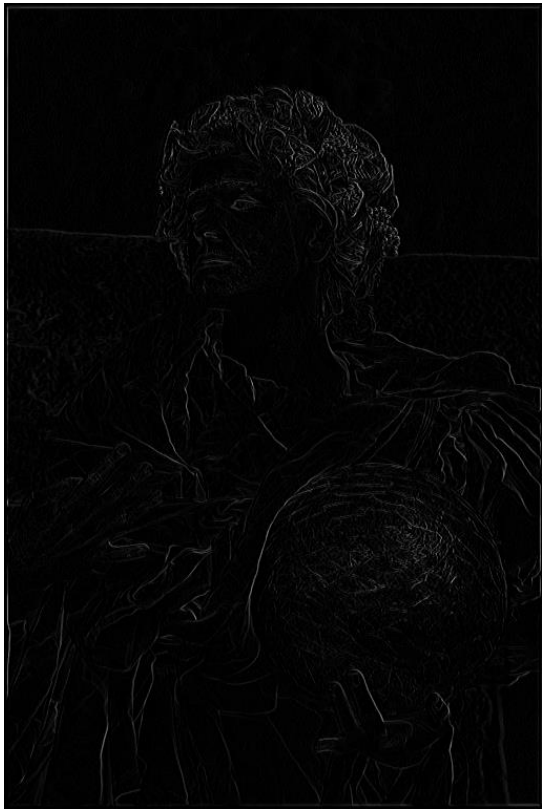


結果圖
(ksize=3)

結果展示

- 二維卷積(Roberts)

來源圖



結果圖

結果展示

- 二維卷積(Prewitt)

來源圖

結果圖

結果展示

- 二維卷積(Kirsch)

來源圖

結果圖

結果展示

- 影像銳利化(LAPLACE_TYPE1)

來源圖

結果圖
(Landa=1.0)

結果展示

- 影像銳利化(LAPLACE_TYPE2)

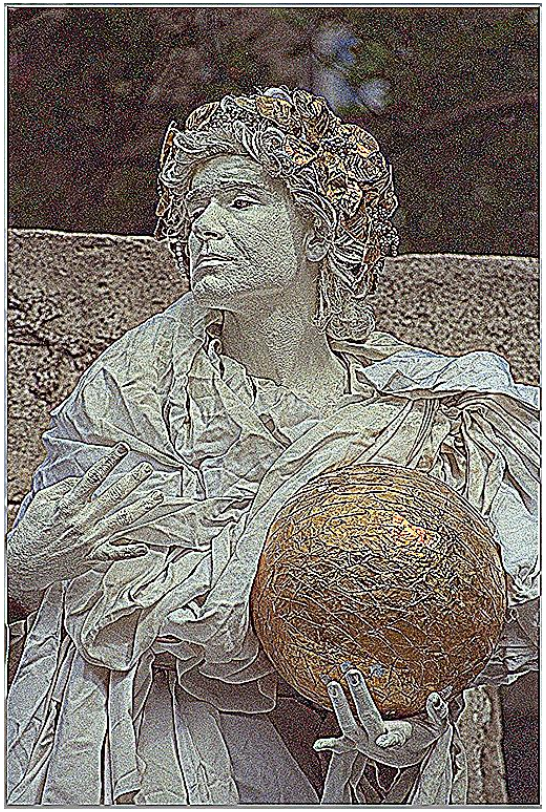
來源圖

結果圖
(Landa=1.0)

結果展示

- 影像銳利化(SECOND_ORDER_LOG)

來源圖



結果圖
(Landa=1.0)

結果展示

- 影像銳利化(UNSHARP_MASK, BLUR)

來源圖

結果圖
(ksize=5,Landa=1.0)

結果展示

- 影像銳利化(UNSHARP_MASK, BOX)

來源圖

結果圖
(ksize=5,Landa=1.0)

結果展示

- 影像銳利化(UNSHARP_MASK, GAUSSIAN)

來源圖

結果圖
(ksize=5,Landa=1.0)

結果展示

- 影像銳利化(UNSHARP_MASK, MEDIAN)

來源圖

結果圖
(ksize=5,Landa=1.0)

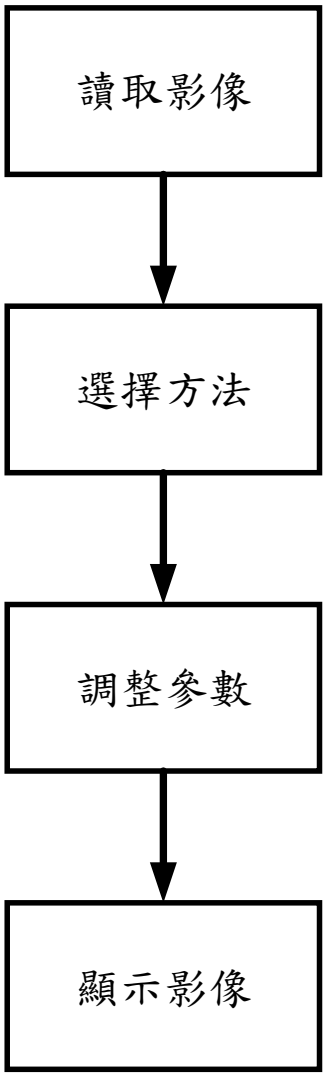
結果展示

- 影像銳利化(UNSHARP_MASK, BILATERAL)

來源圖

結果圖
(d=9,sigma=75,Landa=1.0)

獨特設計之處--動作流程圖



獨特設計之處(Interface.py)

- 匯入的套件

```
import cv2
import numpy as np
import enum
import cv2IP
import tkinter as tk
```

獨特設計之處(Interface.py)

