Proceedings of the IASTED International Conference
Computer Graphics and Imaging (CGIM 2012)
June 18 - 20, 2012 Crete, Greece

DISPARITY CALCULATION: A PROBABILISTIC APPROACH

Diego Montoya, Benjamin Menhorn and Frank Slomka
Institute of Embedded Systems/Real-Time-Systems
Ulm University, Germany
{diego.montoya|benjamin.menhorn|frank.slomka}@uni-ulm.de

ABSTRACT
This paper presents a passive method to calculate the disparity of stereoscopic images. A probabilistic approach is adopted, where two different criteria are used to find a probability density function (PDF) from which the final result can be calculated. The first criterion is a dissimilarity function used to compare the images pixel by pixel. The second is a "sharpness" criterion: Both images are overlapped with an offset in direction of the axis separating the two recording devices. As the objects in both images align better, edges overlap. The energy of the high frequency band increases while the energy in the low frequency band decreases. Due to differences in the perspective, this is only valid locally and therefore only applied to a given neighborhood. The results of both criteria are merged to a final PDF providing the final disparity. The main advantage of our approach is the parallelization ability, and thus a scalable implementation, achieving a compromise between accuracy and ease of realization. Furthermore, with minor adjustments our approach can also be used for software refocusing.

KEY WORDS
stereo vision, disparity, parallel computation

1 Introduction

With a pair of two-dimensional (2D) images from two different perspectives of the same object or scene a stereoscopic illustration can be created. This pair of images is called a stereogram. It is assumed that the images are aligned vertically, and that the images are taken with parallel lines of sight to assure that corresponding pixels lie on the same horizontal line. Therefore the terms first and second image are equivalent to left and right image. The slightly different images of the same scene can not only be used for a stereoscopic illusion but also to create a disparity map. A (stereo) disparity map shows the difference between two views. A depth map can be obtained from the disparity map by an inversion (depth $\propto 1/\text{disparity}$). The depth map is an image with information about the relative distance of the scenes' objects. It is common to represent the distance from the cameras' viewpoint either in a gray or color image. If the size of the object is known, its absolute position from the viewpoint can also be determined. Depth maps are especially useful for navigation. At our institute, a depth map from two cameras will be used together with a sonar to control an autonomous underwater vehicle. Our requirements for processing image data are towards robustness and reliability of the data. But we also require the data to be processed in real-time. All those requirements will be fulfilled by the here presented approach.

Our submarine is equipped with several field-programmable gate arrays (FPGA), which provide the hardware platform. In order to process the camera data in real-time on our platform, we developed the here presented approach. Almost the whole disparity calculation can be efficiently implemented in hardware. Furthermore, the core calculations can run completely in parallel, making FPGAs a perfect target platform.

2 Related Work

A series of algorithms to calculate disparity of stereoscopic footage have been developed [6]. There are search algorithms [15] which usually don't show a straightforward concurrent implementation. Other algorithms rely on spectral properties of the images [10]. Those algorithms lead to strong dependencies on the calculation parameters and image properties. This is a huge disadvantage for the robustness. Iterative approaches [8] are not able to run in real-time on limited resources, such as microprocessors and field-programmable gate arrays. Another approach is an energy estimation as part of the matching criteria [2]. It has its disadvantage with real-time estimations. A proposed fast algorithm uses a normalized squared error and a (fixed) penalty for its local cost function [5]. But this algorithm is limited to images where the corresponding points are normally distributed around a common true value. Especially in underwater scenarios, this requirement isn't fulfilled. A presented real-time approach uses pipelined structures and condensed logic blocks [4]. But their matching criteria differs from the one chosen in this paper. They only consider intensity and not the spectral content of images. Also, their considered neighborhood only consists of horizontal single-row blocks, while our approach uses a circular window which allows a consideration in all directions..

