图像变换

·AIPKU·

人工智能引论实践课 计算机视觉小班

主讲人：胡越予

Python 补充 0： 数据结构

- tuple
  - (1,2)
  - (3, 'abc’)
- list

```
l = []
l.append(1)
l.append('abc’)
print(l) # [1, 'abc']
```

- dict

```
d = {}
d['three'] = 3
d[3] = 'three'
d[1.23] = [1,2,3]

print(d['th'+'ree’]) # 3
print(d[1+2]) # three
```

Python 补充 1：I/O

- 标准输入 `input()`
- 文件输入输出：
  - 打开文件

    ```
    f = open('workfile', 'r') ['w' 'rb' 'wb']
    ```

  - 读文件

    ```
    s = f.read() [readline]
    ```

  - 写文件

    ```
    f.write('abc')
    ```

  - 关闭文件

    ```
    f.close()
    ```

Python 补充 1：I/O

- Python 对象序列化
  - pickle 模块

```
pickle.dump([1,2,3], open('a.pk','wb'))
a = pickle.load(open('a.pk','rb'))
print(a) # [1,2,3]
```

- numpy.save
- numpy.savetxt # 人类可读，一维或二维数组
- numpy.loadtxt
- savetxt，若逗号分隔，扩展名csv，则可用Excel打开

Python 补充 2：IPython Embed

```python
def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255.).astype(np.uint8)
    import IPython
    IPython.embed() # 嵌入模式，可直接在上下文环境中操作
    result = integral[gray] # Pixel-Wise mapping
    return result

img = cv2.imread('./example.jpg')
b, g, r = cv2.split(img) # 默认按最后一维拆分
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
result = np.stack([nb, ng, nr], -1)
cv2.imwrite('hist_equ.png', result)
```

Python 补充 3： Subprocess

- subprocess 执行子程序调用
- 目录下有可执行程序 ./sum
- ./sum 1 2 <输出 "3" >
- 现实现功能，使用 Python 调用该程序，计算加法

```
import subprocess as sp
a = 5
b = 9
child = sp.Popen(['./sum',str(a),str(b)], stdout=sp.PIPE)
res = child.communicate() # 阻塞
print(res[0]) # 输出 14
```

Python 补充 4： 文件系统操作

- glob 模块

```
[15:38:18] huyueyu:liblinear $ ls
COPYRIGHT              linear.cpp          my_train_back.cpp
Makefile               linear.def          predict
Makefile.win           linear.h            predict.c
README                 linear.o            python
README.multicore       make_label.py       train
blas                   mass_train.cpp      train.c
convert_to_solid       matlab              tron.cpp
convert_to_solid.cpp   my_train            tron.h
handle_vgg.cpp         my_train.cpp        tron.o
heart_scale            my_train2           windows
```

```
In [2]: glob.glob('./*.cpp')
Out[2]:
['./my_train.cpp',
 './linear.cpp',
 './convert_to_solid.cpp',
 './mass_train.cpp',
 './handle_vgg.cpp',
 './tron.cpp',
 './my_train_back.cpp']
```

- os 模块：大部分系统调用
  - mkdir, system, link...

Python 补充 5：进程并行

```
from multiprocessing import Pool
import subprocess as sp

def task(args):
    a = args[0]
    b = args[1]
    child = sp.Popen(['./sum',str(a),str(b)],
      stdout=sp.PIPE)
    res = child.communicate()
    return int(res[0])

pool = Pool(8)
result = pool.map(task, [(x, x**2) for x in range(0,8)])
print(result) # [0, 2, 6, 12, 20, 30, 42, 56]
```

图像变换

- Global
  - Histogram Equalization 直方图均衡化
- Local
  - Filters and Convolution 滤波器与卷积
  - Blurring 模糊
  - Denoising 降噪

