VU Stereo Vision – Exercise 3

1 General Task

The overall goal of the practical part of this course is to implement a local stereo matching algorithm based on the ideas of Hosni et al. [HBR+11]. The practical part is based on one preliminary exercise (i.e., exercise 1) to familiarize yourself with OpenCV and on two actual exercises (i.e., exercises 2 and 3) which are describe here in more detail. In exercise 2, you will implement a simple block-based stereo algorithm. In exercise 3, you will add an adaptive support weight approach to your stereo algorithm and you will apply additional disparity refinement steps. In particular, the following four functions have to be implemented in the exercises 2 and 3 (see Figure 1):

1) computeCostVolume \\ Exercise 2 – Task 2a
   A cost volume is a three-dimensional structure and is derived by computing the color dissimilarity for each pixel at each allowed disparity value (see Figure 1b).

2) aggregateCostVolume \\ Exercise 3 – Task 3a
   Smoothing is applied on two-dimensional (x, y) slices of the cost volume at a fixed disparity value (see Figure 1c).

3) selectDisparity \\ Exercise 2 – Task 2b
   The best disparity value for each pixel is selected from the cost volume, i.e., the disparity with the lowest cost (see Figure 1d).

4) refineDisparity \\ Exercise 3 – Task 3b
   Inconsistent pixels are detected in the disparity maps and these pixels are filled with the disparity of the closest consistent pixel (see Figure 1e).

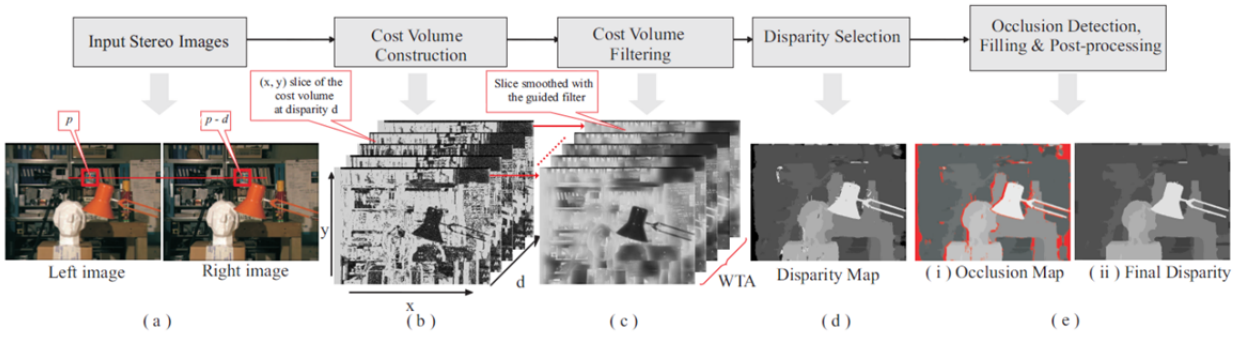

Figure 1: Outline of the algorithm to be implemented. (a) Input stereo images. (b) Cost volume construction. Each *(x, y)* slice holds the dissimilarity costs at a given disparity value *d*. (c) Cost volume filtering. Filtering is applied on each of the *(x, y)* slices. (d) Disparity selection. For each pixel, the disparity of lowest costs in the cost volume is selected. (e) Disparity refinement. Inconsistent pixels which are marked red in the occlusion map are filled with the disparity of the closest consistent pixel in the final disparity map (Figure taken from [HBR+11]).

2 General Instructions

The exercises have to be implemented in C++ using the OpenCV library. In order to set up a C++ project with OpenCV in Visual Studio, a HOWTO is provided in the instructions for exercise 1. Moreover, useful links concerning OpenCV can be found at the section “Useful Links” in this document.

2.1 Recommended Setup

It is recommended to use the following setup:

- Development environment: Visual Studio 2010
- OpenCV: 2.4.10
- Operating system: Windows

Note that for different setups (especially regarding the operation system) either no or only limited support can be provided.

2.2 Submission

The deadline for exercise 3 is **28.06.2015**.

Your submission must include:

- The source code (i.e., the main.cpp file).
- A self-alone executable .exe file which computes and shows the left disparity map. Make sure that (i) you compile the .exe file with the parameters that give the best results and (ii) all necessary files in order to run the .exe file are enclosed.

Your submission may include:

- If you want to earn extra credits, hand in a report on the results which you obtain for the Middlebury evaluation (see Section “Extra Credits” in this document).

All files are to be sent by e-mail in a single .zip attachment to “nezveda@ims.tuwien.ac.at”. The subject of the e-mail is to be “Stereo Vision VU – Group ##: Exercise 3 Submission”. One submission per group is sufficient.

