**Computer vision – Final Report**

[Question 1 2](#_Toc150343781)

[A、 Source code 2](#_Toc150343782)

[B、 Result map 2](#_Toc150343783)

[C、 Original file 3](#_Toc150343784)

[Question 2 4](#_Toc150343785)

[A、 Source code 4](#_Toc150343786)

[B、 Analyze 5](#_Toc150343787)

[C、 Result map 6](#_Toc150343788)

[Question 3 7](#_Toc150343789)

[A、 Source code 7](#_Toc150343790)

[B、 Analyze 9](#_Toc150343791)

[C、 Result map 11](#_Toc150343792)

[D、 Original file 12](#_Toc150343793)

[Appendix(utils.py) 16](#_Toc150343794)

在此交期末報告, 題目須與電腦視覺和影像處理相關,使用OpenCV, 程式用Python 或C++皆可

I. Word文字報告須包含

1.介紹說明,2.運用原理說明,3.程式流程簡要說明與程式註記,4.程式執行結果螢幕截圖,將程式運作拍成短片上傳到YouTube,提供該短片YouTube網址5.結果與討論,簡要說明你的程式可應用於何處及你從中學到什麼和未來的可增加擴充的功能

|  |  |
| --- | --- |
| Name: | WeiWen Wu |
| Student ID: | 10901007A |

# Question 1

## Source code

|  |
| --- |
| **from** numpy **import** float32**,**ones  **from** utils **import** cv2**,**plot\_from\_matplotlib  img **=** cv2**.**imread**(**"./Lib2.jpg"**)**  plot **=** plot\_from\_matplotlib**(**"Question 1(Lib)"**)**  plot**.**col\_row **=** **(**2**,**1**)**  kernel **=** ones**((**5**,** 5**),** float32**)/**81  dst **=** cv2**.**filter2D**(**img**,** **-**1**,** kernel**)**  plot**(**"Original"**,**img**)**  plot**(**"Filter2D"**,**dst**)**  plot**.**show**()** |

## Result map



## Original file



Figure 1 Original



Figure 2 Filter2D

# Question 2

## Source code

|  |
| --- |
| **from** utils **import** cv2**,**plot\_from\_matplotlib**,**bgr2gray**,**histogram  img **=** bgr2gray**(**"La\_Tour\_Eiffel\_480x700.png"**)**  ### Test threshold ###  # histogram(img).calhist(True)  **def** blur**(**name**:str=**"Question 2(La Tour Eiffel)"**,**threshold**:int=**127**):**  plot **=** plot\_from\_matplotlib**(**name**)**  plot**.**col\_row **=** **(**3**,**2**)**  plot**(**"Original"**,**img**)**  ### Otsu's thresholding ###  \_**,**TB **=** cv2**.**threshold**(**img**,**threshold**,**255**,**cv2**.**THRESH\_BINARY**)**  plot**(**"thresholding"**,**TB**)**  ### Otsu's thresholding ###  \_**,**OTSU **=** cv2**.**threshold**(**img**,**threshold**,**255**,**cv2**.**THRESH\_BINARY**+**cv2**.**THRESH\_OTSU**)**  plot**(**"Otsu's thresholding"**,**OTSU**)**  ### Otsu's thresholding (Gaussian) ### Otsu's thresholding after Gaussian filtering  blur **=** cv2**.**GaussianBlur**(**img**,(**5**,**5**),**0**)**  \_**,**OTSUG **=** cv2**.**threshold**(**blur**,**threshold**,**255**,**cv2**.**THRESH\_BINARY**+**cv2**.**THRESH\_OTSU**)**  plot**(**"Otsu's thresholding (Gaussian)"**,**OTSUG**)**  ### ADAPTIVE\_THRESH\_MEAN\_C ###  ATMC **=** cv2**.**adaptiveThreshold**(**blur**,**threshold**,**cv2**.**ADAPTIVE\_THRESH\_MEAN\_C**,**cv2**.**THRESH\_BINARY**,**11**,**2**)**  plot**(**"ADAPTIVE\_THRESH\_MEAN\_C"**,**ATMC**)**  ### ADAPTIVE\_THRESH\_GAUSSIAN\_C ###  ATGC **=** cv2**.**adaptiveThreshold**(**blur**,**threshold**,**cv2**.**ADAPTIVE\_THRESH\_GAUSSIAN\_C**,**cv2**.**THRESH\_BINARY**,**11**,**2**)**  plot**(**"ADAPTIVE\_THRESH\_GAUSSIAN\_C"**,**ATGC**)**  blur**(**"threshold(127)"**,**127**)**  blur**(**"threshold(175)"**,**175**)**  plot\_from\_matplotlib**.**show**()** |

