Trinocular Stereo Vision Using a Multi Level Hierarchical Classification Structure*

Andy Motten[1], Luc Claesen[1], and Yun Pan[2]

[1] Expertise Centre for Digital Media, Hasselt University – tUL – iMinds
Wetenschapspark 2, 3590 Diepenbeek, Belgium
firstname.lastname@uhasselt.be
[2] Institute of VLSI Design, Zhejiang University
Hangzhou, China
panyun@vlsi.zju.edu.cn

Abstract. A real-time trinocular stereo vision processor is proposed which combines a window matching architecture with a classification architecture. A pair wise segmented window matching for both the center-right and center-left image pairs as their scaled down image pairs is performed. The resulting cost functions are combined which results into nine different cost curves. A multi level hierarchical classifier is used to select the most promising disparity value. The classifier makes use of features provided by the calculated cost curves and the pixels' spatial neighborhood information. Evaluation and classifier training has been performed using an indoor dataset. The system is prototyped on an FPGA board equipped with three CMOS cameras. Special care has been taken to reduce the latency and the memory footprint.

Keywords: trinocular stereo camera, real-time matching, confidence metric, computer vision, system-on-chip, FPGA, SoC.

1 Introduction

Trinocular vision makes use of three cameras to calculate a disparity space image (DSI). The DSI is generated by pairwise matching the images from the different cameras which is based on a local window based stereo matching architecture.

An improvement of occlusion handling in trinocular vision compared to stereo vision is achieved by Mozerov [1]. The main idea is based on the assumption that any occluded region in a matched stereo pair (center-left images) in general is not occluded in the opposite matched pair (center-right images). They use a global optimization technique to derive the composite DSI. Bidirectional matching using trinocular stereo is used by Ueshiba [2] to detect half-occlusions and to discard false matches. It uses a cumulative cost function derived from a summation of both cost curves.

* This research has been sponsored in part by the BOF (Bijzonder Onderzoeks Fonds uHasselt), Flanders FWO (Fonds voor Wetenschappelijk Onderzoek) and Chinese MOST (Ministry of Science and Technology) project number G.A.063.10.

A. Burg et al. (Eds.): VLSI-SoC 2012, IFIP AICT 418, pp. 45–63, 2013.

The method presented in this paper likewise calculates several DSI's. However, instead of combining them, a hierarchical classifier is used to select the most likely disparity for each pixel in the final DSI. The matching algorithm is based on the adaptive-weight algorithm proposed by Yoon [3], which adjusts the support weight of each pixel in a fixed sized window. The support weights are depending on the color and the spatial difference between each pixel in the window and the center pixel. Dissimilarities are computed based on the support weights and the plain similarity scores. Their experiment indicates that a local based stereo matching algorithm can produce depth maps similar to global algorithms. A hardware implementation using the same ideas is published by Motten in [4].

For each matching result, a confidence metric is calculated. A good comparison between different confidence metrics can be found in the evaluation paper of Hu [5]. Confidence metrics suitable for hardware implementation can be found in [6]. They conclude that neighboring pixels contain valuable information to distinguish good matches from bad ones.

Recently many stereo implementations have been proposed for hardware implementations. A real-time FPGA-based stereo vision system is presented by Jin [7] that makes use of the census transform. Their system includes all the pre- and post-processing functions such as: rectification, LR-check and uniqueness test in a single FPGA. Another extensive implementation can be found in [8]. They divide the problem into two parts: first a rough depth map is constructed using a segmentation based SAD window comparison, second a disparity refinement module identifies false matches and replaces them with new estimates. Hardware implementations of a trinocular disparity processor are limited. An implementation using the summation of SAD's from both image pairs can be found in [9].

This paper combines the strengths of an advanced stereo vision system with a two-scale adaptive window SAD incorporated in a trinocular setup.

2 System Overview

2.1 General Architecture

The trinocular disparity processor takes three images that have been taken by three cameras that have a vertical alignment and a horizontal offset (see Fig. 1). The objective is to calculate a disparity space image (DSI) where dark pixels represent a distance further away from the cameras and a light pixel represents a distance closer to the camera.

Objects will appear on the same horizontal line (the epipolar line) on all images. The horizontal distance between the same objects on the center image and the left (or right) image is called the disparity. If calibrated correctly, the disparity of an object between the center-left and the center-right image pair is the same. This characteristic can be used to discard false matches using bidirectional matching [2] or to improve the quality of the disparity space image (DSI) especially in occluded regions [1].

The architecture consists of three main blocks (see Fig. 2). The first block captures the pixel streams, generates the scaled images and places them in multiple parallel on-chip memories. The second block performs a pair wise window comparison of the different streams using a binary adaptable SAD cost aggregation [8]. The third block calculates its confidence for each data stream and selects the final disparity value.

