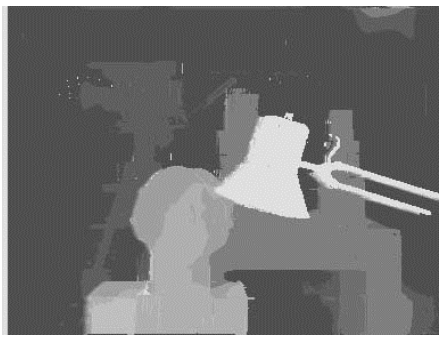
VO Stereo Vision – Exercise 3

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 3 is 11.06.2017

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

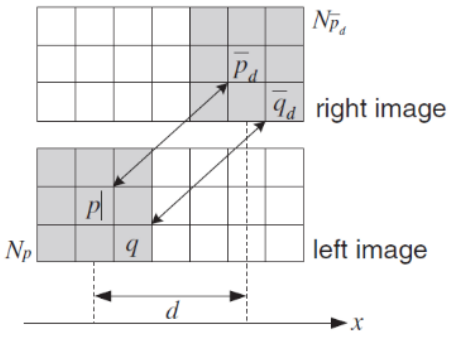
- /source
  - main.cpp
- /exe
  - .exe
  - all necessary files in order to run the .exe file
- /report (OPTIONAL)
  - report.pdf

3 TASK 1 – WEIGHT COMPUTATION

To improve the previously determined disparity map the calculation of the cost volume (`computeCostVolume`) can be enhanced through the consideration of contextual information. This is done by applying a weight function to our cost estimation. The proposed weight function should define how likely it is for two compared pixels to have the same disparity. To construct such a weight function one possibility is to take to contextual clues: similarity and proximity into account based on the assumption that the pixels belonging to the same object have the same color and are close to each other.

$$w(p,q) = \exp\left(-\left(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p}\right)\right)$$

Δc_{pq} … color difference
Δg_{pq} … spatial difference
γ_c, γ_p … constant values (see paper)



The complete cost for a pixel using this function can be calculated by multiplying the absolute differences for the corresponding pixels in the correlation window with their weight for both frames. After that they are summed up and normalized by their total weight.

$$E(p,\bar{p}_d) = \frac{\sum_{q\epsilon N_p,\bar{q}\epsilon N_{\bar{p}_d}} w(p,q)w(\bar{p}_d,\bar{q}_d)e(q,\bar{q}_d)}{\sum_{q\epsilon N_p,\bar{q}\epsilon N_{\bar{p}_d}} w(p,q)w(\bar{p}_d,\bar{q}_d)}$$

4 Task 2 –Refine Disparity Map

To further improve the quality of the disparity map a post processing refinement step can be done. Implement a function taking both disparity maps and the scale factor from exercise 2 as input parameters.

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

To improve the quality of the disparity map check for inconsistent pixels by comparing each pixel from one image with its corresponding pixel in the other image. Mark those pixels as invalid whose corresponding disparity is different by a value larger than one pixel. This errors happen mostly in occluded or mismatched regions. Fill those invalid pixels with the minimum of their closest valid left or right neighbor's disparity.

5 Useful Links

http://opencv-srf.blogspot.co.at/2013/05/installing-configuring-opencv-with-vs.html

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