## Analyze

使用柱狀圖中2個波峰之間的波谷灰階值當門檻值，如Figure 3所示。

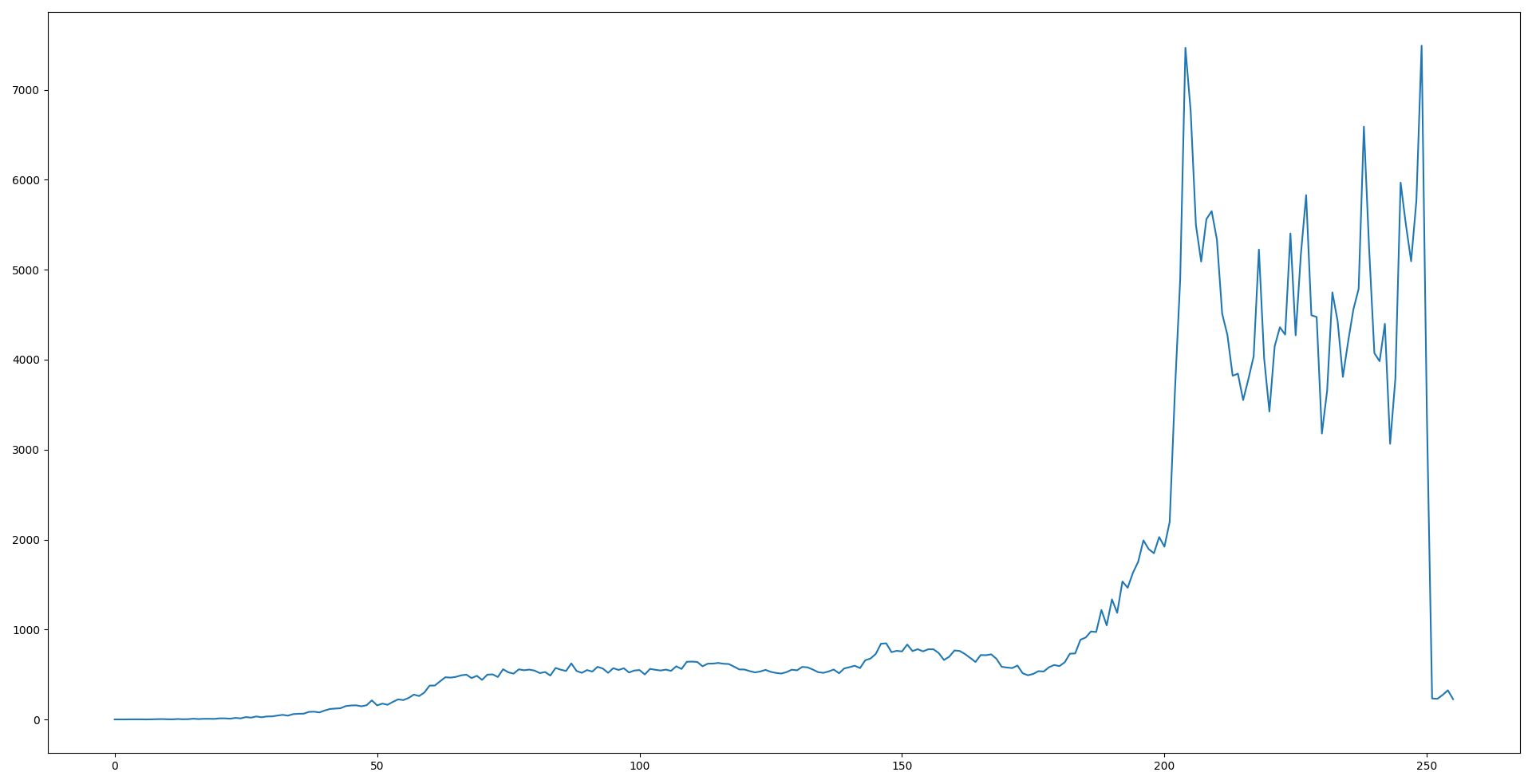


Figure 3 Histogram

## Result map

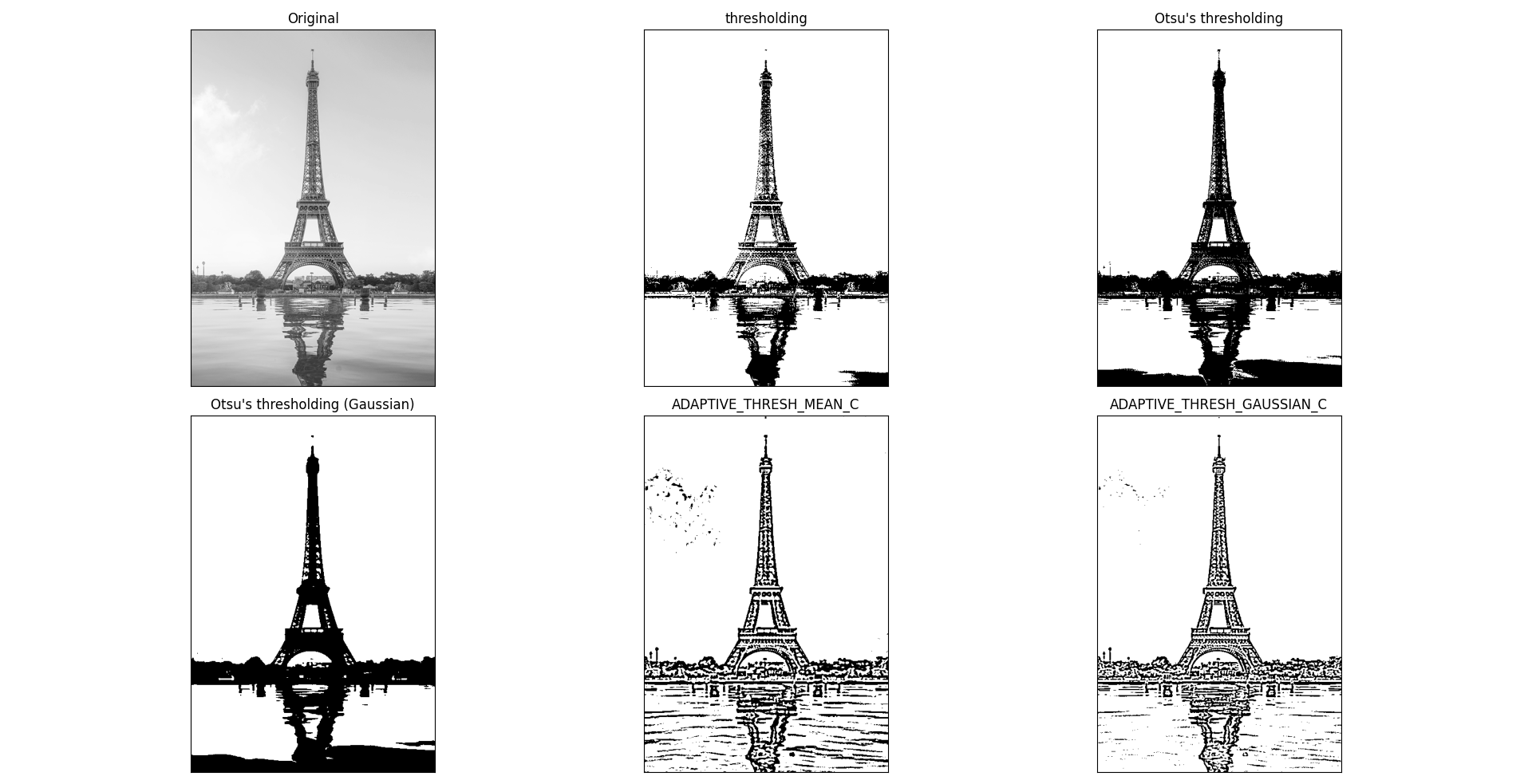


Figure 4 threshold(127)

