International Journal of Computer Vision 47(1/2/3), 229–246, 2002

Real-Time Correlation-Based Stereo Vision with Reduced Border Errors

HEIKO HIRSCHMÜLLER, PETER R. INNOCENT AND JON GARIBALDI
Centre for Computational Intelligence, De Montfort University, Leicester, LE1 9BH, UK
hhm@dmu.ac.uk
pri@dmu.ac.uk
jong@dmu.ac.uk

Abstract. This paper describes a real-time stereo vision system that is required to support high-level object based tasks in a tele-operated environment. Stereo vision is computationally expensive, due to having to find corresponding pixels. Correlation is a fast, standard way to solve the correspondence problem. This paper analyses the behaviour of correlation based stereo to find ways to improve its quality while maintaining its real-time suitability. Three methods are suggested. Two of them aim to improve the disparity image especially at depth discontinuities, while one targets the identification of possible errors in general. Results are given on real stereo images with ground truth. A comparison with five standard correlation methods is provided. All proposed algorithms are described in detail and performance issues and optimisation are discussed. Finally, performance results of individual parts of the stereo algorithm are shown, including rectification, filtering and correlation using all proposed methods. The implemented system shows that errors of simple stereo correlation, especially in object border regions, can be reduced in real-time using non-specialised computer hardware.

Keywords: stereo vision, real-time, correlation problems, multiple correlation windows, object border correction

1. Introduction

1.1. Real Time Stereo Vision

Stereo vision systems determine depth (i.e. distance to real world objects) from two or more images which are taken at the same time from slightly different viewpoints. The most important and time consuming task for a stereo vision system is the registration of both images, i.e. the identification of corresponding pixels. Two pixels are corresponding when they represent the same point in the real world. Area-based stereo attempts to determine the correspondence for every pixel, which results in a dense depth map. Correlation is the basic method used to find corresponding pixels. Several real time systems have been developed using correlation-based stereo (Konolige, 1997; Matthies et al., 1995; Volpe et al., 1996). However, correlation assumes that the depth is equal for all pixels of a correlation window. This assumption is violated at depth discontinuities. The result is that object borders are blurred and small details or objects are removed, depending on the size of the correlation window. Small correlation windows reduce the problem, but increases the influence of noise, which leads to a decrease of correct matches (Kanade and Okutomi, 1994).

1.2. Objectives and Constraints

This research is concerned with the development of a real-time stereo vision system for a tele-operated mobile robot. The system must be suitable for the detection, recognition and tracking of objects and their relative positions, to support high-level object based tasks in the local environment of a tele-operated mobile robot. Furthermore, non-specialised cameras and computer hardware should be employed, because the robot will be used in harsh environments and might be damaged or even destroyed.

Considering these requirements, it has been aimed for a system that is fast (i.e. 5–10 frames/s) but also accurate enough to discriminate objects in the local working environment (i.e. up to several meters) so that they can reliably be detected and recognised.

Correlation-based stereo vision fulfils all given requirements. However, it has in general problems at depth discontinuities. It is assumed that the location of object borders (i.e. depth discontinuities) is important to retrieve proper object shapes for segmentation and recognition purposes. This paper is concerned with the aspect of improving correlation-based stereo, by reducing errors especially near object borders while maintaining its real-time suitability.

As a general rule, it is assumed that it is better to invalidate uncertain matches in order to reduce errors as long as correct matches are not rejected radically.

1.3. Existing Methods

The outcome of correlation is influenced by several parameters. Firstly, the correlation measure determines how the similarity between two areas is determined. Most common is the use of Cross Correlation or the Sum of Absolute or Squared Differences. Zabih and Woodfill introduced the non-parametric Rank and Census measures (Zabih and Woodfill, 1994). These measures rely on the numerical ordering of intensities and not on their values. This makes them less affected by noise and outliers. Results show slight improvements over standard correlation methods. Section 4 uses these five methods as a base for comparison.

Another correlation parameter is the shape of areas, which are correlated. Usually, rectangular windows are used for the sake of computational performance. The size of the correlation window determines the amount of pixels used for correlation. The effect of noise is reduced by increasing the number of pixels and thus the size of the correlation window. However, bigger correlation windows are more likely to cover areas where depth varies. A change in depth results in a change of disparity so that only parts of the windows correspond to each other. This leads to errors at object boundaries. Kanade and Okutomi address this problem by changing the size and shape of rectangular correlation windows, according to local disparity characteristics (Kanade and Okutomi, 1994). This adaptive correlation window approach shows a decrease in errors at object boundaries. However, the algorithm is too slow for real time usage on non-specialised hardware according to results reported by Boykov et al. (1998).

There are computationally efficient multiple window methods, which can be seen as simplifications of the adaptive window approach. A common configuration is the use of 9 rectangular correlation windows that have the same size, but different positions for the point of interest (i.e. in every corner, in the middle of every side and in the middle, as usual). Fusiello et al. and Little report for example about this configuration (Little, 1992; Fusiello et al., 1997). Correlation is done with all 9 windows for every pixel and every disparity, but only the result of the best window is used. This method offers an improved behaviour at depth discontinuities compared to standard correlation and is suitable for real time (comparisons are shown in Section 4).

Boykov et al. (1998) presented a variable window approach, which gives good results at depth discontinuities. The method chooses an arbitrarily shaped window that varies for every pixel. The algorithm seems to be suitable for a real time implementation. However, the method suffers from a systematic error as identified by its authors. It increases the size of objects in some cases by including nearby low texture areas.

Energy optimisation methods like the Maximum Likelihood stereo algorithm (MLMHV) from Cox et al. (1996) perform generally much better (Szeliski and Zabih, 1999). However, the method suffers considerably from horizontal streaking, resulting from a one dimensional optimisation individually along each scan-line rather than a much more time consuming two dimensional optimisation. Dynamic programming has been used to improve the speed. Nevertheless, the algorithm seems to be too slow for a real time implementation.

The cooperative stereo algorithm from Zitnick and Kanade demonstrates that the amount of errors can be reduced dramatically, if execution time is not crucial (Zitnick and Kanade, 1999, 2000). The method results in up to 98% correct matches on the stereo images from the University of Tsukuba. However, the iterative method is far too slow for any current real time application.

This overview can only cover a few methods out of a vast amount published in the stereo vision literature. Nevertheless, several important methods were shown, which address the weak behaviour of stereo correlation at object borders by using different approaches.

1.4. A New Proposal

Simple correlation exhibits a systematic error, i.e. blurring of object borders. However, the assumed location of a computed depth discontinuity is still near (i.e. within the size of the correlation window) to the location of the real depth discontinuity, as long as the object is bigger than the size of the correlation window. Objects, which are in their width or height smaller than the correlation window, might just vanish. Furthermore, correlation has proven to be fast enough for a real time implementation and has a regular structure with fixed execution time, which is independent of the scene contents.

This paper proposes three novel improvements to tackle specific problems of correlation.

1. A multiple window approach that decreases errors at object borders.
2. A correlation function error filter that invalidates uncertain matches and reduces general errors.
3. A border correction method that improves object borders in a post-processing step further.

All these improvements are still suitable for real time applications and can reduce errors by 50% on the used test images.

Section 2 analyses the problems of correlation. Sections 3.1, 3.2 and 3.3 show attempts to tackle typical correlation problems. Section 3.4 gives an overview of the whole algorithm. All methods were analysed for their quality and compared to standard correlation methods, using real stereo images with ground truth (Section 4). Finally, Section 5 discusses an optimised implementation of the whole algorithm and gives detailed results on its performance.

2. Problems of Stereo Correlation

2.1. General Behaviour

Correlation works by using a usually fixed, rectangular window around the pixel of interest in the first image. The window is correlated with a second window, which is moved over all possible positions in the second image. The possible positions are defined by the minimal allowed distance between the camera and an object, which gives the maximum disparity. The position where correlation has the highest value determines the pixel in the second image that corresponds to the pixel of interest. Bigger correlation windows increase the reliability by averaging over a bigger area, thus reducing the effect of noise.

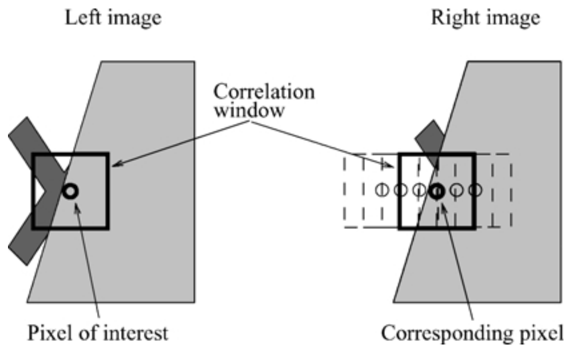

Figure 1. Correlation at object border.

