A STEREO MATCHING DEVELOPMENT PLATFORM WITH GROUND TRUTH EVALUATION AND ALGORITHM TAXONOMY FOR EMBEDDED SYSTEMS

HUMENBERGER, Martin & KUBINGER, Wilfried

Abstract: *This paper presents a part of our work on the development of an embedded stereo vision sensor for mobile robotic platforms. In more detail, it deals with the development and evaluation of stereo matching algorithms. For simulation of matching algorithms we have developed a framework that offers rapid prototyping capabilities. It is PC based and used for algorithm simulation. The resulting disparity maps are evaluated using a pixel wise quality formulation. Furthermore, ideas for a taxonomy with embedded implementation possibilities of algorithms as classification criteria are presented. To reach the maximum processing power, parallel processing on embedded systems is taken under consideration, too.*
Key words: *Stereo Matching, Disparity Map Evaluation, Stereo Vision Test Platform, Algorithm Taxonomy*

1. INTRODUCTION

In the past years, stereo vision has become more and more important in mobile robotic applications. The main benefits are the costs and the physical size of a stereo matching sensor. For us, only a fully embedded solution comes into consideration so the development of a proper algorithm has to deal with this constraint. The following sections briefly describe different steps we go through during development of an embedded stereo matching algorithm. We want to evaluate the most important stereo matching algorithms to find a proper one for our needs. To do this, we first simulate possible candidates on a PC using a proprietary framework. Afterwards, we evaluate the results and try to build a taxonomy by classifying the algorithms with embedded implementation possibilities as criteria.

3. DEVELOPMENT PLATFORM

During our work on stereo vision, the need of an algorithm development and test platform came up and the requirements of this were simple. It should be possible to hook stereo matching algorithms into an existing framework for accelerated development and realizing rapid prototyping. There should be a camera or image input source and an image display and debug output sink. Because the main focus of our research lies on stereo matching algorithms, we dedicated our development platform to this topic and named it *ImProcSim–Image Processing Simulation*.

3.1 Features

The main feature of ImProcSim is the rapid prototyping function for stereo matching algorithms. The framework basically provides all input/output functionality needed for image processing in general. It delivers images as input source, runs the algorithm and displays the results on screen or saves them to memory. This can be done on- or offline, which means that images are captured and processed in a loop or offline loaded from memory and processed afterwards. The workflow is described in sec. 3.2 in more detail. In our case, the image input is a stereo image pair captured from a FireWire stereo head or loaded from memory and a disparity map, represented by an 8 bit image, is the result. An important thing in stereo vision is stereo camera calibration. Calibrated cameras deliver rectified stereo image pairs which ensure epipolar geometry to limit the search for a matching pixel to a single scanline. Stereo calibration and rectification is also implemented in ImProcSim.

Summarizing, ImProcSim offers the possibility of rapid stereo matching development without taking care about image input, camera calibration, parameter adjustment and result visualization. These features of the framework make algorithm implementation and test fundamental easier.

3.2 Workflow

As described above, the framework offers a live mode for continuous stereo matching. The workflow of this mode is given in fig. 1. First, the camera calibration is done offline using the Intel open source computer vision library OpenCV. After that, the processing loop follows. Here, the stereo image pair is captured and rectified at first, the stereo matching algorithm executed afterwards and the results displayed at last.

3.3 User Interface and Results

Fig. 2 shows a screenshot of ImProcSim in offline mode. On the top left, the debug output can be seen. In this example it is the processing time of special parts of the algorithm. On the bottom right, the left source image is located and beside it the resulting disparity map. Our framework can be used to create disparity maps with several already implemented or self developed matching algorithms. The next step in our algorithm development is the evaluation of the resulting disparity maps.

4. RESULT EVALUATION

Once different results have been created, they have to be evaluated to get information about the quality of the calculated data and to appreciate its reliability. A very meaningful possibility to do this is the comparison of resulting disparity maps with their ground truth like presented in (Scharstein et al., 2001). For this, obviously a ground truth reference image has to exist. The authors created their datasets for the tests by illuminating special scenes with structured light (Scharstein & Szeliski, 2003). The introduced datasets are optimized for stereo matching algorithms, that means the images are rare of texture-less and repetitive regions. For our purposes, these are unfortunately not very useful because in natural, real world, indoor scenes such optimal conditions are very rare. This

leaded us to create our own datasets like reported in (Humenberger et al., 2007). This evaluation produces information about how algorithms can deal with certain images and scenes in form of a pixel wise comparison of the disparity maps and the generated ground truth. In our work on stereo vision we actually use this evaluation method to rate the resulting disparity maps from different matching algorithms. After having an idea about the quality, we also want to know more about the embedded implementation possibilities.