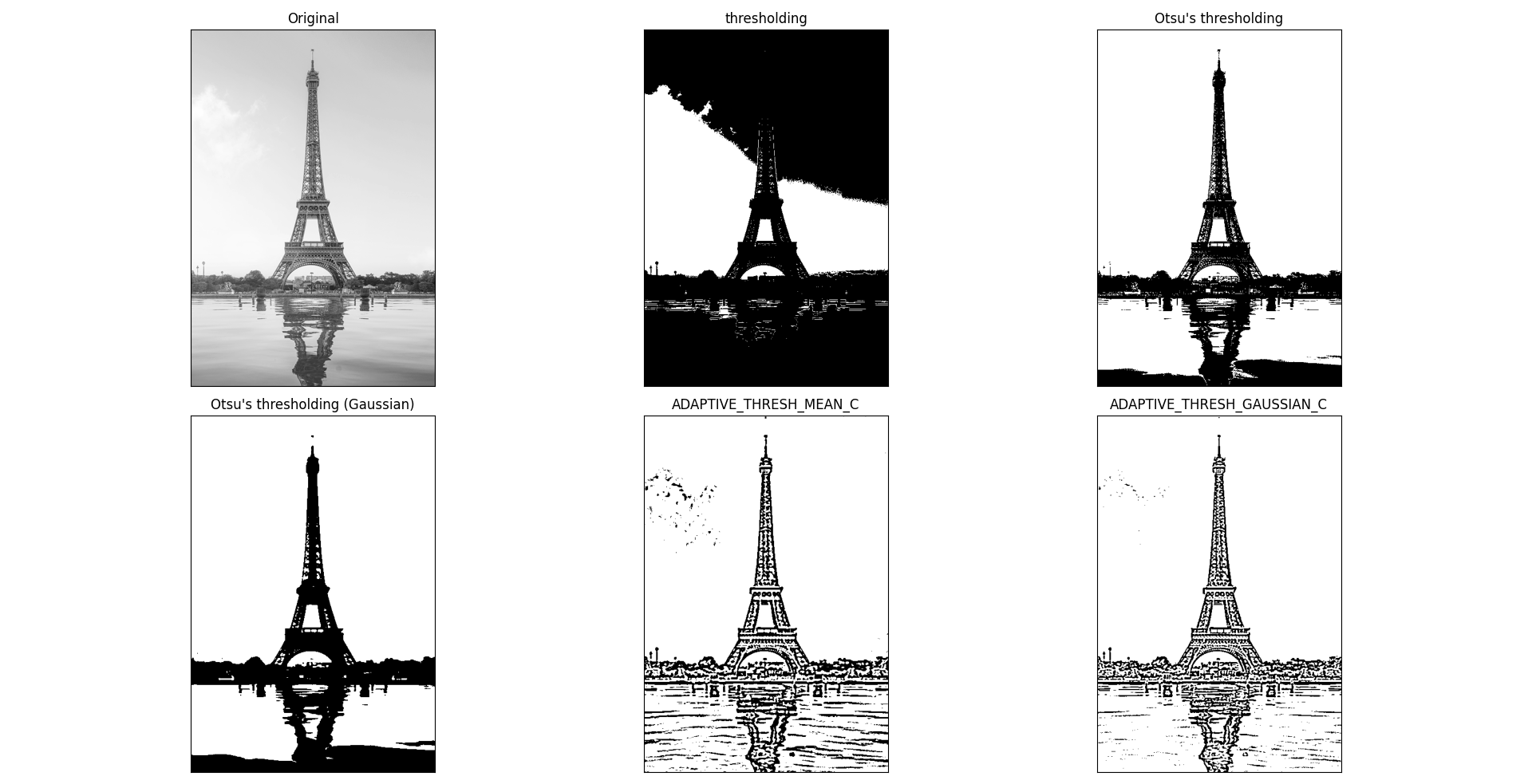


Figure 5 threshold(229)