- 創建ConvIP的物件以及主視窗
  - 使用物件導向方式來呼叫類別中的成員函式

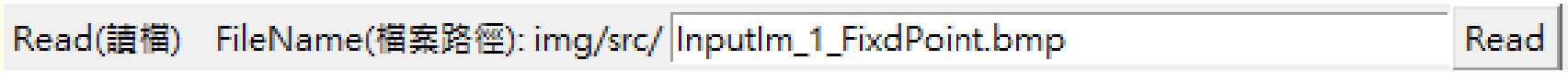
```
class Project3Interface(object):
    Obj_Num = 0

    def __init__(self):
        Project3Interface.Obj_Num += 1
        print("Create 1 obj: Total number of Interface objects is "+ str(Project3Interface.Obj_Num))
        self.IP = cv2IP.ConvIP()

        # 主視窗
        self.window = tk.Tk()
        self.window.title("Project3")
        self.window.geometry("1300x655")
        self.window.configure(background="#FFFFE0")
```

獨特設計之處(Interface.py)

讀檔群組




```
# 以下為 readFile 群組
self.readFileFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
self.readFileFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
self.readFileFrame.propagate(0)
self.readFileFrameLabel = tk.Label(self.readFileFrame, text="Read(讀檔)  ", background="#F0F0F0")
self.readFileFrameLabel.pack(side=tk.LEFT)
self.readFileLabel = tk.Label(self.readFileFrame, text="FileName(檔案路徑): img/src/", background="#F0F0F0")
self.readFileLabel.pack(side=tk.LEFT)
self.readFileString = tk.StringVar()
self.readFileString.set("InputIm_1_FixdPoint.bmp")
self.readFileEntry = tk.Entry(self.readFileFrame, width=40, textvariable=self.readFileString)
self.readFileEntry.pack(side=tk.LEFT)
self.readFileButton = tk.Button(self.readFileFrame, text="Read", command=self.readCommand)
self.readFileButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

讀檔函式

```
#----------------------------------------Read----------------------------------------#
def readCommand(self):
    self.chapterButtonReset()
    # source image
    self.srcImage = self.IP.ImRead("img/src/"+self.readFileString.get())
    if self.srcImage.shape[2] == 4:
        self.srcBGR = np.array(self.srcImage[:,:,:3])
    else:
        self.srcBGR = np.array(self.srcImage)
    self.fileExtensionToPng("img/tmp/src.png", self.srcBGR)
    self.loadSrcImage = tk.PhotoImage(file="img/tmp/src.png")
    self.srcShowLabelImage["image"] = self.loadSrcImage
    self.dstImageReset()
```

```
def dstImageReset(self):
    self.dstImage = self.srcBGR
    self.showImage()
```

獨特設計之處(Interface.py)

顯示群組

來源圖

結果圖

```
# 預設已讀檔的影像
self.srcImage = self.IP.ImRead("img/src/"+self.readFileString.get())
if self.srcImage.shape[2] == 4:
    self.srcBGR = np.array(self.srcImage[:,:,:3])
else:
    self.srcBGR = np.array(self.srcImage)
self.dstImage = self.srcBGR
```

獨特設計之處(Interface.py)

顯示群組

```
# 以下為 show 群組
self.showFrame = tk.Frame(self.window, height=370, background="#F0F0F0")
self.showFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.BOTTOM)
self.showFrame.propagate(0)
self.srcShowFrame = tk.Frame(self.showFrame, width=640, height=370, background="#F0F0F0")
self.srcShowFrame.pack(padx=5, pady=5, side=tk.LEFT)
self.srcShowFrame.propagate(0)
self.srcShowLabelText = tk.Label(self.srcShowFrame, text="來\n源\n圖", font=("microsoft yahei",15), background="#F0F0F0")
self.fileExtensionToPng("img/tmp/src.png", self.srcImage)
self.loadSrcImage = tk.PhotoImage(file="img/tmp/src.png")
self.srcShowLabelImage = tk.Label(self.srcShowFrame, image=self.loadSrcImage)
self.srcShowLabelImage.pack(side=tk.RIGHT, padx=25)
self.srcShowLabelText.pack(side=tk.RIGHT)
self.dstShowFrame = tk.Frame(self.showFrame, width=640, height=370, background="#F0F0F0")
self.dstShowFrame.pack(padx=5, pady=5, ipadx=50, side=tk.LEFT)
self.dstShowFrame.propagate(0)
self.dstShowLabelText = tk.Label(self.dstShowFrame, text="結\n果\n圖", font=("microsoft yahei",15), background="#F0F0F0")
self.fileExtensionToPng("img/tmp/dst.png", self.dstImage)
self.loadDstImage = tk.PhotoImage(file="img/tmp/dst.png")
self.dstShowLabelImage = tk.Label(self.dstShowFrame, image=self.loadDstImage)
self.dstShowLabelText.pack(side=tk.LEFT)
self.dstShowLabelImage.pack(side=tk.LEFT, padx=25)
```

獨特設計之處(Interface.py)

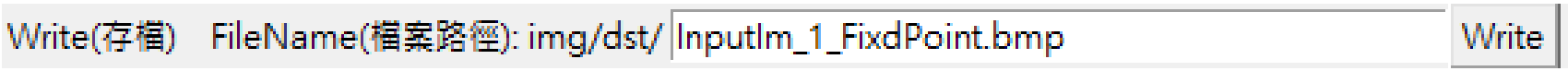
顯示函式

```
#----------------------------------------Show Image---------------------------------#
def fileExtensionToPng(self, filename, tmpImage):
    if tmpImage.shape[0] >= tmpImage.shape[1]:
        if tmpImage.shape[0] > 360:
            columnsRatio = tmpImage.shape[0] / 360
            rows = int(tmpImage.shape[1] / columnsRatio)
            tmpImage = cv2.resize(tmpImage, (rows, 360), interpolation=cv2.INTER_AREA)
    else:
        if tmpImage.shape[1] > 500:
            rowsRatio = tmpImage.shape[1] / 500
            columns = int(tmpImage.shape[0] / rowsRatio)
            tmpImage = cv2.resize(tmpImage, (500, columns), interpolation=cv2.INTER_AREA)
    self.IP.ImWrite(filename, tmpImage)
```

