Real-time embedded system for stereo video processing for multiview displays

R-P. M. Berretty, A.K. Riemens* and P.F. Machado

Philips Research Eindhoven, High Tech Campus, 5656AA Eindhoven, Netherlands

ABSTRACT

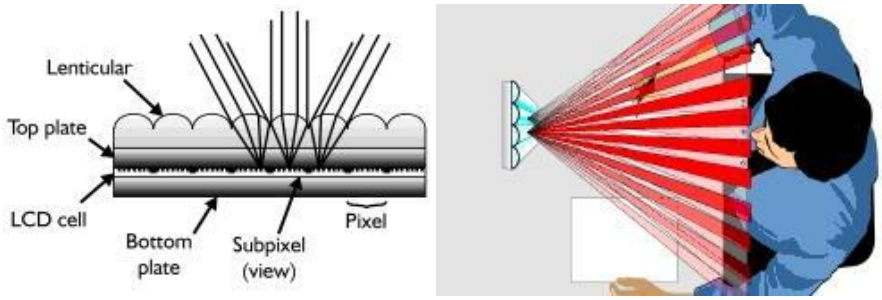
In video systems, the introduction of 3D video might be the next revolution after the introduction of color. Nowadays multiview auto-stereoscopic displays are entering the market. Such displays offer various views at the same time. Depending on its positions, the viewers' eyes see different images. Hence, the viewers' left eye receives a signal that is different from what his right eye gets; this gives –provided the signals have been properly processed– the impression of depth.

New auto-stereoscopic products use an image-plus-depth interface. On the other hand, a growing number of 3D productions from the entertainment industry use a stereo format. In this paper, we show how to compute depth from the stereo signal to comply with the display interface format. Furthermore, we present a realisation suitable for a real-time cost-effective implementation on an embedded media processor.

Keywords: image-plus-depth, stereo to depth conversion, real-time, disparity estimation

1. INTRODUCTION

To enable stereoscopic depth perception, several technologies have been suggested that allow different images to be provided to each eye. Most of them use special glasses†. Auto-stereoscopic displays take a different approach: a different image is provided to each eye without the need of special eye wear. Philips has developed a product line of multiview auto-stereoscopic 3D displays.[1,2] The display offers its different views through the use of a film of lenses in front of the screen. See Figure 1 for details. Other examples of auto-stereoscopic displays are Sharp two view displays,[3] and Opticality barrier displays.[4]



(a) (b)

Figure 1. *Left figure shows a* multiview *LCD. Special attention is given to the lens system. Right hand figure shows how views are spatially distributed on a multiview screen.*

Philips' *multiview* auto-stereoscopic displays can show a high number different views (e.g. 9) at a given moment in time. This fact not only creates a stereoscopic effect but also allows the user to experience horizontal parallax, by moving his head horizontally. Figure 1 *(b)* shows how the different views are spatially distributed.

In general, current multiview displays strongly differ the number of views, size and depth reproduction capabilities. Therefore, a display indepent interface format is required.

*Currently affiliated at Research at NXP Semiconductors, Eindhoven, Netherlands

†Red-blue or shutter glasses, for example.

Stereoscopic Displays and Virtual Reality Systems XIV, edited by Andrew J. Woods, Neil A. Dodgson, John O. Merritt, Mark T. Bolas, Ian E. McDowall, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6490, 649014, © 2007 SPIE-IS&T · 0277-786X/07/$18

The preferred display independent input for these displays is image-plus-depth.[5–7] Being independent of specific display properties, such as number of views, view mapping on pixel grid, etc., this interface format allows optimal multiview visualisation of content from many different sources, while maintaining interoperability between display types. To be more specific, the image-plus-depth format is an extension of the conventional video format. It specifies for each sample point in the input images not only information about the color, but depth related data as well.

Recently, a growing number of new 3D productions from the entertainment industry are aiming at 3D movie theatres. These productions use a stereo format, primarily intended for eye-wear assisted viewing. Stereo material comes in a variety of transmission/storage formats.[8] The two most widely used formats are 'anaglyph' and 'field or frame sequential'. If one would like to view these stereo productions on a multiview display, a conversion step is required.