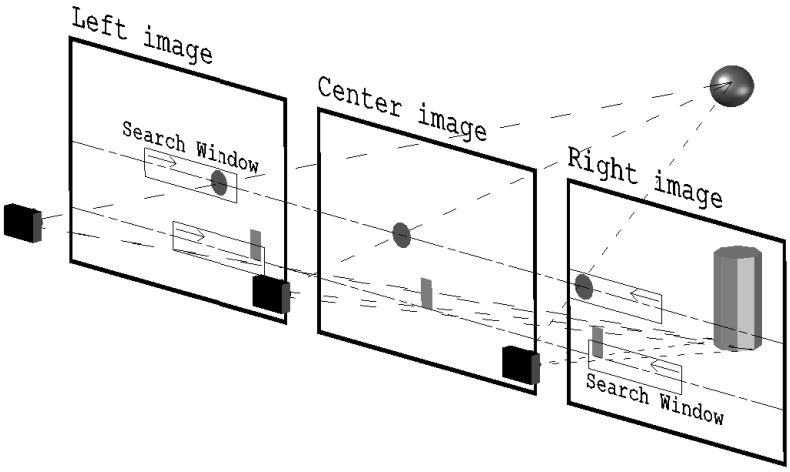

Fig. 1. Trinocular disparity processor setup

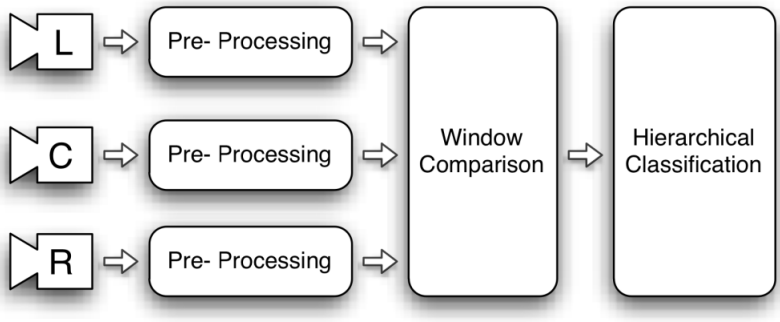

Fig. 2. Global architecture

On several places, this architecture makes use of a binary support window. When using a fixed window shape, depth continuity is implicitly assumed across this window. This assumption is false at depth edges where parts of the window belong to different depth levels. A more conservative assumption is to only assume depth continuity across pixels with similar color. Yoon [3] proposed an adaptive weight algorithm which gives a support weight to each pixel in a window. In order to save system on chip resources, an alternative has been proposed where the support weights are chosen as binary values [4]. A value of '0' means that this pixel doesn't belong to the support window of the center pixel and '1' means that this pixel belongs to the

support window of the center pixel. This is called the binary support window (1). It is calculated by taken the absolute difference of the chroma color components (C_B, C_R) of all pixels q belonging to the rectangular window centered in pixel p.

$$w = \begin{cases} 0 \; if(\; |C_B(q) - C_B(p)| + |C_R(q) - C_R(p)|) > threshold \\ 1 \; otherwise \end{cases} \quad (1)$$

This binary support window is used when comparing different windows with each other (2). Instead of comparing a complete window, only the pixels where the support window is '1' (white) will be taken into account (see Fig.3).

$$SAD = \sum_{i=1}^{window\ size} w * \; absolute\ difference(i) \quad (2)$$

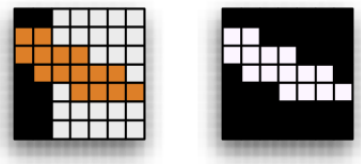

Fig. 3. Window content (left) and resulting binary support window (right)

Each window of the center image needs to be matched with multiple windows of the left or the right camera. For every window that needs to be matched, a SAD calculation is performed. The larger the disparity search width, the more SAD calculations are needed. The result is an array that contains a SAD score for each disparity value (usually starting from 0), this array is also known as the cost curve (see Fig. 4).

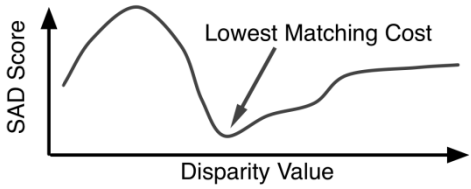

Fig. 4. Cost curve example

In this paper, C1 stands for the lowest SAD score (the minima of the Cost Curve). C2 stands for the second lowest SAD score, and so on. Their corresponding depths are indicated by D1 and D2. Most matching algorithms calculate the disparity from the cost curve using a "Winner Takes All" (WTA) approach. Doing so, the minima of the cost curve (C1) will become the calculated disparity D1.

In this architecture nine different cost curves are calculated for each pixel in the DSI. The first step is to calculate the cost curve the center-right (SAD_{CR1}) and center-left (SAD_{CL1}) image pairs as their scaled down image pairs (SAD_{CR0} and SAD_{CL0}). The second step is to calculate the summation of these cost curves (3).

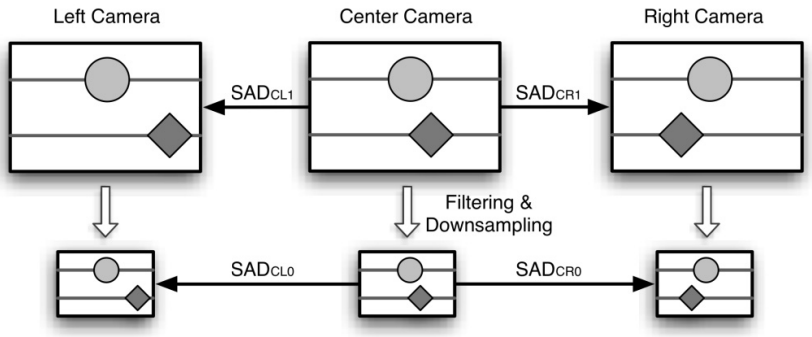

Fig. 5. Different window matching

$$\begin{cases} SAD_{CL1}, SAD_{CR1}, SAD_{CL0}, SAD_{CR0} \\ SAD_{CLR1} = SAD_{CL1} + SAD_{CR1} \\ SAD_{CLR0} = SAD_{CL0} + SAD_{CR0} \\ SAD_{CL01} = SAD_{CL1} + SAD_{CL0} \\ SAD_{CR01} = SAD_{CR1} + SAD_{CR0} \\ SAD_{CLR01} = SAD_{CLR1} + SAD_{CLR0} \end{cases} \tag{3}$$

3 Hierarchical Classification

In the previous section it is explained that nine disparity values are generated for each pixel (3). In order to select one of them for generating the DSI, a two level hierarchical classifier is constructed (see Fig. 6). In the first level of the hierarchy, the disparity values are investigated independently of each other. For each disparity value a binary confidence classifier is constructed using the methods presented in [6]. These confidences are passed on to the second level classifier which selects the disparity to use, or indicates that no disparity has been found.

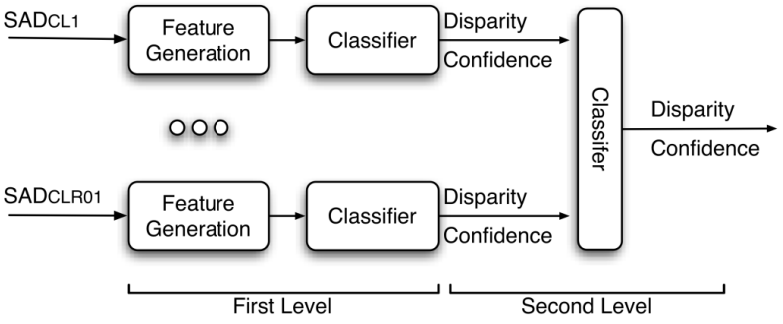

Fig. 6. Hierarchical classification

For each level of the hierarchy, a different set of features is needed for classification. The first level of classifiers uses information obtained from the pixel neighborhood and from its corresponding cost curve. The second level classifier uses the generated binary confidence values together with the agreement between the different disparity values. A binary confidence value of '1' indicates a strong confidence in the correctness of the disparity value. A binary confidence value of '0' indicates a weak confidence in the correctness of the disparity value.

