3D-TV signal conversion algorithms and embedded architectures

de Groot, M.J.G.

Award date:
2009

Eindhoven University of Technology
Department of Electrical Engineering
Signal Processing Systems

Technische Universiteit Eindhoven University of Technology
Where innovation starts

3D-TV Signal Conversion Algorithms and Embedded Architectures

Mark de Groot

September 2009

Confidential from September 2009 until September 2010

Master of Science thesis

Project period: August, 2008 - June, 2009
Report Number: 9-32/06-07-2009

Commissioned by:
Prof.dr.ir. P.H.N. de With, Eindhoven University of Technology, NL
Dr.ir. Erwin B. Bellers, NXP Semiconductors Inc., USA

Supervisors:
Prof.dr.ir. P.H.N. de With, Eindhoven University of Technology, NL
Dr.ir. Erwin B. Bellers, NXP Semiconductors Inc., USA
Ir. Ralf Karge, NXP Semiconductors Inc., USA

This thesis report is confidential from September 2009 until September 2010.

ABSTRACT

Three dimensional television (3D-TV) is considered to be the next logical step in the history of television. From SD to HD, the step to 3D-TV adds a third dimension to the viewing experience. 3D capable display panels are becoming more and more available for the consumer. Therefore, the widely available anaglyph format and the increasing availability of 3D movies creates the need for conversion techniques. First, the anaglyph format suffers from color reproduction artifacts. To display anaglyph images on full color 3D display panels, we have evaluated several color restoration algorithms. Second, the viewing conditions for stereo signals vary among different situations, and can contribute to an uncomfortable viewing experience. To this end, we have developed a stereo signal conversion algorithm, capable of adjusting depth information in the signal. Third, we have investigated the complexity of the view synthesis sub-algorithm (which is part of the stereo conversion algorithm) on a TriMedia TM3282 media-processor and contribute with implementations of complexity-reduced view synthesis algorithms, including a newly developed algorithm. Having performed the complexity analysis for these algorithms, it is concluded that the complexity can be reduced significantly to about one half. As a result, the development of the new view synthesis algorithm has led to an invention disclosure [12].

ACKNOWLEDGMENTS

This project is carried out at NXP Semiconductors Inc. in San Jose, California, USA and would not have been possible without the support of many people. The author wishes to express his gratitude to his supervisors, Prof. Dr. P.H.N. de With, Dr. E.B. Bellers and Ir. R. Karge who were abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude are also due to the members of the Media Processing Center of Excellence, especially Ir. M. Penners and Ir. H. He without whose knowledge and assistance this study would not have been successful. The author wishes to express his love and gratitude to his beloved families; for their understanding & endless love, through the duration of his studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Transitions from analog to digital, and from standard-definition (SD) to high-definition (HD) television, three-dimensional television (3D-TV) is considered to be the next logical step in the history of television. In media and entertainment, the third dimension gives the audience the illusion of depth. Objects seem to pop out of the screen, or they appear far beyond the screen. The film industry is already capable of producing movies in 3D (either from existing 2D movies or from scratch). Currently, more and more cinemas have adjusted to show 3D movies as well.

Figure 1.1: 3D-TV giving the illusion of depth.

However, 3D-TV in the consumer market is one of the more recent developments. For example, flat display panels with 3D capabilities are becoming available.

Stereoscopic vision

Humans are capable of judging depth, as they learned relations between physical positions of objects and the objects in the images seen by the eyes [13]. These relations include *interposition*, *relative size* and *motion parallax* and are only small contributions to the depth perception. Key is that the main mechanism of depth perception involves both eyes.

Each eye will see the same scene from a different perspective. Therefore, images seen by each eye are slightly different. The brain will try to combine them into one picture by matching similarities and using small differences to make judgements on depth. This is known as stereoscopic vision [5].

However, an image on a display is seen by both eyes. Although humans are able to extract depth information, there is no stereoscopic vision possible. To enable stereoscopic vision, it is necessary to provide each eye with a different image. These images should represent two perspectives of the same scene with minor deviations similar to the perspectives that both eyes naturally receive in binocular vision.

Consider Fig. 1.2. Within each eye, points A, B and C produce congruent mappings. At the imaginary position E, both mappings can be shifted to cover each other. However, point P (in front of the plane) causes different mappings, which do not cover each other in E. This

difference causes the brain to see that point P is in front of the ABC plane. In order to reproduce this situation using a 3D display technology, point P must be projected on positions $\tilde{p}$ and $\bar{p}$ for the left-eye image and the right-eye image, respectively [20].

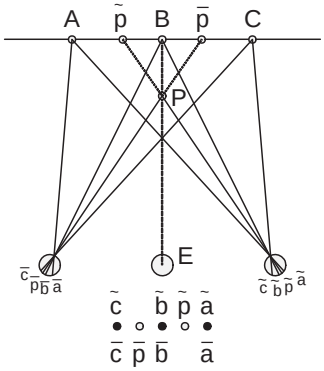


Figure 1.2: Natural viewing with two eyes and resulting perception point E.

Display technology

For many years, displays have been capable of showing only single images at a time. In order to provide each eye with a different image, several methods have been used. One well known and widely used method is to display *anaglyph images* [20].