In order to convert a stereo production, so called disparity estimation[9] needs to be done on the stereo pairs. For most disparity estimators to work properly, a scan line of the left image must correspond to the same scan line in the right image. However, practical recordings pose several difficulties. Firstly, the left and right image can be shot at different time instances.[10] Secondly, stereo pairs are not always aligned, e.g., due to camera misalignment.

To solve the aformentioned difficulties, disparity estimators would require a tedious rectification step. A common approach for rectification involves finding correspondences between feature points in both images and robustly estimating the transformation (fundamental matrix) from the set of correspondences. Often robust estimators like RANSAC[11] or more recently NAPSAC[12] are used. However, Gluckman and Nayar[13] show that careful consideration must be given to the rectification process. A non-optimal rectification can lead to bad disparity matching results. Furthermore, a disparity field estimated on a rectified pair of images must be transformed back to an original image (e.g. the left image) in order to be used for rendering new views. Hence, the transformation must be applied twice.

Recently, Braspenning and Op de Beeck[14] showed how to use scan rate conversion techniques to compute multiple views from uncalibrated stereo material. They render the multiple views without explicitly computing a depth map.

In this paper, we show how to compute *depth* from an uncalibrated stereo signal. Like Braspenning and Op de Beeck, we use techniques derived from scan rate conversion systems. We will present a method to estimate the depth from a stereo input, without the need of a calibration prior to disparity estimation. As a main advantage, our output is the aforementioned display independent image-plus-depth format. Moreover, we present a realisation of our algorithm that is suitable for a real-time cost-effective implementation on an embedded media processor.

In the following we will first present our robust depth estimation algorithm, and subsequently how we mapped our algorithm on the programmable core.

2. ALGORITHMS

2.1. Motion estimation approach

Over more than a decade, Philips has developed products for video scan rate conversion applying motion estimation and motion compensated temporal interpolation.[15,16] This functionality is known in the television market as Natural Motion™. We leverage this experience by applying the *3D recursive search*‡ algorithm[15] to estimate disparity between the left and right image. We have choosen this approach for the following reasons:

- It is an industrially proven approach that delivers real-time high quality true-motion motion vectors at low implementation cost.
- By applying motion estimation, the algorithm is robust for uncalibrated input signals. Hence, there is no need for rectification transformations.
- Existing compute platforms are targeted to a.o. scan rate conversion applications. Therefore, these platforms are also suitable for the disparity estimation algorithm described in this paper. So we expect that the step from algorithm design to productization is relatively small, enabling fast time-to-market.

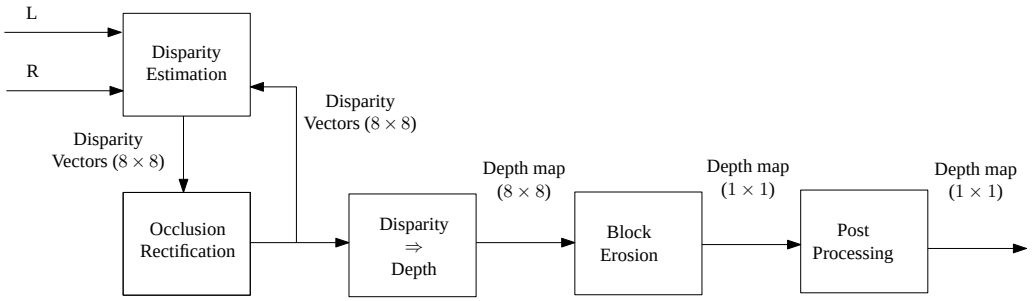


Figure 2. *Depth estimator block diagram*

2.2. Block diagram

Figure 2 shows the block diagram of the depth estimator. The following steps can be distinguished.

Disparity estimation Disparity estimation, using the 3D recursive search algorithm. The left and right inputs are the luminance of the two images. Since this is a block based estimation algorithm, the output of the estimator consists of disparity vectors where each vector is valid for an 8x8 pixel block.