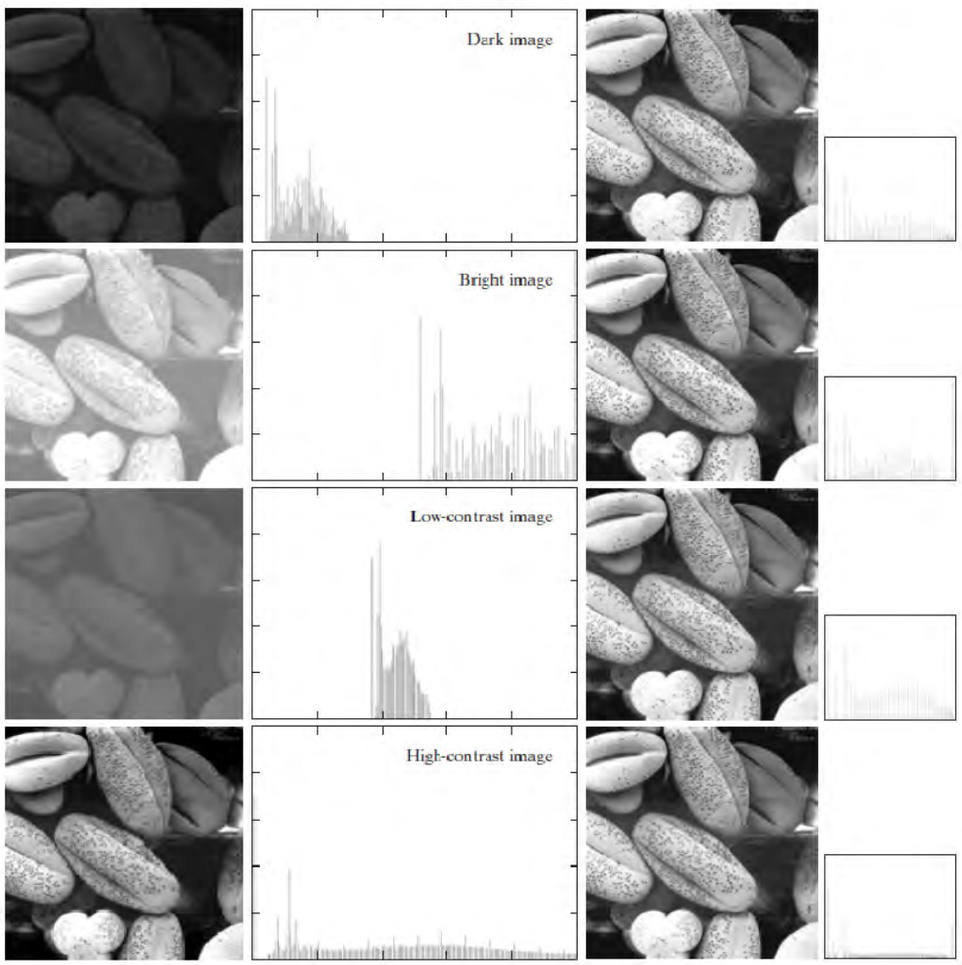
直方图均衡化

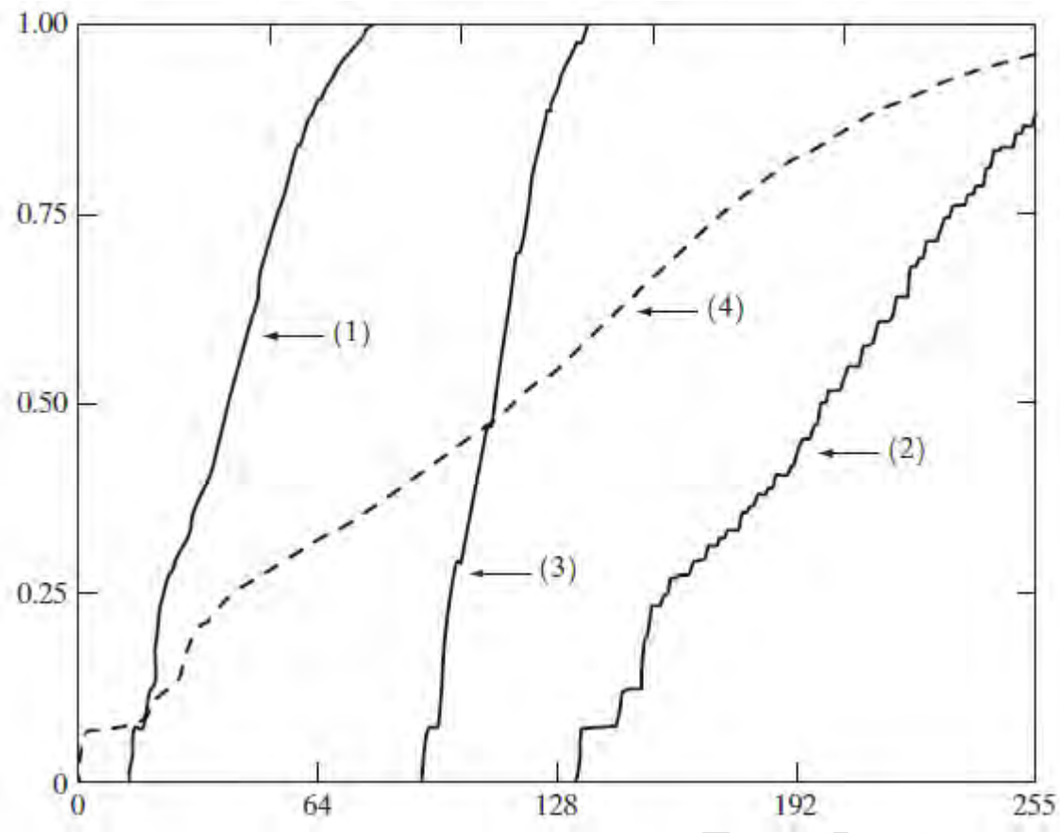
> 直方图均衡化

$$p_r(r_k) = \frac{n_k}{n}$$

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$$

$$k = 0, 1, 2, \ldots, L-1$$





直方图均衡化

> 直方图均衡化

原概率密度：$p_r(r)$

变换后概率密度：$p_s(s)$

$$S_r(x<n) = \int_0^n p_r(r)\,dr$$

$$S_s(x<n) = \int_0^n p_s(s)\,ds$$

排名不变

s.t. $S_r(n) = S_s(n)$ for any n in $[0,1]$

$$\Rightarrow p_r(r)\,dr = p_s(s)\,ds$$

$$s = T(r) = \int_0^r p_r(w)\,dw$$

$$\frac{ds}{dr} = p_r(r)$$

$$p_s(s) = p_r(r)\frac{dr}{ds} = p_r(r)\frac{1}{p_r(r)} = 1$$

直方图均衡化

```python
import cv2
import numpy as np

def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255.).astype(np.uint8)
    # now integral is a transformation function
    result = integral[gray] # Pixel-Wise mapping
    return result

img = cv2.imread('./example.jpg')
b, g, r = cv2.split(img) # 默认按最后一维拆分
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
result = np.stack([nb, ng, nr], -1)
cv2.imwrite('hist_equ.png', result)
```

Example

Transformed

Python List Comprehension

```
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
```

```
print(['#' + str(i) for i in range(0, 18, 3) if i % 2 == 0])
# ['#0', '#6', '#12’]
```

- List Comprehension 允许使用条件
- range 使用半开半闭区间
- str() 可以把对象直接转换成字符串
- int('123') == 123   float('1.2')

Documenting

```python
import cv2
import numpy as np

def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255.).astype(np.uint8)
    # now integral is a transformation function
    result = integral[gray] # Pixel-Wise mapping
    return result

img = cv2.imread('./example.jpg')
b, g, r = cv2.split(img) # 默认按最后一维拆分
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
result = np.stack([nb, ng, nr], -1)
cv2.imwrite('hist_equ.png', result)
```

help

- help(hist_equ)

```
Help on function hist_equ in module __main__:

hist_equ(gray)
    Conduct histogram equalization for gray scale image
```