However, if the correlation window overlaps a depth discontinuity, then a part of the window will affect the result arbitrarily. Figure 1 shows a situation where the left part of the window contains the background, which is different in both windows. Consequently, this part of the window introduces an error in the calculation. Furthermore, it should be noted that the left part of the correlation window in the left image is at least partly occluded in the right image. The size of the occluded part depends on the disparity difference and the size of the correlation window.

The use of a smaller correlation window reduces the problem, because a smaller window does not overlap the depth discontinuity to the same extent. Generally, the choice of the correlation window size is a trade off between increasing reliability in areas with constant depth and decreasing errors in areas where depth changes.

2.2. Behaviour at Depth Discontinuities

Whether the introduced error at a depth discontinuity can be neglected or not depends on the similarity between the object and the occluded and visible part of the background, which is covered by the correlation window. Figure 2 shows a situation where the pixel of interest is just outside the object.

The correct corresponding position for the correlation window R would be L. It is necessary to split the correlation window into two halves to understand why sometimes the correlation of R with $\tilde{L}$ gives a higher response than R with L. This results effectively in an extension of the object at its left border. Let $c(a, b)$ be the correlation value of the area a and b, where a

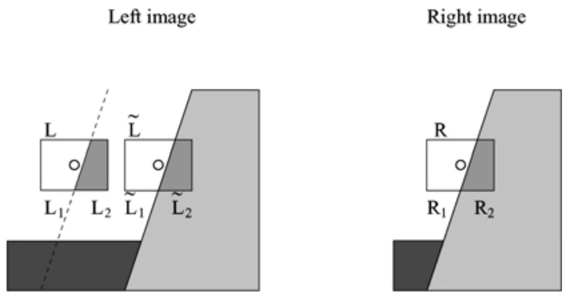

Figure 2. Typical decision conflict at object border.

low value corresponds to a high similarity. The values $c(R_1, L_1)$ and $c(R_2, \tilde{L}_2)$ should be very low, because the corresponding regions are correctly matched. The choice between the position L and $\tilde{L}$ depends on the amount of similarity of R_2 and L_2 and the similarity of R_1 and $\tilde{L}_1$. The areas L_2 and $\tilde{L}_1$ are occluded in the right image. If $c(R_2, L_2)$ is higher than $c(R_1, \tilde{L}_1)$, then the wrong position $\tilde{L}$ will be chosen. The area R_1 is bigger in this example and has a higher effect in the correlation process. However, a small amount of large errors can have a higher effect than a large amount of small errors, depending on the correlation measure. Image noise will affect the choice, but it depends mostly on the similarity between the occluded background, visible background and object.

Usually, the background continues *similarly* and L_1 would be similar to $\tilde{L}_1$, and L_2 dissimilar to $\tilde{L}_2$. This leads to the presumption that objects usually appear bigger. However, shadows or changing texture near object borders can inverse the situation, so that the object would become smaller. The same scenario can be drawn for right borders and leads to fuzzy, blurred object borders.

The situation is slightly different for top or bottom borders of objects, because there is no occluded area in the standard case of rectified images. In this case, epipolar lines correspond to image rows and the object and background shift horizontally against each other. Whether a match is correct depends on similarities between the horizontally shifted background areas and horizontally shifted object areas as well as the influence of noise. The blurring effect is expected to be less severe than at left and right object borders, because the similarity between the background areas is usually high as well as the similarity of object areas. Furthermore, there is no occluded area as found at left or right object borders. Consequently, there is no inherent asymmetry as with vertical object borders. Thus, the matching process is only influenced by image noise.

A single depth discontinuity that crosses the correlation window is only a special case. Generally the depth could change for every pixel in the window. However, it is assumed that the depth varies usually smoothly at most places within real images, except at object borders (Marr and Poggio, 1979). Thus, the case above is an important special case. Slanted surfaces were not especially considered here. However, the theory could be adapted to incorporate several small depth discontinuities within a correlation window as well.

2.3. *Experimental Confirmation*

The predicted behaviour of correlation at object borders can be verified using the stereo image set with ground truth from the University of Tsukuba (see Fig. 3). The disparity image has been calculated by filtering both source images with the Laplacian of Gaussian (LoG) with a standard deviation of 1.0. The Sum of Absolute Differences (SAD) with a window size of 9 × 9 pixels has been used for correlation. The left/right consistency check (Fua, 1993) identified inconsistencies and invalidated the corresponding disparity values. Only valid values have been compared to the ground truth and only values whose disparity differed by more than one have been considered as errors (Szeliski and Zabih, 1999). Each error that appears near a depth discontinuity in the ground truth (i.e. within the size of a correlation window) is considered as a border error. Table 1 shows a summary of results.

Border errors are further categorised according to the kind of border (i.e. left, right, top or bottom) and if the error identified the background wrongly as object (i.e. increased the size of the object) or identified the object wrongly as background. Table 2 shows the categorised border errors. The third column describes the maximum percentage of error that would be possible in one category on the Tsukuba images (i.e. in the worst possible case in which all disparities are wrong). This provides the base to compare different categories with each other as it measures the amount of borders of a certain kind in the images. The last column gives the fraction of the error (i.e. the sum of the first two columns) and the maximal possible error in the considered category.

The fraction shows that the amount of errors at the left and right object borders is indeed higher than the amount of errors at top and bottom borders. Furthermore, most errors identify the background near objects wrongly as object so that objects appear horizontally

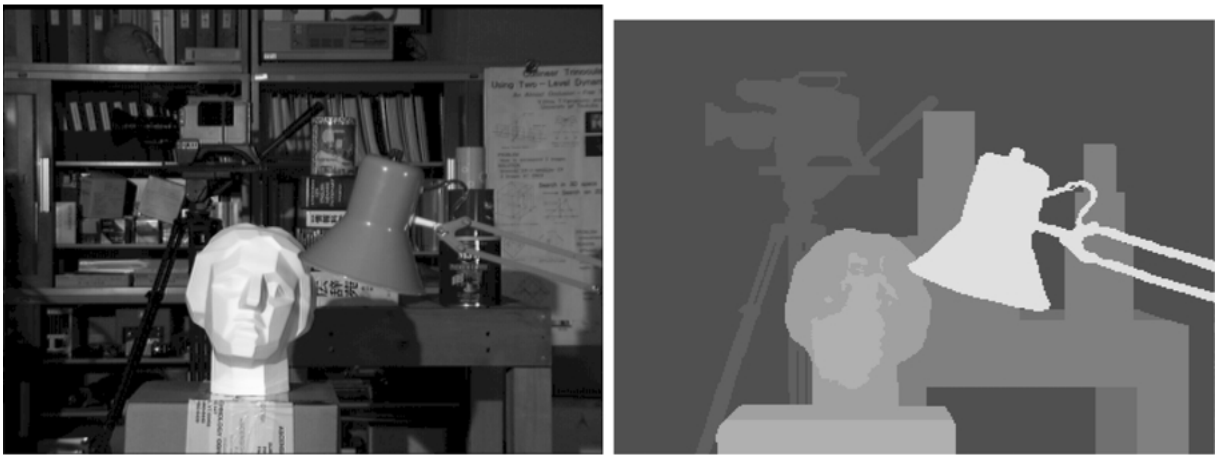

Figure 3. The left image and the ground truth from the University of Tsukuba.

extended. This confirms the prediction of the theoretical analysis. However, this test has only been performed on one stereo image pair. More data would be needed to establish statistically valid results.

The third biggest category are errors which identify the right part of an object wrongly as background. Certain places in the Tsukuba images can be identified, which are prone to this kind of error, like the right side of the upper tin. The background at the right side of the upper tin is a white poster in the left image, while the right image shows a gap between the tin and the poster, which is filled by darker background. The occluded background intensity level is more similar to the tin than to the visible background (i.e. the white poster). According to the theory, this should lead to an object border, which is moved inside the object (i.e. to the left). Figure 10 in Section 4 confirms that the right side of the tin appears to be wrongly shifted to the left.

Table 1. Results of SAD correlation on Tsukuba images.

Correct values	82.97%
Invalid values	11.03%
Errors at borders	4.53%
Other errors	1.47%

Table 2. Errors at borders, using SAD on Tsukuba images.

Border	Wrong object (%)	Wrong background (%)	Maximum error (%)	Fraction (see text)
Left	1.67	0.19	8.35	0.22
Right	1.73	0.40	8.51	0.25
Top	0.14	0.04	3.61	0.05
Bottom	0.19	0.17	3.31	0.11

3. Proposals of Improvements

3.1. Multiple Supporting Windows

Correlation windows that overlap a depth discontinuity introduce an error into the correlation calculation. The error can be reduced by only taking those parts of a correlation window into consideration that do not introduce errors. However, this has to be done systematically and comparably, as described below.