Occlusion rectification The disparity estimator matches pixels from left- and right-hand images. In occlusion areas however, part of the image data is e.g. covered by foreground objects in one of the images. Therefore, the estimator performs weakly in such areas. Based on the known left-to-right relation between the input images, a correction in such areas is carried out. This process is called "occlusion rectification". The improved vector field is fed back into the disparity estimator as previous vector field when processing the next input image pair.

Disparity to depth conversion This step converts a disparity vector into a depth value. First, the x-component of the vector is taken. Then it is scaled to a proper depth signal range. After this step, the result is a depth map, where each depth value is valid for an 8x8 pixel block.

Block erosion The resolution of the depth map is increased to meet the pixel grid in a process called "block erosion".

Post processing In the post processing stage, the alignment between the depth map and the image content is further improved in order to enhance the accuracy and the stability of the depth map. This is subject to ongoing research, therefore it is not extensively described in this paper. Preliminary results are shown in Section 4.

The following sections describe each of these steps in more detail.

2.3. Disparity Estimation

The 3D recursive search algorithm is based on two assumptions[16]:

1. Objects are larger than a block.
2. Objects have inertia.

As a result, the disparity vector for part of an object has usually been calculated before, either in the neighborhood of the current image part, or in the past. Consequently, correct disparity vectors can usually be found by applying a very small set of carefully selected candidate vectors from this spatial or temporal neighborhood. The process to calculate a disparity vector for a current block of pixels is summarized as follows. First the disparity vectors from some of the blocks in the previous image and some surrounding blocks are taken as candidate disparity vectors. Then, each of these candidate vectors is applied on the current pixel block and tested against a match criterion. Finally, the best matching candidate

[‡]Here, "3D" refers to horizontal, vertical and temporal axes from which candidate vectors are obtained. It is unrelated to the "3D" displays that show horizontal, vertical and depth dimensions.

vector is selected as result and assigned to the current block of pixels. To cope with convergence and changing disparities in the scene, a pseudo random update vector is added to some of the candidate vectors.

The algorithm is characterized as follows.

Block size The block size is chosen as 8x8 pixels. This choice proved already since years to be a good compromize between resolution of the vector field and uniqueness of image details.

Match criterion The SAD (sum of absolute differences) is used as match criterion. Also this calculation is already used for years to obtain a good correlation measure at low computational cost.

Vector resolution The resolution for disparity vectors for high quality scan rate conversion is $\frac{1}{4}$ pixel. This implies that sub-pixel vector positions are applied, requiring interpolation of pixel values to calculate image data at the vector position. For the disparity estimator, we found that $\frac{1}{4}$ pixel accuracy in the horizontal direction is beneficial for correctness and smoothness of the disparity map. In the vertical direction however, the vectors are only used to compensate for misalignment of the two input signals. Therefore, an integer accuracy proved sufficient. This simplification results in significantly reduced computational cost because vertical pixel interpolation is not required.

Vector range In a typical scan rate conversion implementation,[15] maximum vector ranges are $[-32 \ldots +32]$ in the horizontal direction, and $[-16 \ldots +16]$ vertically. For the disparity estimation, we found significant different ranges. Horizontal disparities range up to 80 pixels. Vertical disparities are only used to compensate for misalignment of the input data. We found on our test set that a range of $[-6 \ldots +6]$ is sufficient. We considered to avoid any vertical vector component and only allow pure horizontal vectors. However, we found that this significantly detoriates the quality of the vector field. Hence, the vertical vector component is considered essential to compensate for camera misalignment, but its accuracy is not critical.

Random updates Camera misalignment only gradually varies depending on the position in the image. Therefore, neighboring vectors are almost always correctly compensated for camera misalignment. Combined with the few vertical values, we found that only very few random updates need to be applied vertically. This leaves room for substantially more random updates in the horizontal direction. Hence, the distribution of the random update vectors is well tuned to the characteristics of the input signal and the required accuracy of the vector field. In this area, the choices are clearly different for disparity estimation compared to motion estimation for scan rate conversion.

Scanning directions The algorithm is performed by blockwise scanning of the image. New vectors propagate easily in the scan direction. As a side effect, the scanning direction sets the causality constraints for the location of the candidate vectors. After all, a candidate vector must be calculated before it can be used. Scanning is block-wise zigzagging from top to bottom over the image. In successive passes, the scanning starts in alternating at the top or at the bottom of the image.