# Question 3

## Source code

|  |
| --- |
| **from** utils **import** cv2**,**plot\_from\_matplotlib**,**canny**,**histogram**,**ndarray**,**bgr2gray**,**medianBlur**,**GaussianBlur**,**Optional**,**canny\_edge  **def** remove\_noise**(**name**:str,**img**:**ndarray**,**canny\_arg**:**Optional**[tuple[int,int]]=None):**  """Test to use medianBlur or GaussianBlur."""  **if** canny\_arg **==** **None:**  plot\_rn **=** plot\_from\_matplotlib**(**name**)**  plot\_rn**.**col\_row **=** **(**2**,**2**)**  plot\_rn**.**save **=** **False**  ### Median Filtering ###  img\_medianBlur **=** medianBlur**(**img**)**  img\_medianBlur **=** canny**(**img\_medianBlur**)**  plot\_rn**(**"Median Filtering"**,**img\_medianBlur**)**  ### GaussianBlur ###  img\_GaussianBlur **=** GaussianBlur**(**img**)**  img\_GaussianBlur **=** canny**(**img\_GaussianBlur**)**  plot\_rn**(**"GaussianBlur"**,**img\_GaussianBlur**)**  ### histogram ###  img\_hist **=** histogram**(**bgr2gray**(**img**)).**equalization**()**  plot\_rn**(**"histogram"**,**img\_hist**)**  ### canny ###  img\_canny **=** canny**(**img**)**  plot\_rn**(**"canny"**,**img\_canny**)**  plot\_rn**.**show**()**  **return** img\_canny  **else:** **return** canny**(**img**,**canny\_arg**[**0**],**canny\_arg**[**1**])**  **def** Eiffel\_night**()** **->** ndarray**:**  """Eiffel\_night"""  night **=** cv2**.**imread**(**"Eiffel\_night.jpg"**)**  night **=** GaussianBlur**(**night**)**  night **=** GaussianBlur**(**night**)**  night **=** GaussianBlur**(**night**)**  # canny\_edge(night) # 89 255  night**=**remove\_noise**(**"night"**,**night**,(**89**,**255**))**  **return** night  **def** Eiffel\_night\_sp\_noise**()** **->** ndarray**:**  """Eiffel\_night\_sp\_noise"""  night\_noise **=** cv2**.**imread**(**"Eiffel\_night\_sp\_noise.jpg"**)**  night\_noise **=** medianBlur**(**night\_noise**)**  night\_noise **=** GaussianBlur**(**night\_noise**)**  night\_noise **=** GaussianBlur**(**night\_noise**)**  # canny\_edge(night\_noise)#75 244  night\_noise **=** remove\_noise**(**"night\_noise"**,**night\_noise**,(**75**,**244**))**  **return** night\_noise  **def** Eiffel\_night\_gaussian\_noise**(**show**:bool=False)** **->** ndarray**:**  """Eiffel\_night\_gaussian\_noise"""  night\_gnoise **=** cv2**.**imread**(**"Eiffel\_night\_gaussian\_noise.jpg"**)**  night\_gnoise1 **=** GaussianBlur**(**night\_gnoise**)**  night\_gnoise2 **=** GaussianBlur**(**night\_gnoise1**)**  night\_gnoise3 **=** GaussianBlur**(**night\_gnoise2**)**  night\_gnoise4 **=** medianBlur**(**night\_gnoise3**)**  # canny\_edge(night\_gnoise)#35 252  night\_gnoise5 **=** remove\_noise**(**"night\_gnoise"**,**night\_gnoise4**,(**35**,**252**))**  **if** show**:**  plot **=** plot\_from\_matplotlib**(**"Eiffel\_night\_gaussian\_noise"**)**  plot**.**col\_row **=** **(**3**,**2**)**  plot**(**"Original"**,**canny**(**night\_gnoise**))**  plot**(**"1. GaussianBlur"**,**canny**(**night\_gnoise1**))**  plot**(**"2. GaussianBlur"**,**canny**(**night\_gnoise2**))**  plot**(**"3. GaussianBlur"**,**canny**(**night\_gnoise3**))**  plot**(**"4. MedianBlur"**,**canny**(**night\_gnoise4**))**  plot**(**"5. Result "**,**night\_gnoise5**)**  **return** night\_gnoise5  **def** Eiffel\_night\_gaussian\_sp\_noise**(**show**:bool=False)** **->** ndarray**:**  """Eiffel\_night\_gaussian\_sp\_noise"""  night\_gnoise\_sp **=** cv2**.**imread**(**"Eiffel\_night\_gaussian\_sp\_noise.jpg"**)**  night\_gnoise\_sp1 **=** medianBlur**(**night\_gnoise\_sp**)**  night\_gnoise\_sp2 **=** GaussianBlur**(**night\_gnoise\_sp1**)**  night\_gnoise\_sp3 **=** GaussianBlur**(**night\_gnoise\_sp2**)**  # canny\_edge(night\_gnoise\_sp3)#69 255  night\_gnoise\_sp4 **=** remove\_noise**(**"night\_gnoise\_sp"**,**night\_gnoise\_sp3**,(**69**,**255**))**  **if** show**:**  plot **=** plot\_from\_matplotlib**(**"Eiffel\_night\_gaussian\_sp\_noise"**)**  plot**.**col\_row **=** **(**2**,**2**)**  plot**(**"Original"**,**canny**(**night\_gnoise\_sp**))**  plot**(**"1. medianBlur"**,**canny**(**night\_gnoise\_sp1**))**  plot**(**"2. GaussianBlur"**,**canny**(**night\_gnoise\_sp2**))**  plot**(**"3. GaussianBlur"**,**canny**(**night\_gnoise\_sp3**))**  # plot("night\_gnoise\_sp4",canny(night\_gnoise\_sp3))  **return** night\_gnoise\_sp4  plot **=** plot\_from\_matplotlib**(**"Question 3"**)**  plot**.**col\_row **=** **(**2**,**2**)**  plot**.**save**=**0  plot**(**"Eiffel\_night"**,**Eiffel\_night**())**  plot**(**"Eiffel\_night\_sp\_noise"**,**Eiffel\_night\_sp\_noise**())**  plot**(**"Eiffel\_night\_gaussian\_noise"**,**Eiffel\_night\_gaussian\_noise**())**  plot**(**"Eiffel\_night\_gaussian\_sp\_noise"**,**Eiffel\_night\_gaussian\_sp\_noise**())**  plot\_from\_matplotlib**.**show**()** |

## Analyze

使用remove\_noise函數，測試medianBlur或GaussianBlur，哪一個方法較好，如Figure 6所示。最後使用canny\_edge函數決定Canny 上限和下限值，如Figure 7所示。