3 Disparity Map Generation Algorithm

3.1 Approach

This paper intends to show an algorithm that is both robust and easy to put in a real-time implementation. Our ap-

DOI: 10.2316/P.2012.779-016

proach combines two known criteria to our new PDF. The first criterion calculates disparity of pixels from the first image and pixels from the second image using Birchfield's dissimilarity function [3]. The second criterion provides an energy ratio measurement (which corresponds to the sharpness) of both images overlapped with a certain offset δ adjusting the 'high to band pass frequency content ratio' from [14]. Our approach uses the intensity space instead of the Fourier space and applies the criterion to a given neighborhood instead of doing it globally. The results from the dissimilarity function and the high to low frequency energy ratio are combined in our probability density function. The maximum of the PDF is our disparity for a certain point in the image. Applying our algorithm to all pixels creates the disparity map for the whole image. The results of our implemented algorithm are shown at the end of the paper.

Our approach has a number of advantages: First of all, the designer can choose a trade-off between performance or saving resources. Secondly, the processes can be parallelized and pipelined as well as different shift values can be calculated by concurrent computation. Both advantages allow a scalability which allows to process image data in real-time. Third of all, our PDF is also a measure for the reliability of the calculated disparity. Finally, our probabilistic approach can also be used in software refocusing.

3.2 Predefinition

The goal is to produce a mapping from the first image to the other image or vice versa. In the following we consider the first image to be the left image and the second image to be the right image. Our approach also works the other way round. This mapping contains information on how many positions a pixel has to be shifted to the left or right to be on the place of its corresponding pixel on the other image. This shift is the disparity d. Finding the corresponding pixel is done using gray-scale intensity values. In the following we consider 8-bit gray-scale pixels thus intensity value from 0 to 255. For color images, the intensity for each channel (red, green, blue) will be considered separately. This implies that calculating the disparity map for color images can be reduced to a disparity map calculation of (three or more) gray scale image. Our approach can also be used for images with all kinds of color or gray-scale depths. As from here, a mapping from the left image to the right image will be assumed.

Definition 1 (Disparity range)**.** *The disparity range $r = [0, \delta_{\max}]$ is the shift range for which the calculations are done. As only stereoscopic images are regarded, pixels from the left image can only find its correspondence by shifting to the left. $\delta_{\max}$ limits the disparity range and is determined by the designer. For objects close to the camera larger shifts are expected.*

Definition 2 (Starting pixel)**.** *A starting pixel has its coordinates at (x_L, y) in the left image, whose corresponding pixel in the right image has to be found.*

Definition 3 (Matching set)**.** *The matching set consists of those pixels from the right image with which the starting pixel is compared to. This set has $\delta_{\max} + 1$ elements with coordinates $((x_L, y), (x_L + 1, y), \ldots, (x_L + \delta_{\max}, y))$.*

Definition 4 (Matching pixel)**.** *A matching pixel has its coordinates at (x_R, y) from the matching set which is compared with the starting pixel. The resulting disparity is $d \in r$.*

3.3 Birchfield's Dissimilarity Function

Birchfield's dissimilarity function [3] compares each pixel from the first image (starting pixel) to one pixel (matching pixel) out of a set of pixels on a corresponding horizontal line from the second image (matching set). This process is repeated for every pixel in the first image. Hence the function works at pixel resolution. The starting pixel is compared to the pixels from the matching set using Birchfield's dissimilarity function [3]. Birchfield's algorithm finds the minimum distance between the intensity value of the starting pixel $I_L^0 = I(x_L, y)$ and the linearly interpolated intensity curve in an interval $[-\frac{1}{2}, \frac{1}{2}]$ around the matching pixel x_R. The main steps for calculating the dissimilarity are explained in the following paragraphs. Detailed information can be found in Birchfield's article [3]. Figure 1 illustrates Birchfield's dissimilarity calculation. The gray curves are the intensity curves from one line of the left image (I_L) and its corresponding line from the right image (I_R). The black dots represent those pixels considered for the calculation.