Candidate set The candidate set consists of 5 candidate vectors when scanning left-right and top-bottom.

1. Two spatial candidates: this takes vector values from one block above, and from the left hand neigbor of the current block.
2. One temporal candidate: this takes the vector value from the position two blocks to the right and two blocks down the current block.
3. Two update candidates: these are derived from the spatial candidate vectors by adding pseudo random update vectors.

When the scanning direction reverses, candidate vector block positions mirror.

Summarizing, the principle of operation is similar to scan rate conversion application. However, by making specific adaptations, the algorithm is tuned to disparity estimation.

2.4. Occlusion rectification

Parts of one image of a stereo pair can be hidden in the other image. This is where foreground objects occlude the background. Figures 3 (a) and (b) highlight the occluded areas in a stereo pair.

In the disparity estimation, block disparities are found in the original stereo pair. The challenge is to find the correct disparities for the occluded areas. A simple solution could be the assignment of background vectors when those blocks are occluded on the corresponding image.

However the 3DRS algorithm just outputs the local best match using the SAD match criterion. As a consequence, these occluded areas might be filled with the wrong disparities, resulting in visible artefacts.

Figure 3. *Occlusion areas are highlighted in images (a) and (b). Image (c) shows the depth map for image (a) after block erosion.*

Let us analyse what happens with blocks of the left image that are occluded in the right image. First, the disparity estimator will find the best match in the right image. Since this block is occluded, we can deduce that it must be behind a foreground object, and hence be part of the 'background' in the left image. Consequently, best match according to the SAD criterium is likely a part of the right image that contains background texture. Second, there is a block in the background of the left image that is *not occluded* will match to the *same part* of the background of the right image. Figure 4 shows this situation as case *(a)*.

It should be decided which one of the vectors from the left image correspond to occluded image data. We make the decision based on the SADs associated with the matches. The lower SAD values identify the correctly matched background from the left image. The higher SAD values identify the occluded background parts.

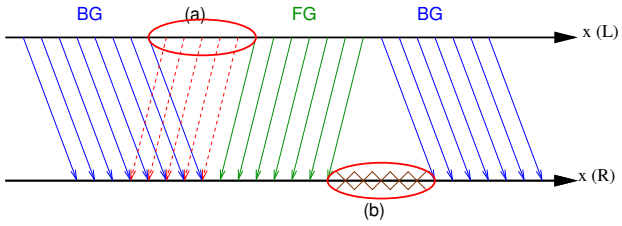


Figure 4. *Matching process when there is no matching information in the corresponding image. Arrows denote vectors for matching blocks from left to right.*

After successful occlusion identification, we need to assign the correct disparity vectors to occluded blocks. We tried several options and it turned out that straightforward background vector extrapolation produces the best results. It produces sharp transitions, which increase the depth perception. However, it can generate visible artifacts, if the occlusion areas are not very well aligned with object borders. Those situation are resolved in the block erosion and post processing stages.

2.5. Vector to depth conversion

The disparity estimator generates a vector field with the block disparities for all blocks. However, for the image to be rendered for 3D screens, depth values are required.

To derive depth values from disparity vectors, we first discard the y-component of the vectors. Next we need to convert the range of disparities to an 8-bit value. To find the optimal conversion, we use a dynamic calculation since the range of disparities can vary over shots. The parameters can adapt to local parts of the sequence, leading to a more effective use of the depth range. Let $\mathcal{X}$ denote the domain of image coordinates, and $D_x(\vec{x}, n)$ the x-coordinate of the disparity vector of frame n at image coordinate $\vec{x}$. Then, the following equations show how the depths $d(\vec{x}, n)$ are derived from the disparity map:

$$M = \frac{1}{c} \sum_{i=n-c+1}^{n} \max\{D_x(\vec{x}, i) | \vec{x} \in \mathcal{X}\}, \;\; m = \frac{1}{c} \sum_{i=n-c+1}^{n} \min\{D_x(\vec{x}, i) | \vec{x} \in \mathcal{X}\}$$

$$\text{gain} = \frac{255}{M - m}, M \neq m$$

$$d(\vec{x}, n) = \text{gain} \cdot (D_x(\vec{x}, n) - m)$$

where c is the number of frames used in the depth range calculation.