```
def showImage(self):
    self.fileExtensionToPng("img/tmp/dst.png", self.dstImage)
    self.loadDstImage = tk.PhotoImage(file="img/tmp/dst.png")
    self.dstShowLabelImage["image"] = self.loadDstImage
```

獨特設計之處(Interface.py)

存檔群組




```
# 以下為 writeFile 群組
self.writeFileFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
self.writeFileFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
self.writeFileFrame.propagate(0)
self.writeFileFrameLabel = tk.Label(self.writeFileFrame, text="Write(存檔)  ", background="#F0F0F0")
self.writeFileFrameLabel.pack(side=tk.LEFT)
self.writeFileLabel = tk.Label(self.writeFileFrame, text="FileName(檔案路徑): img/dst/", background="#F0F0F0")
self.writeFileLabel.pack(side=tk.LEFT)
self.writeFileString = tk.StringVar()
self.writeFileString.set("InputIm_1_FixdPoint.bmp")
self.writeFileEntry = tk.Entry(self.writeFileFrame, width=40, textvariable=self.writeFileString)
self.writeFileEntry.pack(side=tk.LEFT)
self.writeFileButton = tk.Button(self.writeFileFrame, text="Write", command=self.writeCommand)
self.writeFileButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 存檔函式

```
#----------------------------------------Write-----------------------------------------#
def writeCommand(self):
    self.IP.ImWrite("img/dst/"+self.writeFileString.get(), self.dstImage)
```

獨特設計之處(Interface.py)

章節群組

Chapter	Chapter 8(Smooth)	Chapter 9(Edge)	Chapter 10(Conv2D)	Chapter 11(Sharpening)

```
# 以下為 chapter 群組
self.chapterFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
self.chapterFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
self.chapterFrame.propagate(0)
self.chapterFrameLabel = tk.Label(self.chapterFrame, text="Chapter  ", background="#F0F0F0")
self.chapterFrameLabel.pack(side=tk.LEFT)
self.smoothButton = tk.Button(self.chapterFrame, text="Chapter 8(Smooth)", command=self.smoothCommand)
self.smoothButton.pack(side=tk.LEFT)
self.edgeButton = tk.Button(self.chapterFrame, text="Chapter 9(Edge)", command=self.edgeCommand)
self.edgeButton.pack(side=tk.LEFT)
self.conv2DButton = tk.Button(self.chapterFrame, text="Chapter 10(Conv2D)", command=self.conv2DCommand)
self.conv2DButton.pack(side=tk.LEFT)
self.sharpeningButton = tk.Button(self.chapterFrame, text="Chapter 11(Sharpening)", command=self.sharpeningCommand)
self.sharpeningButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 章節的enumclass以及按鈕重置

```
#---------------------------------------Chapter-----------------------------------#
class ChapterType(enum.IntEnum):
    Smooth = 8
    Edge = 9
    Conv2D = 10
    Sharpening = 11

def chapterButtonReset(self):
    self.smoothFrameReset()
    self.edgeFrameReset()
    self.conv2DFrameReset()
    self.sharpeningFrameReset()
    self.chapterButtonValue = None
    self.smoothButton["relief"] = tk.RAISED
    self.edgeButton["relief"] = tk.RAISED
    self.conv2DButton["relief"] = tk.RAISED
    self.sharpeningButton["relief"] = tk.RAISED
```

獨特設計之處(Interface.py)

- 初始化以及運行主程式

```
# 以下為 群組初始化
self.smoothFrame = None
self.smoothValueFrame = None
self.edgeFrame = None
self.edgeValueFrame = None
self.conv2DFrame = None
self.sharpeningFrame = None
self.sharpeningValueFrame = None
self.showFrame = None

# 以下為 按鈕值初始化
self.chapterButtonValue = None
self.smoothButtonValue = None
self.edgeButtonValue = None
self.conv2DButtonValue = None
self.sharpeningButtonValue = None

# 運行主程式
self.window.mainloop()
```

獨特設計之處(Interface.py)

- 解構式

```
def __del__(self):
    Project3Interface.Obj_Num -= 1
    print("Delete 1 obj: Total number of Interface objects is "+ str(Project3Interface.Obj_Num))
    del self.IP
```

獨特設計之處(Interface.py)

- 影像平滑群組重置以及設定章節按鈕值

```
#---------------------------------------Smooth---------------------------------------#
def smoothFrameReset(self):
    self.dstImageReset()
    self.smoothValueFrameReset()
    self.sharpeningValueFrameReset()
    self.smoothButtonValue = None
    if self.smoothFrame != None:
        self.smoothFrame.destroy()
        self.smoothFrame

def smoothCommand(self):
    self.chapterButtonReset()
    self.smoothButton["relief"] = tk.SUNKEN
    self.chapterButtonValue = self.ChapterType.Smooth
    self.createSmoothFrame()
```

獨特設計之處(Interface.py)

- 影像平滑群組

```
def createSmoothFrame(self):
    # 以下為 smooth 群組
    self.smoothFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
    self.smoothFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.smoothFrame.propagate(0)
    self.smoothFrameLabel = tk.Label(self.smoothFrame, text="Smooth  ", background="#F0F0F0")
    self.smoothFrameLabel.pack(side=tk.LEFT)
    self.blurButton = tk.Button(self.smoothFrame, text="BLUR", command=self.blurCommand)
    self.blurButton.pack(side=tk.LEFT)
    self.boxButton = tk.Button(self.smoothFrame, text="BOX", command=self.boxCommand)
    self.boxButton.pack(side=tk.LEFT)
    self.gaussianButton = tk.Button(self.smoothFrame, text="GAUSSIAN", command=self.gaussianCommand)
    self.gaussianButton.pack(side=tk.LEFT)
    self.medianButton = tk.Button(self.smoothFrame, text="MEDIAN", command=self.medianCommand)
    self.medianButton.pack(side=tk.LEFT)
    self.bilateralButton = tk.Button(self.smoothFrame, text="BILATERAL", command=self.bilateralCommand)
    self.bilateralButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 影像平滑按鈕重置

```
def smoothButtonReset(self):
    self.dstImageReset()
    self.smoothValueFrameReset()
    self.sharpeningValueFrameReset()
    self.smoothButtonValue = None
    self.blurButton["relief"] = tk.RAISED
    self.boxButton["relief"] = tk.RAISED
    self.gaussianButton["relief"] = tk.RAISED
    self.medianButton["relief"] = tk.RAISED
    self.bilateralButton["relief"] = tk.RAISED
```

