Homework 4: Binocular Stereo

November 19, 2023

Due Date: December 10 by 23:59:59

Introduction

In this project, you will implement a stereo matching algorithm for rectified stereo pairs. We just consider the simplest case: Image planes of cameras are parallel to each other and to the baseline.

1 Simple Stereo System (40 pts.)

Run the sample code and visualize the results of **cv2.StereoBM**.

Add your own code to the **simple_disparity()** function to implement a simple window-based stereo matching algorithm. Follow the outline given in slides 08: pick a window around each pixel in the first (reference) image, and then search the corresponding scanline in the second image for a matching window with the least difference. The output should be a disparity map with respect to the first view. Report the comparison of your disparity maps and groundtruth maps.

Function **visualize_disparity_map()** can be used to compare your results and groundtruth maps.

2 Algorithm Settings (30 pts.)

After finishing the Task 1, you should experiment the algorithm with the following settings and parameters, and report the visualization and runing-time results:

Search window size: show disparity maps for several window sizes and discuss which window size works the best (or what are the tradeoffs between using different window sizes). How does the running time depend on window size?

Disparity range: what is the range of the scanline in the second image that should be traversed in order to find a match for a given location in the first image? Examine the stereo pair to determine what is the maximum disparity value that makes sense, where to start the search on the scanline, and which direction to search in. Report which settings you ended up using.

Matching function: try sum of squared differences (SSD), sum of absolute differences (SAD), and normalized correlation. Discuss in your report whether there is any difference between using these functions, both in terms of quality of the results and in terms of running time.

3 Visualize the depth map in 3D (20 pts.)

Convert your disparity map to a depth map and attempt to visualize the depth map to pointcloud in **visualize_pointcloud()**. Follow the outline in slides to calculate depth from disparity. XY in image and depth can be converted to XYZ in pointcloud, and the reference image can be viewed as colors in pointcloud. Try to "guess" the depth scaling constant **baseline** and **focal_length** to get a better performance. The sample code stores pointcloud to *ply* file, which can be visualized by MeshLab.

4 Dynamic Programming Algorithm (10 pts.)

Try to incorporate non-local constraints (smoothness, uniqueness, ordering) into your algorithm. You can try to implement the dynamic programming algorithm discussed in slides.

5 Report

You have now completed the entire process of this project. Put all the visualization and running-time results in your report. More experiments and discussions are encouraged and may get a bonus.

6 Hints

Here are some supplemental materials:

- Stereo Matching: https://zhuanlan.zhihu.com/p/161276985
- cv2.StereoBM: https://docs.opencv.org/4.x/d9/dba/classcv_1_1StereoBM.html
- cv2.StereoBeliefPropagation:
- https://docs.opencv.org/4.x/de/d7a/classcv_1_1cuda_1_1StereoBeliefPropagation.html