- help(cv2.imread)

```
Help on built-in function imread

imread(...)
    imread(filename[, flags]) -> retval
    .   @brief Loads an image from a file.
    .
    .   @anchor imread
    .
    .   The function imread loads an image from the specified file and
returns it. If the image cannot be
    .   read (because of missing file, improper permissions, unsupported or
invalid format), the function
    .   returns an empty matrix ( Mat::data==NULL ).
```

dir

dir: returns list of the attributes and methods of any object

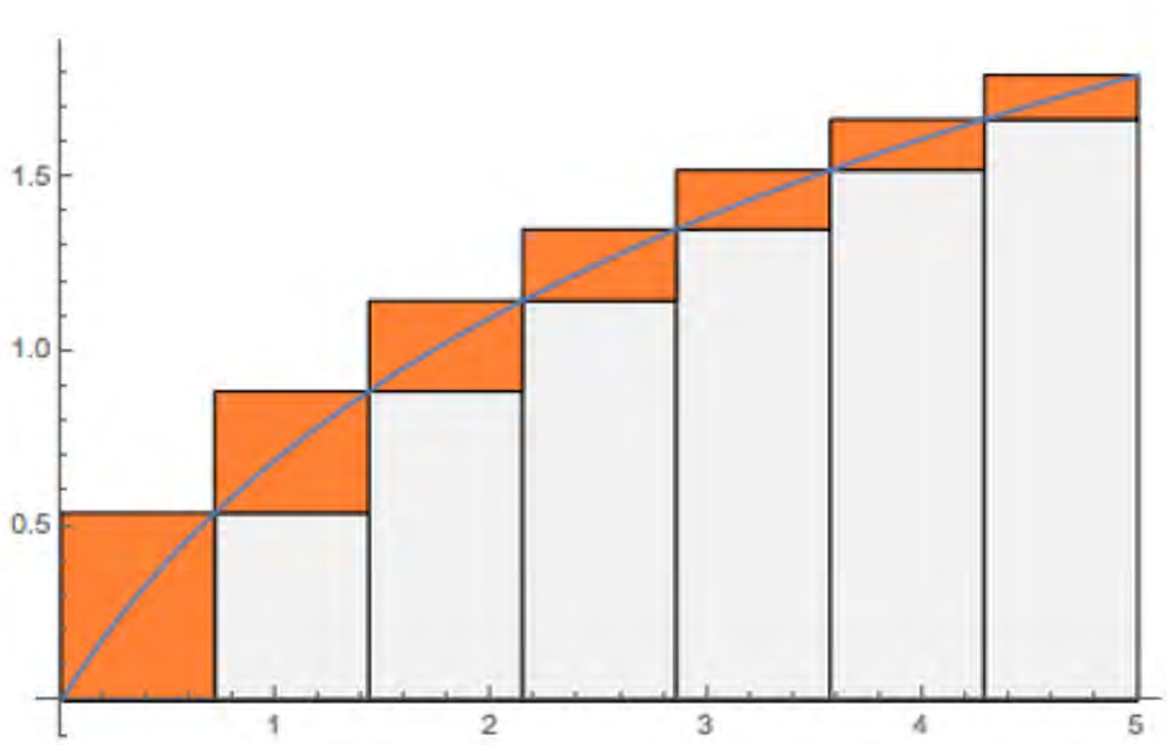
```
In [24]: dir(cv2)
Out[24]:
['',
 'ACCESS_FAST',
 'ACCESS_MASK',
 'ACCESS_READ',
 'ACCESS_RW',
 'ACCESS_WRITE',
 'ADAPTIVE_THRESH_GAUSSIAN_C',
 'ADAPTIVE_THRESH_MEAN_C',
 'AGAST_FEATURE_DETECTOR_AGAST_5_8',
 'AGAST_FEATURE_DETECTOR_AGAST_7_12D',
 'AGAST_FEATURE_DETECTOR_AGAST_7_12S',
 'AGAST_FEATURE_DETECTOR_NONMAX_SUPPRESSION’,
...]
```

离散积分

```
import cv2
import numpy as np

def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255 ).astype(np.uint8)
    # now in
    result =
    return re

img = cv2.im
b, g, r = cv
nb, ng, nr =                                    ]]
result = np.
cv2.imwrite(
```



Indexing

```python
import cv2
import numpy as np

def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255.).astype(np.uint8)
    # now integral is a transformation function
    result = integral[gray] # Pixel-Wise mapping
    return result

img = cv2.imread('./example.jpg')
b, g, r = cv2.split(img) # 默认按最后一维拆分
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
result = np.stack([nb, ng, nr], -1)
cv2.imwrite('hist_equ.png', result)
```