獨特設計之處(Interface.py)

影像平滑按鈕函式

```
def blurCommand(self):
    self.smoothButtonReset()
    self.blurButton["relief"] = tk.SUNKEN
    self.smoothButtonValue = self.IP.SmoothType.BLUR
    self.createSmoothValueFrame()
    self.smoothKsizeValueCommand()

def boxCommand(self):
    self.smoothButtonReset()
    self.boxButton["relief"] = tk.SUNKEN
    self.smoothButtonValue = self.IP.SmoothType.BOX
    self.createSmoothValueFrame()
    self.smoothKsizeValueCommand()

def gaussianCommand(self):
    self.smoothButtonReset()
    self.gaussianButton["relief"] = tk.SUNKEN
    self.smoothButtonValue = self.IP.SmoothType.GAUSSIAN
    self.createSmoothValueFrame()
    self.smoothKsizeValueCommand()
```

```
def medianCommand(self):
    self.smoothButtonReset()
    self.medianButton["relief"] = tk.SUNKEN
    self.smoothButtonValue = self.IP.SmoothType.MEDIAN
    self.createSmoothValueFrame()
    self.smoothKsizeValueCommand()

def bilateralCommand(self):
    self.smoothButtonReset()
    self.bilateralButton["relief"] = tk.SUNKEN
    self.smoothButtonValue = self.IP.SmoothType.BILATERAL
    self.createSmoothValueFrame()
    self.smoothDValueCommand()
    self.smoothSigmaValueCommand()
```

獨特設計之處(Interface.py)

- 影像平滑參數值群組重置

```
def smoothValueFrameReset(self):
    self.smoothSigma = None
    self.smoothD = None
    self.smoothKsize = None
    if self.smoothValueFrame != None:
        self.smoothValueFrame.destroy()
        self.smoothValueFrame = None
```

獨特設計之處(Interface.py)

影像平滑參數值群組

```
def createSmoothValueFrame(self):
    # 以下為 smooth value 群組
    self.smoothValueFrame = tk.Frame(self.window, height=40, background="#F0F0F0")
    self.smoothValueFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.smoothValueFrame.propagate(0)
    self.smoothValueFrameLabel = tk.Label(self.smoothValueFrame, text="Smooth Value  ", background="#F0F0F0")
    self.smoothValueFrameLabel.pack(side=tk.LEFT)
    if self.smoothButtonValue != self.IP.SmoothType.BILATERAL:
        self.smoothKsizeValue = tk.Label(self.smoothValueFrame, text="ksize(濾波視窗尺寸): ", background="#F0F0F0")
        self.smoothKsizeValue.pack(side=tk.LEFT)
        self.smoothKsizeValueScale = tk.Scale(self.smoothValueFrame, orient=tk.HORIZONTAL, from_=1, to=15, resolution=1,
                                              length=200, command=self.smoothKsizeValueCommand)
        self.smoothKsizeValueScale.pack(side=tk.LEFT)
        self.smoothKsizeValueScale.set("5")
    else:
        self.smoothDValue = tk.Label(self.smoothValueFrame, text="d(鄰域直徑): ", background="#F0F0F0")
        self.smoothDValue.pack(side=tk.LEFT)
        self.smoothDValueScale = tk.Scale(self.smoothValueFrame, orient=tk.HORIZONTAL, from_=1, to=15, resolution=1,
                                          length=200, command=self.smoothDValueCommand)
        self.smoothDValueScale.pack(side=tk.LEFT)
        self.smoothDValueScale.set("9")
        self.smoothSigmaValue = tk.Label(self.smoothValueFrame, text="  sigma(標準差): ", background="#F0F0F0")
        self.smoothSigmaValue.pack(side=tk.LEFT)
        self.smoothSigmaValueScale = tk.Scale(self.smoothValueFrame, orient=tk.HORIZONTAL, from_=10, to=150, resolution=1,
                                              length=200, command=self.smoothSigmaValueCommand)
        self.smoothSigmaValueScale.pack(side=tk.LEFT)
        self.smoothSigmaValueScale.set("75")
```

獨特設計之處(Interface.py)

- 影像平滑參數值函式(ksize)

```
def smoothKsizeValueCommand(self, number = 5):
    if self.smoothButtonValue == self.IP.SmoothType.GAUSSIAN or self.smoothButtonValue == self.IP.SmoothType.MEDIAN:
        number = int(number)
        if not number % 2:
            self.smoothKsizeValueScale.set(number+1 if number > self.smoothKsize else number-1)
    self.smoothKsize = self.smoothKsizeValueScale.get()
    if self.chapterButtonValue == self.ChapterType.Smooth:
        self.doSmooth()
    elif self.chapterButtonValue == self.ChapterType.Sharpening and self.sharpeningLanda != None:
        self.doSharpening()
    elif self.sharpeningLanda == None:
        self.createSharpeningValueFrame()
        self.sharpeningLandaValueCommand()
```

獨特設計之處(Interface.py)