Three different datasets have been used to verify the results:

- Tsukuba [10]: 384 x 288 (Maximal disparity of 30).
- Teddy [11]: 450 x 375 (Maximal disparity of 30).
- Art [12]: 695 x 555 (Maximal disparity of 30).

In order to train a classifier, it is needed to define a target output. In this case, the preferable output would be a Boolean value indicating the correctness of the disparity value (the confidence value). A pixel is defined to be correctly matched with its corresponding disparity when the calculated disparity (D_c) and the real disparity (D_r) do not differ more than one unit disparity value (4, 5).

$$Error\ map(i) = \begin{cases} 1\ if\ D_c\ (i) \notin\ [D_r(i), D_r(i) \pm 1] \\ 0\ otherwise \end{cases} \tag{4}$$

$$Error_{Th1} = \sum_{i=1}^{image\ size} Error\ map(i) \tag{5}$$

3.1 Feature Generation

The features for the first level of classification are proposed in [6]. Their objective lies in accommodating the classification of the disparity stream for the first level of classification.

The matching cost (MC) is the minimum value of the cost curve. A high score will be a good indication of a wrong depth value.

$$MC = C_1 \tag{6}$$

The texture (TEX) uses a fixed window of color information (C_i) around the investigated pixel and measures the amount of texture it contains. The intuition behind it is that textureless regions will provide more incorrect depth values.

$$TEX = max_{window}(C_i) - min_{window}(C_i) \tag{7}$$

The segmentation size (SEG) calculates the sum of the binary support window (1). This binary support window can be the same as the one used in the cost aggregation phase.

$$SEG = \sum_{i=1}^{window\ size} w \tag{8}$$

The following two features make use of neighborhood information of the disparity space image (DSI). In order to calculate them, a buffer is needed to store several lines

of the DSI. The size of this window depends on the size of the neighborhood and the width of the image.

The sum of neighboring depths differences (SNDD) uses a fixed window of depths around the investigated pixel and calculates the depth differences in this window.

$$SNDD = \sum_{i=1}^{window\ size} |D_1(i) - D_1(center)| \tag{9}$$

The sum of neighboring depths differences binary window (SNDDBW) is similar to SNDD, but instead of using a fixed window it uses only the neighboring pixels, which have a similar color. This is a different usage of the binary support window (1).

$$SNDDBW = \frac{\sum_{i=1}^{window\ size} w*|D_1(i)-D_1(center)|}{\sum_{i=1}^{window\ size} w} \tag{10}$$

The following features are designed for multi stream classification. They take the confidence value generated from the first level of classification and provide a feature which objective lies in accommodating the selection of the best disparity stream.

The sum of streaming depths differences (SSDD) calculates the depth difference between the different disparity streams taking the confidence value into account.

$$SSDD = \sum_{i=1}^{streams} confidence_i * |D_1 - D_1(i)| \tag{11}$$

The sum of streaming confidences (SSC) calculates the number of streams which have a positive confidence.

$$SSC = \sum_{i=1}^{streams} confidence_i \tag{12}$$

3.2 Classification Methods

The first level classifier consists of a decision tree (DT) for each disparity stream individually. The decision tree is a top-down tree structure consisting of internal nodes, leaf nodes, and branches. Each internal node represents a decision on a feature, and each outgoing branch corresponds to a possible outcome. Each leaf node represents a class (0 or 1 in this case). The main advantage of a decision tree is the ease of interpretation and implementation, while still being able to separate hard to separate classes. In the example of Fig. 7, two classes are separated by the class boundary which is constructed using a small DT.

The second level classifier chooses the final disparity from the different disparity streams by choosing the one with the lowest SSDD. A decision tree classifier is trained to construct a confidence value for the final disparity value.

Both classification methods are easily implemented in hardware without using many resources.

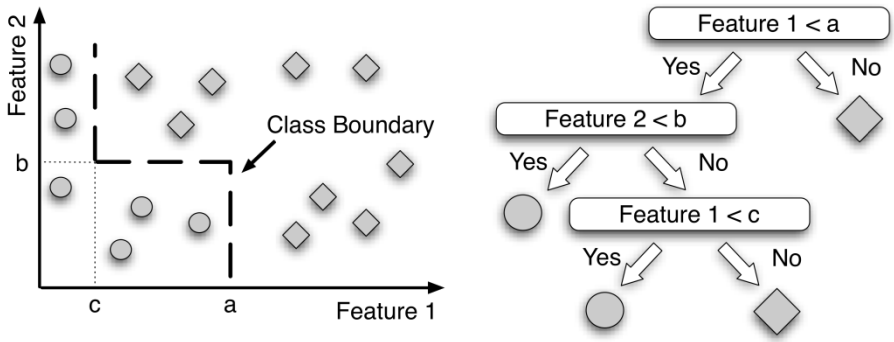

Fig. 7. Decision tree example: two-dimensional feature space (left) and resulting DT (right)

3.3 First Level Classification Evaluation

Matlab has been used to generate and classify the different features using five-fold cross validation. The dataset is split into five sets, where four sets are used for training the classifier and one set is used for validating the results. This is repeated five times, such that each of the five sets is used exactly once as validation set. The five validation results are averaged to generate the final result.

For each dataset, cost curves have been calculated using the SAD aggregation method for a binary adaptable window with four different selection thresholds and a window size of 7x7.

The features are indicated in the following table by their acronym and by the subscript of their parameters: e.g. $SNDDBW_{21,8}$ means SNDDBW with a window size of 21x21 and a Chroma threshold of 8.

The results of the first level classification can be seen in table 1. For every test, the feature is shown which is most important in the construction of the tree. From this table, we can see that the error rate between the different disparity streams is different; by summation of the SAD's a more correct DSI is constructed.