2.6. Block erosion

After vector to depth conversion, we have a map with depth values for each 8×8 block of the left image. In this section, we show how to derive a pixel dense depth map. An effective solution to compute pixel dense maps is *block erosion*.[16] It is a recursive process executed in three steps.

In each step, each block is subdivided into four sub blocks. The depth value of a sub block is computed using *median* operations over the values in its neighboring blocks as shown in Figure 5.

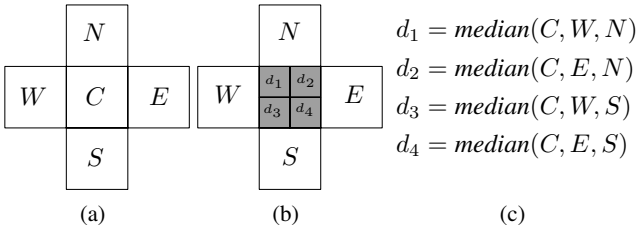


Figure 5. *Block erosion step. (a) Blocks at the lower resolution. (b) Subdivision of the center block into four sub blocks. (c) Median operations to compute the depth values of the sub blocks.*

Hence, in each step the resolution of the depth map is increased by a factor of four, leading to a pixel dense depth map after the three steps. Figure 3 (c) shows the resulting pixel dense depth map after block erosion.

3. REALISATION

After initial algorithm design, the software was mapped and optimised for a TM3270 embedded media processor.[17] This process involved algorithmic adaptations and extensive performance (cost) versus image quality evaluation. At the time of algorithmic design, no silicon was available yet. So computational load was determined using cycle accurate simulation, whereas image quality is determined by visual inspection of the results. This section elaborates on architectural elements of the processor that are of particular interest to the problem at hand and on the interactions between the algorithms and the architecture.

3.1. TM3270 characteristics

The TM3270 embedded media processor is developed by NXP semiconductors. This embedded processor is available in the PNX4103 IC[18] and is scheduled for application in other products.

The main characteristics of the TM3270 can be summarized as follows.

- VLIW architecture with 5 issue slots, allowing up to 5 RISC-like operations to be executed at every processor clock cycle.

- Fully pipelined architecture, with pipeline depth of 7-12 cycles, depending on the particular operation.
- Data width and address width are 32 bits.
- Unified register file with 128 general purpose registers of 32 bits.
- Hardware support for IEEE-754 floating point operations.
- Support for standard operations with two operand and one result register, but also for "super operations" that occupy two issue slots and allow operations with four operand and two result registers.
- 64 kbyte instruction cache, 128-byte lines, 8-way set associative, LRU replacement policy.
- 128 kbyte data cache, 128-byte lines, 4-way set associative, LUR replacement policy, allocate-on-write miss policy.
- Specialized SIMD vector instruction set with support for 4 x 8 bit, 2 x 16 bit vectors and filtered load instructions.

The extensive instruction set of the processor is the most clearly visible performance tool for the application programmer. It can be summarized as:

Multimedia operations An extensive set of specialized operations provides support for multimedia applications.

Saturated arithmetic For many operations, automatic clipping is provided on over- or underflow conditions. This avoids the need for explicit checking and clipping operations.

SIMD operations The SIMD vector operations allow processing of up to four data values in a single operation. This is particularly effective when operating at 8 bit video pixel data.

Filtered loads A new feature in the TM3270 core is the possibility to apply FIR filtering during the load instruction. In a single operation, five bytes are loaded from memory, a linear interpolation is carried out, and the four byte result is saved into a 32 bit register.

The first design of this processor is realized in 90 nm standard CMOS technology and occupies $8.1\mathrm{mm}^2$. Typical supply voltage is $1.2V$, but for low power applications (requiring lower computational load) this can be reduced to $0.8V$. At nominal supply, the power consumption is typically $0.935\frac{\mathrm{mW}}{\mathrm{MHz}}$. The core runs at a clock speed of $350\mathrm{MHz}$.

