Documentation for Computer Vision Challenge 2019

Group 33
Manuel Lengl
Christoph Preisinger
Marcus Schmidt
Felix Oberhansl
Henrique Cabral Meneses de Almeida e Sousa

Prof. Dr.-Ing. Klaus Diepold
M.Sc. Stefan Röhrl

Chair for Data Processing
Technical University Munich
18.07.2019

Abstract

Presentation and discussion of our group's solution to the computer vision challenge summer term 2019. The goal was to compute a Disparity Map and an euclidean movement from a pair of stereo images.

1. Introduction

While computing a Disparity Map is still an active research topic with various flavours regarding exactness, execution time etc. all proposed algorithms rely on the same method to compute disparity of stereo images. They try to match parts (pixels or blocks) of one image to parts of the other and use the relative shift of an object as disparity. In quantifiable metrics, disparity is the inverse depth of an object in the image. The other part of the challenge was to compute the euclidean movement of the camera.
In the following, we will present our solution and discuss it shortly.

2. Matching Algorithm

In [6] the challenge of matching a pair of Stereo Images is subdivided into 4 steps: matching cost computation, cost support aggregation, disparity computation/optimization and disparity refinement. In the following chapter we will present our solution for each of these 4 steps and explain why we decided to go that way.

2.1. Matching Cost Computation

The Matching Cost is introduced as a metric for how likely it is, that a pixel in the left Stereo Image corresponds to a pixel in the right Stereo Image. The most common approaches for cost computation include squared intensity differences (SD) and absolute intensity differences (AD) [6]. Both approaches are sensitive to sampling, since both use absolute intensity values, which depend on the exact environment conditions (e.g. lighting) at the time point of sampling. Examples for methods that are insensitive to sampling are the CENSUS matching [7], where not the absolute intensities are compared but rather if an intensity in a matching block is higher or lower than the intensity at the center of the block, and using linearly interpolated intensity functions surrounding two pixels to measure their dissimilarity [1]. We concluded for both approaches that they don't yield much better results compared to the computational overhead they introduce. Instead we decided to use a fast and computational efficient method of absolute intensity difference calculation. To make the cost computation even more robust and efficient we can limit the search space.

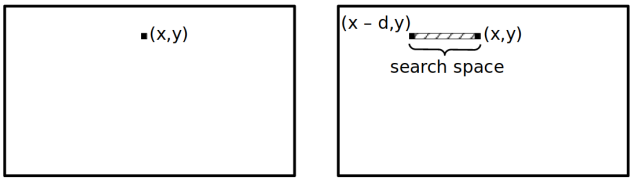

Figure 1: Illustration of search space: The left image functions as reference image. Since the camera was only shifted horizontally and we can assume a maximum disparity d, the marked pixel has to be located in the marked search space in the right image.

We know that between recording the stereo images, the camera was only shifted horizontally, which allows us to introduce a constraint, that matching pixels of one image have to be found at the same height in the other image. Also we limit the search to a maximum horizontal shift (disparity) which depends on the width of the images.

2.2. Cost Support Aggregation

To match pixels of one stereo image to pixels of another in a robust but also efficient manner, a cost aggregation is introduced, which allows us to match blocks instead of single pixels. This has the benefit of additional robustness, since outliers introduced by sampling errors in one image do not have to be matched to one single pixel but are included in one block, with overall low cost, which therefore can be matched correctly. The other major advantage is the reduced computation time, since we do not need to calculate the minimal cost for each pixel but for blocks. For our implementation we decided to use a three dimensional space, with the dimensions $width - height - disparity$, which is proposed in [6]. Over this search space we perform a cost aggregation over a specified window size and for all possible disparity values. In [6] a moving average filter of a structure that makes the computation time of the aggregation process insensitive to window size. For a window size of 3 it looks like this:

$$f = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$WindowCost(x, y, d) = PixelCost(x, y, d) * f \quad (1)$$

This allows a very fast implementation using matrix operations in MATLAB.