- 影像平滑參數值函式(d, sigma)

```
def smoothDValueCommand(self, number = 9):
    self.smoothD = self.smoothDValueScale.get()
    if self.chapterButtonValue == self.ChapterType.Smooth and self.smoothSigma != None:
        self.doSmooth()
    elif self.chapterButtonValue == self.ChapterType.Sharpening and self.smoothSigma != None:
        self.doSharpening()

def smoothSigmaValueCommand(self, number = 75):
    self.smoothSigma = self.smoothSigmaValueScale.get()
    if self.chapterButtonValue == self.ChapterType.Smooth:
        self.doSmooth()
    elif self.chapterButtonValue == self.ChapterType.Sharpening and self.sharpeningLanda != None:
        self.doSharpening()
    elif self.sharpeningLanda == None:
        self.createSharpeningValueFrame()
        self.sharpeningLandaValueCommand()
```

獨特設計之處(Interface.py)

- 執行影像平滑程式以及顯示結果圖

```
def doSmooth(self):
    self.dstImage = self.IP.Smooth2D(self.srcBGR, self.smoothKsize, self.smoothButtonValue,
                                     self.smoothD, self.smoothSigma)
    self.showImage()
```

獨特設計之處(Interface.py)

- 邊緣偵測群組重置以及設定章節按鈕值

```
#-----------------------------------------Edge-----------------------------------------#
def edgeFrameReset(self):
    self.dstImageReset()
    self.edgeValueFrameReset()
    self.edgeButtonValue = None
    if self.edgeFrame != None:
        self.edgeFrame.destroy()
        self.edgeFrame

def edgeCommand(self):
    self.chapterButtonReset()
    self.edgeButton["relief"] = tk.SUNKEN
    self.chapterButtonValue = self.ChapterType.Edge
    self.createEdgeFrame()
```

獨特設計之處(Interface.py)

- 邊緣偵測群組

```
def createEdgeFrame(self):
    # 以下為 edge 群組
    self.edgeFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
    self.edgeFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.edgeFrame.propagate(0)
    self.edgeFrameLabel = tk.Label(self.edgeFrame, text="Edge  ", background="#F0F0F0")
    self.edgeFrameLabel.pack(side=tk.LEFT)
    self.sobelButton = tk.Button(self.edgeFrame, text="SOBEL", command=self.sobelCommand)
    self.sobelButton.pack(side=tk.LEFT)
    self.cannyButton = tk.Button(self.edgeFrame, text="CANNY", command=self.cannyCommand)
    self.cannyButton.pack(side=tk.LEFT)
    self.scharrButton = tk.Button(self.edgeFrame, text="SCHARR", command=self.scharrCommand)
    self.scharrButton.pack(side=tk.LEFT)
    self.laplaceButton = tk.Button(self.edgeFrame, text="LAPLACE", command=self.laplaceCommand)
    self.laplaceButton.pack(side=tk.LEFT)
    self.colorSobelButton = tk.Button(self.edgeFrame, text="COLOR_SOBEL", command=self.colorSobelCommand)
    self.colorSobelButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 邊緣偵測按鈕重置

```
def edgeButtonReset(self):
    self.dstImageReset()
    self.edgeValueFrameReset()
    self.edgeButtonValue = None
    self.sobelButton["relief"] = tk.RAISED
    self.cannyButton["relief"] = tk.RAISED
    self.scharrButton["relief"] = tk.RAISED
    self.laplaceButton["relief"] = tk.RAISED
    self.colorSobelButton["relief"] = tk.RAISED
```

獨特設計之處(Interface.py)

邊緣偵測按鈕函式

```
def sobelCommand(self):
    self.edgeButtonReset()
    self.sobelButton["relief"] = tk.SUNKEN
    self.edgeButtonValue = self.IP.EdgeType.SOBEL
    self.createEdgeValueFrame()
    self.edgeKsizeValueCommand()

def cannyCommand(self):
    self.edgeButtonReset()
    self.cannyButton["relief"] = tk.SUNKEN
    self.edgeButtonValue = self.IP.EdgeType.CANNY
    self.createEdgeValueFrame()
    self.edgeKsizeValueCommand()
    self.edgeLowThresholdValueCommand()
    self.edgeHighThresholdValueCommand()

def scharrCommand(self):
    self.edgeButtonReset()
    self.scharrButton["relief"] = tk.SUNKEN
    self.edgeButtonValue = self.IP.EdgeType.SCHARR
    self.doEdge()
```

```
def laplaceCommand(self):
    self.edgeButtonReset()
    self.laplaceButton["relief"] = tk.SUNKEN
    self.edgeButtonValue = self.IP.EdgeType.LAPLACE
    self.createEdgeValueFrame()
    self.edgeKsizeValueCommand()

def colorSobelCommand(self):
    self.edgeButtonReset()
    self.colorSobelButton["relief"] = tk.SUNKEN
    self.edgeButtonValue = self.IP.EdgeType.COLOR_SOBEL
    self.createEdgeValueFrame()
    self.edgeKsizeValueCommand()
```

獨特設計之處(Interface.py)

- 邊緣偵測參數值群組重置

```
def edgeValueFrameReset(self):
    self.edgeHighThreshold = None
    self.edgeLowThreshold = None
    self.edgeKsize = None
    if self.edgeValueFrame != None:
        self.edgeValueFrame.destroy()
        self.edgeValueFrame = None
```

獨特設計之處(Interface.py)

邊緣偵測參數值群組