3.2. Computational load

For embedded systems, the computational load is a prime cost factor. As a first step, we optimized the generic ANSI-C code to the TM3270 instruction set by applying the appropriate custom multimedia operations. Since the actual processor load is only known with sufficient accuracy after optimization, this is also the development phase where algorithmic complexity can be balanced against the computational cost. This assessement led to several trade-offs, which are discussed below.

The computational load is measured on a cycle accurate simulator, since at the time of this project, no silicon was available. Furthermore, the simulator shows in-depth performance information at a detail level that is not readily available when executing a program on the chip. The simulation ran at a 4:7 memory-processor clock ratio, to simulate a processor clock of 350 MHz with memory at 200 MHz. The simulation only incorporates a simplified system model, consisting of the CPU, a memory bus and the background memory. Since no other bus clients are taken into account, bus requests are always immediately granted. In reality, other bus clients also require bus capacity. Therefore, the current simulation results may be too optimistic with respect to data and instruction cache stall cycles.

During the optimization process, several trade-offs became apparent. First of all, we considered subsampling the image data when computing the SAD in the disparity estimator. By subsampling, only one of every two pixels in the block is used, the other pixel value is simply ignored. Although this pixel dropping introduces aliasing, the quality of the disparity field is hardly affected due to the inherent spatial consistency of the 3DRS algorithm. Horizontal subsampling proved to match the instruction set very well, resulting in a reduction of 38% of computational load. The disparity estimator runs at approx. 20 cycles per block match (including vector construction, address calculations, $\frac{1}{4}$ pixel accurate data loading and SAD calculations).

As second trade-off, we experimented with the number of invokations of the disparity estimator on each input image pair. More scans result in faster convergence of the estimator. We found that two scans is a good design point, balancing fast convergence with reasonable computational cost. Also note that the two scans are in alternating directions, so there is no preference for the direction in which a change propagates on a single image pair.

Function	kcycles	%
Disparity estimation (2 scans)	1218	74%
Occusion rectification and depth conversion	84	5%
Block erosion (3 steps)	346	21%
Total	1648	100%

Figure 6. *Processor load breakdown showing the various processing steps after code optimization when calculating a single image pair of 720x480 pixels.*

Figure 6 shows the processor load of the various processing steps. The code of the "occlusion rectification" and "disparity to depth conversion" steps (section 2.2) is merged into a single function and therefore listed as a single value. The table shows the results for a single image pair of 720x480 pixels, operating on the luminance values. Issue slot filling is approximately 80%, which is considered a good result.

When operating on 30 frames per second, this results in a total of $30 \cdot 1.648 = 50$ MHz.

4. DISCUSSION AND RESULTS

In this paper, we have presented an algorithm to compute depth maps from stereo input images. The resulting depth maps can be computed in real time on an embedded platform, and can be used on multiview 3D displays. The depth maps are derived from a block based disparity field, and refined by block erosion. Consequently, the pixel dense depth maps are not optimally aligned with objects in the images. To improve this, we have applied additional object based postprocessing, see also.[19] The result, after postprocessing is shown in Figure 7. From this illustration, it can clearly be observed that the boundaries in the depth map are nicely matched with the image. As consequence of the alignment of the depth map to object borders, the visual perception of the result on the 3D display is excellent.