From Fig. 8 we can see that the different DSI's have different areas which are indicated as correct. The center-left image comparison (CL1) provides a good result across borders where the area on the left side of the border is further away then the area on the right side of the border. The center-right image (CR1) provides good results when the border is reversed. A summation of both SAD's (CLR1) gives a lower global error rate, but the borders are less clear.

As expected, the scaled down image pair comparisons (CL0 – CR0 – CLR0) provides better results on large texture less areas compared with the normal size image pair comparisons (CL1 – CR1 – CLR1). This is particular true for the background. However they have problems finding the correct disparity value for small objects, like the bars on the lamp. The summation of the SAD's generated by the normal size and the scaled downs image pairs (CL01 – CR01) gives a lower global error rate, keeps small details and has a better result with texture less areas.

Table 1. First level classification

Data Stream	*Data*	*Feature Name*	*Error Rate DSI (Th1)*	*Misclassification Binary Classifier*
CL0	Tsukuba	$SNDDBW_{21,8}$	27.30%	17.13%
	Teddy	$SNDDBW_{21,8}$	13.35%	5.35%
	Art	$SNDDBW_{21,8}$	19.65%	10.60%
CR0	Tsukuba	$SNDDBW_{21,8}$	29.43%	21.15%
	Teddy	$SNDDBW_{21,8}$	18.26%	6.12%
	Art	$SNDDBW_{21,8}$	19.27%	10.64%
CL1	Tsukuba	$SNDDBW_{21,8}$	22.98%	11.51%
	Teddy	$SNDDBW_{21,8}$	14.53%	13.39%
	Art	$SNDDBW_{21,8}$	23.74%	18.12%
CR1	Tsukuba	$SNDDBW_{21,8}$	25.05%	11.88%
	Teddy	$SNDDBW_{21,8}$	23.55%	16.25%
	Art	$SNDDBW_{21,8}$	24.45%	19.53%
CL01	Tsukuba	$SNDDBW_{21,8}$	22.92%	14.77%
	Teddy	$SNDDBW_{21,8}$	11.60%	6.05%
	Art	$SNDDBW_{21,2}$	19.11%	10.21%
CR01	Tsukuba	$SNDDBW_{21,8}$	24.83%	18.53%
	Teddy	$SNDDBW_{21,8}$	17.06%	6.83%
	Art	$SNDDBW_{21,2}$	18.99%	11.50%
CLR0	Tsukuba	$SNDDBW_{21,8}$	25.19%	20.06%
	Teddy	$SNDDBW_{21,8}$	14.02%	7.47%
	Art	$SNDDBW_{21,8}$	16.30%	10.13%
CLR1	Tsukuba	$SNDDBW_{21,8}$	22.40%	11.85%
	Teddy	$SNDDBW_{21,8}$	18.15%	21.59%
	Art	$SNDDBW_{21,8}$	19.12%	14.06%
CLR01	Tsukuba	$SNDDBW_{21,2}$	22.52%	16.67%
	Teddy	$SNDDBW_{21,8}$	14.17%	8.10%
	Art	$SNDDBW_{21,8}$	16.73%	10.45%

This indicates that by combining the DSI's, we could obtain a higher quality DSI. However before combining them, we need to know which part of each individual DSI is correct. A binary classifier is constructed to provide a confidence value for each DSI. The more correct this classifier, the more success we will have with the combined DSI. Depending on the dataset, a misclassification rate between 5% and 20% is obtained. This could be improved by using a more discriminative classifier like an artificial neural network [6].

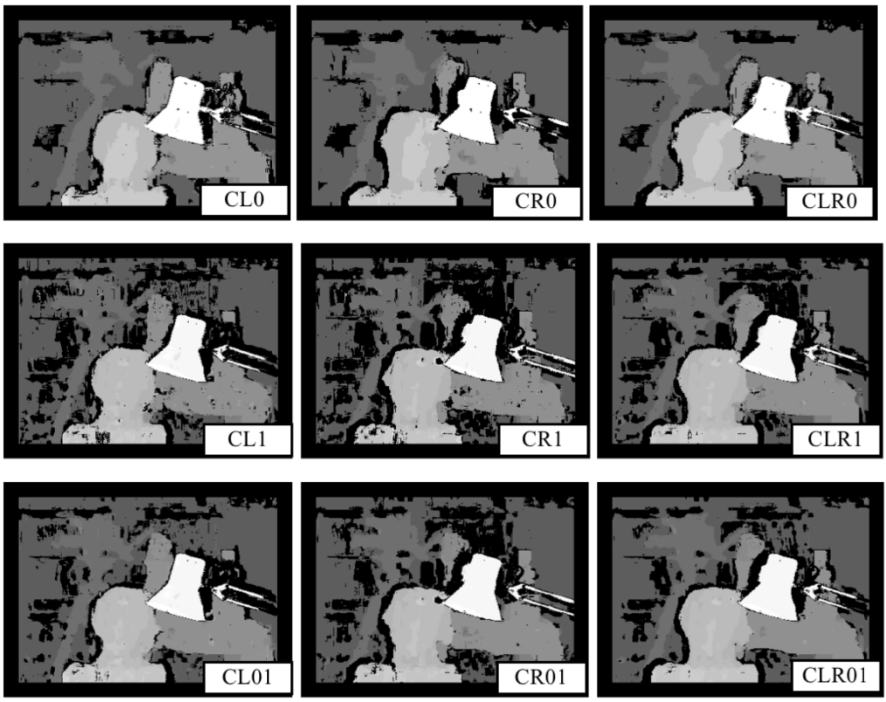

Fig. 8. Depth map quality of the Tsukuba dataset for a fixed window size of 7x7 after the first level of classification (black pixels indicate a confidence value of zero)

3.4 Second Level Classification Evaluation

The goal of this classification level is to select the most promising disparity value (13). The input of this classification level is the SSDD for each disparity stream. The output of this classification level is a disparity selection. A confidence value is afterwards generated using the same method as for each individual stream although using SSC as an extra input feature.

$$Disparity = \min_{i:1 \rightarrow streams} \text{SSDD (i)} \tag{13}$$

An exhaustive search is performed in order to know which combination of streams provides the highest disparity improvement. A selected set of results can be seen in table 2. The results indicate that, by combining extra streams, the classification rate for all investigated datasets are improved. From Fig. 9 we can see that the addition of extra streams improves the quality of the DSI. The trinocular setup improves the DSI most noticeably at occluded regions. The scaled image improves the disparity map at parts with little texture.