Nowadays, displays are entering the market, which are capable of showing multiple images simultaneously to produce stereoscopic vision. The most common display technologies are *lenticular displays*, *polarized displays*, *projection displays* and *time-sequential displays*. Except for the latter technology (which does not display stereoscopic images truly simultaneously), these are all time-parallel displays. If these displays are also capable of showing more than two images simultaneously, the viewer can also experience horizontal parallax.

Viewing conditions and perception of depth

The viewing conditions can be very different among various situations, i.e. cinema versus home theater. Consequently, the perception of depth is also very different between these situations. There are two aspects that play a key role in depth perception:

- *Viewing distance.* Increasing the distance between viewer and display will also increase the depth perception. This means that the distance between perceived objects increases. Consequently, decreasing the viewing distance decreases the perceived depth.
- *Display size.* The display size together with its resolution determines the pixel pitch. The distance between two neighboring pixels is different for larger or smaller displays with an equal resolution. The horizontal inter-pixel distance can even be different from vertically adjacent pixels. Image points match the pixel grid. If the distance between neighboring pixels increases, the real screen disparity also increases. Therefore, the depth perception is

also different. Here, disparity is defined as the pixel-shift between the same object in the left and the right images.

The physical condition of the viewer also affects depth perception. If the eyes are tired, they need more effort to perceive depth.

1.1 PROBLEM DESCRIPTION AND OBJECTIVES

3D-TV signals come in many flavors (i.e. anaglyphs, stereo signal, etc.). A stereo signal (or two-view signal) is composed of two signals. These two signals are in full color and are placed side-by-side. Each signal is intended for one eye only. However, an anaglyph signal has two views combined in a single signal. Each perceived view for one eye has reduced color information, because each eye obtains only color-filtered information. For our study, we consider both types of input signals: full-color stereo signal and an anaglyph signal. Both possible input signals are subject to conversion, such that they can be displayed on full-color stereo display. The displayed signal is perceived by the user employing polarized glasses. The problem statement is now detailed in two ways. First, for anaglyph images, we aim at a conversion of the color shifted information towards a disparity oriented shift suitable for feeding the left and right stereo channel. Second, for stereo images, we aim at a conversion into another stereo signal with which the user can control the depth perception of the 3D signal.

In the following subsections, for each of the individual problems, specific requirements and objectives are specified.

Anaglyph format conversion

The anaglyph image format is a well known and still widely used technique to produce stereoscopic vision. An anaglyph image is a composition of two images, each in different color. The main advantage of anaglyph images is that any color display can be used to render it.

The separation of both images is realized by color filters in the goggles. However, each eye will only see a partial color image, which is not seen by the other eye (i.e. with red-cyan anaglyph, the left eye obtains a red color image and the right eye obtains a cyan color image). Since chromatically opposite colors are used to separate each view in the anaglyph, original color information is lost.

However, display panels with 3D capabilities are able to display full-color images for each eye. Since anaglyphs have only partial color information for each view, conversion to full color images is necessary to display anaglyph images on these 3D display panels.

The first objective is to determine an anaglyph conversion algorithm, which is capable of restoring full color signals from a given anaglyph signal. We address the objective in the sequence of the following steps. Firstly, the color information in the anaglyph signal is separated. Secondly, the missing color components are restored. Then, the components are combined to form two full color signals.

Stereo signal conversion

Depth perception of 3D content (or stereo content in particular) depends on the viewing conditions. Visual discomfort can occur with incorrect viewing conditions. Therefore, adjusting the perception of depth to a comfortable viewing experience, or enhancing the perception of depth are several use cases of adjusting depth information in stereo content.

However, display panels are driven by dedicated high-end video processing platforms. These embedded architectures are specifically designed for video processing. To add this feature to 3D capable displays, a stereo signal conversion algorithm is proposed. This algorithm will be integrated in the next generation platform and imposes several requirements:

- *Dedicated video processing platform.* This platform consists of, one or more powerful TriMedia TM3282 media-processors, a dedicated co-processor and memory. The conversion algorithm is required to run on this platform.
- *Algorithm complexity.* Since a single media-processor has limited processing capacity, the complexity of this algorithm should be reduced to a minimum. This algorithm is assumed to take its place in a multitasking environment.
- *Memory bandwidth.* Memory bandwidth should be limited to a minimum.
- *Real-time processing.* A frame rate of 60 frames per second is required for real-time processing.
- *Intermediate format.* To cope with different input and output formats, an intermediate format is defined. This format consists of a 2D reference image and depth image.
- *3D display panel.* The stereo signal is displayed on a 3D capable display panel with an effective resolution of 960x540.

The second objective is to determine a stereo signal conversion algorithm which is capable of adjusting depth information contained in stereo signals. The stated requirements should be satisfied. We address this objective in the sequence of the following steps. Firstly, the signal conversion algorithm is decomposed into smaller existing algorithms. Secondly, these existing sub-algorithms are adjusted together to form this stereo conversion algorithm for stereo content.