```
def createEdgeValueFrame(self):
    # 以下為 edge value 群組
    self.edgeValueFrame = tk.Frame(self.window, height=40, background="#F0F0F0")
    self.edgeValueFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.edgeValueFrame.propagate(0)
    self.edgeValueFrameLabel = tk.Label(self.edgeValueFrame, text="Edge Value  ", background="#F0F0F0")
    self.edgeValueFrameLabel.pack(side=tk.LEFT)
    self.edgeKsizeValue = tk.Label(self.edgeValueFrame, text="ksize(濾波視窗尺寸): ", background="#F0F0F0")
    self.edgeKsizeValue.pack(side=tk.LEFT)
    self.edgeKsizeValueScale = tk.Scale(self.edgeValueFrame, orient=tk.HORIZONTAL, from_=1, to=7, resolution=1,
                                        length=200, command=self.edgeKsizeValueCommand)
    self.edgeKsizeValueScale.pack(side=tk.LEFT)
    self.edgeKsizeValueScale.set("3")
    if self.edgeButtonValue == self.IP.EdgeType.CANNY:
        self.edgeKsizeValueScale["to"] = 15
        self.edgeLowThresholdValue = tk.Label(self.edgeValueFrame, text="  lowThreshold(下邊界): ", background="#F0F0F0")
        self.edgeLowThresholdValue.pack(side=tk.LEFT)
        self.edgeLowThresholdValueScale = tk.Scale(self.edgeValueFrame, orient=tk.HORIZONTAL, from_=0, to=255, resolution=1,
                                                   length=200, command=self.edgeLowThresholdValueCommand)
        self.edgeLowThresholdValueScale.pack(side=tk.LEFT)
        self.edgeLowThresholdValueScale.set("80")
        self.edgeHighThresholdValue = tk.Label(self.edgeValueFrame, text="  highThreshold(上邊界): ", background="#F0F0F0")
        self.edgeHighThresholdValue.pack(side=tk.LEFT)
        self.edgeHighThresholdValueScale = tk.Scale(self.edgeValueFrame, orient=tk.HORIZONTAL, from_=0, to=255, resolution=1,
                                                    length=200, command=self.edgeHighThresholdValueCommand)
        self.edgeHighThresholdValueScale.pack(side=tk.LEFT)
        self.edgeHighThresholdValueScale.set("150")
```

獨特設計之處(Interface.py)

- 邊緣偵測參數值函式(ksize)

```
def edgeKsizeValueCommand(self, number = 3):
    number = int(number)
    if not number % 2:
        self.edgeKsizeValueScale.set(number+1 if number > self.edgeKsize else number-1)
    self.edgeKsize = self.edgeKsizeValueScale.get()
    if self.edgeButtonValue != self.IP.EdgeType.CANNY or (self.edgeLowThreshold != None and self.edgeHighThreshold != None):
        self.doEdge()
```

獨特設計之處(Interface.py)

- 邊緣偵測參數值函式(lowThresHold, highThresHold)

```
def edgeLowThresholdValueCommand(self, number = 80):
    self.edgeLowThreshold = self.edgeLowThresholdValueScale.get()
    if self.edgeHighThresholdValueScale.get() < self.edgeLowThreshold:
        self.edgeHighThresholdValueScale.set(self.edgeLowThreshold)
    if self.edgeHighThreshold != None:
        self.doEdge()

def edgeHighThresholdValueCommand(self, number = 150):
    self.edgeHighThreshold = self.edgeHighThresholdValueScale.get()
    if self.edgeLowThresholdValueScale.get() > self.edgeHighThreshold:
        self.edgeLowThresholdValueScale.set(self.edgeHighThreshold)
    self.doEdge()
```

獨特設計之處(Interface.py)

- 執行邊緣偵測程式以及顯示結果圖

```
def doEdge(self):
    self.dstImage = self.IP.EdgeDetect(self.srcBGR, self.edgeButtonValue, self.edgeKsize,
                                       self.edgeLowThreshold, self.edgeHighThreshold)
    self.showImage()
```

獨特設計之處(Interface.py)

- 建立enumclass
- 二維卷積群組重置以及設定章節按鈕值

```
#---------------------------------------Conv2D---------------------------------------#
class Conv2DType(enum.IntEnum):
    Roberts = 1
    Prewitt = 2
    Kirsch = 3

def conv2DFrameReset(self):
    self.dstImageReset()
    self.conv2DButtonValue = None
    if self.conv2DFrame != None:
        self.conv2DFrame.destroy()
        self.conv2DFrame

def conv2DCommand(self):
    self.chapterButtonReset()
    self.conv2DButton["relief"] = tk.SUNKEN
    self.chapterButtonValue = self.ChapterType.Conv2D
    self.createConv2DFrame()
```

獨特設計之處(Interface.py)

- 二維卷積群組

```
def createConv2DFrame(self):
    # 以下為 conv2D 群組
    self.conv2DFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
    self.conv2DFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.conv2DFrame.propagate(0)
    self.conv2DFrameLabel = tk.Label(self.conv2DFrame, text="Conv2D  ", background="#F0F0F0")
    self.conv2DFrameLabel.pack(side=tk.LEFT)
    self.robertsButton = tk.Button(self.conv2DFrame, text="ROBERTS", command=self.robertsCommand)
    self.robertsButton.pack(side=tk.LEFT)
    self.prewittButton = tk.Button(self.conv2DFrame, text="PREWITT", command=self.prewittCommand)
    self.prewittButton.pack(side=tk.LEFT)
    self.kirschButton = tk.Button(self.conv2DFrame, text="KIRSCH", command=self.kirschCommand)
    self.kirschButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 二維卷積按鈕重置

```
def conv2DButtonReset(self):
    self.dstImageReset()
    self.conv2DButtonValue = None
    self.robertsButton["relief"] = tk.RAISED
    self.prewittButton["relief"] = tk.RAISED
    self.kirschButton["relief"] = tk.RAISED
```

獨特設計之處(Interface.py)

- 二維卷積按鈕函式

```
def robertsCommand(self):
    self.conv2DButtonReset()
    self.robertsButton["relief"] = tk.SUNKEN
    self.conv2DButtonValue = self.Conv2DType.Roberts
    self.doConv2D()

def prewittCommand(self):
    self.conv2DButtonReset()
    self.prewittButton["relief"] = tk.SUNKEN
    self.conv2DButtonValue = self.Conv2DType.Prewitt
    self.doConv2D()

def kirschCommand(self):
    self.conv2DButtonReset()
    self.kirschButton["relief"] = tk.SUNKEN
    self.conv2DButtonValue = self.Conv2DType.Kirsch
    self.doConv2D()
```

獨特設計之處(Interface.py)