In order to calculate the dissimilarity, the first step is to interpolate the values:

$$I_R^- \equiv \frac{1}{2}\left(I_R^0 + I_R^{-1}\right) \text{ and } I_R^+ \equiv \frac{1}{2}\left(I_R^0 + I_R^{+1}\right) \quad (1)$$

Within equation 1, $I_R^{-1} = I(x_R - 1, y)$ and $I_R^{+1} = I(x_R + 1, y)$ are those pixels left and right of the matching pixel. The second step computes the minimum intensity value $I_{R_{\min}}$ and the maximum intensity value $I_{R_{\max}}$:

$$I_{R_{\min}} \equiv \min\{I_R^0, I_R^-, I_R^+\} \quad (2)$$

$$I_{R_{\max}} \equiv \max\{I_R^0, I_R^-, I_R^+\} \quad (3)$$

The last step performs a symmetric computation $\bar{D}(I_L^0, I_R^0, I_R^-, I_R^+)$, which leads to the dissimilarity $D(x_L, x_R, y) \in \mathbb{N}_0$:

$$\bar{D} = \max\{0, I_L^0 - I_{R_{\max}}, I_{R_{\min}} - I_L^0\} \quad (4)$$

$$D = \min\{\bar{D}(I_L^0, I_R^0, I_R^-, I_R^+), \bar{D}(I_R^0, I_L^0, I_L^-, I_L^+)\} \quad (5)$$

3.4 High to Low Frequency Energy Ratio

In order to obtain the high to low frequency energy ratio, first the left and right images are overlapped (averaged) with a horizontal offset $\delta \in \mathbb{N}_0$, starting with $\delta = 0$ to $\delta_{\max}$. This composed image is convolved with a Gaussian kernel to obtain its low-pass image LP. Subtracting

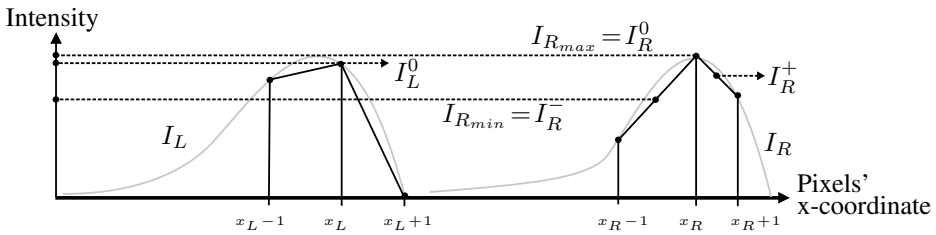


Figure 1. An example of Birchfield's dissimilarity function. In this case, I_L^0 lies between I_R^+ and I_R^0, thus $D = 0$

this low-pass image from the composed image obtains its high-pass image HP. The high to low frequency energy ratio ($HTLR$) for a given pixel at the position (x, y) and a given offset δ is defined as the sum of all squared intensity values in the high frequency band divided by the sum of all squared intensity values in the low frequency band as shown in equation (6).

$$HTLR(x, y, \delta) = \frac{\sum\limits_{x_i, y_i \in \eta} HP^2(x_i, y_i, \delta)}{\sum\limits_{x_i, y_i \in \eta} LP^2(x_i, y_i, \delta)} \qquad (6)$$

This summation is performed for every pixel inside a neighborhood η of the considered pixel, in our case a round window. Our calculation in equation (6) bases on the 'high to band pass frequency content ratio' explained in [14]. The difference is that our calculation is in the intensity space instead of the Fourier space. According to Plancherel's theorem the result is the same and a Fourier transformation is avoided [9]. The theorem, which was proven by Michel Plancherel, states that the Fourier transformation within L^2 of a quadratically integrable function is an isometry. In other words, a function and its Fourier transformation have the same norm in L^2.

Figure 3 shows how the $HTLR$ varies within the neighborhood of the starting pixel from figure 2 depending on the offset δ. The first column shows the overlapped images with a offset of δ. The second column shows the corresponding low-pass image and the third column the corresponding high-pass image. The high-pass image is the difference of the composed image and its low-pass image. In the top row, the right image is overlapped with a shift $\delta = 3$ pixels to the left image. Objects are not aligned and the sharpness $HTLR$ equals $4.5 \cdot 10^{-6}$. In the middle row the shift is $\delta = 12$ pixels. The portion of the head is aligned and a higher value is obtained, $HTLR = 2.6 \cdot 10^{-2}$. The bottom row uses a shift of $\delta = 16$. The $HTLR$ is again lower, as most of the edges don't match. The maximum is reached for a shift of $\delta = 12$. This is the correct shift and therefore the disparity for the starting pixel.