存在的问题：颜色异常

```python
import cv2
import numpy as np

def hist_equ(gray):
    '''Conduct histogram equalization for gray scale image'''
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist) # Cumulative Sum
    integral = (integral * 255.).astype(np.uint8)
    # now integral is a transformation function
    result = integral[gray] # Pixel-Wise mapping
    return result

img = cv2.imread('./example.jpg')
b, g, r = cv2.split(img)
nb, ng, nr = [hist_equ(gray) for gray in [b, g, r]]
result = np.stack([nb, ng, nr], -1)
cv2.imwrite('hist_equ.png', result)
```

换用HSV色彩空间

```python
import cv2
import numpy as np

def hist_equ(gray):
    hist = np.histogram(gray, 256, [0, 256])
    norm_hist = hist[0] / (gray.shape[0] * gray.shape[1])
    integral = np.cumsum(norm_hist)
    integral = (integral * 255.).astype(np.uint8)
    result = integral[gray]
    return result

img = cv2.imread('./example.jpg')
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)
new_v = hist_equ(v)
new_hsv = np.stack([h, s, new_v], -1)
result = cv2.cvtColor(new_hsv, cv2.COLOR_HSV2BGR)
cv2.imwrite('hist_equ_hsv.png', result)
```

Transformed HSV

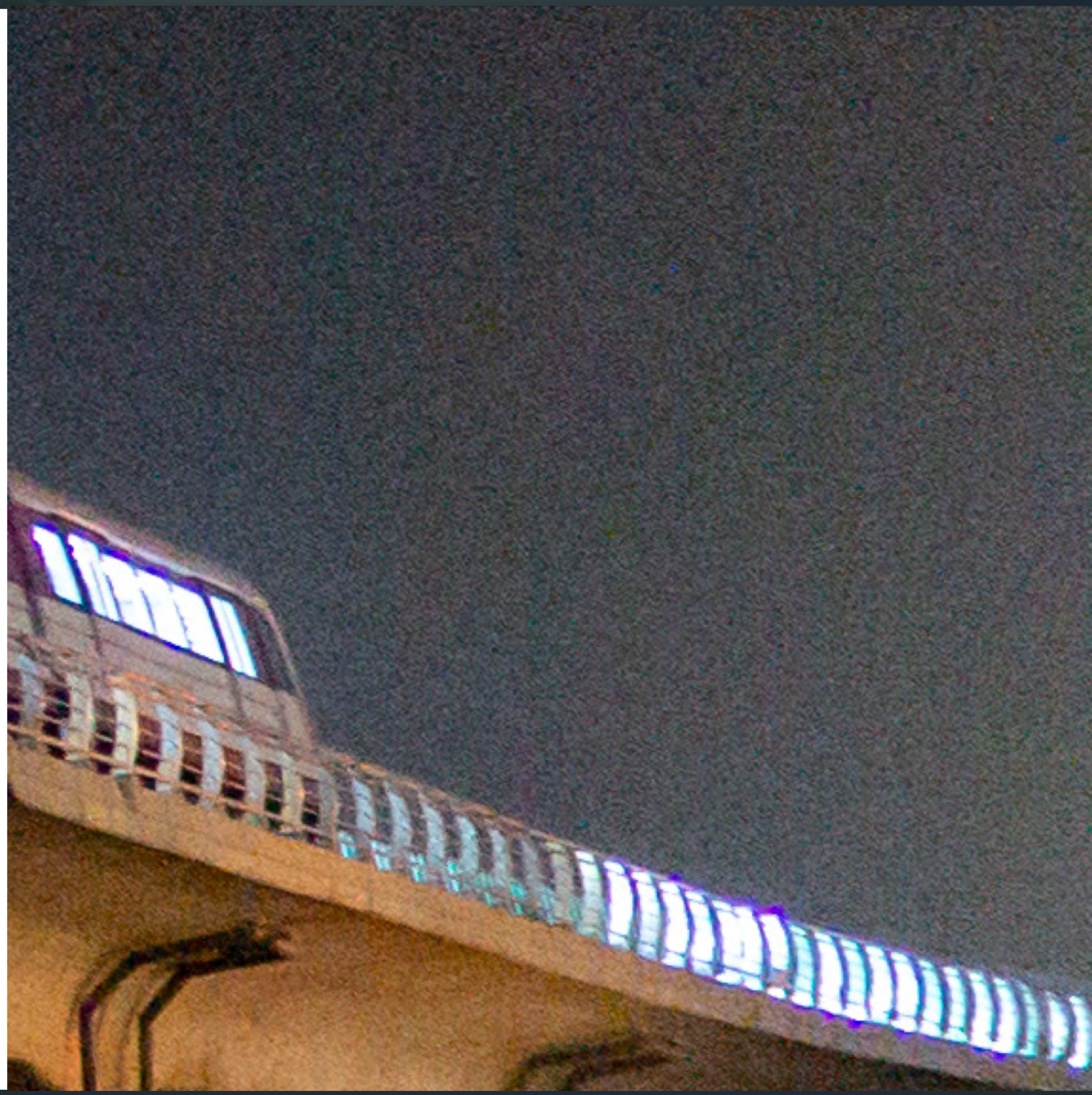
Transformed RGB

仍然存在的问题：噪声

噪声

- 图像中随机亮度/颜色改变
- 原因：物理过程，电路…
- 影响视觉质量和识别性能
- 非图像成分
- 滤波可一定程度消除噪声

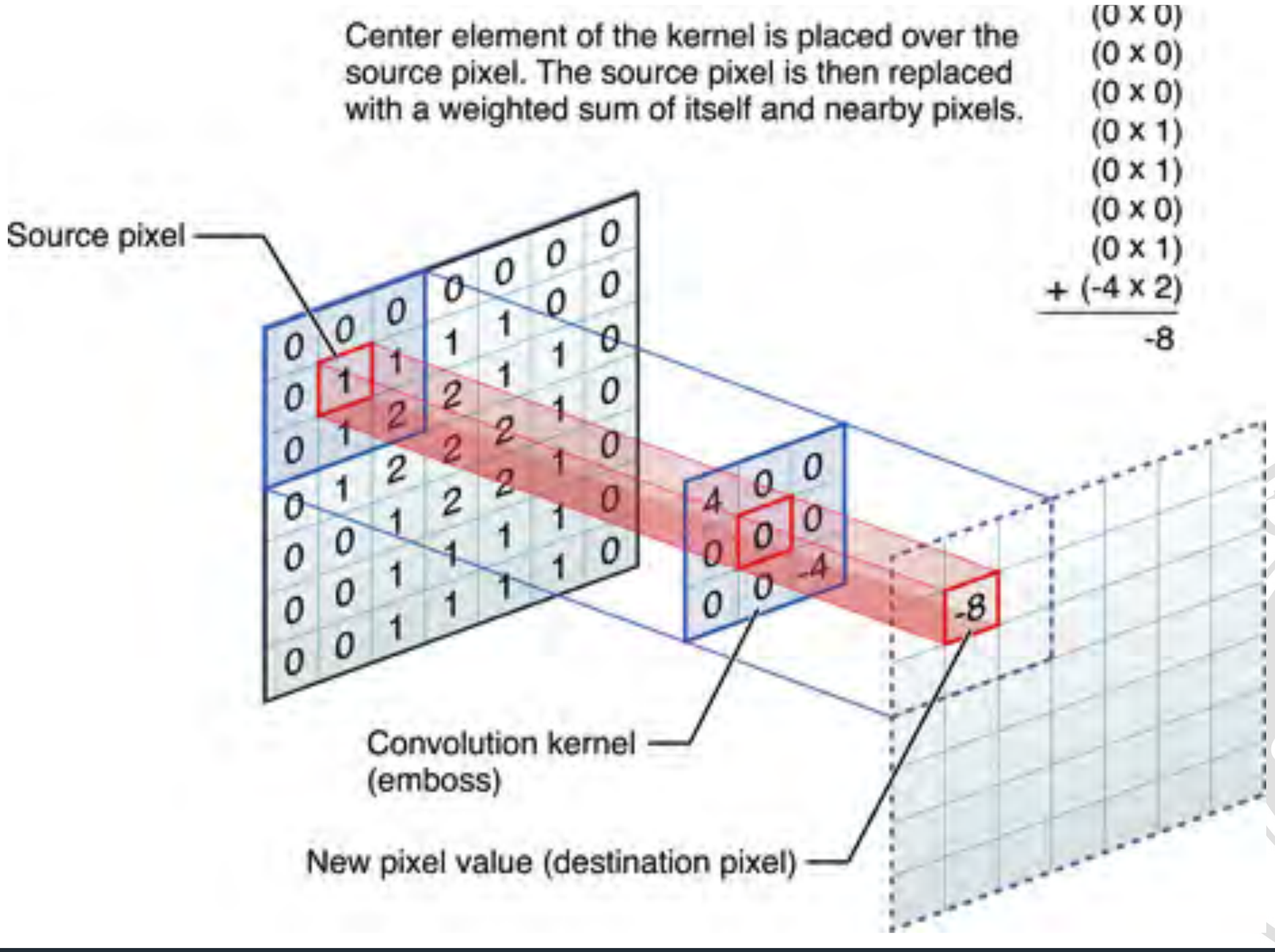
- 为什么小屏幕上不容易看出明显的噪点？



重采样

局部变换

- Convolution 卷积（空间滤波器）



卷积

- 输入：图像
- 参数：Kernel
- 输出：图像

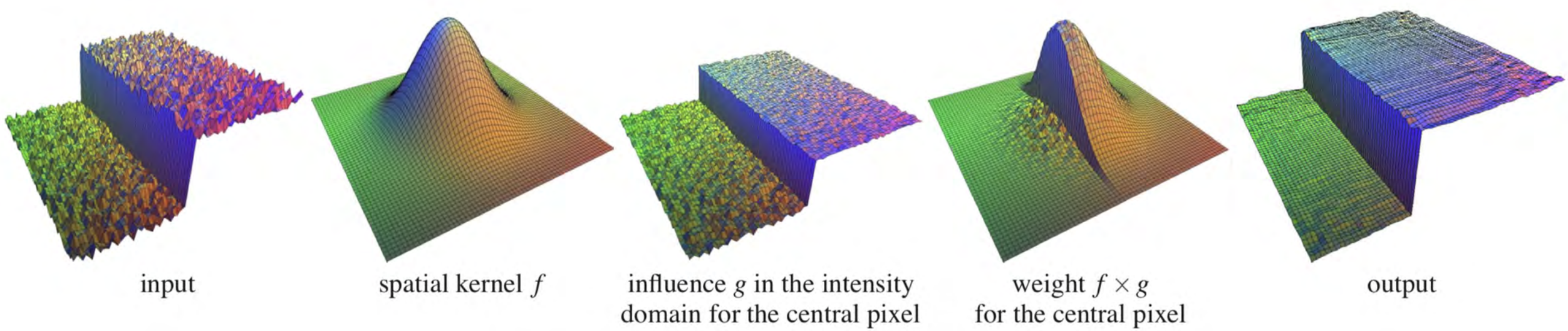
```
cv2.filter2D(
        src, # 输入图像
        ddepth, # 控制输出图像数据类型，-1 默认相同类型
        kernel # 卷积核
        [, dst[, anchor[,
         delta[, borderType]]]]
) → dst
```

高斯模糊

```
cv2.GaussianBlur(
    src,
    ksize, # 卷积核大小
    sigmaX # 标准差大小，影响模糊程度
    [, dst[, sigmaY[, borderType]]]
) → dst
```

双边滤波

```
cv2.bilateralFilter(
    src,
    d, # 邻域直径
    sigmaColor, #
    sigmaSpace
[, dst[, borderType]]) → dst
```



input　spatial kernel f　influence g in the intensity domain for the central pixel　weight $f \times g$ for the central pixel　output

Bilateral Filter and Gaussian Filter

噪声

- 图像中随机亮度/颜色改变
- 原因：物理过程，电路，有损压缩
- JPEG 量化噪声 `cv2.imwrite('q10.jpg', img, params=[cv2.IMWRITE_JPEG_QUALITY, 10])`

Further Reading

- Median Filter 中值滤波
- Non-Local Means
  - Buades, Antoni, Bartomeu Coll, and J-M. Morel. "A non-local algorithm for image denoising." *CVPR 2005*.
- BM3D
  - http://www.cs.tut.fi/~foi/GCF-BM3D/