2.3. Disparity Computation and Optimization

For the actual computation of disparity, we use a simple local method, where windows of a reference picture are matched to windows with the lowest cost. While this technique is rather easy to implement and computational effective, it has the disadvantage, that multiple windows of the reference picture might be matched to the same window of the other picture. Global or Semi Global Methods [3] would avoid this, by adding constraints to prevent unrealistic matches or enforce smoothness, and minimize this function globally. Since we were already able to limit the search space to a horizontal shift and a maximum disparity, we decided to go with a simple local optimization, which, again, could be implemented very time efficient in MATLAB.

2.4. Disparity Refinement

Disparity refinement is already closely related to post processing, since it tries to optimize the calculated disparity values. Our general post processing approaches are discussed in the next chapter. However, here we want to discuss a special technique, that improves the resolution of Disparity Maps, targeted at block-matched images, that is proposed in [6] as disparity refinement technique. Block Matching can lead to unappealing Disparity Maps, with block quantified sections, that are imprecise at edges and mess with the optical flow. Fitting a curve to the matching cost at discrete disparity levels and using it to smooth the disparities can improve the result. While we used this technique during test phase, we decided against including it in the final application. Due to high resolution pictures and small block sizes we use, the improvements were not significant enough.

3. Pre- and Post-Processing

To optimize the results obtained from the rather simple stereo matching algorithm, we employed different techniques of Pre- and Post-Processing:

3.1. Pre-Processing

- Image Re-sizing: In chapter 2.1 we introduced a narrowed search space along horizontal scan lines. This requires the camera to keep the exact vertical position during taking stereo pictures. As this may not always be the case, we scaled all images down. This has also the benefit of reduced computation time.
- Gaussian Filter: In [1] is proposed to slightly defocus the lens before taking an image. This helps in ensuring that a point in a scene looks identical on each image, which is required for our simple Block Matching. However this does not hold true for reflections (Non-Lambertian Surfaces) and aliasing. Slightly changing the focus can help avoiding errors introduced by these effects. Since we can not influence the exposure itself, we try to achieve the same with a Gaussian Filter.

3.2. Post-Processing

- Hybrid Median Filter: Our simple Block Matching algorithm produces a significant amount of mismatches, due to a numerous of reasons, which can be abstracted as noise. To reduce noise in our Disparity Map, we use a Median Filter. In [4] a Hybrid Median Filter is introduced, which works on a given block size and takes the median over values at specified positions, in a way, that it better preserves edges.

- Image Re-sizing: To get a Disparity Map with the same sizes as the input images, we have to reverse the re-sizing we did during the Pre-Processing steps.

4. Results

The following pictures in figures 2 to 5 show the resulting Disparity Maps for the four given input scenes.