3.5 Objective Function

The lower the dissimilarity D the more likely the chosen pixels correspond to the same point in both images. This is, however, susceptible to noise and also won't work well

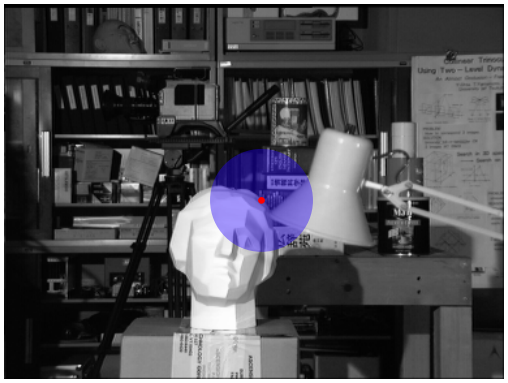


Figure 2. Original left image from [7] with the starting pixel (red dot) and its neighborhood (blue circle) with 75 pixels of diameter.

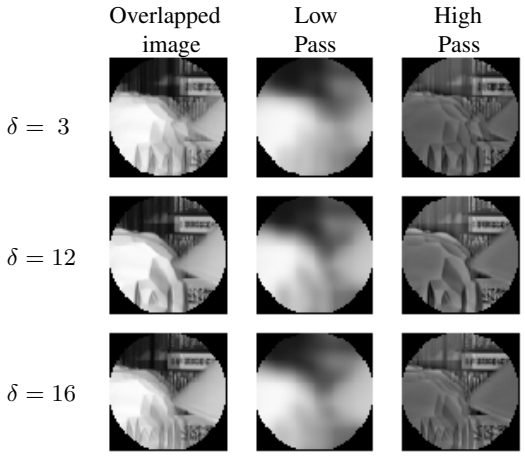


Figure 3. Comparison of the $HTLR$ with different offsets δ.

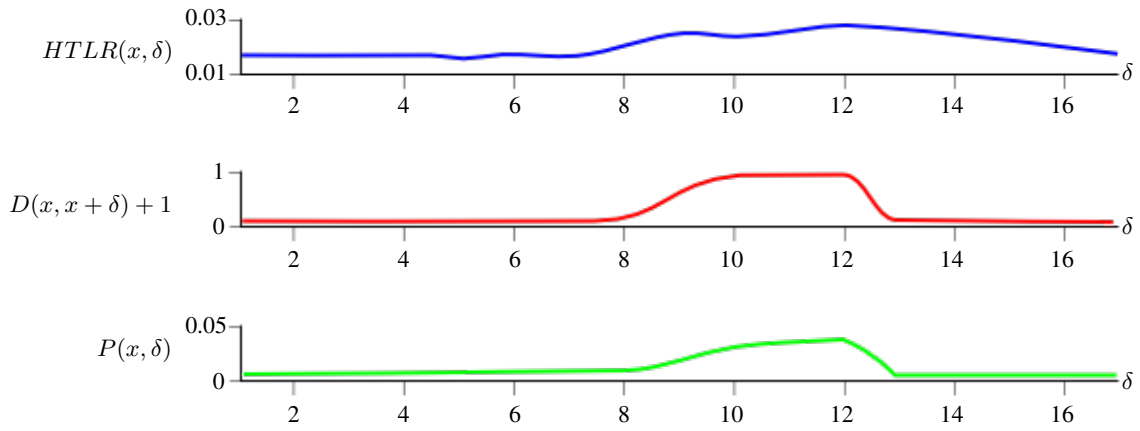


Figure 4. $HTLR$, dissimilarity D and disparity d for the starting pixel from figure 2 within a offset $\delta \in [1, 17]$.

either with plain surfaces or with patterns with frequencies higher than the inverse of the disparity range. On the other hand, $HTLR$ is very stable, but will fail when most of the window is occupied with objects not in the plane of the central pixels and gives wrong results at object edges. A formula that can fuse the two former criteria is needed using them as probability density functions $P(x, y, \delta)$. It is assumed, that both criteria are independent.

$$P(x, y, \delta) = \frac{HTLR\,(x, y, \delta)}{D\,(x, x + \delta, y) + 1} \tag{7}$$

