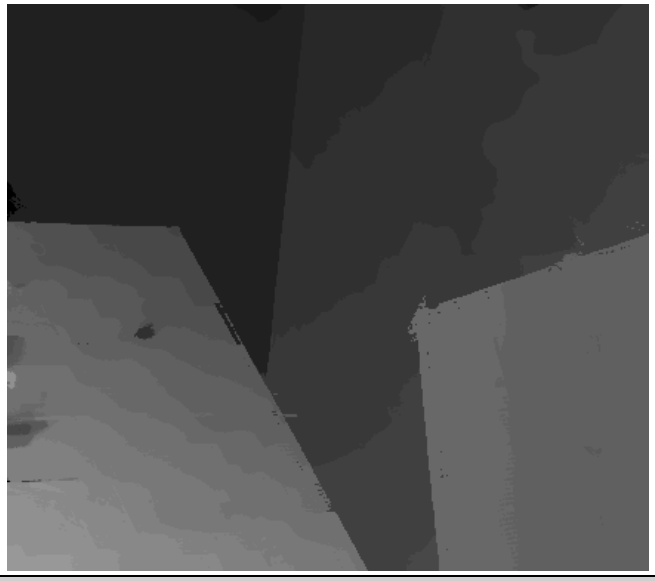
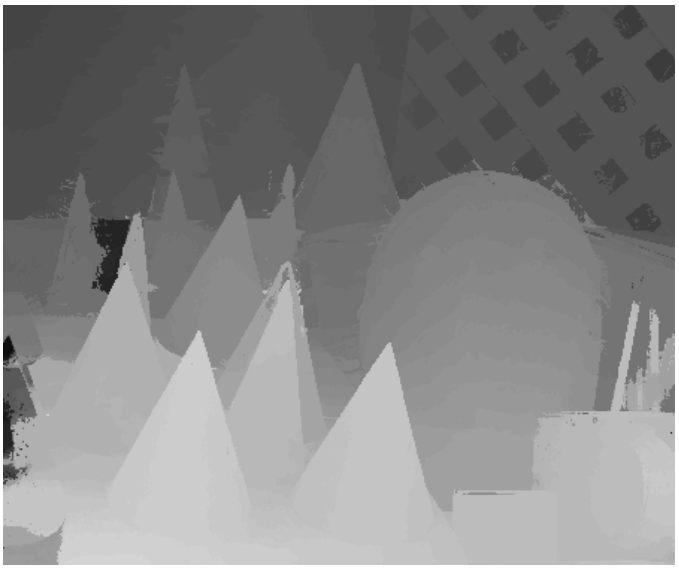
Computer Vision HW4 Report

Student ID: R10921A36
Name: 石子仙

Visualize the disparity map of 4 testing images.

Tsukuba	Venus
	
Teddy	**Cones**
	

Report the bad pixel ratio of 2 testing images with given ground truth (Tsukuba/Teddy).

	bad pixel ratio
Tsukuba	3.59%
Teddy	10.48%

Describe your algorithm in terms of 4-step pipeline.

- 運行時間：

	Processing Time(sec)
Tsukuba	2.7943
Teddy	11.6472
Venus	4.9055
Cones	11.9582

- Cost computation
  - 前處理：先將每個 pixel 的 local binary pattern 算出，使後續步驟可以查表、進行平行運算加速

  1. 使用 cv2.copyMakeBorder 將圖片取 windows 時，會超出原圖範圍的部分 padding 成 value=0。
  2. 使用 np.lib.stride_tricks.sliding_window_view 來得到各個 window，回傳值 windows_array 的 shape 為(h, w, window_size, window_size, channel)。例如當取 windows_array[r,c]，可以得到 image[r,c]的 window pattern(shape 為 window_size*window_size*channel)。
  3. 針對每個 window 將其轉成 local binary pattern：

  ```
  for row in image.h:
      for col in image.w:
          local_binary_pattern[r,c] = windows_array[r,c] >= image[r,c]
  ```

  如此一來就得到每個 pixel 的 local binary pattern。

- 計算 Cost：因為每次 Disparity 都只會影響 column 取值的範圍，不影響 row，所以可以針對所有 row 平行運算。

```
for col in image.w:
    for disp in max_disp:
        if (not out of bound):
            ### "Il to Ir"
            census_cost_l_to_r[:, col, disp] = Hamming distance of Il[:, col] to Ir[:, col - disp]
            ### "Ir to Il"
            census_cost_r_to_l[:, col, disp] = Hamming distance of Il[:, col + disp] to Ir[:, col]
```

每次計算會有一些 out of bound，當 Disparity=d 時，Il to Ir，Ir to Il 分別會在頭尾有 d 個 columns out of bound，在這邊把 out of bound 補成最近的值。

```
for disp in max_disp:
    census_cost_l_to_r[:,:disp, disp] = census_cost_l_to_r[:,disp+1,disp][..., None]
    census_cost_r_to_l[:,-disp:,disp] = census_cost_r_to_l[:,-(disp+1),disp][..., None]
```

完成 Cost 的計算。

- Cost aggregation

使用 Joint bilateral filter 來 refine cost，這裡用到的套件是是 cv2.xmgproc.jointBilateralFilter。

```
for disp in range(max_disp):
        # to both Il->Ir Ir->Il
        refine_cost[:, :, disp] = xip.jointBilateralFilter(Il, census_cost[:, :, disp])
```

- Disparity optimization

使用 np.argmin 進行 Winner-take-all，得到 Disparity map D_L 和 D_r。

- Disparity refinement

依序進行 Left-right consistency check、Hole filling、Weighted median filtering

1. Left-right consistency

   檢查是否 $D_L(x,y) = D_R(x - D_L(x,y),y)$，若不一致則標記為 hole。

for row in image.h:

for col in image.w:

if not $D_L(x,y) = D_R(x - D_L(x,y), y)$

mark as hole

2. Hole filling

for row in image.h:

for col in image.w:

if hole:

initial left_nearest_value and right_nearest_value as max_disparity

left_nearest_value = the nearest left value (if out of bound, stop search)

right_nearest_value = the nearest right value (if out of bound, stop search)

disparity[row, col] = min(left_nearest_value, right_nearest_value)

3. Weighted median filtering

使用 cv2.xmgproc.weightedMedianFilter 進行 Weighted median filtering。