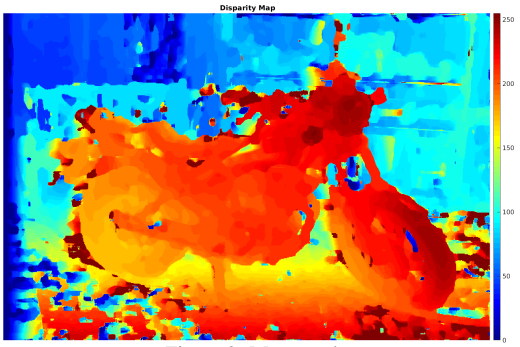

Figure 2: Motorcycle

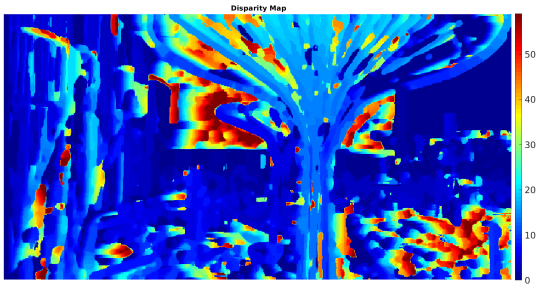

Figure 3: Playground

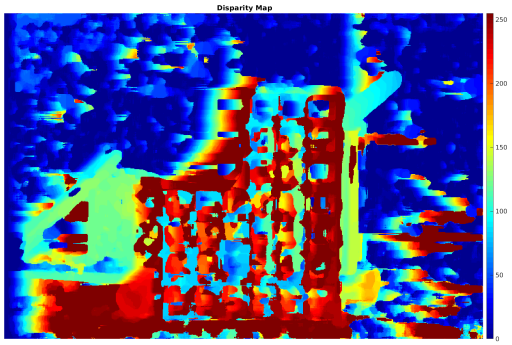

Figure 4: Sword

5. Calculation of the Translation and Rotation matrices

The Translation and Rotation matrices can be obtained by looking for robust correspondences in the two images.

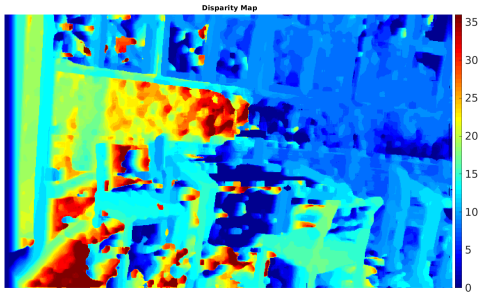

Figure 5: Terrace

To get these correspondences at first Sobel-Filtering is applied in order to get an image with emphasized edges and corners. With the Harris-Detector these features can then be detected and matched in both images. For robuster correspondences, we additionally applied the RanSaC-Algorithm to decrease the probability of mismatches. With at least eight matched points the essential matrix E can then be estimated using the Eight-Point-Algorithm. The SVD of E leads to four solutions for T and R, of which only one is geometrically possible.
In the special case of the Middlebury test set there is no calculation necessary, because the given images are all rectified (i.e. there is no Rotation and Translation only along the x-axis). The obtained Rotation and Rranslation all look like this:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

6. Evaluation

In the following we shortly discuss our solution and compare it to other algorithms.

6.1. Quality of Disparity Map

Evaluating our Disparity Map Calculation on the Middlebury test set, we made the observation, that it works quite well for clearly distinguishable objects in the center of a frame, like the motorcycle image (Figure 2). A sufficient amount of pixels can be matched, such that blocks too will be matched correctly. Our used Pre- and Post-Processing techniques help to improve the result in a way that it can compete with far more complex algorithms. However our algorithm fails to deliver consistent results for untextured regions with similar intensities. For these conditions it is hard to match pixels correctly, using just the intensity value, even for our reduced search space.

Global or Semi Global Methods [3] or matching along Scanlines with a Sequential Matching Cost [1], which are used in advanced image processing libraries (e.g. OpenCV) yield far better results for these regions. This is due to the fact, that these approaches avoid to match one pixel of a reference image to multiple pixels of the other image by introducing smoothness constraints to perform a global minimization.

A simple and quantifiable metric to compare the quality of a Disparity Map to a Ground Truth is the Peak Signal to Noise Ratio, which can be defined with the mean squared error:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (2)$$

$$PSNR = 10 \log_{10}(\frac{MAX_I^2}{MSE}) \quad (3)$$

Where I is the noise-free $m \times n$ Ground Truth, K its noisy approximation, calculated by our algorithm and MAX_I the maximal possible pixel value, in our case 255.

For the provided test images, we computed the following values:

	Motorcycle	Playground	Sword	Terrace
MSE	2561.45	297.04	6932.89	114.07
PSNR	14.05	23.40	9.72	27.56

Table 1: Resulting values for PSNR and MSE

One interesting observation is, that, while the Motorcycle-image looks like the best result, the Terrace-image has the lowest MSE and highest PSNR. This is due to the fact, that even if the overall structure is recognizable, there may still be mistakes in the Disparity Level. In general PSNR as a metric is not fitted to describe the quality of an image as perceived by humans, but it can be used to interpret the information loss for compression/transmission. A specific PSNR value has no absolute meaning, but for image compression the PSNR typically ranges between 20 and 40. That means, that our Disparity Map loses slightly more information, compared to the Ground Truth, than a picture that is compressed using for example MPEG (information threshold $PSNR = 0.5$, but typically $PSNR = 20...40$). [5] A comparison of our results to the results obtained from MATLAB ComputerVision Toolbox was not drawn, because the yielded Disparity Maps by MATLAB require additional processing.