The third objective is to determine the implementation and computational complexity of a sub-algorithm onto the dedicated video processing platform. We address this objective as follows. Firstly, this part of the algorithm is mapped onto the processing platform. Secondly, complexity analysis is performed for this algorithm part. And last but not least, we investigate ways to reduce the computational complexity of this algorithm part.

1.2 THESIS LAYOUT

This thesis consists of two parts.

The first part elaborates on anaglyph conversion algorithms in Chapter 2. This chapter describes how anaglyph images are produced and

is followed by an evaluation of several methods to restore full color images from them.

The second part elaborates on the stereo signal conversion algorithm. This algorithm is detailed in Chapter 3. Each sub-algorithm is explained in depth. Chapter 4 discusses the mapping and complexity analysis of the last sub-algorithm of the conversion algorithm. Methods to reduce the computational complexity are also investigated. Chapter 5 elaborates on a new algorithm, which replaces the last sub-algorithm. Computational complexity is reduced significantly, and the development of this algorithm has led to an invention disclosure [12]. Chapter 6 provides a summary of the previous chapters and concludes with conclusions and future challenges.

2 ANAGLYPH CONVERSION

Anaglyph images provide stereoscopic vision. An anaglyph image is a composition of two input images [20] (image from the left-view position and image from the right-view position). Fig. 2.1 shows how an anaglyph is created from two full color input images. To perceive three-dimensional objects, the left eye should only receive the left input image and the right eye should only receive the right input image. Therefore, each input image is shown in chromatically opposite colors (i.e. red is chromatically opposite to green and blue). In conjunction with appropriate color spectral glasses, each eye receives only one of its views. Consequently, each eye will only receive a reduced color image.

2.1 ANAGLYPHS

The stereoscopic effect of anaglyph images is only provided using appropriate anaglyph glasses. Each lens has a chromatically opposite color filter. These special glasses are necessary to separate the left and right views contained in anaglyph images. The image intended for the left eye should not be received by the right eye and vice versa, otherwise interference occurs.

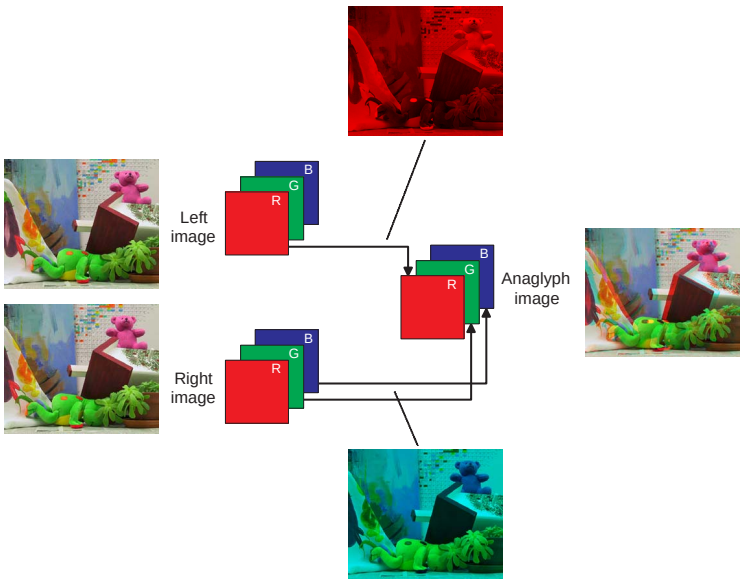


Figure 2.1: Composition of an anaglyph image.

Red-cyan anaglyphs

The most common anaglyphs are represented in red and cyan colors. Consequently, the appropriate glasses have also red and cyan color filters, which are positioned over the left and right eye, respectively.

More recently, the so-called ColorCode anaglyphs are introduced. The views in these types of anaglyphs are shown in amber and blue colors. Consequently, the glasses have also amber and blue color filters. The color information is seen through the amber filter and parallax information is seen through the blue filter [4].

Color perception

Since each eye only receives a reduced color image, color is perceived by combining the two images. Therefore, perception of color depends on the input images. Color input images produce anaglyphs, in which the viewer perceives color. However, if gray input images are used to produce anaglyphs, no color is perceived. This leads to *color* and *gray* anaglyphs.

Consider a color anaglyph image. Color is perceived by combining colors from each separate view. The left eye only receives the red colors and the right eye only receives cyan colors. The brain has to combine the separate color information to perceive the true colors of the perceived image.

Side effects

There are some side effects that can degrade the quality of viewing anaglyphs. These side effects include:

- *Color reproduction.* Each eye receives a reduced color image. The brain has to combine the color images received in both eyes. Gray anaglyphs contain no color information.
- *Ghosting.* Non-ideal filters in the color spectral glasses cause information intended for one eye, to be received by the other eye also (this is also known as interference).
- *Retinal rivalry.* The images for left and right eye are too different, matching similarities fails.
- *Brightness.* If the brightness difference of same objects in both views becomes too much, it causes unpleasant viewing.

Regardless of these effects, anaglyphs have been widely used in science and design where depth perception is useful. Any color display technology is capable of showing anaglyph images. The viewing glasses are inexpensive to create.

Producing anaglyphs

In the past, stereo camera systems have been extended by placing a red color filter in front of the left camera, and a cyan color filter in front of the right camera. These two images are then projected together as a single image to form the anaglyph image. Nowadays, an image processing algorithm is typically used that simulates these color filters.