- 執行二維卷積程式以及顯示結果圖

```
def doConv2D(self):
    srcGray = self.IP.ImBGR2Gray(self.srcBGR)
    if self.conv2DButtonValue == self.Conv2DType.Roberts:
        kernels = self.IP.GetRobertsKernel()
    elif self.conv2DButtonValue == self.Conv2DType.Prewitt:
        kernels = self.IP.GetPrewittKernel()
    elif self.conv2DButtonValue == self.Conv2DType.Kirsch:
        kernels = self.IP.GetKirschKernel()
    gradPlanes = []
    for i in range(0, len(kernels)):
        gradPlanes.append(self.IP.Conv2D(srcGray, kernels[i]))
        gradPlanes[i] = cv2.convertScaleAbs(gradPlanes[i])

    if self.conv2DButtonValue != self.Conv2DType.Kirsch:
        self.dstImage = cv2.convertScaleAbs(gradPlanes[0] * 0.5 + gradPlanes[1] * 0.5)
    else:
        self.dstImage = cv2.max(gradPlanes[0], gradPlanes[1])
        for i in range(2, len(kernels)):
            self.dstImage = cv2.max(self.dstImage, gradPlanes[i])
    self.showImage()
```

獨特設計之處(Interface.py)

- 影像銳利化群組重置以及設定章節按鈕值

```
#-------------------------------------Sharpening---------------------------------#
def sharpeningFrameReset(self):
    self.dstImageReset()
    self.sharpeningValueFrameReset()
    self.sharpeningButtonValue = None
    if self.sharpeningFrame != None:
        self.sharpeningFrame.destroy()
        self.sharpeningFrame

def sharpeningCommand(self):
    self.chapterButtonReset()
    self.sharpeningButton["relief"] = tk.SUNKEN
    self.chapterButtonValue = self.ChapterType.Sharpening
    self.createSharpeningFrame()
```

獨特設計之處(Interface.py)

- 影像銳利化參數值群組

```
def createSharpeningFrame(self):
    # 以下為 sharpening 群組
    self.sharpeningFrame = tk.Frame(self.window, height=25, background="#F0F0F0")
    self.sharpeningFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.sharpeningFrame.propagate(0)
    self.sharpeningFrameLabel = tk.Label(self.sharpeningFrame, text="Sharpening  ", background="#F0F0F0")
    self.sharpeningFrameLabel.pack(side=tk.LEFT)
    self.laplaceType1Button = tk.Button(self.sharpeningFrame, text="LAPLACE_TYPE1", command=self.laplaceType1Command)
    self.laplaceType1Button.pack(side=tk.LEFT)
    self.laplaceType2Button = tk.Button(self.sharpeningFrame, text="LAPLACE_TYPE2", command=self.laplaceType2Command)
    self.laplaceType2Button.pack(side=tk.LEFT)
    self.secondOrderLogButton = tk.Button(self.sharpeningFrame, text="SECOND_ORDER_LOG", command=self.secondOrderLogCommand)
    self.secondOrderLogButton.pack(side=tk.LEFT)
    self.unsharpMaskButton = tk.Button(self.sharpeningFrame, text="UNSHARP_MASK", command=self.unsharpMaskCommand)
    self.unsharpMaskButton.pack(side=tk.LEFT)
```

獨特設計之處(Interface.py)

- 影像銳利化按鈕重置

```
def sharpeningButtonReset(self):
    self.dstImageReset()
    self.smoothFrameReset()
    self.sharpeningValueFrameReset()
    self.sharpeningButtonValue = None
    self.laplaceType1Button["relief"] = tk.RAISED
    self.laplaceType2Button["relief"] = tk.RAISED
    self.secondOrderLogButton["relief"] = tk.RAISED
    self.unsharpMaskButton["relief"] = tk.RAISED
```

獨特設計之處(Interface.py)

- 影像銳利化按鈕函式
  (LAPLACE_TYPE1以及LAPLACE_TYPE2)

```
def laplaceType1Command(self):
    self.sharpeningButtonReset()
    self.laplaceType1Button["relief"] = tk.SUNKEN
    self.sharpeningButtonValue = self.IP.SharpType.LAPLACE_TYPE1
    self.createSharpeningValueFrame()
    self.sharpeningLandaValueCommand()

def laplaceType2Command(self):
    self.sharpeningButtonReset()
    self.laplaceType2Button["relief"] = tk.SUNKEN
    self.sharpeningButtonValue = self.IP.SharpType.LAPLACE_TYPE2
    self.createSharpeningValueFrame()
    self.sharpeningLandaValueCommand()
```

獨特設計之處(Interface.py)

- 影像銳利化按鈕函式
  (SECOND_ORDER_LOG、UNSHARP_MASK)

```
def secondOrderLogCommand(self):
    self.sharpeningButtonReset()
    self.secondOrderLogButton["relief"] = tk.SUNKEN
    self.sharpeningButtonValue = self.IP.SharpType.SECOND_ORDER_LOG
    self.createSharpeningValueFrame()
    self.sharpeningLandaValueCommand()

def unsharpMaskCommand(self):
    self.sharpeningButtonReset()
    self.unsharpMaskButton["relief"] = tk.SUNKEN
    self.sharpeningButtonValue = self.IP.SharpType.UNSHARP_MASK
    self.createSmoothFrame()
```

獨特設計之處(Interface.py)

- 影像銳利化參數值群組重置

```
def sharpeningValueFrameReset(self):
    self.sharpeningLanda = None
    if self.sharpeningValueFrame != None:
        self.sharpeningValueFrame.destroy()
        self.sharpeningValueFrame = None
```

獨特設計之處(Interface.py)