Figure 4(b) shows a configuration with one small window in the middle (C_0), surrounded by four partly overlapping windows (C_{1i}). The correlation value C can be computed by adding the values of the two best surrounding correlation windows C_{1i_1} and C_{1i_2} to the middle one. This approach can also be seen as using a small window C_0 and supporting the correlation decision by four nearby windows.

$$C = C_0 + C_{1i_1} + C_{1i_2} \tag{1}$$

Another configuration using 9 supporting windows is shown in Fig. 4(c). The correlation value in this case is calculated by adding the four best surrounding correlation values to the middle one.

$$C = C_0 + C_{1i_1} + C_{1i_2} + C_{1i_3} + C_{1i_4} \tag{2}$$

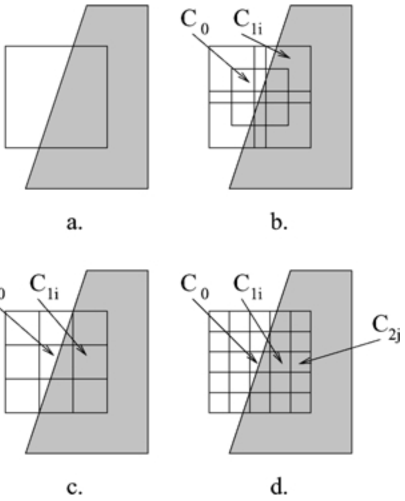

Figure 4. Configurations with multiple windows.

The approach can be extended by adding another ring of surrounding windows as shown in Fig. 4(d). The correlation value for the 25 supporting windows configuration is calculated by using the four best values of the middle surrounding ring and the eight best values of the outer ring.

$$C = C_0 + C_{1i_1} + \cdots + C_{1i_4} + C_{2k_1} + \cdots + C_{2k_8} \quad (3)$$

It can be seen that it is possible for these correlation windows to adapt to the local environment by assembling a big correlation window out of smaller ones. The blurring effect should be reduced as only the small middle window C_0 is always used and may overlap the depth discontinuity. All other parts can adapt to avoid an overlap with the depth discontinuity. Nevertheless, a good correlation behaviour is still maintained because of the big area that is covered using the best neighbouring windows.

The measure for calculating the correlation value of the individual windows can be selected as needed. The Sum of Absolute Differences (SAD) is very fast to calculate as it does not require multiplications compared to Sum of Squared Differences (SSD) or Normalised Cross Correlation. Furthermore, it gives good results and was therefore chosen for other real time stereo systems (Konolige, 1997; Matthies et al., 1995).

The calculation of C seems to be computationally costly as it needs to be done for all image pixels at all possible disparities. However, an implementation can make use of the same optimisations proposed for standard correlation (Faugeras et al., 1993) to compute the individual windows, which are all the same size. The correlation step alone involves to calculate for every pixel at every disparity 2 absolute differences for the SAD measure and additionally 4 additions and subtractions to calculate the final correlation value. Operations for loading data from memory and storing results back are not considered, as well as the overhead of the loop.

The multiple window approach requires additionally to select the best surrounding correlation windows and to calculate a sum. The selection of the best windows is costly, as it requires a sorting algorithm. However, an implementation can take advantage of the fact that the best values do not need to be sorted themselves. Selecting the two best values out of four as required by the configuration with 5 windows can be implemented with 4 comparisons and 2 additions. The configuration using 9 windows would require 16 comparisons and 4 additions and with 25 windows, 80 comparisons and 12 additions (i.e. see Appendix A.1).

The configuration using 5 windows seems to require the same amount of time as the correlation phase, according to this theoretical consideration. The configuration using 9 or 25 windows would require several times more processing time. Consequently, the configuration using 5 windows is suitable for a real time implementation.

Appendix A.1 shows the algorithm in detail and explains parallel implementations on modern processor architectures.

3.2. Filtering of General Errors

The determination of a disparity value involves correlating the window in the first image with windows at all disparities d in the second image. The resulting correlation values C form a correlation function as shown in Fig. 5. The disparity at which the correlation function is lowest corresponds with the place of highest similarity.[1] The left/right consistency check (Fua, 1993) uses the place of highest similarity in the second image and then moves the correlation window of the first image over all possible disparities, which gives another correlation function. The disparity is considered to be valid if the minimum of the second correlation function corresponds to the same disparity as the minimum of the first correlation function.

The left/right consistency check is a very effective mean to identify places where correlation is contradictory and thus uncertain. This is usually the case at occlusions (Fusiello et al., 1997). An analysis of the

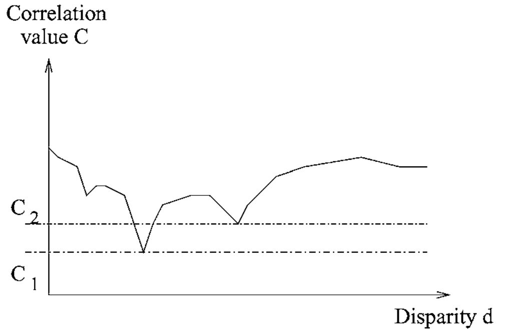

Figure 5. A typical correlation function. The minima C_1 is the place of highest similarity.

correlation function can further help to identify uncertainties. A nearly flat correlation function corresponds to areas with low texture. A function with several minima indicates several good places which can be caused by repetitive texture. In these cases image noise can easily lead to wrong decisions. Let C_1 be the minimum correlation value and C_2 the second lowest correlation value. C_2 should not be a direct neighbour of C_1, because the best place for correlation usually lies between pixels. If C_1 and C_2 are direct neighbours, then they would represent neighbouring pixel positions of the same minimum and not the position of the lowest and second lowest minimum. The relative difference C_d can be calculated as:

$$C_d = \frac{C_2 - C_1}{C_1} \tag{4}$$

A low C_d indicates possible problems. It is assumed that many errors will be caught by invalidating all values whose C_d is below a certain threshold for the correlation function. However, the threshold needs to be set empirically, depending on the constraints of the application.

Moravec's 'Interest Operator' offers a way of invalidating low texture areas before correlation is performed (Moravec, 1977). However, the method described above considers the image directly through the correlation function, which should be more accurate. Secondly, problems with repetitive like texture are treated at the same time.

An implementation of the error filter needs to select the second best correlation value, to calculate the relative difference between the best and second best value and to use the threshold to reject uncertain values. The selection of the second best correlation value is as expensive as the search of the best correlation value that has always to be done. The operation can be well implemented in parallel on modern processor architectures.

3.3. Border Correction Filter

The behaviour of stereo correlation at object borders depends on local similarities. In Section 2.2 was shown that most errors appear at left and right object borders and extend the size of objects. This is a systematic error that is typical for correlation. A correction of this error would improve the shapes of objects significantly.

After the disparity image is calculated, vertical disparity gradients can be discovered by comparing horizontally neighbouring disparity values. A positive disparity step represents a calculated left object border, while a negative step represents a calculated right object border. The real position of the object border is usually within the distance of half the size of a correlation window, according to the theory in Section 2.2. However, usually some filters are used. For example the left/right consistency check, which invalidates many occluded disparity values near left object borders (Fua, 1993). For the purpose of identifying disparity steps, the lowest neighbouring value of an invalidated area is propagated through the invalid area (Fusiello et al., 1997).

Figure 6 shows a situation of a positive disparity step. The dotted line marks the position of the calculated left object border. The calculated object border is assumed to go always vertical through the correlation window, for simplicity of calculation. The pixel of interest in the middle of the correlation window corresponds to the higher disparity of the object, while all pixels to its left have the lower disparity of the background. If the calculated border is correct, then only the correlation $c(R_2, \tilde{L}_2)$ is correct for a correlation of R with $\tilde{L}$. The correct partner for R_1 would be L_1, which is shifted to

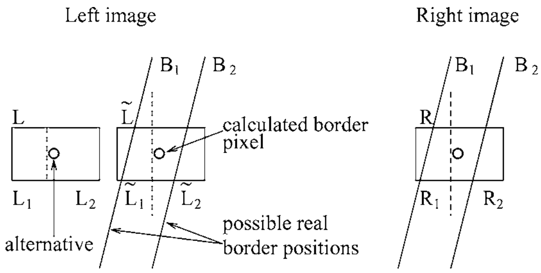

Figure 6. Situation where $\tilde{L}$ has been chosen. This is correct, if the real border is at B_1, but wrong if it is at B_2.

the left by a distance that corresponds to the disparity difference between the object and the background. All pixels between the right border of L_1 and the left border of $\tilde{L}_2$ would be occluded.