Consider producing a red-cyan (color) anaglyph image. Eq. (2.1) is used to create such an anaglyph image and is defined as:

$$\begin{bmatrix} r_A \\ g_A \\ b_A \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_L \\ g_L \\ b_L \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_R \\ g_R \\ b_R \end{bmatrix}, \quad (2.1)$$

where

r_A, g_A, b_A	are the R, G and B color components of the anaglyph image,
r_L, g_L, b_L	are the R, G and B color components of the left input image,
r_R, g_R, b_R	are the R, G and B color components of the right input image.

This equation calculates the intensities for the separate primary color components red (R), green (G) and blue (B). See [22] for a summary of similar equations. These equations essentially try to reduce the side effects. This is achieved by changing the contribution of each color component.

Restoring an anaglyph image back to full color images

Consider a red-cyan (color) anaglyph, created according to Eq. (2.1). The available color information are the separate color components in the anaglyph image. The red color component originates from the left-view image, while the green and blue color components originate from the right-view image. Consequently, the green and blue color components are missing for the left-view image and the red color component is missing for the right-view image.

Restoring the missing color components of each view is based on the assumption of stereoscopic vision. The images received by the eyes should represent two perspectives of the same scene with minor deviations, similar to the perspectives that both eyes naturally receive in binocular vision. Consequently, the missing color components of the left-view image can be found in the right-view image. The missing color component in the right-view image can be found in the left-view image.

2.2 RESTORATION METHODS

The missing color components are restored by retrieving them from the images in each other's view. A possible restoration method is depicted in Fig. 2.2.

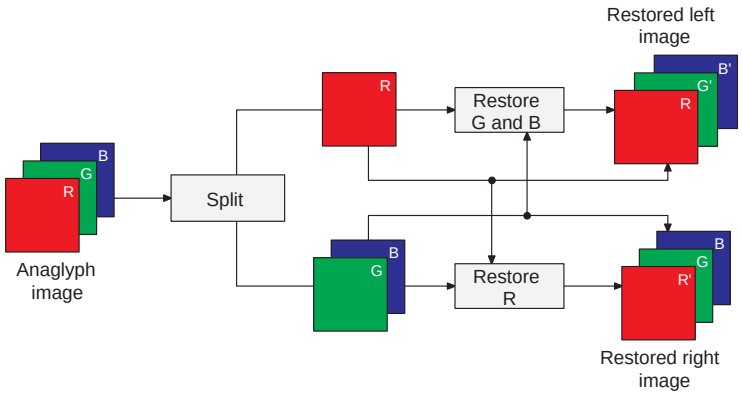


Figure 2.2: Proposed separation process of anaglyph images.

Objects which appear closer in one viewpoint, occlude objects further away. Since the perspective of both views are slightly different, occluded parts may become visible in the other viewpoint. Consequently, missing color information of these parts cannot be found in the image of the other view.

Two restoration methods have been evaluated. The first method involves a global transformation model. The second method evaluates the use of a local motion estimation method.

2.2.1 *Affine transformation model*

In [6], a solution is proposed to improve the quality of colonoscopy videos. The colonoscope has a monochrome chipset, which records red, green and blue color components successively and merges them into one video signal. Due to movements of the camera, misalignment of the color channels occur. The proposed solution restores this artifact by first equalizing the color channels, followed by performing camera motion estimation and compensation.

Approach

The production of an anaglyph can be depicted as the recording of a colonoscopy video. First, the red color channel is recorded. Second, a movement of the camera follows, specifically in a horizontal, right direction. Then, the green and blue color channels are recorded successively without camera movement in between.

The proposed restoration method is evaluated to restore anaglyph images.

Implementation aspects

Histogram specification is used to equalize the histogram of red color channel to the green color channel. This is implemented according to [21].

To estimate the 6 parameters for the camera motion model, the framework applied in [6] is used. This framework is available at [16].

Motion of the camera is compensated for each color channel, using the 6-parameter affine transformation model. The transformed image may not exactly match the output pixel grid. Therefore, bilinear interpolation is used to re-sample the transformed image.

2.2.2 *3-D Recursive search*

3-D recursive search (3DRS) block matching algorithm ([9] and [8]) is an efficient motion estimation method and is extensively used in motion compensated Frame Rate Conversion (FRC) systems. Instead of estimating motion between two successive frames (time-domain), 3DRS is used to estimate screen disparity (spatial domain) between left and right input images.[1]

1 For more details regarding 3-D Recursive Search as a disparity estimator, read Section 3.2 on page 18.

This 'motion' of objects between the left and right input images is better known as screen disparity. Screen disparity is the horizontal distance between two corresponding points in stereoscopic content.

In this solution, 3DRS is used to estimate the screen disparity. This block-based matching algorithm compares blocks of one image (i.e. left-view image) to one block of the other image (i.e. right-view image). The similarity of blocks is determined by the sum of absolute differences (SAD) and the minimum value depicts the best match. This similarity measure depends on the brightness constancy assumption. Brightness of a small region in an image is assumed to remain the same, although the location may change. However, an anaglyph has only red color information of the left view, and green and blue color information of the right view. Depending on the color of the object, the red, green and blue color components have different intensity levels. Therefore, the brightness constancy assumption does not hold anymore.