Table 2. Second level classification

Data Stream	*Data*	*Error Rate DSI (Th1)*	*Misclassification Binary Classifier*
CL1 - CR1 - ...	Tsukuba	16.50%	12.90%
CL01 - CR01 - ...	Teddy	7.82%	7.18%
CLR1 - CLR0	Art	11.86%	10.09%
CL1 - CR1 -...	Tsukuba	16.52%	12.47%
CL0 - CR0	Teddy	7.74%	6.71%
	Art	12.43%	11.50%
CL1 - CR1 - ...	Tsukuba	16.85%	12.57%
CL01 - CL01	Teddy	8.05%	7.23%
	Art	13.09%	11.38%
CL01 - CR01 - ...	Tsukuba	17.25%	13.22%
CLR1 - CLR0	Teddy	7.58%	6.47%
	Art	11.82%	10.99%

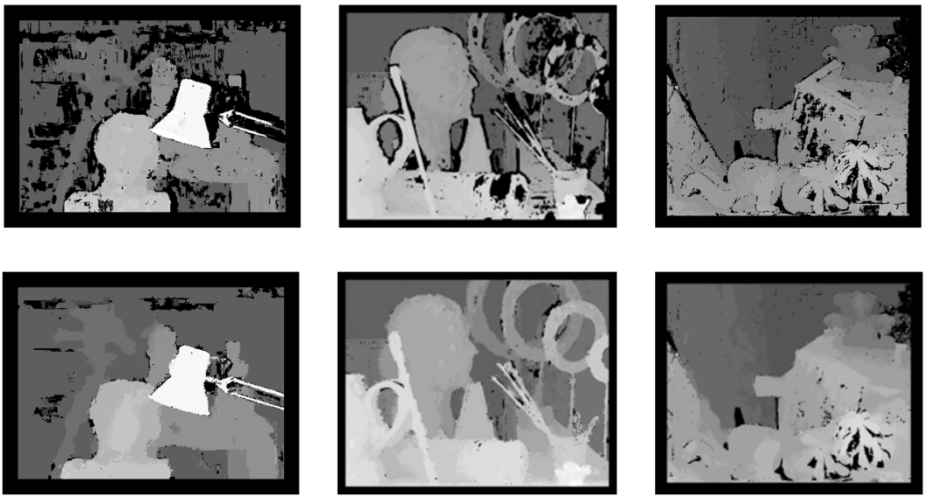

Fig. 9. Depth map quality of the investigated datasets (Tsukuba, Teddy, Art). Comparison of DSI generated from CL0 data stream (Top row) and DSI generated from the combination of CL0, CL1, CR0 and CR1 data streams (Bottom row).

4 System Design

The hardware architecture consists of three main modules. First a filter and sub sampling module has been added to the pre-processing module [8] so that a scaled image is generated with one-fourth the size of the original image. Second the window matching module is modified from [8] to allow for multiple data stream matching. Third a hierarchic classification module is constructed to select the most promising disparity from the different disparity results.

4.1 Pre-Processing Module

The pre-processing module (see Fig. 10) consists of four different entities for each pixel stream: first a Bayer demosaicing algorithm is used to reconstruct the color image, next a rectification module is used to remove lens distortion and perform trinocular calibration and lastly the image is filtered and down sampled to generate a scaled image.

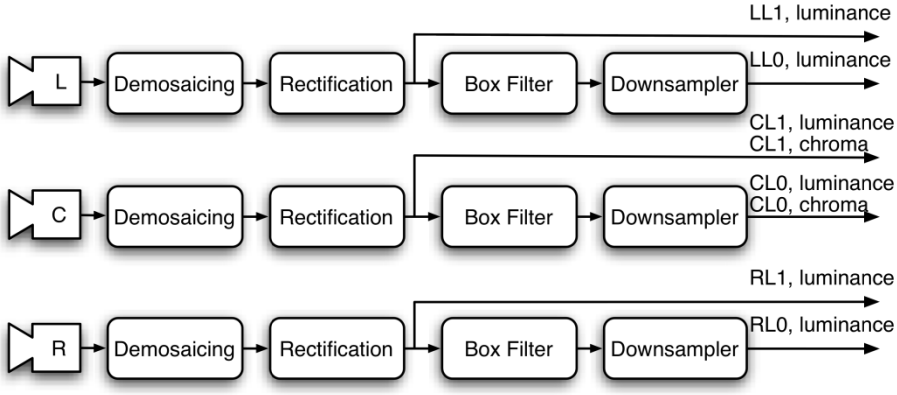

Fig. 10. Pre-processing module

Pixels generated by the camera are formatted in a Bayer pattern consisting of the four colors: Red (R), Green1 (G1), Blue (B) and Green2 (G2), representing the three color filters. The high quality linear interpolation demosaicing algorithm [13] is used to estimate the color components for each pixel.

The proposed architecture makes use of the YC_BC_R color space. The Luminance (Y) values are used to compare the two input streams. While the chrominance values (C_B, C_R) are used to construct the binary support window. Hence, the reconstructed RGB color space needs to be transformed into the YC_BC_R color space (14).

$$\begin{cases} Y = 16 + (66 \cdot R + 129 \cdot G + 25 \cdot B) \\ C_B = 128 + (-38 \cdot R - 74 \cdot G + 112 \cdot B) \\ C_R = 128 + (112 \cdot R - 94 \cdot G - 18 \cdot B) \end{cases} \tag{14}$$

Two different kinds of distortions are present in a trinocular camera setup. The first kind consists of the lens distortions; the second kind consists of the misalignment of the three cameras. Since the search space is only located on the epipolar line, both distortions should be resolved before matching can be performed. The intrinsic and extrinsic parameters of the cameras individually and the transformation matrix of the trinocular setup are determined offline using images of checkerboard patterns [14]. These parameters are hence used to construct the x and y mapping coordinates for each pixel in the image. Those parameters are called the reverse mapping coordinates.

