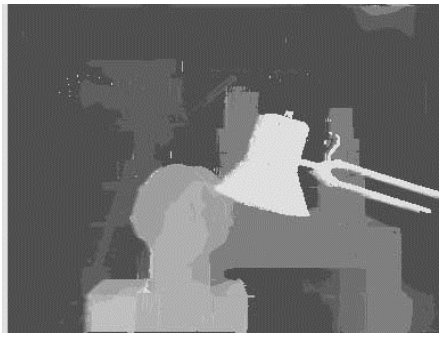
VO Stereo Vision – Exercise 2

1 General Task

The goal of the practical part of this course is the implementation of a local stereo matching algorithm based on the paper "*Locally Adaptive Support-Weight Approach for Visual Correspondence Search*". Although this paper is rather old and many newer algorithms produce far better results, it is still relevant as its idea builds the basis for many other works. There are three exercises while the first one only servers as a short introduction to OpenCV. The following two exercises deal with the more complex subject of implementing a stereo matching algorithm. In exercise 2 a simple window-based algorithm gets implemented which in turn is improved during exercise 3 using a weight function and an additional refinement step.



2 General Instructions

The exercises have to be implemented in C++ using OpenCV. The basic explanations for setting up a C++ project using this library can be found in exercise 1. Although the exercises are designed to be self-explanatory it might aid your understanding of the tasks to read the underlying paper. Additionally useful links can be found at the end of this document.

Not only is each submission mandatory, but also each exercise is built upon the previous ones.

2.1 Recommended Setup

- Development environment: Visual Studio
- OpenCV: 2.4.13
- Operating system: Windows

Note that for different setups (especially regarding the operating system) no support can be provided.

2.2 SUBMISSION

The deadline for exercise 2 is 30.04.2017

The submission has to be a zip file containing:

- Your source code
- A stand-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.

The file must be uploaded via the TUWEL course. Only upload one solution per group.

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 1 – COMPUTE COST VOLUME

The goal of this task is to implement a function taking a pair of rectified stereo images L and R together with a correlation window and an upper bound for the maximal disparity as input parameters to calculate a left and a right cost volume.

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

The cost for a pixel p at a disparity d is calculated through the dissimilarity to its corresponding pixel in the second image. The dissimilarity is the sum of the absolute pixel color differences between the two given correlation windows N_p:

$$c(p,d) = \sum_{q \in N_p} \sum_{c \in \{r,g,b\}} |L_c(q) - R_c(q-d)|$$

knowing disparity just means we know

The resulting cost volume should be a vector containing a cost matrix for each possible disparity, with the first element being the cost volume of disparity 0, the second of disparity 1 and so on.

Test your function with a maximum disparity of 15 and a window size of 5. Before you upload your exercise, try to find the best possible value for the window size.

Mind the datatype of your matrices as it can easily happen that the saved costs exceed its bounds.

4 TASK 2 – SELECT DISPARITY

The goal of this task is to implement a function taking the previously calculated cost volumes to compute a left and a right disparity map as input parameters.

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

This is commonly done by applying a winner takes all strategy, meaning that for each pixel the disparity with lowest cost is selected.

$$d_p = \min_{d \in D} C(p, d)$$

To visualize the results multiply the calculated disparity map with an appropriate scale factor in regards to the maximum disparity (e.g. 16 for max. disparity 15).

A possible left disparity map for the provided images could look like this:

5 OPTIONAL EVALUATION

Optionally try to evaluate the quality of your results. You can do this by calculating the number of wrong pixels. The ground truth for this set of images and many other test data sets are provided by the Middlebury stereo evaluation website. If you are extremely ambitious, you can even try to evaluate your algorithm on the Middlebury website using their framework.

Submit your evaluation results together with an explanation in your report.

USEFUL LINKS

http://ieeexplore.ieee.org/document/1467541/

http://vision.middlebury.edu/stereo/

http://docs.opencv.org/2.4/modules/refman.html

http://docs.opencv.org/2.4/doc/tutorials/tutorials.html

http://docs.opencv.org/2.4/doc/user_guide/user_guide.html