Exercise 4

Deadline: Friday, February 03, 2017

Please send your solutions to threedcv@dfki.uni-kl.de

Dense 3D Stereo Reconstruction - Theory

Normally, in dense 3D reconstruction, one of the challenges is to find the correct correspondences among all pixels between the images. As explained in the lecture, if the epipolar geometry is known, the correspondence search reduces to a 1-D search problem. In a stereo setup (two images only), image rectification may be used to further simplify the search for correspondences.

Questions
Q_1 : What is the advantage of a rectified image pair regarding correspondence search?
Q_2 : What is the advantage of a rectified image pair regarding triangulation?
Q_3 : Is image rectification also a good approach in case of multi-view dense reconstruction? Why? (*Hint: Consider a scenario where several images were taken around a statue.*)

Implementation

Goal

The goal of this exercise is to perform dense matching on *rectified* images and reconstruct the original 3D scene.

Requirements

The tasks below are to be implemented in Python and the following packages are required:

- numpy
- scipy
- python-opencv
- MeshLab: installation **sudo apt-get install meshlab** for ubuntu . For windows users you can download it form source. [1]

Datasets

There are two pairs of rectified images. The first pair, from the KITTI dataset [1], consists of two grayscale images. The second pair, from the Middlebury dataset [2], consists of color images. The depth is obtained by

$$Z = \frac{b \times f}{d + doffs},$$

where b is the baseline, f the focal length, d the disparity and $doffs$ the x-difference of principal points, that is $doffs = cx_1 - cx_0$. In case of the KITTI dataset, $doffs = 0$. For more details, check the explanation in the provided source code and the readme files of the datasets in the data folder.

[1] www.meshlab.net

Template

Use the provided template code to implement a dense matching and reconstruction algorithm according to the tasks below (fill in the missing gaps in the code). Note that it is necessary to account for the number of channels depending on the target dataset.

Flowers Dataset

You are given a stereo pair from flowers dataset, (left image)im0.png and (right image) im1.png
$Task_1$:

✓ 1. For each pixel in the left image, find the corresponding pixel on the right image using a *pixel-based* cost function, i.e. the difference between individual left and right pixels

✓ 2. Compute the disparity between the left and right images, then display it as a gray scale image.

Questions
Q_1 : Does the recovered depth map look a good representation of the original scene? Why?

$Task_2$:

✓ 1. For each pixel in the left image, find the corresponding pixel on the right image using a *window-based* NCC (Normalized Cross Correlation) cost function. Use window size of 7.

2. Compute the disparity for each pixel in the left image and display it as a gray scale image.

3. Experiment with different window sizes: 3x3, 5x5, 9x9 pixels, etc. See if they can give better result compared to window size of 5.

4. Compared to the result of the pixel-wise approach of $Task_1$, has the quality of the depth map improved when using window based comparison ? explain why?

KITTI Dataset

Given, the left and right images from the KITTI dataset folder.
$Task_1$:

✓ 1. For each pixel in the left image, find the corresponding pixel on the right image using a *pixel-based* cost function, i.e. the difference between individual left and right pixels

✓ 2. Triangulate all valid correspondences to reconstruct the original 3D scene

✓ 3. Display the reconstructed point cloud using **MeshLab or a python package of your choice**

Questions
Q_3 : Are the reconstructed point clouds a good representation of the original scene? Why?

$Task_2$:

1. For each pixel in the left image, find the corresponding pixel on the right image using a *window-based* SSD (Sum of Squared Differences) cost function. Use window size 5.

2. For a given pixel in the left image, if 2 points in the right image have SSD scores less than $1.5 \times$ *min_ssd_score*, consider it as an outlier and reject the pixel. This will serve as a simple outlier filtering scheme.

3. Display the disparity as a gray scale image.

4. Triangulate all valid correspondences to reconstruct the original 3D scene.

5. Display the reconstructed point cloud

6. Experiment with different window sizes: 3x3, 7x7, 9x9 pixels, etc. See if they can give better result compared to window size of 5. What was the best window size you got? (You can compare the results visually and decide the best size)

Questions
From you observations from both, Flowers and KITTI experiments.
Q_1 : What is the influence of the size of window on the reconstruction quality?
Q_2 : What is the influence of the size of window on the processing time?

Hand in the answers to all questions and tasks, the complete source code and screenshots of the 3D reconstructions (also for at least two window sizes, one for the best window size and for non optimal window size).
If your algorithm is not fast enough, for the Flowers dataset you can send us reconstructions for areas as small as $200x200$ pixels, for the KITTI dataset however try to reconstruct the full image. As explained in the template code, you should work with downsized versions of the datasets.

Good Luck!

References

[1] `http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo`.

[2] `http://vision.middlebury.edu/stereo/data/scenes2014`.