$P(x, y, \delta)$ gives the probability of a starting pixel in the coordinates (x, y) in the left picture matching a pixel in the coordinates $(x+\delta, y)$ in the correspondent picture. The probability $P(x, y, \delta)$ is directly proportional to $HTLR$ and inversely proportional to the dissimilarity D between the two pixels, as given in equation (7). $HTLR$ is in an interval $(0, \infty)$ and $D \in \mathbb{N}_0 \rightarrow D \geq 0$. In order to avoid a zero value in the denominator $+1$ is added. Thus $D+1 \geq 1$ and the not normalized probability P in an interval $(0, \infty)$. If required, $P(x, y, \delta)$ can be normalized by equation (8).

$$P_n(x, y, \delta) = \frac{P\,(x, y, \delta)}{\sum_{\Delta=0}^{\delta_{\max}} P\,(x, y, \Delta)} \tag{8}$$

The $+1$ in equation (7) avoids the singularity, but also distorts the relation between $HTLR$ and D. Because only the maximum of the probability P is relevant for the disparity d, and both $\frac{1}{x}$ and $\frac{1}{x+1}$ are both monotonically decreasing functions, the disorientation has no negative effect. Therefore, the probability P from equation (7) does not need to be normalized according to equation (8). Finally, our disparity $d(x, y)$ is given by equation (9).

$$d(x, y) = \underset{\delta \in r}{\operatorname{argmax}}\{P(x, y, \delta)\} \tag{9}$$

Figure 4 shows the values for $HTLR(x, \delta)$, $D\,(x, x + \delta) + 1$ and the resulting probability $P(x, \delta)$ for the starting pixel (red dot) from figure 2 within a shift (offset) of $\delta \in [1, 17]$, The maximum and therefore according to equation (9) the disparity $d(x)$ is found at an offset of $\delta = 12$ with a non normalized probability P of 0.026.

In order to generate the disparity map for a color scene, the single results from each color are averaged. Pixels with a probability below a predefined threshold are not considered for averaging.

3.6 Reliability

Equation (7) not only expresses the probability but can also be used as a measure for the reliability of the calculated disparity. A threshold can be predefined and if there is not enough information in this part of the picture to calculate a reliable disparity value, the non-normalized probability will be lower than the threshold. In this case there will be no disparity value assigned to those pixels as they are considered unreliable. If the number of pixels with no value assigned exceeds a predefined limit, the whole disparity map is unreliable. In our case the map will then not be used for underwater navigation of our submarine.

3.7 Real-Time Implementation

With the here presented approach a real-time implementation is realized by parallelizing and pipelining the processes. For our FPGA target platform, most of the blocks can be realized in hardware. Figure 5 shows the proposed architecture. From the left and right images, regions are extracted and overlapped and stored in window buffers. The low and high-pass filter, implemented as finite impulse response filters, as well as the dissimilarity calculation can run in parallel. The Gaussian kernel is set to the same size as the window w and a standard deviation of $0.05 \cdot w$. For each offset value δ these tasks can be further parallelized, illustrated by the shaded blocks in the background. Because our filters are faster than the dissimilarity calculation, the additional block for the $HTLR$ calculation

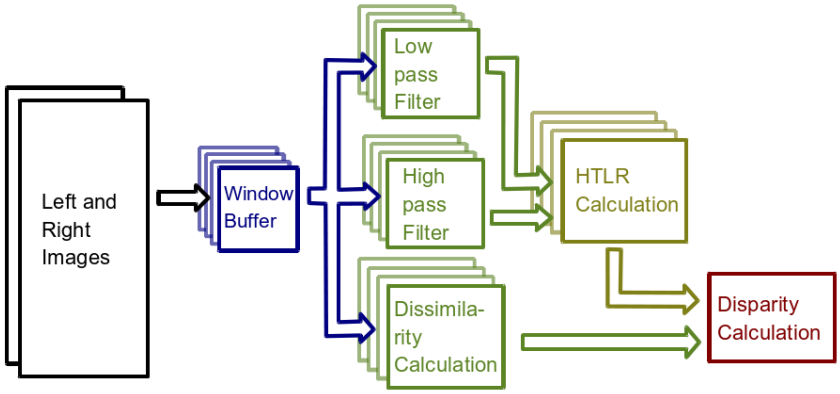


Figure 5. Data-flow model for real-time implementation.

doesn't slow down the whole process. The last block provides the disparity calculation. Post-processing elements can be added at the end of the pipeline. In our case, a median filter of 7×7 pixels, which can be also realized as a parallel functional block. The whole task can be further accelerated by adding concurrent computation for the different shift values. It is up to the designer what is more important, performance or saving resources.