Approach

To cope with intensity differences between color components, two approaches with the 3DRS algorithm are evaluated:

- *Modify pixel intensities of input images.* The input images are transformed, either by histogram specification or by filtering using a high-pass filter. This is prior to disparity estimation.
- *Use a different similarity measure.* Instead of using the SAD value, Mutual Information (MI) is used to measure similarity of blocks. Input images are not changed. See [10] and [14] for more details on this concept.

MODIFY PIXEL INTENSITIES The input images are filtered, either by histogram specification or by a high-pass filter.

With histogram specification the red color component is adjusted, such that its histogram matches with the histogram of the green color component. Histogram specification is implemented according to [21].

The high-pass filter is a 3×3 spatial box-filter [7] and is defined as:

$$h_{HPF} = \frac{\alpha}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2.2}$$

The gain $\alpha = 4$ for clearer display and offset of 128 is chosen to adopt to the range of the input image (8-bit unsigned).

The result of both filters is given in Fig. 2.3. These images have a resolution of 448x368 pixels. Certain cones appear very dark in the green color component in Fig. 2.3(d), but appear bright in the red color component in Fig. 2.3(a). These parts are difficult to match using the SAD similarity measure. The matching of blocks remains difficult when intensity levels are adjusted after applying histogram specification. Alternatively, applying a high-pass filter removes low frequencies and keeps high-frequency (noisy) artifacts. Since the SAD measure relies on these low frequencies, block matching can be difficult. Nevertheless, we have adopted the filtering for implementation, as it yields a more accurate vector field for 3DRS.

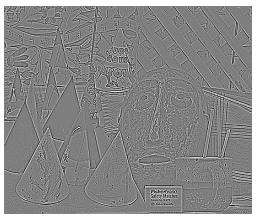
(a) Red color component

(b) Histogram specification of (a) adjusted to (d)

(c) High-pass filtered of (a)

(d) Green color component

(e) Histogram specification of (d) adjusted to (a)



(f) High-pass filtered of (d)

Figure 2.3: Original and filtered color components.

DIFFERENT SIMILARITY MEASURE The 3DRS algorithm uses the similarity measure Sum of Absolute Differences (SAD). This measure is based on the brightness constancy assumption. Due to brightness differences between red and green color components, this assumption does not hold anymore. To cope with these differences, another similarity measure is used, which is Mutual Information (MI).

Mutual Information depends on the entropy and joint entropy of two random variables. With stereo images, the random variables are image pixels taken from each image in a stereo frame. In case of anaglyphs, pixels are taken from the red color component and from the green color component. This means that the input images for 3DRS are the unaltered color components, i.e. no transformation is applied.

Entropy is defined as:

$$H(A) = -\sum_{i \in A} p(i) \cdot \log_2 p(i) \,, \tag{2.3}$$

where

$H(A)$ is the entropy of image A,

$p(i)$ is the probability of i occurring in image A.

The joint entropy $H(A, B)$ is defined similar to the single entropy of Eq. (2.3). The probability function $p(i)$ is replaced with the joint probability function $p(i, j)$. Using the definition of entropy and joint entropy, the Mutual Information is defined as:

$$MI(A, B) = H(A) + H(B) - H(A, B) \,, \tag{2.4}$$

where

$MI(A, B)$ is the Mutual Information between images A and B,

$H(A)$ is the entropy of image A,

$H(A, B)$ is the joint entropy of images A and B.

The 3DRS algorithm is based on block matching. For every block in the matching image, several candidate blocks are evaluated using this similarity measure.

Consider an evaluation of one matching block with a candidate block. First, the entropy of the matching block is calculated, followed by calculating the entropy of the candidate block. Then, the joint entropy is calculated of both blocks. Finally, the Mutual Information of this pair is calculated.

The entropy for an 8×8 block is calculated according to Eq. (2.3), where A is limited to a small window, which is positioned such that the reference block is in the middle. This is depicted as a bold rectangle centered surrounding candidate block C in Fig. 2.4. The size of this window can be adjusted to make a tradeoff in disparity-vector quality and performance. With respect to the implementation, when a window is larger than a block, it automatically overlaps image data. Consequently, image data has to be accessed more than once.

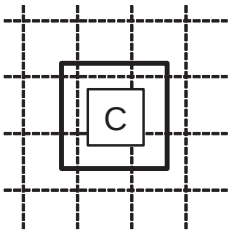


Figure 2.4: Example of entropy window, symmetrically positioned.

Implementation aspects

In order to use MI as a similarity measure in 3DRS, the probability functions need to be estimated in the matching images. A normalized histogram with 20 bins is used, proposed by [10].

The 3DRS algorithm calculates the similarity values between a block in the matching image and several candidate blocks in the matched image. The vector associated with the best match is assigned to the block in matching image. The size of these blocks is 8×8. To calculate the entropies for the MI similarity measure, a window of 15×15 pixels is used [10].