The rectification module proposed in [15] consists of three main parts (see Fig.11). The reverse mapping coordinates are stored in a LUT. When the mapping coordinates

do not change drastically from pixel to pixel, it suffices to only store the mapping coordinates of certain pixels. These pixels are chosen to be located on a regular grid. The desired grid size depends on the amount of distortion in the image. Bilinear interpolation is used to reconstruct the mapping coordinates for the complete image. The warped image is constructed by selecting the pixels from the source image whose coordinates are provided by the reverse mapping LUT. In order to perform reverse mapping, the source pixels need to be stored in an input buffer. This input buffer is especially designed to keep the memory usage low so that no external memory is needed. Third, the output pixels are resampled in order to get sub-pixel accuracy.

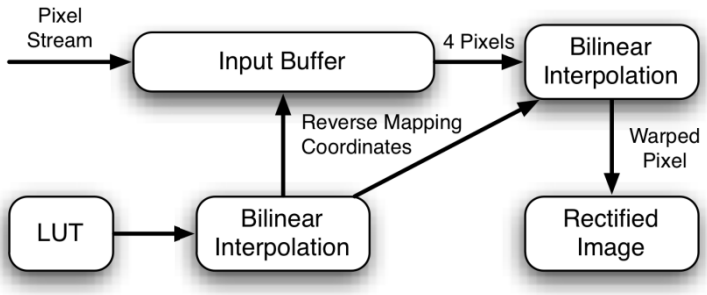

Fig. 11. Image rectification module [15]

The rectified pixel stream is passed through a 3x3 mean filter and down sampled by a factor of two. The original pixel stream is annotated with level 1 (L1) while the scaled pixel stream is annotated with level 0 (L0).

4.2 Window Comparison Module

The pixel streams originating from the right and left camera are compared with the center camera using segmentation based SAD calculation (see Fig. 12). During every clock cycle a window of the center camera is compared with four windows of the left or right camera. Since four successive pixels are stored in one memory location, one memory read accesses four pixels; hence four comparison modules are running in parallel.

On every clock cycle, the stream selection unit (SSU) determines where each data stream is written to and which windows are compared.

The frequency of the window matching module directly controls the possible disparity search width of the trinocular matching architecture and can be adapted to the available resources. The higher the frequency difference between the pixel stream and the window matching module, the more comparisons can be executed. In the example on Fig. 13 the window matching module is clocked twenty-four times higher than the pixel streams.

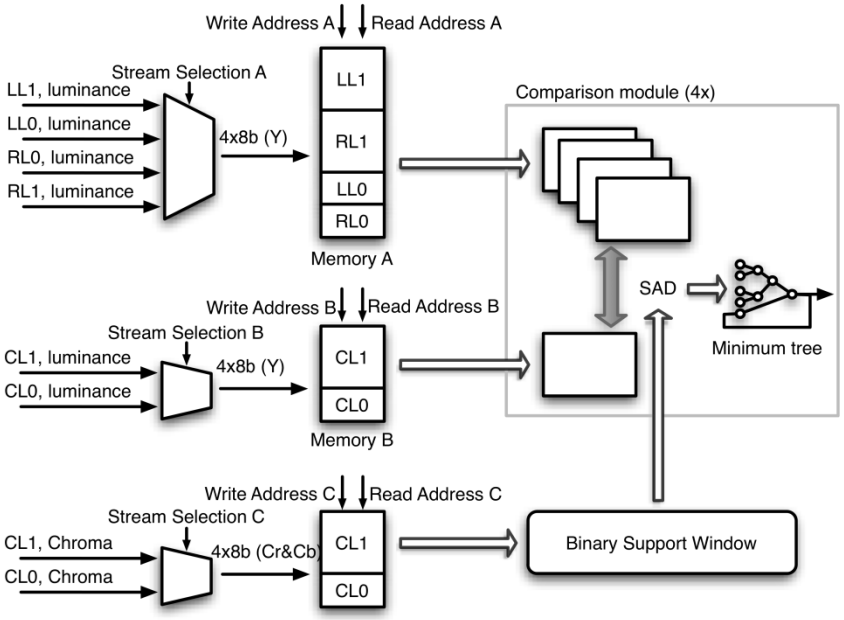

Fig. 12. Window comparison module

On each clock cycle, the comparison module compares the reference window with four consecutive windows (see Fig. 13). The lowest SAD score and its corresponding index are saved in a register, so that on the next clock cycle this lowest SAD score can be compared against the SAD scores of the next four windows. When the end of a search window is reached, the index indicates the disparity result and a new search window is initiated. In our example, in the first eight clock cycles, the center image is compared with the right image. In the next eight clock cycles the center image is compared with the left image.

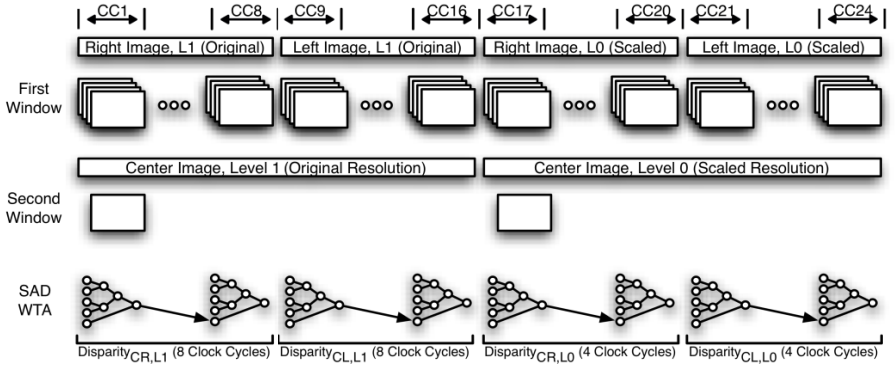

Fig. 13. Window comparison of different data streams

In the following four clock cycles the scaled center image is compared with the scaled right image and in the last four clock cycles the scaled center image is compared with the scaled left image. This leads to a combined disparity search width of thirty-two.

This architecture makes it possible to easily change the disparity search width and comparison data streams for each pixel in the DSI. By adapting the SSU it is possible to switch between a trinocular and a stereo disparity search. The trade-off is the disparity range; on each clock cycle, four comparisons can be performed. When using only two cameras, all clock cycles can be used for this camera pair. While with three cameras, only half of the clock cycles remain for each camera pair, this will lead to a reduction of the disparity search width.

