







Computer Vision HW1 Report



Student ID: B11901067

Name: 董家愷


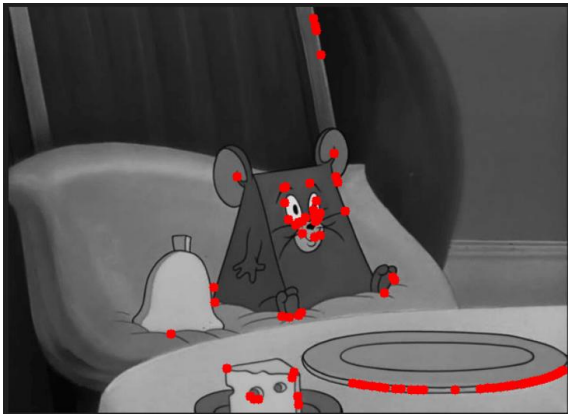
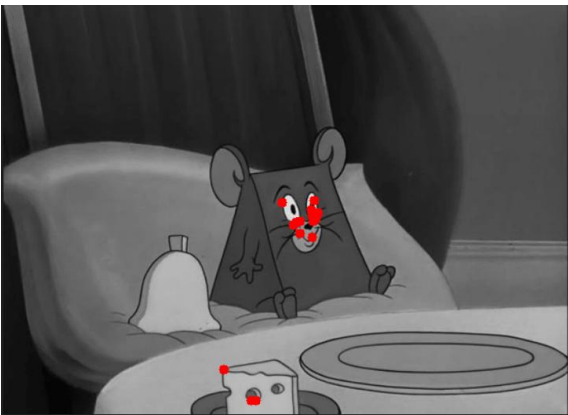
Part 1.

- Visualize the DoG images of 1.png.

	DoG Image (threshold = 5)		DoG Image (threshold = 5)
DoG1-1.png		DoG2-1.png	
DoG1-2.png		DoG2-2.png	
DoG1-3.png		DoG2-3.png	

DoG1-4.png		DoG2-4.png	
------------	---	------------	---

- Use three thresholds (1,2,3) on 2.png and describe the difference.

Threshold	Image with detected keypoints on 2.png		
2			
5			
7			

(describe the difference)

Threshold = 2: 因為 threshold 較低而顯示較多的特徵點，遍布各個物體及區域，幾乎能完整框出主角傑利鼠，但也標記了不太重要的背景如椅背、桌緣

Threshold = 5: 相較於第一張圖片，背景區域的特徵點較少，主要集中在傑利鼠面部、餐盤邊緣。

Threshold = 7: 相較上一張特徵點又更少，所代表的資訊若要用做後續訓練模型可能不太夠，只剩下傑利鼠面部及乳酪。






Part 2.

- Report the cost for each filtered image.

Gray Scale Setting	Cost (1.png)
cv2.COLOR_BGR2GRAY	1207799
$R*0.0+G*0.0+B*1.0$	1439568
$R*0.0+G*1.0+B*0.0$	1305961
$R*0.1+G*0.0+B*0.9$	1393620
$R*0.1+G*0.4+B*0.5$	1279697
$R*0.8+G*0.2+B*0.0$	1127913

Gray Scale Setting	Cost (2.png)
cv2.COLOR_BGR2GRAY	436673
$R*0.1+G*0.0+B*0.9$	143870
$R*0.2+G*0.0+B*0.8$	178405
$R*0.2+G*0.8+B*0.0$	432475
$R*0.4+G*0.0+B*0.6$	284221
$R*1.0+G*0.0+B*0.0$	321971

- Show original RGB image / two filtered RGB images and two grayscale images with highest and lowest cost.