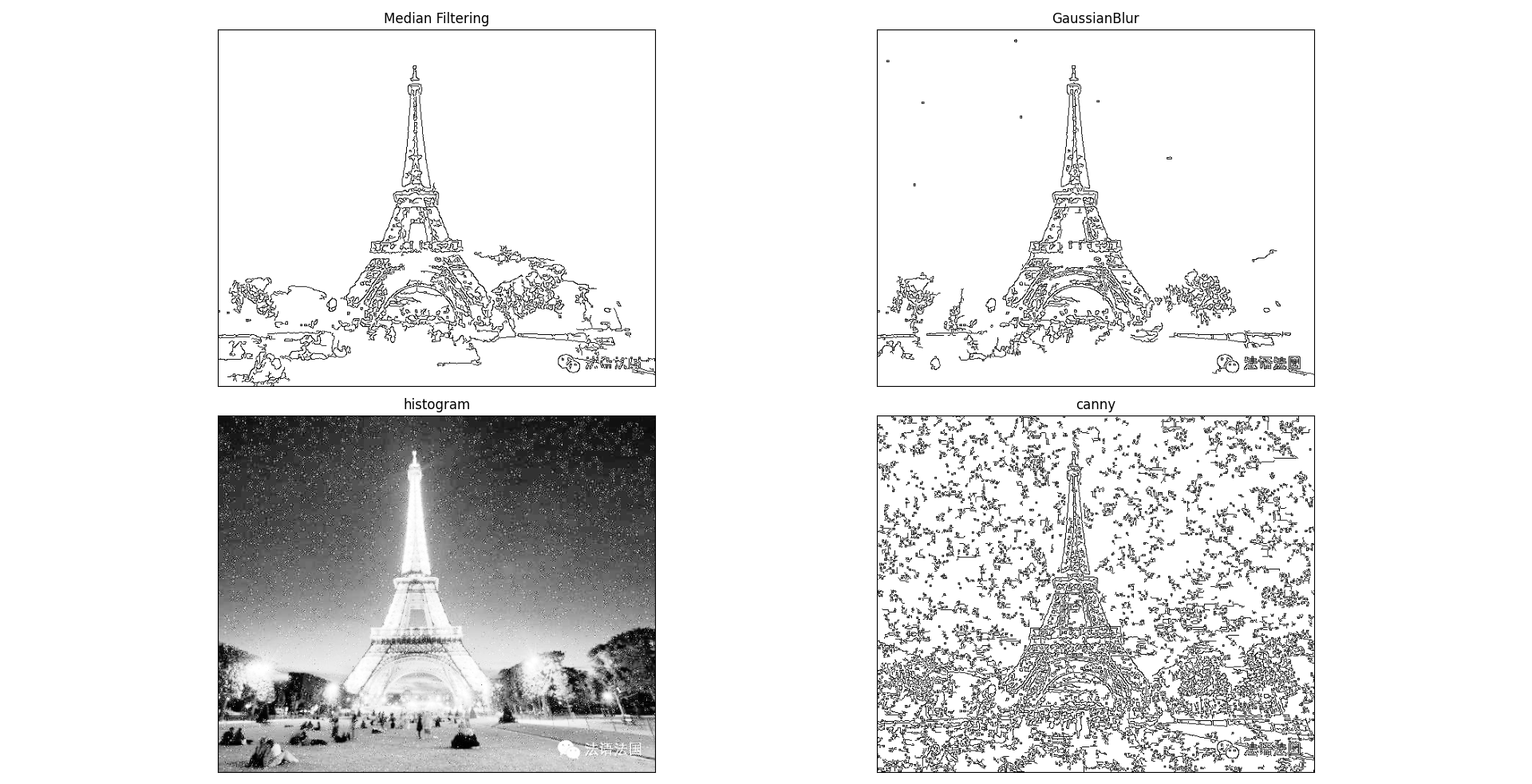


Figure 6 Test to use medianBlur or GaussianBlur.