4.3 Hierarchical Classification Module

The hierarchical classification module consists of the generation of the features used during the classification phase and the two classification steps. The first level classifier calculates the confidence of each stream in the selection (15, 16). For each stream, different thresholds are selected. However, the main structure of the classifier remains the same. The second level classifier selects the most promising disparity stream for the final DSI (17, 18, 19).

$$\text{streams} = \{\text{CL0,CL1,CR0,CR1}\} \tag{15}$$

$$\begin{cases} Conf_x = \ if\big(SNDDBW_{21,10} < a\big) then \\ [if(TEX > b) then\ 1\ else\ 0\ ]\ else\ 0 \\ x \in streams \end{cases} \tag{16}$$

$$\text{SSDD}_x = \sum_{y \in \text{streams}} \text{Conf}_y * \left|\text{D}_x\text{-D}_y\right|, x \in \text{streams} \tag{17}$$

$$Disparity\_selection = \min_{y \in \text{streams}} SSDD_y \tag{18}$$

$$Disparity = \text{streams}\ (Disparity\_selection) \tag{19}$$

The features are calculated at different moments in the streaming pipeline. For the first level of classification, three main timing zones have been specified.

- Zone 1: Features based on the luminance window e.g. TEX.
- Zone 2: Features based on the cost curve e.g. MC.
- Zone 3: Features based on the disparity window e.g. SNDD.

Features of different timing zones are expensive to synchronize. Each feature needs a buffer to temporally store its value. This is of particular importance when combining features from zone 1 and 3. Zone one features are calculated even before the calculation of the depth value, while zone three features are based on a window of disparity values around the disparity value of which the feature is calculated. This means that a delay of several image lines is to be expected.

A solution is to split the classification method into separate classification methods for each zone. For a decision tree classifier, no major changes are needed, each branch of the DT can be pre-calculated in a different zone (see Fig. 14). Since only the results need to be buffered, the width of the buffer is reduced to one.

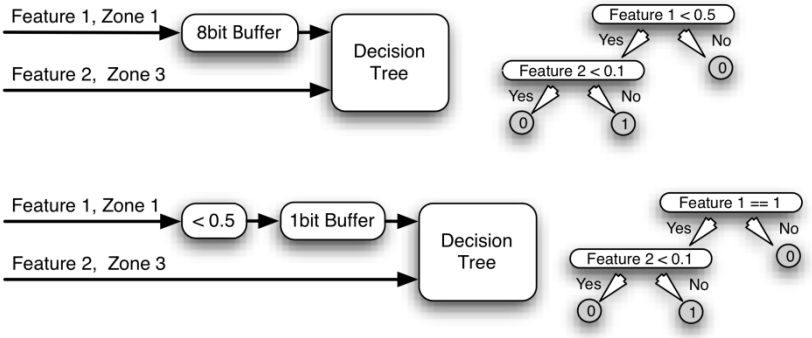

Fig. 14. Example of different timing zones for classification

5 Implementation

The architecture and methods presented in this paper have been implemented on an FPGA system (Terasic DE2-115 development board) , based on an Altera Cyclone IV (EP4CE115F29C7N) with 114,480 logic elements and 432 memory blocks. The sources of the input streams are three cameras with a resolution of 640x480 and a pixel clock of 16 MHz resulting in a refresh rate of 52 Hz. The current implementation consists of the proposed design using a 7x7 binary adaptive window SAD with a window matching clock of 96 MHz. The hardware block diagram can be found on Fig. 15.

The architecture has been constructed to reduce memory usage. Hence there is no need for external memories. The reduction of external memory usage has the additional advantage that the latency between input frame and output frame becomes minimal. This makes this system suitable to be incorporated in real-time control loops. In addition to the evaluation presented in section 2, the system has also been tested in real life environments.

The synthesis results can be found in Table 3.

