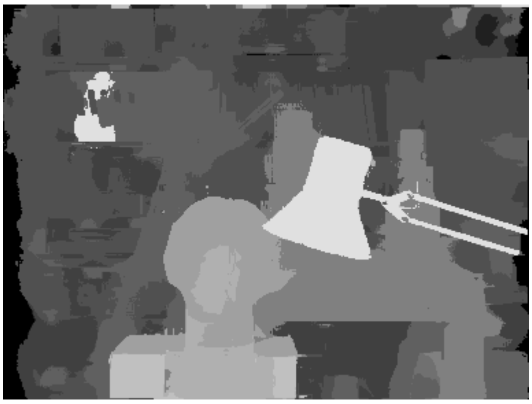
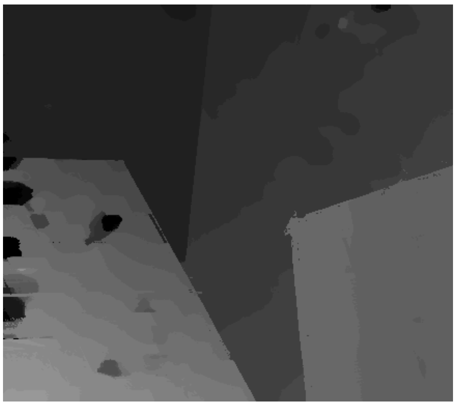
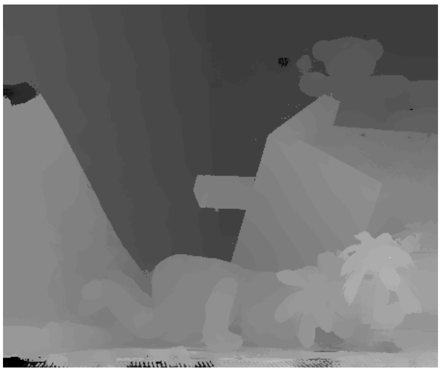
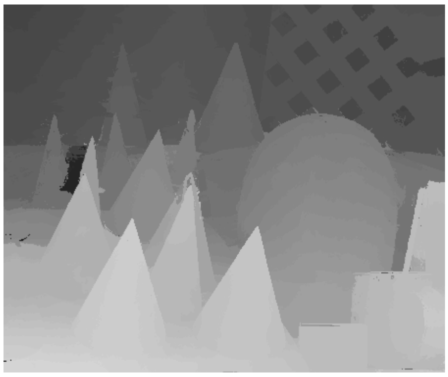
Computer Vision HW4 Report

Student ID: R12921A10
Name: 謝宗翰

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	3.91%
Teddy	10.58%

Describe your algorithm in terms of 4-step pipeline.

1. Cost computation

   Using matrix operation to calculate the local binary matrix from the left and right images.

```python
for i in range(h):
        for j in range(w):
            IL_window = img_L[i : i + window_size, j : j + window_size, :].copy()
            IR_window = img_R[i : i + window_size, j : j + window_size, :].copy()

            for c in range(ch):
                apply_threshold(IL_window[:, :, c],IL_window[pad_size, pad_size, c])
                apply_threshold(IR_window[:, :, c],IR_window[pad_size, pad_size, c])

            local_binary_IL[i, j] = IL_window
            local_binary_IR[i, j] = IR_window
```

2. Cost aggregation

   Using the filters to filter out the disparity maps. I tried many versions of the filter, the best filter I found is bilateral filter. Second, the bilateral filter's parameter will affect the bad pixel ratio a lot. Thus, I tried to use the brute force method to test every combination of the parameter.

```

Processing image Teddy ...
[Time] 11.7684 sec
i,j,k = 29,29,16
[Bad Pixel Ratio] 11.30%
best i,j,k = 17,13,12 / error = 10.52593381072189

Processing image Teddy ...
[Time] 11.6156 sec
i,j,k = 29,29,17
[Bad Pixel Ratio] 11.32%
best i,j,k = 17,13,12 / error = 10.52593381072189

Processing image Teddy ...
[Time] 11.7037 sec
i,j,k = 29,29,18
[Bad Pixel Ratio] 11.56%
best i,j,k = 17,13,12 / error = 10.52593381072189

Processing image Teddy ...
[Time] 11.6423 sec
i,j,k = 29,29,19
[Bad Pixel Ratio] 11.69%
best i,j,k = 17,13,12 / error = 10.52593381072189

```

3. Disparity optimization

   Using the winner take all's way to choose the min value.

```python
win_L = np.argmin(cost_L, axis=2)
win_R = np.argmin(cost_R, axis=2)
```

4. Disparity refinement

   Check the consistency of the left-right to improve the disparity map quality and fill out the hole which is invalid disparity map. In the end, I used the weightedMedianFilter to improve the disparity map

```
for i in range(h):
        for j in range(w):
            if win_L[i, j] != win_R[i, j - win_L[i, j]]:
                win_L[i, j]=-1

    for i in range(h):
        for j in range(w):
            if win_L[i, j] == -1:
                idx_L = j - 1
                while idx_L >= 0 and win_L[i, idx_L] == -1:
                    idx_L -= 1

                FL = win_L[i, idx_L] if idx_L >= 0 else float('inf')

                idx_R = j + 1
                while idx_R < w and win_L[i, idx_R] == -1:
                    idx_R += 1

                FR = win_L[i, idx_R] if idx_R < w else float('inf')

                win_L[i, j] = min(FL, FR)
```