VU Stereo Vision – Exercise 2

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).

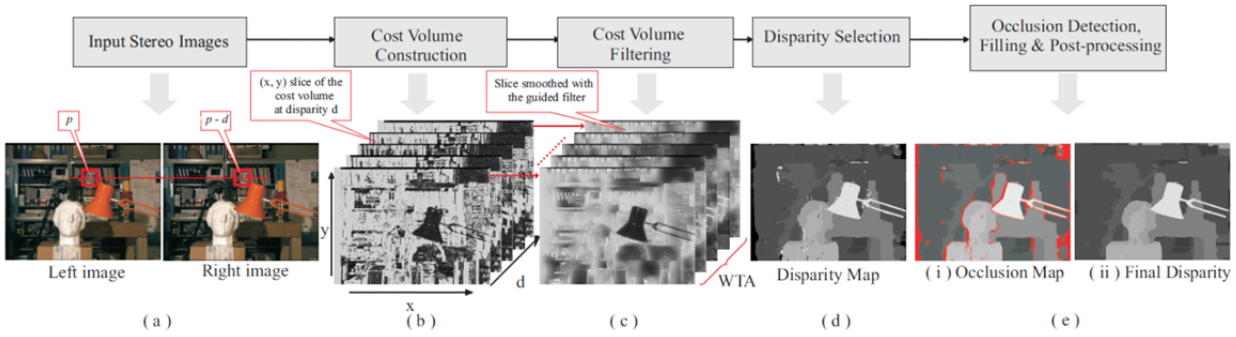

Figure 1: Outline of the algorithm to be implemented. (a) Input stereo images. (b) Cost volume construction. Each *(x, y)* slice holds the dissimilarity costs at a given disparity value *d*. (c) Cost volume filtering. Filtering is applied on each of the *(x, y)* slices. (d) Disparity selection. For each 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 2 is **24.05.2015**.

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 1 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 2a:** Compute cost volume

Implement a function that takes as input (i) a pair of rectified stereo images “imgLeft” and “imgRight”, (ii) a correlation window size “windowSize” and (iii) an upper bound on the disparity “maxDisp”, and outputs both “costVolumeLeft” and “costVolumeRight”.

The cost value *C(p, d)* for pixel *p* at disparity *d* is derived by measuring the dissimilarity in appearance of pixel *p* of the left image and pixel *p – d* of the right image. Use the sum of absolute difference *SAD()* of gray scale images to compute the pixel dissimilarity based on a correlation window N_p:

$$SAD(p,d) = \sum_{q \in Np} \left| I_{left}(q) - I_{right}(q-d) \right|.$$

Your function should look like this:

```
void computeCostVolume(const cv::Mat &imgLeft, const cv::Mat &imgRight, std::vector<cv::Mat>
&costVolumeLeft, std::vector<cv::Mat> &costVolumeRight, int windowSize, int maxDisp)
```

Hint 1: Sample rectified left and right images with corresponding ground truth disparity maps can be found on the TUWEL course page.

Hint 2: The parameter “windowSize” should be an odd value. Start with “windowSize = 5” which gives you a correlation window of 5x5 pixels.

Hint 3: To avoid handling boundary problems, discard correlation windows that would intersect the image boundary.

Hint 4: Start with a value of 15 for the parameter “maxDisp”.

Hint 5: In order to ease the implementation of the further tasks, use a vector to store the whole cost volume where the vectors elements are of type cv::Mat (it is recommended to use a mat type of “CV_32FC1”). In these vectors, the element at position “0” holds the computed cost values for all pixels at disparity “0”, the element at position “1” holds the computed cost values for all pixels at disparity “1”, etc. Note that if you have for example a maximum disparity of “15”, the number of elements in the vector holding the cost volume is “16”, because the allowed disparities range from “0 - 15”. See Figure 1(b) for a schematically illustration of a cost volume.

4 **Task 2b:** Select best disparity match from cost volume

Implement a function that takes as input (i) both "costVolumeLeft" and "costVolumeRight" and (ii) a factor to scale the disparity values, and outputs both disparity maps "dispLeft" and "dispRight".

The winner-takes-all strategy is applied to choose the best disparity value for each pixel *p* (i.e., the disparity value with the lowest cost value *C(p, d)*):

$$d_p = \text{argmin}_{d \in D} C(p, d),$$

where *D* represents the set of all allowed disparities.

Your function should look like this:

```
void selectDisparity(cv::Mat &dispLeft, cv::Mat &dispRight, std::vector<cv::Mat>
&costVolumeLeft, std::vector<cv::Mat> &costVolumeRight, int scaleDispFactor)
```

Hint 1: The parameter "scaleDispFactor" is used in order to scale the disparity values and thus stretch the disparities to a higher range. This is especially useful for the visualization of the disparity maps. Note that "maxDisp * scaleDispFactor" must be below a value of "256" in order to be visualized as a gray scale image properly. Use a "scaleDispFactor" of "16" for a "maxDisp" of "15". Make sure that you change the "scaleDispFactor" properly if you change the "maxDisp".

Hint 2: Look at the slides of the pre-lecture meeting on TUWEL in order to see disparity maps which you can expect to obtain with your implemented stereo algorithm.

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.

6 Questions

Be prepared to discuss the implications of the following modifications:

- Try different values for the parameter "maxDisp". What happens if you increase/decrease this parameter? How can this parameter be determined for a new scene?

- Try different values for the parameter "windowSize". Which block size gives the best results? Is there a correlation between the type of image content and the block size regarding the quality of the disparity maps?

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

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