6.2. Computational Effort

Due to the down scaling introduced in chapter 3 and our reduced search space, the execution time of our algorithm scales good over the image size. While the size of the Motorcycle-image (figure 2) has almost 20 times the size of the Terrace-image (figure 5) its computation takes only twice as long (8.8 s for the Motorcycle-image and 4.4 seconds for the Terrace-image). For our implementation we used efficient techniques proposed in [6] like an average box filter.

As far as possible, operations like Pixel Cost Computation, Cost Aggregation and Disparity Computation were implemented using matrix operations, which allow MATLAB to internally parallelize computation.

6.3. Pre- and Post Processing Techniques

To show the effect of our Pre- and Post-Processing methods we provide a Disparity Map calculated without these methods:

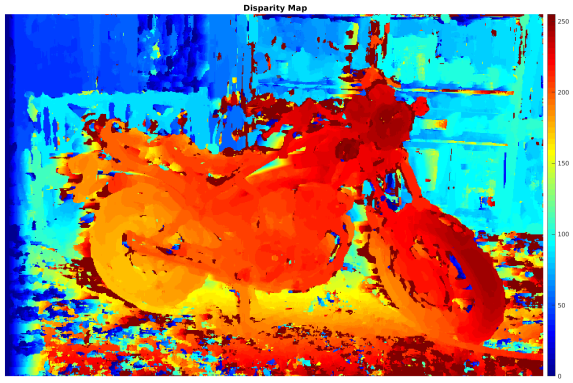

Figure 6: Motorcycle without Pre- and Post-Processing methods

It can be seen, that our used methods allow to optimize a Disparity Map, if a sufficient amount of blocks could be matched for correct disparities. However, inherent errors due to inconsistent matching can't be corrected as can be seen in Chapter 5. More advanced Post-Processing techniques might make that possible. For example in [2] a method is proposed, that tries to find segments in an image. These segments are individually evaluated for outliers. This has the potential to greatly improve our results, since untextured regions like walls and floors, where our matching fails, could be processed as one. But since the computational effort for this method alone is a lot higher than the Disparity Computation itself, we decided to not continue looking into it.

6.4. Calculation of T and R

For the calculation of T and R it is necessary to find at least eight correspondences in the images. This works most of the time, but in the case of the Sword-image the features in the two images are mismatched, which leads to a false

calculation. Due to a similar structure in the Sword-image it is hard to find true correspondences. During implementation we looked at the found robust correspondences obtained from RanSaC and most of them were mismatched. Due to the fact, that calculation of T and R is unnecessary for the Middlebury test set, we decided to not look further into different correspondence matching methods.

7. Conclusion

We recognize, that our chosen path to calculate a Disparity Map may not be ideal for the exact computation, that is required to achieve a high PSNR. It rather has its benefits for a rough but fast depth estimation or even real time computation of Disparity Maps. In retrospective we would have chosen another path, however we learned a lot about time efficient computation and improving results with Pre- and Post-Processing.

References

[1] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. volume 20, pages 401–406, April 1998.

[2] G. da Silva Vieira, F. Soares, G. Teodoro Laureano, R. Parreira, and J. C. Ferreira. Disparity map adjustment: a post-processing technique. 06 2018.

[3] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. volume 30, pages 328–341, Feb 2008.

[4] R. M.R, B. Ajeya, and M. A.R. Hybrid median filter for impulse noiseremoval of an image in image restoration. 2013.

[5] D. Salomon. *Data Compression - The Complete Reference*. Springer Science Business Media, Berlin Heidelberg, 2007.

[6] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, Dec 2001.

[7] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ECCV '94, pages 151–158, Berlin, Heidelberg, 1994. Springer-Verlag.