Spatial candidate vectors are preferred over temporal and updated spatial candidates. Therefore, temporal and updated spatial candidates are given a penalty. Because the candidate vector with maximum MI value is selected as the best candidate, penalties given to the candidates should lower the MI value. The range of similarity values of MI differs from SAD. Therefore, further investigation of the penalties for each candidate is needed. Currently, a penalty of 0.125 is given to every temporal and updated spatial candidate.

Estimated computational complexity

Table 2.1 shows the number of operations needed to evaluate the similarity between the current block and the block pointed by a candidate vector.

OPERATION	ADD/SUB	ABS	MUL	DIV
Sum of Absolute Differences (SAD)	128	64	0	0
Mutual Information (MI)	1118	0	225	0

Table 2.1: Operation cost estimation for calculating similarity of two 8x8 blocks.

The cost estimation of the similarity measure MI is based on the implementation with 20 bins and a window of 15×15. Except for the joint histogram, bin selection for the histogram and the calculation of the entropies are based on implementation with lookup tables.

The similarity measure MI needs a larger memory. In contrary to SAD (only two times 64 read accesses occur), MI has to store intermediate histograms in memory. Also the required lookup tables are stored in memory. Consequently, more read and write accesses are needed.

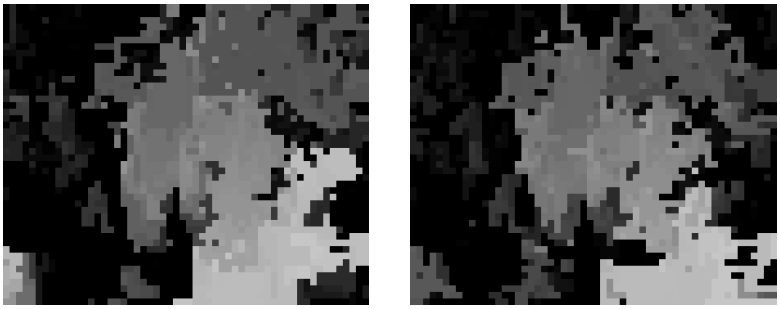
2.3 EXPERIMENTAL RESULTS

Determining a 6-parameter motion model for stereo content, results in several cases that the transformed color components appear slanted. In stereo content, objects which are closer to the viewpoint are usually in the bottom of that image. Objects further away from the viewpoint, appear usually in the middle or top of that image. Since closer objects have a larger screen disparity, the global parametric model shifts the bottom of the image more in horizontal direction compared to the top of the image. Consequently, the transformed image appears slanted.

Modifying the input images, prior to 3DRS-based disparity estimation, such as histogram specification, does not always reduce the large difference of pixel intensities between corresponding objects sufficiently. Consequently, the block-matching algorithm has difficulties with estimating disparities. However, removing low frequency components using a high-pass filter leaves in the high frequencies, but removed the low ones. See Fig. 2.3(c) and 2.3(f). In Fig. 2.5 disparity images are shown, where the pixel intensity represents the screen disparity between matched points. A brighter pixel means a larger screen disparity. Disparity images are up-sampled to match the frame-resolution. Results in Fig. 2.5(a), 2.5(b) and 2.5(c) are produced by 3DRS with SAD as the similarity measure. The latter two figures show results of 3DRS with transformed input images. Fig. 2.5(d) shows the result of 3DRS with MI as the new similarity measure and untransformed input images.

However, 3DRS-based disparity estimation with MI as the similarity measure produces equal or better results in comparison to 3DRS-based disparity estimation with SAD as the similarity measure on high-pass filtered input.

In the Skydiving sequence, Fig. 2.6 with a resolution of 720×480 pixels, Mutual Information as the similarity measure contributes to find

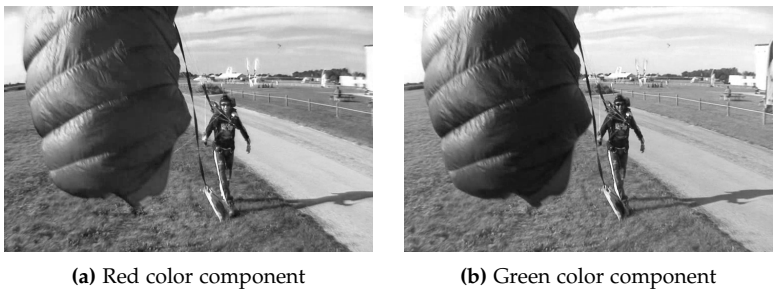


(a) No transformation applied **(b)** Histogram specification

(c) High-pass filtered **(d)** Mutual Information

Figure 2.5: Results of 3DRS-based disparity estimation on transformed Cones images.

better correspondences. The input images are shown at the top of this figure.



(a) Red color component **(b)** Green color component

(c) High-pass filtered **(d)** Mutual Information

Figure 2.6: Original frames and results of 3DRS-based disparity estimation, using either pre-filtering (c) or alternative similarity measure (d).

2.4 DISCUSSION

Affine transformation model

Estimating and compensating camera motion according to a parameterized affine motion model gives very poor results. The restored color components appear slanted, due to the nature of images. Objects close to the viewpoint usually appears at the bottom of the image and have more screen disparity than objects further away from the viewpoint.