The structure of the submission should look like this:

.zip

- /source
  - main.cpp
- /exe
  - .exe
  - all necessary files in order to run the .exe file
- /report (OPTIONAL)
  - report.pdf

3 **Task 3a:** Filter cost volume with guided filter

Implement a function that takes as input (i) a pair of rectified stereo images “imgLeft” and “imgRight”, (ii) both “costVolumeLeft” and “costVolumeRight”, (iii) and the radius *r* and the regularization parameter *eps* of the guided filter, and outputs both filtered “costVolumeLeft” and “costVolumeRight”.

The guided filter [HST10] is used to filter each *(x, y)* slice of the cost volume *C(p, d)*. The output of the filtering process at pixel *p* and disparity *d* is a weighted average of all pixels in the same slice:

$$C'(p,d) = \sum_q W_{p,q}(I)C(q,d),$$

where *C’(p, d)* denotes the filtered cost value at pixel *p* and disparity *d*, and $W_{p,q}$ denotes the filter weights of the guided filter on guidance image *I*.

Your function should look like this:

```
void aggregateCostVolume(const cv::Mat &imgLeft, const cv::Mat &imgRight, std::vector<cv::Mat>
&costVolumeLeft, std::vector<cv::Mat> &costVolumeRight, int r, double eps)
```

Hint 1: You do not have to implement the guided filter on your own. Instead, go to the link provided at the end of this hint, download the source code of the guided filter and add it to your project. Use the mentioned implementation of the guided filter in order to filter each slice of the cost volume.

Link: https://github.com/atilimcetin/guided-filter

Hint 2: The window size of the guided filter is $2 * r + 1$.

Hint 3: The quality of the disparity map after filtering the cost volume with the guided filter strongly depends on the radius *r*, the regularization parameter *eps* and on the range of the values stored in the cost volume. Thus, you have to find the best settings empirically. However, start with the settings for radius *r* and regularization parameter *eps* as described in [HBR+11].

4 **Task 3b:** Refine disparity map

Implement a function that takes (i) both disparity maps “dispLeft” and “dispRight” and (ii) a factor that was used to scale the disparity values “scaleDispFactor”, and outputs both refined disparity maps “dispLeft” and “dispRight”.

In order to refine the left disparity map, you have to do the following steps. First, apply left-right consistency checking and mark each pixel in the left disparity map as inconsistent, if the disparity value of its matching pixel in the right disparity map differs by a value larger than one pixel. This consistency check typically fails for occluded and mismatched pixels. Next, fill all inconsistent pixels by the disparity of the closest consistent pixel. To this end, record the disparity d_l of the closest consistent pixel to the left of the inconsistent pixel. Record also the disparity d_r of the closest consistent pixel to the right of the inconsistent pixel. The inconsistent pixel is then assigned to the disparity $min(d_l, d_r)$.

Your function should look like this:

```
void refineDisparity(cv::Mat &dispLeft, cv::Mat &dispRight, int scaleDispFactor)
```

Hint 1: You can use also additional refinement steps in order to increase the quality of the disparity map (see lecture slides).

5 Extra Credits

The Middlebury stereo evaluation website[1] constitutes a commonly agreed benchmark for state-of-the-art stereo research. Make yourself familiar with the evaluation website and run your developed stereo algorithm on the four image pairs “tsukuba”, “venus”, “teddy” and “cones”. Submit the computed disparity maps on the evaluation website and report the obtained results. In particular, compare the results you obtain with your block-based approach and your adaptive support weight approach.

6 Questions

Be prepared to discuss the following questions during the exercise interview:

- What is the difference of adaptive support weight approaches compared to standard block-based approaches?

- Why are the authors of [HBR+11] using the guided filter? What is the advantage of the approach of [HBR+11] compared to other adaptive support weight approaches?

7 Useful Links

Note that all these links refer to OpenCV 2.4.10:

- http://docs.opencv.org/2.4.10/modules/refman.html
- http://docs.opencv.org/2.4.10/doc/user_guide/user_guide.html
- http://docs.opencv.org/2.4.10/doc/tutorials/tutorials.html

References

[HBR+11] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz and C. Rother: Real-time local stereo matching using guided image filtering. In *IEEE International Conference on Multimedia and Expo*, pp. 1-6, 2011

[HST10] K. He, J. Sun and X. Tang: Guided image filtering. In *European Conference on Computer Vision*, pp. 1.14, 2010

[1] http://vision.middlebury.edu/stereo