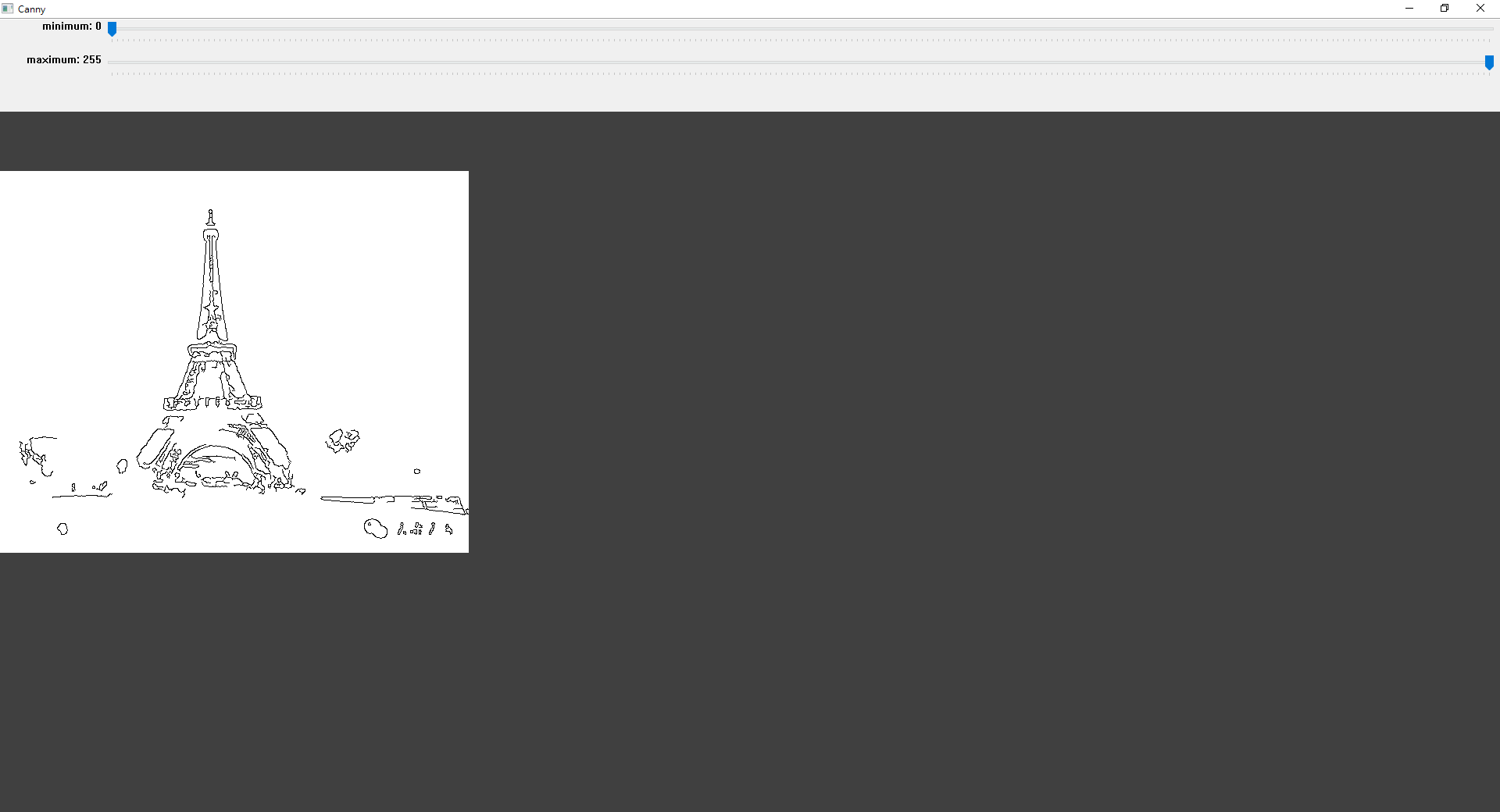
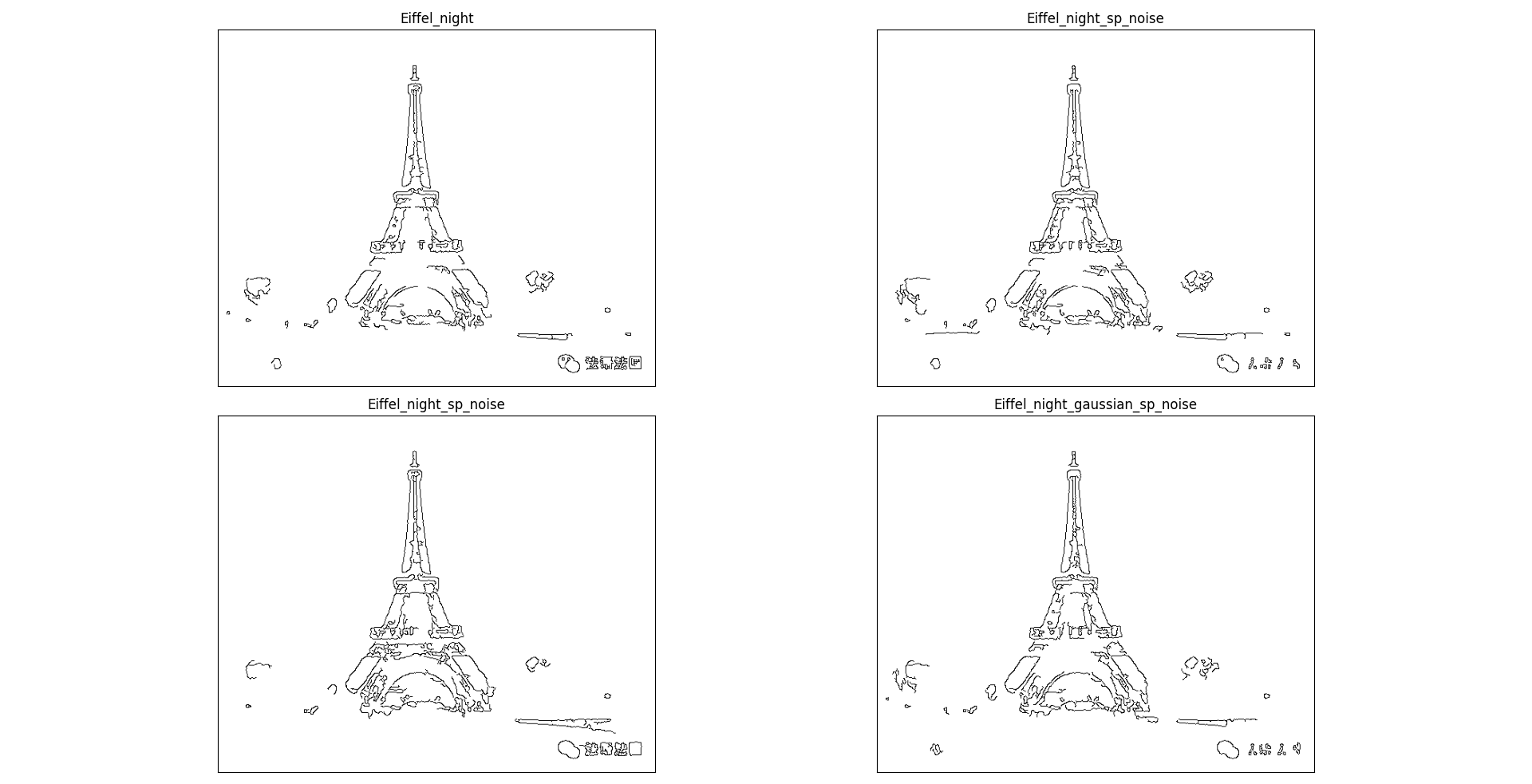


Figure 7 Test Canny upper and lower values.