However, the real object border is usually a few pixels further left or right and in general not vertical. The direction in which the real object border is, can be identified by comparing $c(R_1, L_1)$ and $c(R_2, \tilde{L}_2)$. Correlation windows commonly have an odd size so that they are symmetric around its point of interest. To compare both values properly, the size of both halves of the correlation window is made equal by increasing the width of the left half window by one pixel. If the real border corresponds to position B_1, then the value $c(R_2, \tilde{L}_2)$ should be low because it is completely correct, while $c(R_1, L_1)$ should be high because only a part of R_1 does really correspond to L_1. The situation is vice versa if the real position of the border corresponds with B_2. Finally, if the position of the real border goes through the middle of the correlation window, both correlation values are equally low apart from image noise.

Consequently, the values $c(R_1, L_1)$ and $c(R_2, \tilde{L}_2)$ are calculated, while moving the windows in both images simultaneously to the left and right. The position where $c(R_1, L_1)$ has the same amount as $c(R_2, \tilde{L}_2)$ is searched. However, this position is in general between pixel coordinates. As an approximation, the pixel position where the difference between $c(R_1, L_1)$ and $c(R_2, \tilde{L}_2)$ is lowest is used as the position of the correct object border. The disparity values need to be corrected accordingly.

In practise the situation can be much more complex. The depth might vary not only once, but several times within a small area, due to slanted objects. This might confuse the correction algorithm as the assumption of constant depth within half of a correlation window is again violated. However, the case above is assumed to occur often and thus justifies this special treatment.

The computational expense is quite low compared to the correlation stage, because only places where object borders are assumed need to be inspected. Typically, processing the Tsukuba stereo image pair results in less than 5% of the pixels, which are assumed to be object borders. Some of these are actual border pixels and the rest are errors. In contrast, correlation is performed at every pixel and for all possible disparities.

The structure of the border correction algorithm is outlined in Appendix A.2 in pseudo code. The calculated disparity image as well as both rectified source images and the size of the used correlation window serve as input to the algorithm.

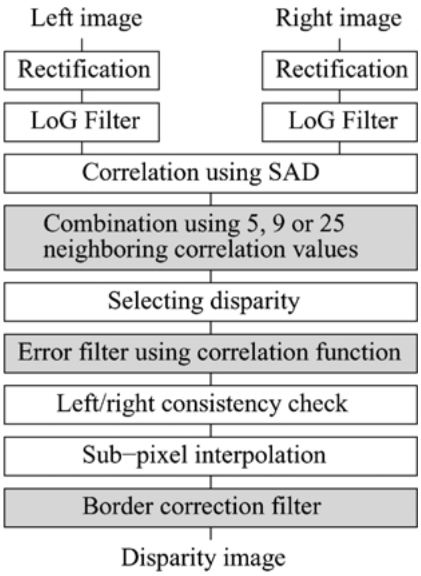

Figure 7. Overview of a standard correlation algorithm with new methods shown in grey (see text for description).

3.4. Summary of the Whole Algorithm

The improvements, which have been suggested in the last sections can be included into the framework of a standard correlation algorithm as shown in Fig. 7. The source images are first rectified and aligned, so that the epipolar lines correspond to image rows and the image rows with the same number correspond to each other. Next, the Laplacian of Gaussian is used as a pre-filter.

Correlation can be done by using optimisation techniques as suggested by Faugeras et al. (1993). The correlation values are calculated row by row for all disparities at all pixels and stored temporarily for the combination step. In the combination step, the correlation values of the neighbouring 5, 9 or 25 windows are used to calculate the combined correlation value as proposed in Section 3.1. The results are stored in two dimensional arrays for every image row. Each array contains the combined correlation value for all pixels and all disparities.

The disparity of a pixel is selected by searching the lowest correlation value for one pixel. The error filter that was proposed in Section 3.2 additionally searches for the second lowest value and calculates the relative difference as a measure of uncertainty. Disparity values whose difference are below a threshold are rejected as uncertain.

The left/right consistency check that was introduced by Fua matches pixels from the right image back to the left image and identifies many errors, which are caused by occluded pixels near left object borders (Fua, 1993). Sub-pixel interpolation is done to increase the depth resolution, by using the three correlation values around the position of the calculated disparity and fitting a quadratic curve through them. The minimum of the curve corresponds to the sub-pixel disparity.

Finally, the border correction method modifies the disparity image by horizontally shifting assumed object borders. This was proposed in Section 3.3.

The overview above shows all functions as separate steps. However, for memory efficiency, it is useful to interleave the steps from correlation until sub-pixel interpolation, so that one image row is processed by all steps before the next image row is considered.

4. Qualitative Assessment

4.1. Experimental Setup and Analysis

A stereo image pair from the University of Tsukuba (Fig. 3 in Section 2.3) and an image of a slanted object (Fig. 8) from Szeliski and Zabih (1999) have been used for evaluation. Both are provided on Szeliski's web-page.[2] The image of the slanted object is very simple. However, it is expected to compensate for the lack of slanted objects in the Tsukuba images.

All given disparity images are enhanced for visual analysis by using the full intensity range for showing the used disparities range (i.e. 32 disparities). Light grey is used for high disparities (i.e. close objects), whereas darker grey corresponds to smaller disparities. Black is used in the disparity images for values that are rejected by the algorithm as being invalid.

All disparities that are marked as invalid have been ignored for comparison with the ground truth. Disparities that differ by only one from the ground truth are considered to be still correct (Szeliski and Zabih, 1999). The amount of errors at object borders is calculated as explained in Section 2.3 and shown separately.

The difference images, which are provided next to the disparity images show the difference of calculated disparity and ground truth. Correct matches appear in white as well as invalid matches, which are ignored for comparison. All errors (i.e. disparity values that differ by more than one from the ground truth) are shown in black.

The range of possible disparities has been set to 32 in all cases. For every method, all combinations of *meaningful* parameters were computed to find the best possible combination for the Tsukuba images. The horizontal and vertical window size was usually varied between 1 and 19. The standard deviation of the LOG filter was varied in steps of 0.4 between 0.6 and 2.6. All together almost 20000 combinations were computed for the Tsukuba image set, which took several days using mainly non-optimised code.

4.2. Results of Standard Correlation Methods

The results of the best parameter combination (i.e. which gives the lowest error) for some standard

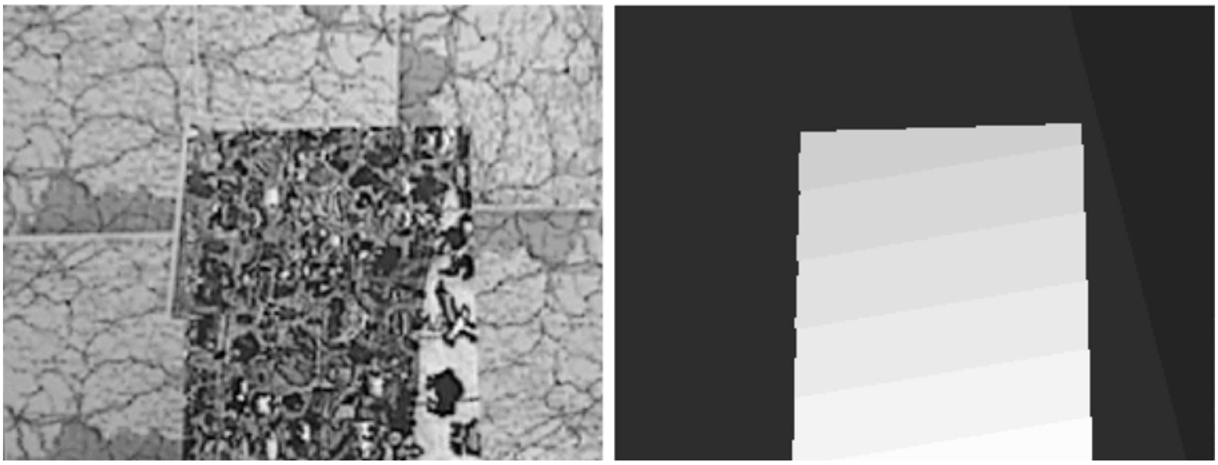

Figure 8. The left image and the ground truth of a slanted object from Szeliski and Zabih.

Table 3. Results of standard methods (first part), proposed methods (second part) and combinations of proposed methods (third part) on Tsukuba images.