- 影像銳利化參數值群組

```
def createSharpeningValueFrame(self):
    # 以下為 edge value 群組
    self.sharpeningValueFrame = tk.Frame(self.window, height=40, background="#F0F0F0")
    self.sharpeningValueFrame.pack(fill=tk.X, padx=5, pady=5, side=tk.TOP)
    self.sharpeningValueFrame.propagate(0)
    self.sharpeningValueFrameLabel = tk.Label(self.sharpeningValueFrame, text="Sharpening Value  ", background="#F0F0F0")
    self.sharpeningValueFrameLabel.pack(side=tk.LEFT)
    self.sharpeningLandaValue = tk.Label(self.sharpeningValueFrame, text="Landa(增強的強度): ", background="#F0F0F0")
    self.sharpeningLandaValue.pack(side=tk.LEFT)
    self.sharpeningLandaValueScale = tk.Scale(self.sharpeningValueFrame, orient=tk.HORIZONTAL, from_=0, to=2, resolution=0.1,
                                              length=200, command=self.sharpeningLandaValueCommand)
    self.sharpeningLandaValueScale.pack(side=tk.LEFT)
    self.sharpeningLandaValueScale.set("1.0")
```

獨特設計之處(Interface.py)

- 影像銳利化參數值函式(Landa)

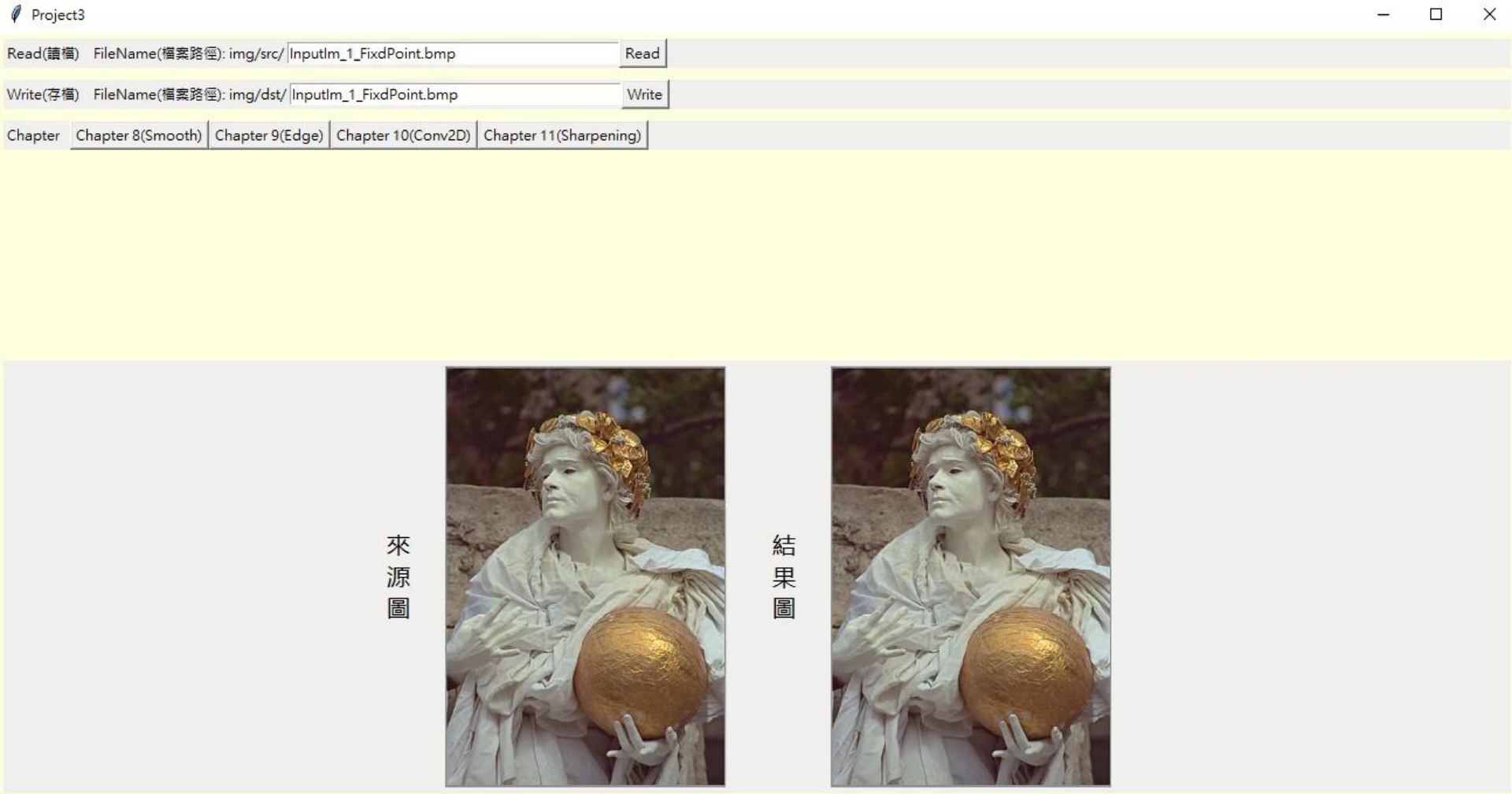
```
def sharpeningLandaValueCommand(self, number = 1.0):
    self.sharpeningLanda = self.sharpeningLandaValueScale.get()
    self.doSharpening()
```

獨特設計之處(Interface.py)

- 執行影像銳利化程式以及顯示結果圖

```
def doSharpening(self):
    self.dstImage = self.IP.ImSharpening(self.srcBGR, self.sharpeningButtonValue, self.smoothButtonValue,
                                         self.smoothKsize, self.smoothD, self.smoothSigma, self.sharpeningLanda)
    self.showImage()
```

獨特設計之處(影片)


Project3
Read(讀檔) FileName(檔案路徑): img/src/ InputIm_1_FixdPoint.bmp Read
Write(存檔) FileName(檔案路徑): img/dst/ InputIm_1_FixdPoint.bmp Write
Chapter Chapter 8(Smooth) Chapter 9(Edge) Chapter 10(Conv2D) Chapter 11(Sharpening)
來源圖
結果圖

組員分工表

組員	工作分配比重	內容
李宗晏	100%	程式、報告

Thanks for your attention