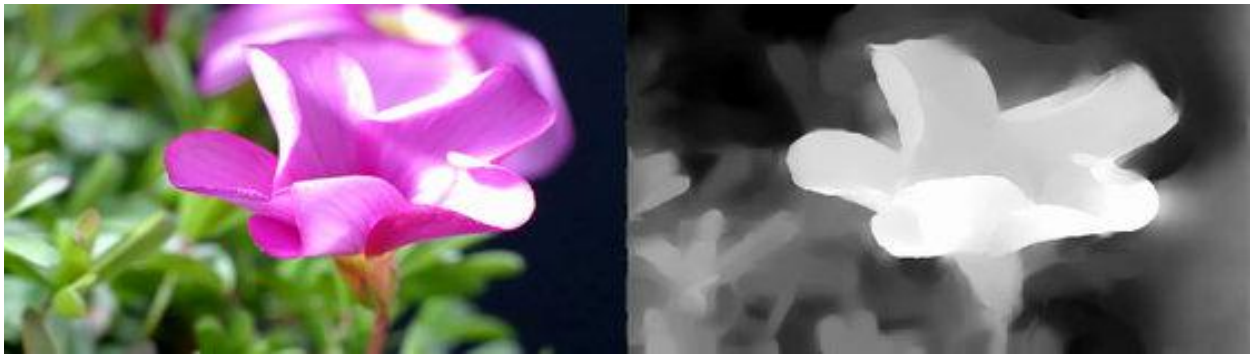


Figure 7. *The result of image based filtering: (left) the original left image and (right) the resulting depth map*

5. CONCLUSIONS

In the paper, we presented a complete system for computing depth from uncalibrated stereo video signals. We have also included novel algorithmic improvements. We illustrate excellent depth map quality. Processing standard definition input signals (i.e. 2 x 720x480@30p), we consume only 50 MHz of the available 350 MHz of a single embedded core. We expect that the complete system, including the postprocessing, will run in real time as well. Therefore, this system is suitable as real-time, cost-effective realisation for future products.

Acknowledgements

We thank Bart Barenbrug, Ralph Braspenning and Chris Varekamp for shaping our thoughts. We thank Eric Funke, Om Prakash Gangwal, and Erik van der Tol for their comments on draft versions of this paper. Furthermore, we would like to thank our 3D colleagues within Philips for providing a stimulating collaborative work environment.

REFERENCES

1. "Philips lenticular autostereoscopic 3d display `http://www.philips.com/3dsolutions/`."
2. C. van Berkel and J. Clarke, "Characterisation and optimisation of 3d-lcd module design,," in *Proc. SPIE*, **3012**, pp. 179–186, 1997.
3. "Sharp two view barrier display `http://www.sharp3d.com/`."
4. " Opticality multiview barrier display `http://www.opticalitycorporation.com/technology/technology_core.%html`."
5. C. Fehn, "Depth-image-based rendering (DIBR) compression and transmission for a new approach on 3D-TV ," in *Proc. Stereoscopic Displays and Applications*, 2004.
6. B. Barenbrug, "3D throughout the video chain," in *Proceedings of Int. Congress of Imaging Science*, pp. 366–369, 2006.
7. W. Bruls, "Enabling introduction of stereoscopic 3D video: compression standards, displays and content generation." Submitted for publication at ICCE 2007.
8. L. Lipton, "Stereo vision formats for video and graphics." http://www.stereographics.com/support/body_stereo_formats.html.
9. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal on Computer Vision* **47(1/2/3)**, pp. 7–42, April-June 2002.
10. NuView Camcorder Adapter. http://www.i-glassesstore.com/nuview.html.
11. M. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**, pp. 381–395, 1981.
12. D. Myatt et al, "NAPSAC: High noise, high dimensional robust estimation," in *Proceedings of the British Machine Vision Conference*, pp. 458–467, 2002.
13. J. Gluckman and S. Nayar, "Rectifying transformations that minimize resampling effects," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, **1**, pp. I–111–I–117, 2001.
14. R. Braspenning and M. O. de Beeck, "Efficient view synthesis from uncalibrated stereo," in *Proceedings of the SPIE*, **6055**, pp. 189–199, 2006.
15. G. de Haan, "IC for motion-compensated de-interlacing, noise reduction, and picture-rate conversion," *IEEE Transactions on Consumer Electronics* **45**(3), pp. 617–624, 1999.
16. G. de Haan, P. Biezen, H. Huijgen, and O. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Transactions on Circuits and Systems for Video Technology* **3**(5), pp. 368–379, 1993.
17. J. van de Waerdt, "The tm3270 media-processor," in *Proceedings 38th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 331–342, 2005.
18. "NXP PNX4103 `http://www.nxp.com/products/pnx4103/index.html`."
19. P. Redert, R. Berretty, C. Varekamp, B. Geest, J. Bruijns, R. Braspenning, and Q. Wei, "Challenges in 3DTV image processing," in *Proceedings of the VCIP*, 2007. to appear.