3.8 Software Refocusing

The same probabilistic approach can also be used for software refocusing. A series of photographs of the same scene are taken from exactly the same perspective. The only variable is the depth of field. Starting at the closest point to the camera, it is progressively shifted back for every consecutive picture. To apply our method, the matching criteria and the objective function have to be adapted. Pixel to pixel comparison makes no sense, as the images already correspond and their pixels are aligned. Therefore the dissimilarity function is not needed. To adjust the $HTLR$ function the images won't be shifted and overlaid. In order to efficiently estimate the relation between low and high frequency content the square of the magnitude of the gradient is used. Leading to equation (10) for the high to low frequency energy ratio $HTLR(x, y, k)$ where k is the k-th image because there is no δ shift.

$$HTLR = \left(\frac{\partial I(x,y,k)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y,k)}{\partial y}\right)^2 \quad (10)$$

$$P(x, y, k) = HTLR(x, y, k) \quad (11)$$

$$d(x, y) = \underset{k \in r}{\operatorname{argmax}}\{P(x, y, k)\} \quad (12)$$

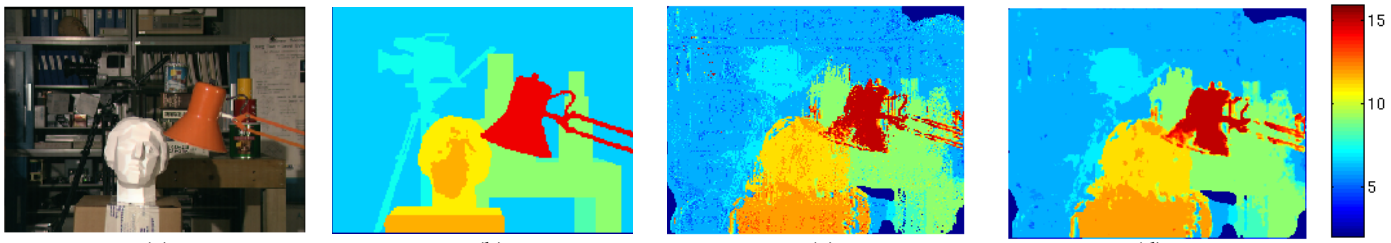
Equation (11) shows the adjusted objective function and equation (12) the disparity equation. In this case r is not the disparity range, but the indexes of the progressive depth of field pictures. The result is not disparity *per se*, but as the disparity does, it carries information about the depth of the scene. A real-time calculation of the gradient calculation is of course possible.

4 Results

In order to demonstrate our approach, the whole algorithm including a median filter was implemented in Matlab. The Matlab implementation allows to easily generate VHDL-Code which can be programmed on out FPGA target platform. Figures 6, 7 and 8 show some obtained results. The sub-image (a) always shows the original left image. Because of the almost invisible differences in the right scene, only the left image is shown here. The right image can be found at the cited sources. Sub-image (b) shows the ground truth, also obtained from the cited sources. (c) is our calculated disparity map. For the colored scenes of figure 6 and 7 the three color channels (red, green and blue) were regarded as independent gray-scale images, and the disparity map was computed averaging the single previous results. Pixels with a probability below the threshold (dark blue pixels) are not considered for averaging. The sub-image (d) is the result of applying a median filter of 7×7 pixels. Figure 9 shows the result for our implementation of software refocusing. The first sub-image is the first image of the scene. The second sub-image is our obtained disparity map and the most right sub-image shows the disparity map after applying the median filter.