Since objects are mostly horizontally offset, the 6-parameter model seems more than sufficient. This transformation also includes scaling and rotation, which is for stereo content usually not applicable.

The proposed restoration method is specifically designed for these colonoscopy videos. These videos contain very little or no depth information. A horizontal movement of an object most likely applies to all other objects within the same image. The colonoscopy camera moves in arbitrary directions. Therefore, this parametric model is sufficient. However, for restoring images from an anaglyph, this method is not sufficient. The discontinuities in screen disparities makes this parametric transformation model insufficient.

3DRS-based disparity estimator

Modifying pixel intensities by high-pass filtering (prior to disparity estimation) produces significantly better results in comparison to modifying images by histogram specification.

Using a different similarity measure gives also promising results. However, further improvement by tuning this block-matcher is necessary. More investigation of determining the optimal window size for entropy calculations, block sizes and the preference of spatial candidate vectors over other candidate vectors is necessary to increase the robustness of this algorithm.

A disadvantage of 3DRS with Mutual Information as a similarity measure is the computational complexity. For each block, a normalized histogram must be calculated, and its window is usually larger than the block. Calculating entropy (given a histogram) and Mutual Information can be implemented by using a lookup table. Compared to calculating SAD values, this similarity measure is much more complex.

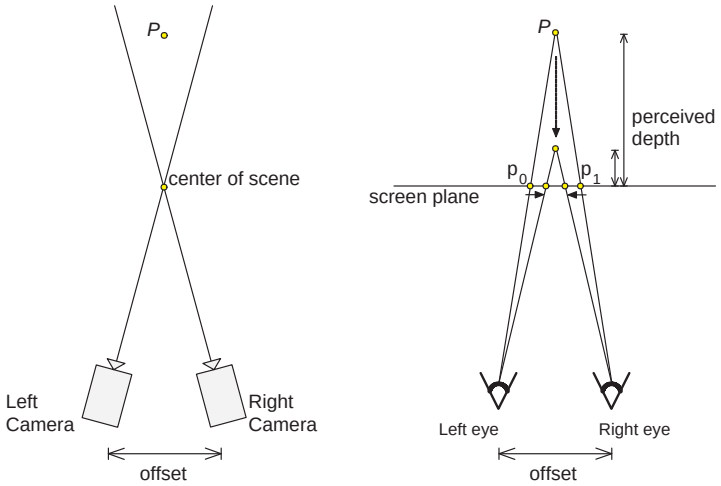
Both approaches have their strengths and weaknesses. From a complexity point of view, using Mutual Information as a similarity measure is much more complex, but produces similar or better results.

3 STEREO SIGNAL CONVERSION

This chapter elaborates on the 3D-TV conversion algorithm introduced in the first chapter. This particular conversion algorithm extracts depth information from the stereo signal. With this depth information and a reference image, a new stereo signal is created. First, the concept of adjusting depth information is explained. Subsequently, each of the steps will be explained in more detail.

3.1 STEREO CONTENT

In a stereo-view signal, a scene is captured by two cameras. The optical axes are pointing to the center of the scene, i.e. their optical axes are converging. In Fig. 3.1(a), such a camera setup is illustrated.



(a) Typical camera setup **(b)** Depth conversion

Figure 3.1: Camera setup (a) and depth conversion (b) in stereo content.

Consider Fig. 3.1(a) with a point P, which is behind the center of the scene. Point P is captured by both cameras. Since the cameras are horizontally offset, point P appears also horizontally offset in the captured images. Point P is projected in the left image on the left, compared to the position where point P is projected in the right image. Projection of a point which is in front of the center, appears in opposite orientation. Projection of a point in the center of the scene, overlap each other exactly. Projections of point P are depicted in Fig. 3.1(b) as p_0 and p_1 on the screen plane.

Consider Fig. 3.1(b) with a point P and its projections p_0 and p_1. If it is possible to decrease (or increase) the distance between these projections, the perceived depth of point P is also decreased (or increased).

This stereo signal conversion algorithm is composed of three steps. First, the screen disparity is estimated from stereo content. The screen

disparity is the distance between the projections of point P in the left and right image, respectively p_0 and p_1 . Second, the low resolution disparity image is filtered and up-sampled to a depth image with frame-resolution. In the final step, a view synthesis algorithm produces stereo images from a reference and the frame-resolution depth image. These step are depicted in Fig. 3.2.