Method	Window	Rank/Census	LOG (σ)	Correct (%)	All errors (%)	Border errors (%)	Invalid (%)
Normalised cross correlation (NCC)	9 × 19	–	0.0	82.37	8.15	7.05	9.49
Sum of absolute differences (SAD)	9 × 9	–	1.0	82.97	6.00	4.39	11.03
Sum of squared differences (SSD)	9 × 9	–	1.0	81.42	6.55	4.88	12.03
Non-parametric Rank	11 × 11	9 × 7	–	85.68	4.58	3.96	9.74
Non-parametric Census	9 × 11	9 × 7	–	84.86	4.65	3.87	10.49
SAD with mult. windows (MW-SAD)	11 × 9	–	0.0	80.88	4.91	2.92	14.21
SAD with 5 windows config. (SAD5)	7 × 9	–	0.0	85.12	4.56	3.36	10.32
SAD with 9 windows configuration	5 × 5	–	0.0	83.65	4.39	2.89	11.96
SAD with 25 windows configuration	3 × 5	–	1.0	83.36	4.89	3.36	14.67
SAD with 10% error filtering	9 × 9	–	1.0	78.96	4.14	3.61	16.89
SAD with border correction	9 × 9	–	1.0	85.63	6.10	4.04	8.26
SAD5 with 10% error filtering	7 × 9	–	0.0	80.70	3.02	2.59	16.28
SAD5 with 10% error filt. and border corr.	7 × 9	–	0.0	82.24	3.26	2.45	14.50

correlation methods can be found in the first part of Table 3. The MW-SAD approach performs correlation at every disparity with 9 windows with asymmetrically shifted points of interest and uses the best resulting value. Algorithms which are based on this configuration have been proposed in the literature for improving object borders (Little, 1992; Fusiello et al., 1997). Results are discussed in Section 4.3.

The best parameter combinations of the Tsukuba images have been used on the slanted object images as well. Almost all errors occur near object borders on this simple image set. This is probably due to the evenly strong texture and the lack of any reflections, etc. It is interesting that the slanted nature of the object, which appears as several small depth changes, is generally well handled. However, the weak slant is not really a challenge for correlation.

Figure 9 shows the errors of all methods on the Tsukuba (left) and slanted object images (right) as a graph. It can be seen that the amount of errors is different for each method for both image sets. However, the graphs show almost the same tendency by comparing the methods with each other (i.e. NCC is worse

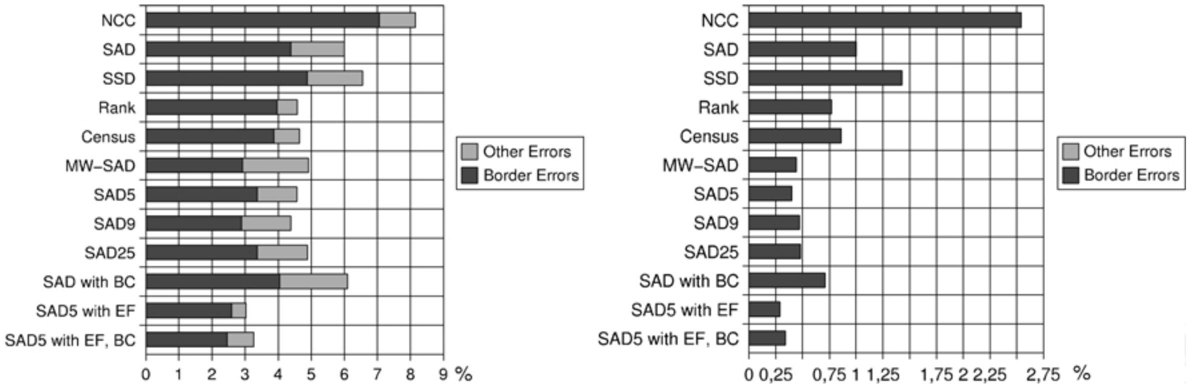

Figure 9. Errors of all methods on the images from University of Tsukuba (left) and the slanted object (right). The slanted object shows only errors at object borders due to its evenly strong texture. BC is border correction and EF is error filtering.

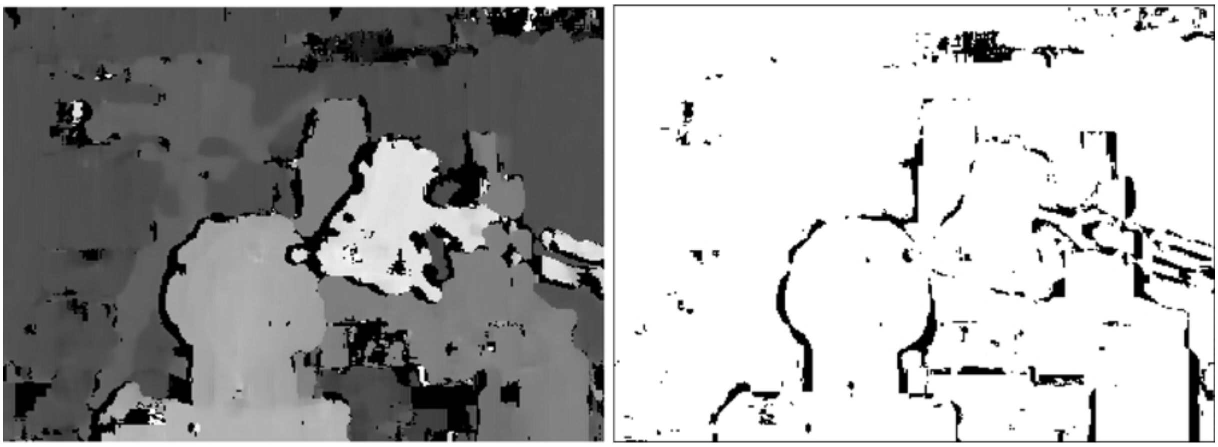

Figure 10. Result from SAD correlation.

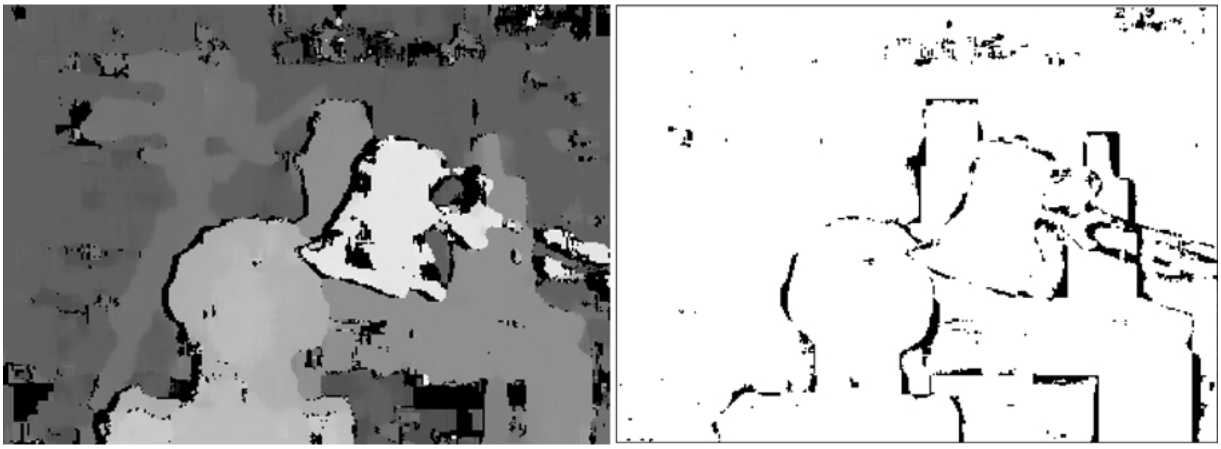

Figure 11. Result from Rank correlation.

than SSD and SAD is slightly better than SSD for both image sets).

The SAD correlation (Fig. 10) was chosen as the basis for an evaluation of the proposed improvements. It is the fastest in computation and shows advantages over NCC and SSD. The non-parametric Rank and Census transform (Figs. 11 and 12) give better results because they are more tolerant against outliers (Boykov et al., 1998). However, Census is expensive to compute due to the calculation of the Hamming distance and Rank is rather seen as a filter, like LOG, that transforms the source images before a SAD correlation is performed.

4.3. Results of Proposed Methods

All suggested improvements have been evaluated using SAD correlation. The results of the best parameter combinations are shown in the second part of Table 3. The error filter and the border correction was only applied to the best parameter combination of SAD. The same parameter combinations on the slanted object images show again very similar results. This can easily be seen in Fig. 9.