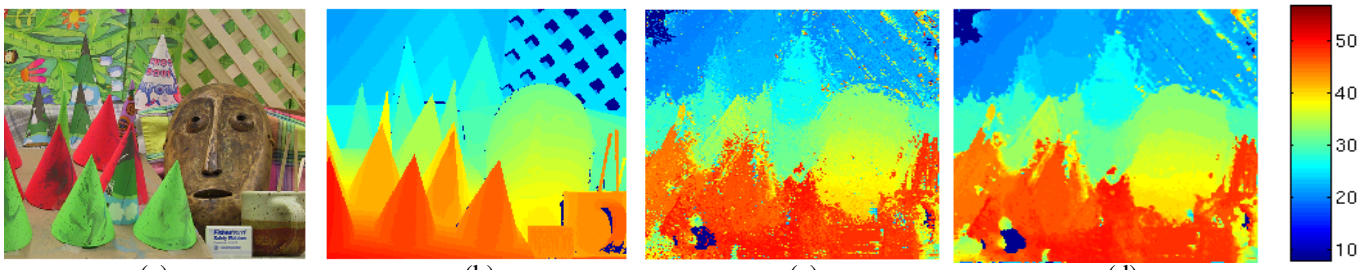
We used Matlab 2010a, Version 7.20.0.499, 32-bit under Windows XP 2002 with SP3 with an AMD Phenom II X4 955 3.21 GHz processor and of 3.50 GB RAM. The following computations times were measured for figure 7. A non parallelized and non pipelined implementation takes 4 seconds per channel. Thus the generation of the disparity map need 12 seconds.

To estimate the computation time of a parallel and pipelined implementation, we measured the time for each individual block from figure 5. In order to load the 225x225



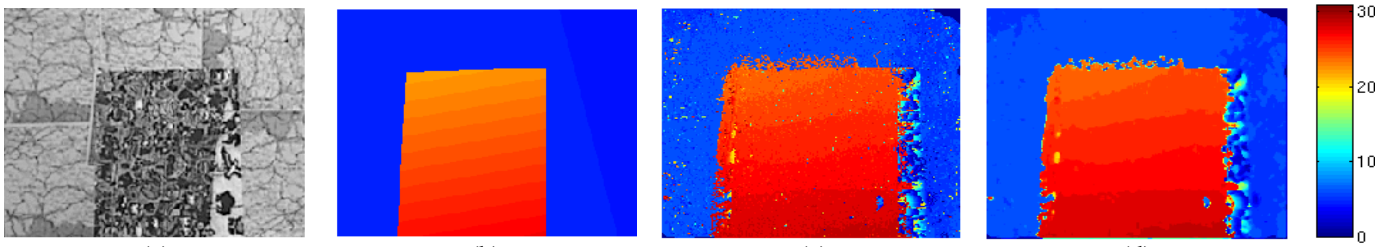
(a) (b) (c) (d)

Figure 6. Image set from the University of Tsukuba [7]. Window size is 75 pixels of diameter, disparity range is 16 pixels and the threshold is set to $3.4 \cdot 10^5$.



(a) (b) (c) (d)

Figure 7. Image set also from the University of Tsukuba [13]. Window size is 75 pixels of diameter, disparity range is 50 pixels and the threshold is set to $1.7 \cdot 10^5$.



(a) (b) (c) (d)

Figure 8. Image set from Microsoft Research [12]. Window size is 50 pixels of diameter, disparity range is 30 pixels and the threshold is set to $3.4 \cdot 10^5$.

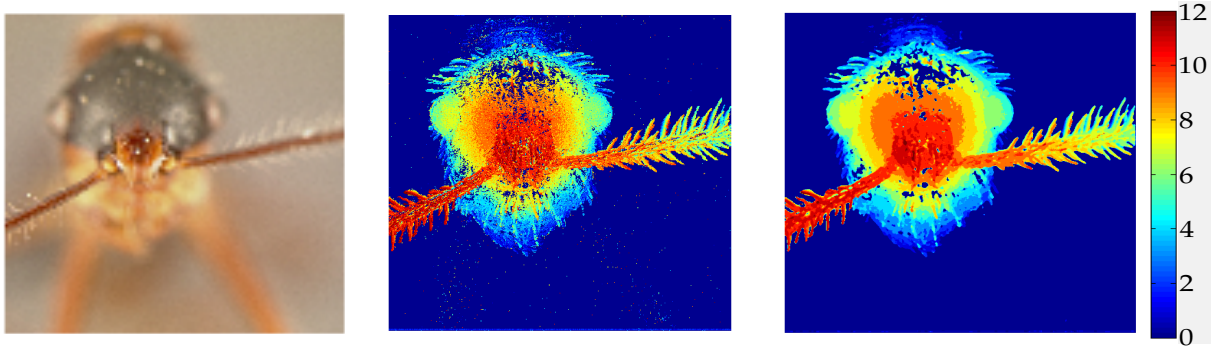


Figure 9. Image set from the University of Washington [1]. The threshold is set to 0.5.