## Result map



## Original file

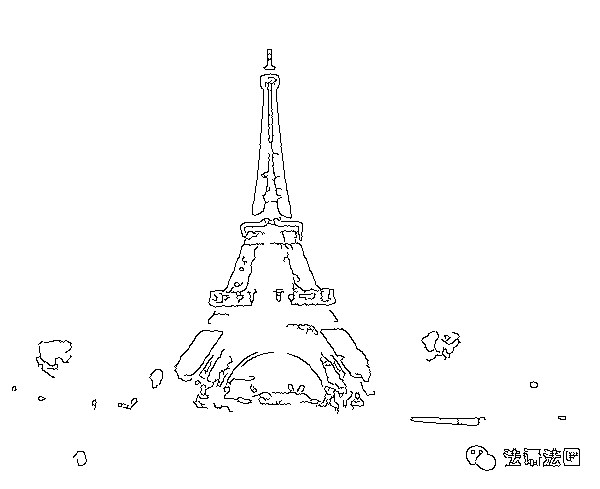


Figure 8 Eiffel\_night

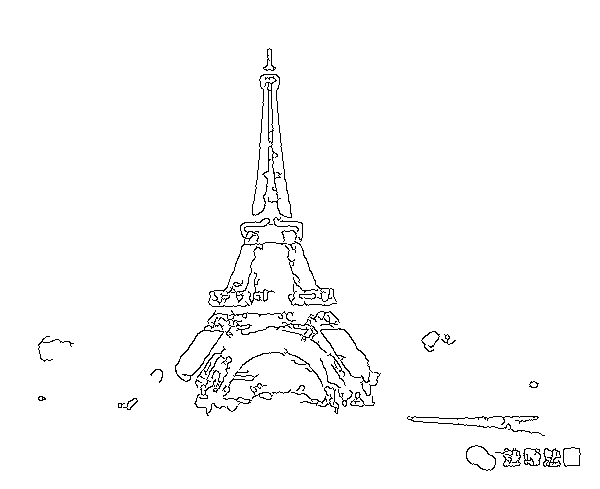


Figure 9 Eiffel\_night\_gaussian\_noise

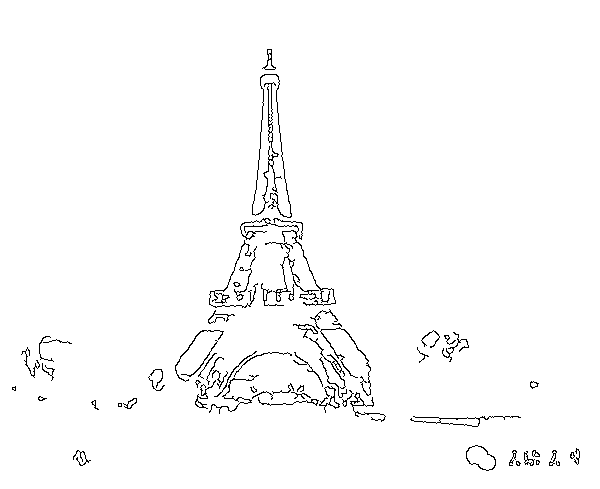


Figure 10 Eiffel\_night\_gaussian\_sp\_noise

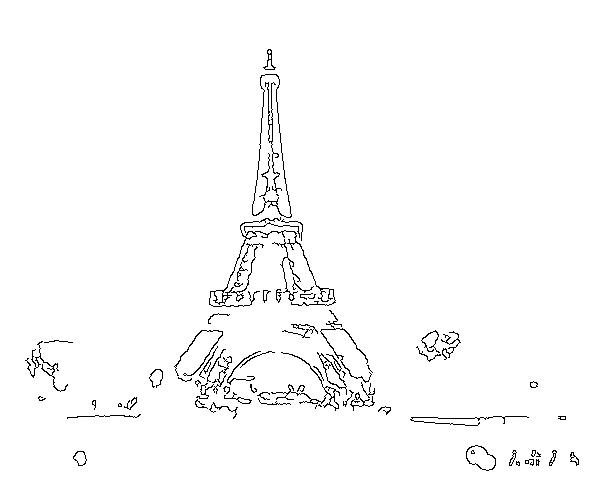


Figure 11 Eiffel\_night\_sp\_noise

# Appendix(utils.py)

|  |
| --- |
| # utils.py WeiWen Wu  **from** matplotlib **import** pyplot **as** plt  **import** cv2  **from** typing **import** Literal**,**Union**,**Optional  **from** numpy **import** ndarray  **class** **plot\_from\_matplotlib:**  """  Make pictures from matplotlib.  # Example  ```  plot = plot\_from\_matplotlib("title")  plot.col\_row = (2,2)  plot.save = True  plot("Original",img)  plot\_from\_matplotlib.show()  ```  """  n**:int=**1  col\_row**:tuple[int,int]**  save**:bool=False**  **def** \_\_init\_\_**(**self**,**name**:str)** **->** **None:**  fig **=** plt**.**figure**()**  fig**.**canvas**.**manager**.**window**.**setWindowTitle**(**name**)**  fig**.**tight\_layout**()**  # fig.subplots\_adjust(top=0.95, bottom=0.05, left=0.0, right=1.0)  self**.**fig **=** fig  **def** \_\_call\_\_**(**self**,**name**:str,**dst**,**cmap**:str=**'gray'**,**save**:bool=**0**):** # Make pictures.  n **=** self**.**n  col**,**row **=** self**.**col\_row  path **=** \_\_file\_\_**.**split**(**"\\"**)[-**2**]**  plt**.**subplot**(**row**,**col**,**n**),** plt**.**imshow**(**dst**,**cmap**),** plt**.**title**(**name**)**  plt**.**xticks**([]),** plt**.**yticks**([])** # Do not show scale.  **if** save **or** self**.**save**:** cv2**.**imwrite**(**f"./{path}/{name}.png"**,**dst**,[int(**cv2**.**IMWRITE\_PNG\_COMPRESSION**),**0**])**  self**.**n**+=**1  # @staticmethod  **def** show**(**self**):**  """Display image."""  self**.**fig**.**tight\_layout**()**  plt**.**show**()**    *@staticmethod*  **def** show**():**  """Display image."""  plt**.**show**()**  **class** **histogram:**  """  Imgage Histogram - Find, Plot and Analyze  It which gives you an overall idea about the intensity distribution of an image.  It distribute data along x and y axis.  x - axis contain range of color vlaues.  y - axis contain numbers of pixels in an image.  With histogram to extrct information about contast, brigthness and intensity etc.  plot histomgram using matplotlib  # Example  ```  hist = histogram()  histogram.plot(hist.calhist())  imshow("hist",hist.equalization())  ```  """  **def** \_\_init\_\_**(**self**,**img**:**Union**[str,**ndarray**]=**"lena.jpg"**)** **->** **None:**  **if** **isinstance(**img**,str):**  img **=** cv2**.**imread**(**img**)**  img **=** bgr2gray**(**img**)**  self**.**img **=** img    # img = cv2.resize(img,(500,650))    *@staticmethod*  **def** plot**(**img**:**ndarray**):**  fig **=** plt**.**figure**()**  fig**.**canvas**.**manager**.**window**.**setWindowTitle**(**"Histogram"**)**  plt**.**plot**(**img**)**  plt**.**show**()**  **def** calhist**(**self**,**show\_hist**:bool=False)** **->** ndarray**:**  """Plotting with calhist method."""  # It accept parameters like ([img],[channel],mask,[histsize],range[0-255]).  hist **=** cv2**.**calcHist**([**self**.**img**],** **[**0**],** **None,** **[**256**],** **[**0**,** 256**])**  **if** show\_hist**:** histogram**.**plot**(**hist**)**  **return** hist  **def** plot\_bgr\_hist**(**self**):**  b**,** g**,** r **=** cv2**.