The multiple correlation window configuration showed improvements in the number of correct matches as well as errors compared to SAD. The performance seems to be especially good at object borders. Figure 13 shows the results from the 5 windows configuration. The rings of errors around objects look smaller compared to Fig. 10. This means that there are less errors in border areas. Additionally, the rings of errors appear more even, which means that although the object appears wrongly bigger, its shape is much less fuzzy. This can also be seen by comparing

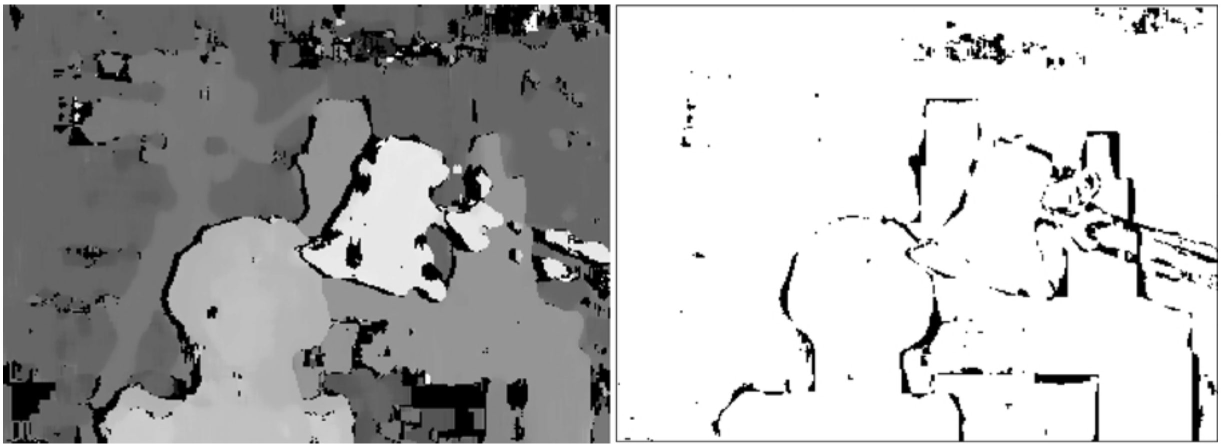

Figure 12. Result from Census correlation.

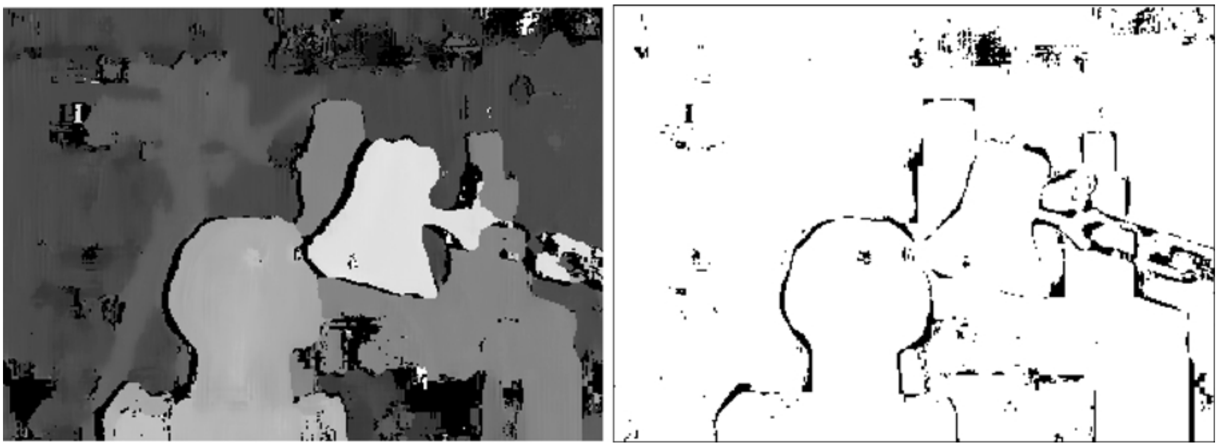

Figure 13. Result from the 5 window configuration.

the disparity images (i.e. left images) in Fig. 13 with 10.

Rank and Census (Figs. 11 and 12) produce similar improved numerical results. However, a visual comparison again unveils a slightly more fuzzy object border, compared to the 5 windows configuration in Fig. 13. This can be best seen by comparing the shape of the lamp or the pile of tins.

A comparison with the MW-SAD shows that MW-SAD performs better in the synthetic case of horizontal or vertical object borders, but performs worse at general border shapes (i.e. introduces steps). This can be seen in Fig. 14. Additionally, MW-SAD is less stable in general, which increases general errors as well as invalid matches. It is assumed that the middle window, which is always used in the suggested 5 window configuration, serves as a stabilising factor in the calculation.

The error filter that was tested for different thresholds on the best parameter configuration of SAD exhibits an expected characteristic. The graph in Fig. 15 shows that many errors can be caught at the risk of filtering correct matches out as well. However, the amount of filtered errors compared to filtered correct matches is quite high when the ratio between errors and correct matches is considered. A threshold of 10% filters for example almost 2% errors out, at the expense of loosing 4% correct matches. Furthermore, filtered correct matches are distributed all over the image, so that their disappearance can be compensated by interpolation. In the end the amount of lost correct matches that is acceptable depends on the application.

The threshold for error filtering is difficult to choose. One strategy in practice without having a ground truth could be to set the threshold so that the number of

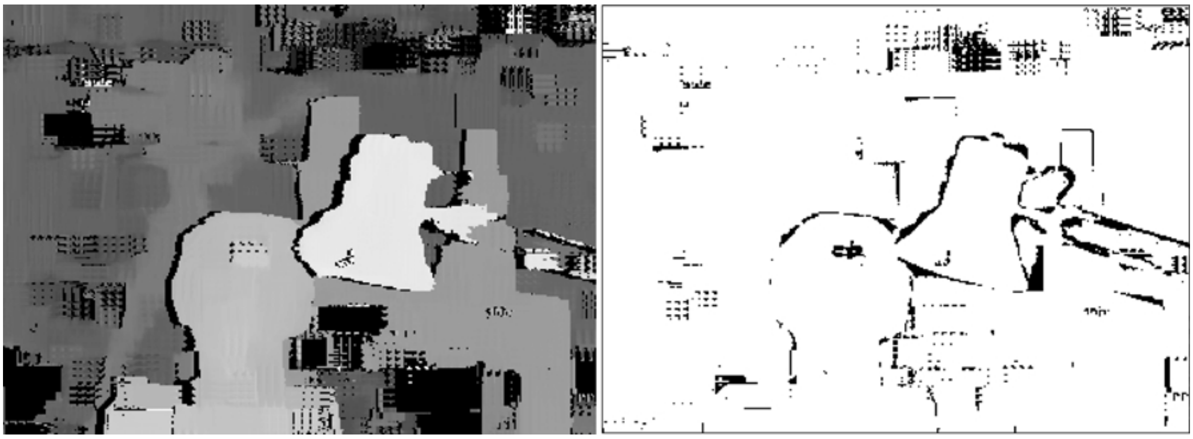

Figure 14. Result from MW-SAD correlation.

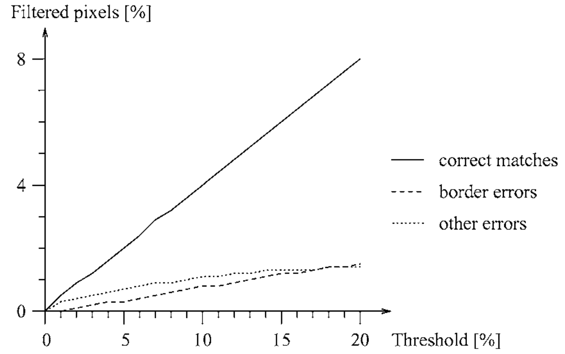

Figure 15. Filtered correct matches and errors at certain thresholds, using SAD on the Tsukuba images.

invalid matches is increased by a fixed amount. Another strategy would involve to point the cameras to a large texture-less area and to set the threshold high enough so that the whole texture-less area is just invalidated. Thus, the threshold would be just high enough so that arbitrary matches due to image noise from the cameras, frame grabbers, etc. are prevented.

Finally, an evaluation of the border correction shows only a slight decrease in errors at object borders and an unexpected increase of errors at other places. Nevertheless, the number of correct matches is in this example increased by 2.66% compared to SAD without border correction. The situation can be explained using Fig. 16. The borders of objects are in fact improved compared to Fig. 10 (i.e. rings of errors around objects appear much smaller), which results in the decrease of border errors. The increase in correct matches results from changing many invalid values near object borders into valid, correct values.