pixel images into the windows buffer, it took 285 μs. With a pipelined windows buffer, the loading time was negligible. The low and high pass filters took $38,7\ ms$ without pipelining and $0,138\ \mu s$ with pipelining within the filters. The calculation of the disparity took $0.5\ \mu s$ and the $HTLR$ calculation $0.03\ \mu s$. The disparity calculation took $0,87\ \mu s$. With a completely parallelized and pipelined implementation, the whole calculation would take as little as $1.538\mu s$ per pixel in hardware. With a pipeline like the one in figure 5 only the first pixel takes $1.538\mu s$, the rest would only need as much as the slowest step ($0.87\mu s$) allowing a 15-fps display for a gray scale $320x240$ image. A FPGA implementation is expected to be a lot faster than Matlab, thus allowing higher frame rates and RGB image processing capabilities.

We compared our disparity results from figure 7 with the benchmark tool Middlebury Stereo Evaluation [11]. Our result has an error of 18.9% in the non-occluded regions, 20.8% errors in the whole image and 48% error in the discontinuities. These results are more than satisfactory for our purpose, especially the run-time for our calculation. Our focus was towards a fully parallelizable architecture, which was achieved.

5 Conclusion

This paper introduced a passive method to calculate the disparity of stereoscopic footage. The probability density function adopted Birchfield's dissimilarity function. The sharpness criterion was applied to the overlapping images on a given neighborhood. Our approach is a stable and deterministic solution to the disparity calculation problem. Our probability measurement also serves as a reliability measurement. Depending on the application, this measurement can be adjusted by a threshold to get only usable results. As shown, our method allows parallelizing and pipelining. High frame rates can be obtained using graphics cards or FPGAs for a parallel implementation. This allows to obtain results in real-time. Furthermore, with slight adjustments our approach can also be used for software refocusing. For future work, the probability density function can be extended to include calculations from other methods to improve results. If done adequately, the pipeline can be extended and allows for real-time results.

References

[1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23:294–302, Aug. 2004.

[2] L. Alvarez, R. Deriche, J. Sánchez, and J. Weickert. Dense disparity map estimation respecting image discontinuities: A pde and scale-space based approach. *Journal of Visual Communication and Image Representation*, 13(1-2):3 – 21, 2002.

[3] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:401–406, 1998.

[4] G. Calin and V. Roda. Real-time disparity map extraction in a dual head stereo vision system. *Latin American Applied Research*, 1:21 ff., 2007.

[5] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63:542–567, May 1996.

[6] U. R. Dhond and K. K. Aggarwal. Structure from stereo: A review. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation (Vol. 1)*, pages 25–46. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[7] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1:103, 2001. Supplemental material.

[8] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, Oct. 1976.

[9] M. Plancherel and M. Leffler. Contribution à l'étude de la représentation d'une fonction arbitraire par des intégrales définies. *Rendiconti del Circolo Matematico di Palermo (1884 - 1940)*, 30:289–335, 1910.

[10] B. Porr, A. Cozzi, and F. Wörgötter. How to "hear" visual disparities: Real-time stereoscopic spatial depth analysis using temporal resonance, 1999.

[11] D. Scharstein and A. Blasiak. Middlebury stereo evaluation - version 2. http://vision.middlebury.edu/stereo/eval/.

[12] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2001.

[13] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1:195, 2003. Suppliment datasets with ground truth.

[14] D. Shaked and I. Tastl. Sharpness measure: towards automatic image enhancement. In *ICIP (1)'05*, pages 937–940, 2005.

[15] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. *Int. J. Comput. Vision*, 47:275–285, April 2002.