**split**(**self**.**img**)**  #cv2.imshow("img", img)  # cv2.imshow("b", b)  # cv2.imshow("g", g)  # cv2.imshow("r", r)  #Plotting different channel with hist  plt**.**hist**(**b**.**ravel**(),** 256**,** **[**0**,** 256**])**  plt**.**hist**(**g**.**ravel**(),** 256**,** **[**0**,** 256**])**  plt**.**hist**(**r**.**ravel**(),** 256**,** **[**0**,** 256**])**  plt**.**title**(**"ColorFull Image"**)**  plt**.**show**()**  **def** equalization**(**self**)** **->** ndarray**:**  """Histogram equalization is good when of the image is confined to a particular region."""  img\_gray **=** self**.**img  equ **=** cv2**.**equalizeHist**(**img\_gray**)**  # res = hstack((img\_gray,equ)) #stacking images side-by-side  **return** equ  **def** imshow**(**title**:str,**img**:**ndarray**):**  """Show image."""  cv2**.**imshow**(**title**,**img**)**  cv2**.**waitKey**(**0**)**  cv2**.**destroyAllWindows**()**  **def** bgr2gray**(**img**:**Union**[str,**ndarray**],**blur**:bool=False)** **->** ndarray**:**  """Convert bgr to gray and remove noise points."""  **if** **isinstance(**img**,str):**img **=** cv2**.**imread**(**img**)** # Read image.    **if** blur**:**  img **=** cv2**.**GaussianBlur**(**img**,** **(**3**,** 3**),**0**)** # GaussianBlur  img **=** cv2**.**medianBlur**(**img**,** 3**)** # Median Filtering  img **=** cv2**.**GaussianBlur**(**img**,** **(**3**,** 3**),**0**)** # GaussianBlur  img **=** cv2**.**medianBlur**(**img**,** 3**)** # Median Filtering  **return** cv2**.**cvtColor**(**img**,**cv2**.**COLOR\_BGR2GRAY**)** # BGR to gray.  **def** plot\_histogram**(**img**:**ndarray**)** **->** **None:**  """Plot histogram."""  fig **=** plt**.**figure**()**  fig**.**canvas**.**manager**.**window**.**setWindowTitle**(**"Histogram"**)**  hist **=** cv2**.**calcHist**([**img**],** **[**0**],** **None,** **[**256**],** **[**0**,** 256**])**  plt**.**plot**(**hist**)**  plt**.**show**()**  **def** canny**(**img**:**ndarray**,min:int=**0**,max:int=**255**)** **->** ndarray**:**  """cv2.Canny (input gray)"""  **return** cv2**.**bitwise\_not**(**cv2**.**Canny**(**bgr2gray**(**img**),min,max))**    **def** canny\_edge**(**img**:**ndarray**)** **->** **tuple[int,int]:**  """Test Canny upper and lower values."""  cv2**.**namedWindow**(**"Canny"**)**  ### Create a track bar ### cv2.createTrackbar('Slider name', 'window name', min, max, fn)  cv2**.**createTrackbar**(**"minimum"**,** "Canny"**,** 0**,** 255**,** **lambda** \_**:**\_**)**  cv2**.**createTrackbar**(**"maximum"**,** "Canny"**,** 0**,** 255**,** **lambda** \_**:**\_**)**  cv2**.**setTrackbarPos**(**"minimum"**,** "Canny"**,** 0**)**  cv2**.**setTrackbarPos**(**"maximum"**,** "Canny"**,** 255**)**    img **=** bgr2gray**(**img**)**  **while** **True:**  ### Get trackbar position. ### cv2.setTrackbarPos('Slider name', 'window name', default)  **min=** cv2**.**getTrackbarPos**(**'minimum'**,**'Canny'**)**  **max=** cv2**.**getTrackbarPos**(**'maximum'**,**'Canny'**)**    cv2**.**imshow**(**"Canny"**,**canny**(**img**,min,max))**  key **=** cv2**.**waitKey**(**1**)**  **if** key **==** **ord(**'q'**)** **or** key **==** 27**:**  **break**  cv2**.**destroyAllWindows**()**  **return** **min,max**  **def** medianBlur**(**img**)** **->** ndarray**:return** cv2**.**medianBlur**(**img**,** 3**)**  **def** GaussianBlur**(**img**)** **->** ndarray**:** **return** cv2**.**GaussianBlur**(**img**,** **(**3**,** 3**),**0**)** |