
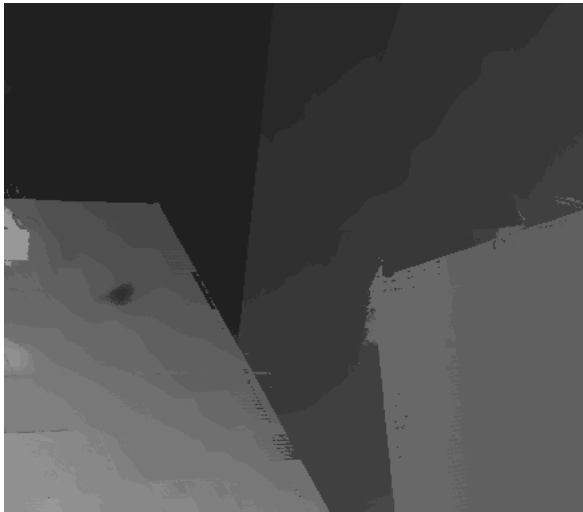

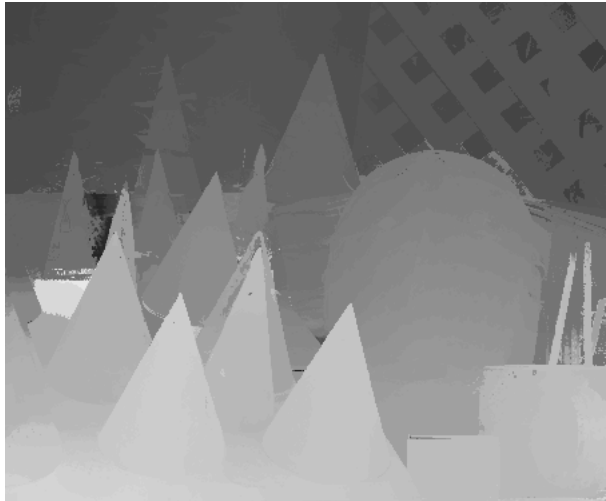


Computer Vision HW4 Report

Student ID: B09901075

Name: 陳駿瑋

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 | 4.40% |
| Teddy | 11.86% |

Describe your algorithm in terms of 4-step pipeline.

Step 1: Compute Matching Cost

一開始先計算 matching cost, 對於每個 pixel, 要計算 $\max_disp + 1$ 個 costs。這裡我照著投影片使用 Census Cost。為此我先對原圖做 padding, 讓邊界的 pixels 在與 neighbors 比較時不會出錯。而經過測試, padding 的方法使用 `cv2.BORDER_REPLICATE` (複製邊界 pixel 值) 產生出的 bad pixel ratio 值較低。Padding 後, 計算出兩個 $8 \times \text{image_size}$ 的 binary 矩陣, 代表每個 pixel 與 8 個 neighbors 比大小後的結果 (大於=1)。最後對 \max_disp 做 iteration, 每圈對應的 disp 值決定右圖橫移距離, 並在右圖橫移後, 每個 pixel 與左圖對應, 用左右圖各 8 個值計算出 hamming distance (同時會把 3 個 channel 的 hamming distance 加起來), 如此一個 iteration 會產生兩個 $h \times w$ 的 cost 矩陣 (左到右和右到左)。最後能產生兩個 shape 為 $(\max_disp + 1, h, w)$ 的 cost 矩陣。

Step 2: Cost Aggregation

這一步中，雖然是 step 2，但在我的程式中，融合進上一步的 for 迴圈。在每個迴圈，可以計算出兩個 $h * w$ 的 cost 矩陣，為了減少雜訊，可以再對這個 cost 做 filter，我選擇的是[Tips]建議的 `jointBilateralFilter`。值得注意的是，`jointBilateralFilter` 的參數是影響結果 bad pixel ratio 一個很重要的因素。這裡的參數（`diameter`, `sigmaColor`, `sigmaSpace` = 22, 10, 26）是先隨便設定，最後慢慢依照 bad pixel ratio 的結果調整得出的結果。

Step 3: Disparity Optimization

這步很單純的把前面每個 pixel 算出的 `max_disp+1` 個 costs，取最小值的 index，並記錄下來，產生 disparity map。

Step 4: Disparity Refinement

最後是加強第三步產生的 disparity map。首先是檢查 $DL(x, y) = DR(x - DL(x, y), y)$ ，看左到右和右到左的結果是否相同，若是不同，就在左到右圖的 map 挖一個 hole（將 index 值設為-1）。之後 hole filling 的部分，利用 for 迴圈掃過左到右圖的 map，如果有 hole，就從它左右延伸（while loop），找到 pixel 距離最近的兩個點（一左一右），並選擇值較小的填入。而如果往左或往右時，沒有找到非 hole 的 pixel 就到邊界了，那值就設為 `max_disp`。最後在填完 holes 之後，對其做 `weightedMedianFilter` 去改善結果。