Original RGB image (1.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		
		


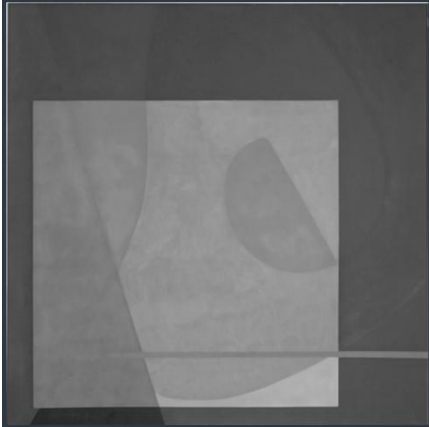



(Describe the difference between those two grayscale images)

從上圖可以看出，使用 $RGB=(0, 0, 1)$ 作為 guidance 的圖片，其 cost function 最高（即相似度最低）。這可能是因為原圖主要由紅色和綠色構成，而 guidance 的藍色與之無關，導致 JBF 在處理時錯誤地平滑了一些區域，或無法有效保留原圖的紅綠色調，從而降低了相似度。

相對地，使用 $RGB=(0.8, 0.2, 0)$ 作為 guidance 的圖片，cost function 最低（即相似度最高）。這可能是

因為 guidance 的色彩與原圖更為匹配，使得 JBF 能夠更好地保留楓葉的細節和對比度。同時，紅色與綠色的比例為 0.8:0.2，使得在兩者交界處能進行適當的平滑處理，而不會使邊界模糊。

此外，也可以看出，使用前者 RGB 係數的灰階圖片無法有效區分楓葉與背景綠地，而使用後者 RGB 係數的灰階圖片則能清晰地區分楓葉與綠地，並保持一定的亮度。

Original RGB image (2.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		
		

(Describe the difference between those two grayscale images)
這張圖片包括大面積的紫色、紫紅色、藍色，以及少量的橘色和黃色。根據三原色合成原理，我們可以推斷，含綠色較多的導引圖會造成較高的 cost。從上面的列表中可以看到，RGB=(0.2, 0.8, 0) 的 cost 與標準灰階相似，皆顯示出較高的 loss function，與其他 RGB 設置相比較不適合作為 guidance。

從圖中可以看出，標準灰階未能有效區分各色塊的位置，甚至無法清晰分辨外圍的深藍和紫紅色區域。相較之下，RGB=(0.1, 0, 0.9) 能夠更精確地區分各個色塊，因此在作為 guidance 時，能夠提供最高的 perceptual similarity。

Describe how to speed up the implementation of bilateral filter.
1. Use look-up table for spatial kernel and range kernel

```
x = np.arange(-self.pad_w, self.pad_w + 1)
X, Y = np.meshgrid(x, x)
s_kernel_2D = np.exp(-0.5 * (X**2 + Y**2) / self.sigma_s**2)
r_kernel = np.exp(-0.5*(np.arange(256)/255)**2/self.sigma_r**2)
```

後續計算時直接索引兩個 kernel 取值避免重複計算

```
for y in range(-self.pad_w, self.pad_w + 1):
    for x in range(-self.pad_w, self.pad_w + 1):
        offsets.append((y, x))
        spatial_weights.append(s_kernel_2D[self.pad_w + y, self.pad_w + x])
```

```
if guidance_diff.ndim == 2: # 灰階
    r_w = r_kernel[guidance_diff]
else: # 彩色
    r_w = np.prod([r_kernel[guidance_diff[:, :, c]] for c in range(guidance_diff.shape[2])], axis=0)
```

2. 預先計算所有 offset 組合再對每個偏移組合作運算

```
offsets = []
spatial_weights = []

for y in range(-self.pad_w, self.pad_w + 1):
    for x in range(-self.pad_w, self.pad_w + 1):
        offsets.append((y, x))
```

```
for i, ((y, x), s_w) in enumerate(zip(offsets, spatial_weights)):
    shifted_guidance = np.roll(padded_guidance, [y, x], axis=[0, 1])
    guidance_diff = np.abs(shifted_guidance - padded_guidance).astype(np.float32)

    if guidance_diff.ndim == 2: # 灰階
        r_w = r_kernel[guidance_diff]
    else: # 彩色
        r_w = np.prod([r_kernel[guidance_diff[:, :, c]] for c in range(guidance_diff.shape[2])], axis=0)

    t_w = s_w * r_w
```