The increase in errors at other places is due to the fact that the algorithm tried to correct object borders that resulted from previous errors, leading to a randomly

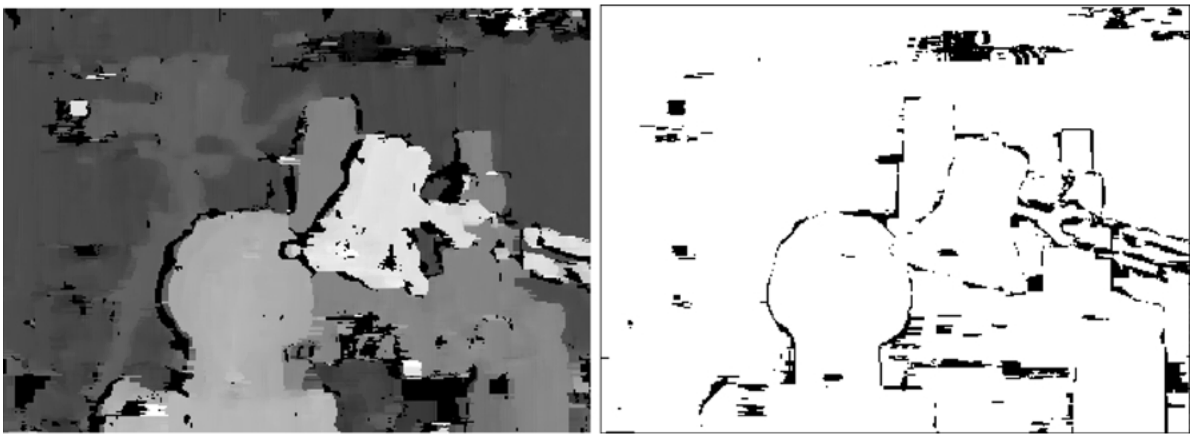

Figure 16. Result from SAD correlation with border correction.

stretching or shifting of error patches. It is unfortunately not possible for the border correction algorithm to differentiate between correct but shifted and completely incorrect object borders. A reduction of general errors would be advantageous to prevent this behaviour. The error filter would be appropriate for this purpose. A combination of these methods is discussed in Section 4.4.

Although borders are improved, small details which were lost during the correlation phase, like the cable of the lamp, cannot be recovered using this method. Finally, it can be concluded that the effect of noise gets stronger, the further the border is moved towards the real object border, due to the design of the calculation. The method leads to reduced border errors, but usually not to a complete removal. A remedy could be a more accurate consideration of the distribution of neighbouring disparities within the window.

4.4. Results of Combinations of Proposed Methods

The third part of Table 3 shows results of combinations of several methods. The best parameter combinations established previously have been used. The result is also shown in Fig. 17. Comparing these results visually and in their numbers against any of the standard correlation methods clearly shows an improvement for certain applications.

Not only general errors were reduced, but especially errors in border areas of objects. However, invalid values are increased due to the error filter. Nevertheless, it is acceptable for some applications to increase the amount invalid values slightly in order to reduce errors.

A comparison between the SAD correlation that was chosen as a base and the combination of all proposed methods shows on the example images that errors were reduced by almost 50% and the number of correct matches was maintained.

Finally, the results of the same combinations of methods and parameters on the slanted object images show almost no improvement. A look at the disparity difference image reveals that the border error before was almost only one pixel, which is already very low. Other errors have not been detected. There is not very much room for further improvements. The border correction algorithm corrected the depth discontinuity slightly to much, which results in an object that appears slightly smaller than it really is.

5. Performance of the Real-Time System

5.1. Experimental Setup

All the methods described have been implemented in optimised C and contain optional inline assembler sections to make use of MMX commands. MMX is the Multi-Media Extension that was introduced by Intel in the Pentium processors. It allows parallel processing of logical and arithmetical integer operations, which can increase the performance several times. However, only the critical loops in the whole process were optimised to save development effort. All performance tests were accomplished on a Pentium II, 450 MHz using the Linux (Kernel 2.4) operating system. Two BT878 based frame grabbers were used to capture stereo images from a self-made medium resolution stereo camera.[3]

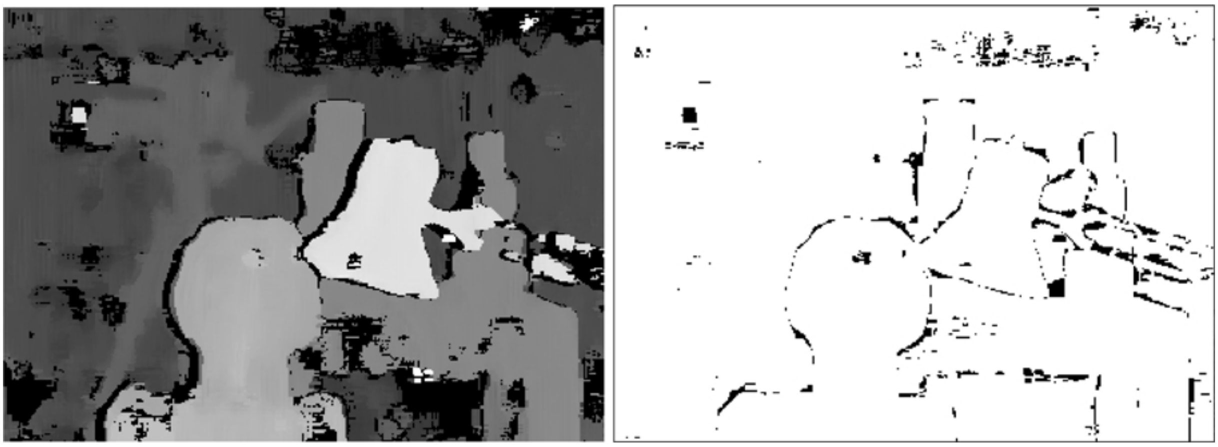

Figure 17. Result from 5 windows configuration, 10% error filtering and border correction.

All measurements were done by grabbing, correlating and painting the disparity image 600 times. The results represent the average time.

5.2. Performance Results

The size of the source images and the disparity range are the main factors that influence the speed of execution. However, the execution time of the border correction filter depends partly on the amount of detected gradients in the disparity image. All gradients need to be analysed and corrected. Table 4 lists the parameters and shows the effective frame rate that was measured including sub-pixel interpolation and painting the disparity image on the screen.

Finally, Table 5 shows how much of the execution time was spent in individual parts of the process and the language that was used for implementation.

The suggested improvements, which are shown in italic, require of course some additional computation time and slow down the frame rate. If only the standard algorithm without all suggested improvements is used, then the frame rate increases from 4.7 frames/s to 7 frames/s, by loosing a major reduction in errors, especially at object borders.

Nevertheless, the results show clearly that all proposed methods are suitable for real time usage.

Table 4. Parameters and frame rate using all methods.

Used hardware	P II, 450 MHz
Overall speed (C and MMX)	4.7 frames/s
Overall speed (optimised C)	1.6 frames/s
Size of images	320 × 240 pixel
Size of correlation window	7 × 7 pixel
Disparity range	32 pixel
Sub-pixel interpolation	1/8 pixel
Laplacian of Gaussian	1.0

Table 5. Time spend in individual parts.

Function	Language	Time (ms)
Rectification	C, MMX	11
Laplacian of Gaussian	C, MMX	25
SAD correlation	C, MMX	53
Combination of 5 windows	C, MMX	30
Determine disparity	C, MMX	17
Error filtering	C, MMX	14
Left/Right consistency check	C, MMX	20
Sub-pixel interpolation	C	6
Border correction	C	18
Painting disparity image	C, Java	11

6. Conclusion

It has been shown that it is possible to improve simple correlation by understanding the source of its weakness. Three methods have been proposed, which tackle specific problems of correlation. A novel multiple window approach decreases errors at object borders and increases correct matches. A general error filter uses the correlation function to invalidate uncertain matches. Finally, a border correction method improves object borders further in a post-processing step. It was shown that all improvements are suitable for real-time applications. All methods were explained in detail, including their integration into a standard correlation algorithm. Optimisation issues and parallel implementation was discussed as well.

Every method shows clear improvements, but also weaknesses. The main weakness of the multiple correlation window configuration is its computational cost. However, an optimised implementation of the configuration using 5 windows is possible and very effective. The error filtering requires a threshold, which is difficult to choose in practise and reduces the number of correct matches as well. Finally, the border correction improves object borders, although previous general errors can be slightly increased.

Nevertheless, the combination of suggested methods improves the quality of real-time correlation based stereo significantly. In the example images the errors have been reduced to 50%, while the number of correct matches has been maintained. Further research in this area could bring even better results.

The whole stereo algorithm has been implemented and optimised, including all the proposed methods. A detailed performance evaluation shows that all the proposed methods require one third of the whole processing time. The system grabs stereo images, rectifies, filters and correlates them almost 5 times a second on a Pentium II with 450 MHz. Current hardware can increase the speed to processing at frame rate. Some further optimisation is possible as only the critical loops were optimised.

Furthermore, the proposed methods can be used individually or in combination. This allows a suitable quality and speed selection for an application. The current system is the base for further research on a teleoperated robot. It will be used for processing the local working environment of the robot on a higher, object based level.

Appendix: Algorithms in Pseudo Code

A.1. Multiple Supporting Windows Algorithm

The multiple supporting correlation window algorithm (Section 3.1) is an additional step between the calculation of the correlation values and the selection of the disparity value for every pixel.