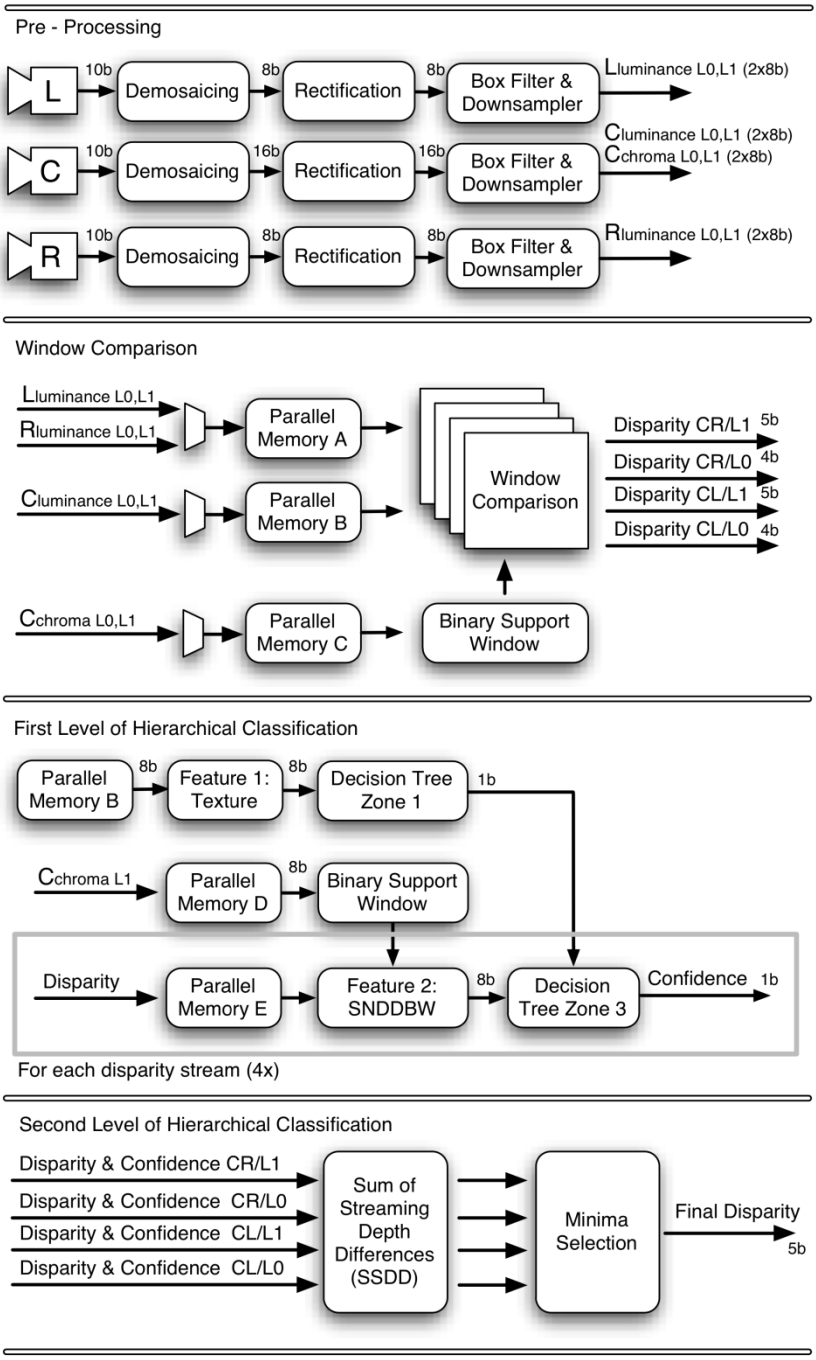

Fig. 15. Hardware block diagram

Table 3. Synthesis results overview

	Module Name	#	Logic Elements		Memory Elements			
					Nr of Blocks		Kilobits	
			Single	Total	Single	Total	Single	Total
Pre-Processing	Demosaicing	3	861	2,583	6	18	31	92
	Rectification	3	4,870	14,610	44	132	331	993
	Box Filter & Downsampling	3	688	2,064	5	15	28	84
Window Matching	Address & Stream Selection A	1	296	296				
	Parallel Memory A	1	659	659	22	22	119	119
	Address & Stream Selection B	1	340	340				
	Parallel Memory B	1	1,557	1,557	22	22	59	59
	Address & Stream Selection C	1	310	310				
	Parallel Memory C	1	661	661	22	22	59	59
	Binary Support Window	1	5,014	5,014				
	Comparison Module	1	20,893	20,893	1	1	0	0
Hierarchical Classification	Feature 1: Texture	4	854	3,416				
	Feature 2: Parallel Memory D	1	2,115	2,115	18	18	102	102
	Feature 2: Binary Suppport Window	1	5,675	5,675				
	Feature 2: Parallel Memory E	4	2,115	8,460	12	48	64	255
	Feature 2: SNDBBW (21x21)	4	8,350	33,400				
	Feature 3: SSDD & Minima Selection	1	212	212				
Total:				**102,265**		**298**		**1,764**

6 Conclusions and Future Work

A trinocular disparity processor has been proposed. We investigated nine cost curves resulting from pairwise comparison of three cameras. Each data stream has been investigated independently from one another and ultimately a hierarchic classification algorithm selects the most promising disparity value.

For each of the nine cost curves, a classification algorithm is trained in order to provide a confidence indication for their disparity values. These confidences are passed on to the second level classifier which selects the disparity to use, or indicates that no disparity has been found.

The selection of classification algorithms has been used as guideline for the implementation in an FPGA. From the results we can conclude that the quality of the disparity space image increases by using more cost curves from a trinocular camera.

Due to the adaptability of the window matching module and the hierarchic classification structure, the system can easily be expanded with more data streams to further improve the disparity space image.

References

1. Mozerov, M., Gonzalez, J., Roca, X., Villanueva, J.J.: Trinocular stereo matching with composite disparity space image. In: 16th IEEE International Conference on Image Processing, Proceedings IEEE ICIP 2009, pp. 2089–2092 (2009)
2. Ueshiba, T.: An efficient implementation technique of bidirectional matching for real-time trinocular stereo vision. In: 18th International Conference on Pattern Recognition, Proceedings IEEE ICPR 2006, pp. 1076–1079 (2006)
3. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Trans. PAMI 28(4), 650–656 (2006)
4. Motten, A., Claesen, L.: A Binary Adaptable Window SoC Architecture for a Stereo Based Depth Field Processor. In: Proceedings IEEE VLSI-SOC 2010, 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, Madrid, September 27-29, pp. 25–30 (2010)
5. Hu, X., Mordohai, P.: Evaluation of stereo confidence indoors and outdoors. In: Proceedings IEEE CVPR 2010, 23rd IEEE Conference on Computer Vision and Pattern Recognition, pp. 1466–1473 (2010)
6. Motten, A., Claesen, L., Pan, Y.: Binary confidence evaluation for a stereo vision based depth field processor SoC. In: Proceedings IEEE ACPR 2011, 1st Asian Conference on Pattern Recognition, Beijing, November 28-30, pp. 456–460 (2011)
7. Jin, S., Cho, J., Pham, X.D., Lee, K.M., Park, S.-K., Jeon, J.W.: FPGA Design and Implementation of a Real-Time Stereo Vision System. IEEE Transactions on Circuits and Systems for Video Technology 20(1), 15–26 (2010)
8. Motten, A., Claesen, L.: Low-cost real-time stereo vision hardware with binary confidence metric and disparity refinement. In: Proceedings IEEE ICMT 2011, International Conference on Multimedia Technology, pp. 3559–3562 (2011)
9. Li , M., Jia, Y.: Stereo vision system on programmable chip (SVSoC) for small robot navigation. In: Proceedings IEEE/RSJ IROS 2006, International Conference on Intelligent Robots and Systems, pp. 1359–1365 (2006)
10. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision 47(1), 7–42 (2002)
11. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: Proceedings IEEE CVPR 2003, IEEE Conference on Computer Vision and Pattern Recognition, pp. 195–202 (2003)
12. Hirschmüller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. In: Proceedings IEEE CVPR 2007, International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
13. Malvar, H., He, L., Cutler, R.: High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In: Proceedings IEEE ICASSP 2004, IEEE International Conference on Acoustics, Speech, and Signal Processing, May 17-21, pp. 485–488 (2004)
14. Zhang, Z.: Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In: Proceedings IEEE ICCV 1999, 7th IEEE International Conference on Computer Vision, Kerkyra, September 20-25, pp. 666–673 (1999)
15. Motten, A., Claesen, L., Pan, Y.: Adaptive Memory Architecture for Real-Time Image Warping. In: Proceedings IEEE ICCD 2010, 30th IEEE International Conference on Computer Design, Montreal, September 30-October 3, pp. 466–471 (2010)