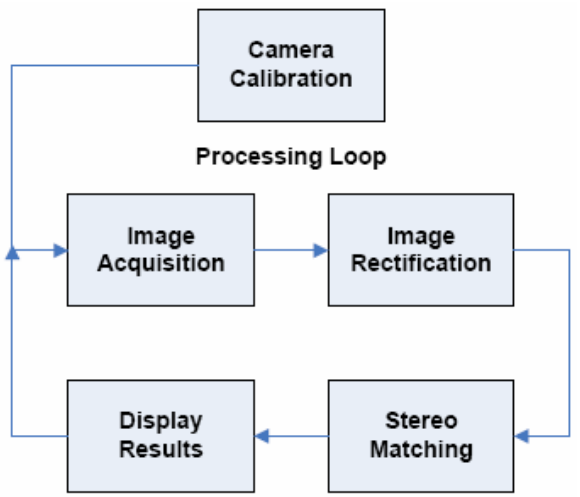

Fig. 1. Workflow of live mode.

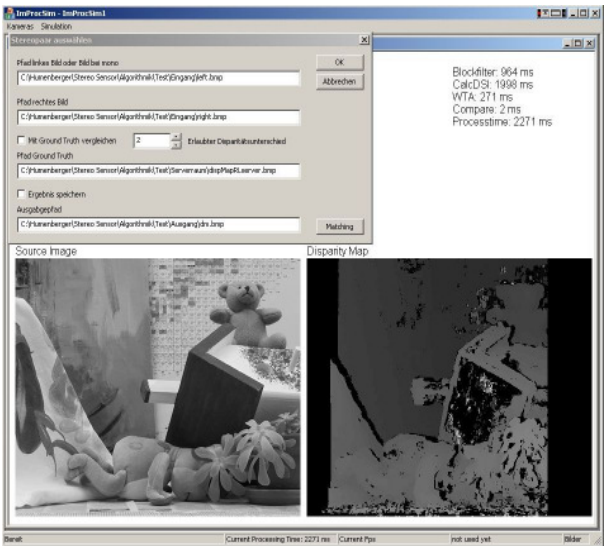

Fig. 2. Screenshot of ImProcSim

5. ALGORITHM TAXONOMY

For development of an embedded stereo sensor for mobile robotic platforms we plan to completely implement a stereo matching algorithm on an embedded system. Up to now, all development is done on a PC without seriously taking care about processing time and memory consumption. This changes here because we now spend our most advertence to real-time capability and embedded implementation needs. Because of the big amount of processing power stereo matching algorithms need, all advantages of embedded systems have to be taken into consideration. The most important one is the possibility of parallel processing. Basically, stereo matching algorithms are very well suited for parallel processing because the calculation of all matching costs (Bleyer, 2006) can be done for each disparity level independently. Because of that, parallel processing is the most important attribute we take care for our algorithm taxonomy. What we try to do is to build an estimation method for rating algorithms on their possibility of parallel processing and embedded implementation based on experiences gained from former implementations of matching algorithms. As a result, we try to build a taxonomy that classifies algorithms with their capability of embedded implementation as criteria, but this is not finished yet. This taxonomy should make it much easier to decide which algorithm could deliver the best results and fastest processing time on an embedded system without fully implementing each one. Of course, after this rough screening a detailed implementation and processing time analyses of the related one follows. We use these two introduced techniques for result evaluation and we develop a stereo matching algorithm which is suitable for embedded systems, fast in execution and delivers a reliable, dense disparity map.

6. CONCLUSION AND OUTLOOK

Up to now, we have implemented several different stereo matching algorithms on PC using ImProcSim, and one on FPGA (Ambrosch et al., 2007). This FPGA implementation was a great success in processing time because the time gain was about 20 times in respect to the PC. Of course, a PC solution is much more flexible but with the possibility of parallel processing the FPGA is a very powerful alternative. Concluding, we have developed a PC based stereo matching development framework called ImProcSim. We use it in our stereo matching research for development of a fully embedded stereo sensor. The quality of the results is evaluated with a pixel wise comparison of the disparity map with its ground truth. On the one hand, we use the common datasets from middlebury and on the other we use our own and more realistic ones. After evaluating the results of the algorithms, we try to classify them in respect to their capability of embedded implementation. If an algorithm delivers quite good and dense disparity maps and seems to be a proper candidate, we choose it for a detailed analysis.

In future, we plan to evaluate the most important stereo matching algorithms to find a proper one for our needs.

7. ACKNOWLEDGEMENTS

This research has been supported by the European Union project ROBOTS@HOME under grant FP6-2006-IST-6-045350.

8. REFERENCES

Ambrosch, K.; Humenberger, M. & Kubinger, W. & Steininger A. (2007). Hardware implementation of an SAD based stereo vision algorithm. *Proceedings of IEEE Computer Vision and Pattern Recognition, Workshop,* ISBN 1-4244-1180-7

Bleyer, M. (2006). Segmentation-based Stereo and Motion with Occlusion, *PhD Thesis*, *Technical University of Vienna*

Humenberger, M.; Hartermann, D. & Kubinger, W. (2007). Evaluation of Stereo Matching Systems for Real World Applications Using Structured Light for Ground Truth Estimation. *Proceedings of IAPR Conference on Machine Vision Applications,* ISBN 978-4-901122-07-8

Scharstein, D.; Szeliski, R. & Zahib, R. (2001). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision,* ISBN 0-7695-1327-1

Scharstein, D. & Szeliski, R. (2003). High-Accuracy Stereo Depth Maps Using Structured Light. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Vol. 1,* ISBN 0-7695-1900-8