The combination of five correlation values is shown below in pseudo code. The algorithm produces the combined correlation value for all pixels of row k at all disparities. It expects that the correlation values have been calculated for all pixels for the three image rows $k - w_y$, k and $k + w_y$, were $2w_y + 1$ is the height of the correlation window. $c_{in}(i, k, d)$ refers to the correlation value in the image column i, image row k and disparity d. The results will be stored in $c_{out}(i, k, d)$, which uses the same syntax.

The algorithm can make use of the fact that the order of the lowest m out of n values is not important. This results in $m(n - m)$ comparisons between values. Only 4 comparisons are required for the five supporting windows configuration.

Furthermore, the SIMD architecture[4] of modern processors can be used to process data in parallel. Comparison and selection can make use of saturated arithmetic to avoid jumps. The code below can be encoded in 20 Pentium II assembler instructions (i.e. including the transfer from memory into registers and the transfer back to memory, but excluding loop overhead) and processes 4 values in parallel. This results in only 5 assembler instructions for every pixel at every disparity.

```
for all pixels $i$ in row $k$ do
  for all disparities $d$ do
    $c = c_{in}(i, k, d)$;

    $c_1 = c_{in}(i - w_x, k - w_y, d)$;
    $c_2 = c_{in}(i + w_x, k - w_y, d)$;
    $c_3 = c_{in}(i - w_x, k + w_y, d)$;
    $c_4 = c_{in}(i + w_x, k + w_y, d)$;

    $c_{l1}$ = lowest value of $c_1, c_2, c_3, c_4$;
    $c_{l2}$ = second lowest value of $c_1, c_2, c_3, c_4$;
    $c_{out} = c + c_{l1} + c_{l2}$
  end
end
```

A.2. Border Correction Algorithm

Section 3.3 explained the theory behind object border correction and gave an overview of the algorithm. This section shows the algorithm in pseudo code.

Firstly, some definitions are required. Figure 18 shows a correlation window of the size $2w_x + 2$, $2w_y + 1$ (i.e. the width of the window is increased by 1 as explained in Section 3.3). The area that is covered by the left half of the the window at the position i, k in the image X is defined as $A_1(X, i, k)$ and the right half as $A_2(X, i, k)$. L and R refer to the left and right rectified image. $D(i, k)$ is the calculated disparity at i, k. If i is the position of a positive disparity step, then $d_b = D(i - 1, k)$ is the disparity of the background and $d_o = D(i, k)$ is the disparity of the object. The correlation windows that were shown in Fig. 6 in Section 3.3 can now formally be defined.

$$L_j = L_j(i, k) = A_j(L, i - d_o + d_b, k) \tag{5}$$

$$\tilde{L}_j = \tilde{L}_j(i, k) = A_j(L, i, k) \tag{6}$$

$$R_j = R_j(i, k) = A_j(R, i - d_o, k) \tag{7}$$

$$\tilde{R}_j = \tilde{R}_j(i, k) = A_j(R, i - d_b, k) \tag{8}$$

The following pseudo code contains only the short forms on at the left side of these definitions (e.g. $\tilde{L}_1$), because they are always used at the position i, k. The algorithm that corrects all left object borders can now be written as:

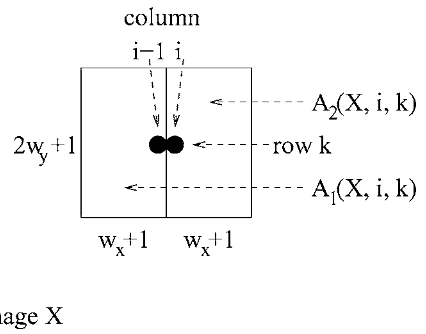

Figure 18. Definition of the areas A_1 and A_2 of a correlation window.

```
for $k = 1$ to number of rows do

  $i = 0$;
  while $i <=$ number of columns do
    if $D(i, k)$ is invalid then
      use lowest valid disparity, either to the
      left or right of the invalid disparity area
      as $D(i, k)$
    end

      // search positive disparity step

    if $D(i - 1, k) < D(i, k)$ then

      // identify disparity of background and object

      $d_b = D(i - 1, k)$;
      $d_o = D(i, k)$;
      $v_1 = c(R_1, L_1) - c(R_2, \tilde{L}_2)$;

      if $v_1 < 0$ then

        // shift left border to the right until
        // the correct position is found

        $j = i + 1$;
        $n = i + w_x$;

        while $j <= n$ and $v_1 < 0$ do

          $v_2 = c(R_1, L_1) - c(R_2, \tilde{L}_2)$;

          if $v_2 < 0$ or $-v_1 > v_2$ then
            $D(i, k) = d_b$;
            $i = i + 1$;
          end

          $v_1 = v_2$;
          $j = j + 1$;
        end
      else

        // shift left border to the left until
        // the correct position is found

        $j = i - 1$;
        $n = i - w_x$;

        while $j >= n$ and $v_1 > 0$ do

          $v_2 = c(R_1, L_1) - c(R_2, \tilde{L}_2)$;

          if $v_2 > 0$ or $v_1 > -v_2$ then
            $D(j, k) = d_o$;
          end

          $v_1 = v_2$;
          $j = j - 1$;
        end
      end
    end

    $i = i + 1$;
  end
end
```

The correction of all right object borders requires a second pass over the whole disparity image. The algorithm is the same apart from some minor differences and is therefor not explicitely given here. The first difference is that a negative disparity step is searched (i.e. $D(i - 1, k) > D(i, k)$) instead of a positive step. Next, the disparity of the object d_o is $D(i - 1, k)$ and the disparity of the background d_b is $D(i, k)$. Finally, the calculation of the difference between both halves of the correlation window is done by using $c(\tilde{R}_1, \tilde{L}_1) - c(R_2, \tilde{L}_2)$ instead of $c(R_1, L_1) - c(R_2, \tilde{L}_2)$.

Correction of the left and right object border could be done within one pass over the image. However, special care is required to adjust the loop counter i and the values that depend on it (i.e. R_1, L_1, etc.), because the i is increased in some inner loops too.

Acknowledgments

The authors would like to thank QinetiQ (i.e. formerly known as DERA) for their financial support and the University of Tsukuba and R. Szeliski and R. Zabih for making their stereo images with ground truth available for evaluation.

Notes

1. The SAD correlation has low values if the similarity is high.
2. http://www.research.microsoft.com/∼szeliski/stereo/.
3. http://www.cse.dmu.ac.uk/∼hhm/research.html.
4. Single Instruction Multiple Data.

References

Boykov, Y., Veksler, O., and Zabih, R. 1998. A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1283–1294.

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

Faugeras, O., Hotz, B., Mathieu, H., Viville, T., Zhang, Z., Fua, P., Thron, E., Moll, L., Berry, G., Vuillemin, J., Bertin, P., and Proy, C. 1993. Real time correlation-based stereo: Algorithm, implementations and application. INRIA, Tech. Rep. 2013.

Fua, P. 1993. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6:35–49.

Fusiello, A., Roberto, V., and Trucco, E. 1997. Efficient stereo with multiple windowing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Puerto Rico. IEEE: Piscataway, NJ, pp. 858–863.

Kanade, T. and Okutomi, M. 1994. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:920.

Konolige, K. 1997. Small vision systems: Hardware and implementation. In *Eighth International Symposium on Robotics Research*, Hayama, Japan. Springer: London, pp. 203–212.

Little, J.J. 1992. Accurate early detection of discontinuities. In *Proceedings of the Vision Interface Conference VI'92*, Vancouver, Canada, pp. 97–102.

Marr, D. and Poggio, T. 1979. A computational theory of human stereo vision. *Proceedings of the Royal Society*, B-204:301–328.

Matthies, L., Kelly, A., and Litwin, T. 1995. Obstacle detection for unmanned ground vehicles: A progress report. In *International Symposium of Robotics Research*, Munich, Germany.

Moravec, H. 1997. Toward automatic visual obstacle avoidance. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 584–590.

Szeliski, R. and Zabih, R. 1999. An experimental comparison of stereo algorithms. In *Vision Algorithms: Theory and Practice*, Springer-Verlag: Corfu, Greece, pp. 1–19.

Volpe, R., Balaram, J., Ohm, T., and Ivlev, R. 1996. The rocky 7 mars rover prototype. In *International Conference on Intelligent Robots and Systems*, Osaka, Japan, vol. 3, pp. 1558–1564.

Zabih, R. and Woodfill, J. 1994. Non-parametric local transforms for computing visual correspondance. In *Proceedings of the European Conference of Computer Vision 94*, pp. 151–158.

Zitnick, C. and Kanade, T. 1999. A cooperative algorithm for stereo matching and occlusion detection. Carnegie Mellon University, Robotics Institute, Pittsburgh, PA,Tech. Rep. CMU-RI-TR-99-35.

Zitnick, C.L. and Kanade, T. 2000. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:675–684.