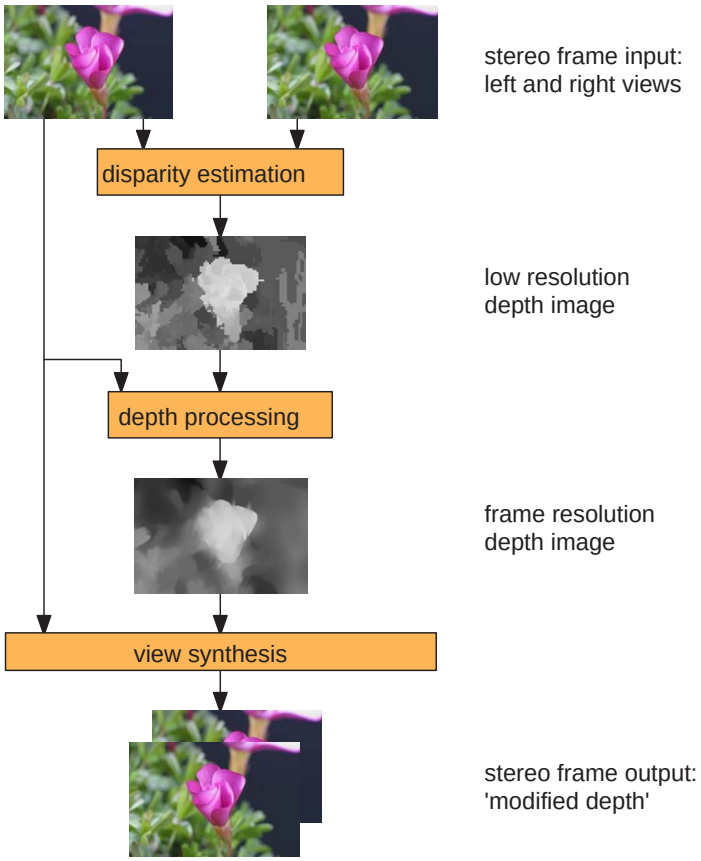


Figure 3.2: Overview of stereo signal conversion algorithm.

3.2 DISPARITY ESTIMATION

To extract disparity information from a stereo image, a disparity estimator is used. This is also known as the *stereo correspondence* problem. The stereo correspondence problem is to find a set of points in one image, which can be identified as the same points in another image. There are many algorithms available in literature trying to solve this problem. A large set of algorithms is evaluated in [19].

3.2.1 *3-D Recursive Search*

3-D recursive search (3DRS) block matching algorithm ([9] and [8]) is an efficient motion estimation method and is extensively used in motion compensated Frame Rate Conversion (FRC) systems. Its efficiency has proven to satisfy the real-time requirement in consumer electronics.

Instead of estimating motion between two successive frames (in the time-domain), 3DRS is used to estimate screen disparity (in the spatial domain) between left and right input images.

The 3DRS algorithm is based on two assumptions. First, objects are larger than a block. This means that each block within the same object should have the same disparity vector. Therefore, the disparity vector can be found in one of the neighboring blocks. These disparity vectors are spatial vectors. Second, objects have inertia. The disparity that each object has, is assumed to continue over time and can change gradually, not instantaneously. Hence, disparity vectors of this object in nearby frames are likely to be the same too. These disparity vectors are referred to as temporal vectors. Next to the spatial and temporal vectors, there is also a set with update vectors. The update vectors allow the algorithm to detect discontinuities in screen disparity. Instead of taking all possible candidate vectors, this algorithm defines a small set of candidate vectors, consisting of spatial, temporal and updated vectors.

The 3DRS algorithm in motion estimation finds motion vectors, that can be in arbitrary directions. However, 3DRS in disparity estimation finds disparity vectors, which should be only in the horizontal direction. This is a consequence of the horizontal offset in the specific camera setup.

Parameters

In [15] and [2] 3DRS is used as a disparity estimator in stereo video processing.

- *Block size.* The block size is chosen as 8x8 pixels.
- *Match criterion.* The similarity measure is sum of absolute differences.
- *Vector resolution.* $\frac{1}{4}$ pixel accuracy is used in the horizontal direction of the disparity image. However, the accuracy in the vertical direction is limited to integer accuracy only.
- *Vector range.* Disparities in evaluated stereo content range up to 80 pixels. Vertical disparities are only used to compensate for camera misalignment of the input images.
- *Random updates.* Camera misalignment varies only gradually and therefore, only few random updates in the vertical direction are needed. In the horizontal direction more random updates are needed. The distribution of these updates is different from the 3DRS-based motion estimator.
- *Scanning directions.* Block-wise scanning is performed, zigzagging from top-to-bottom over the image. In successive passes, scanning starts at the top or at the bottom of the image.
- *Candidate set.* Basically, the candidate set consists of 5 candidate vectors (when scanning from left-to-right and top-to-bottom, see Eq. (3.1)):
  - Two spatial candidates: vectors taken from the block above and from the block on the left hand side of current block.
  - One temporal candidate: a vector taken from position two blocks to the right and two blocks down the current block.

– Two update candidates: each spatial candidate is updated with a pseudo random update vector.

While the scanning direction changes, the positions of candidate vector blocks are mirrored.

The candidate set **CS** of the 3DRS-based disparity estimator for a block at position $\vec{x} = (x, y)^T$ in frame n, is defined as:

$$\mathbf{CS}(\vec{x}, n) = \left\{ \begin{array}{l} \vec{d}\left(\vec{x} + (0,-1)^T, n\right), \\ \vec{d}\left(\vec{x} + (-1,0)^T, n\right), \\ \vec{d}\left(\vec{x} + (2,2)^T, n-1\right), \\ \vec{d}\left(\vec{x} + (0,-1)^T, n\right) + \vec{u}, \\ \vec{d}\left(\vec{x} + (-1,0)^T, n\right) + \vec{u} \end{array} \right\}, \tag{3.1}$$

where $\vec{d}(\vec{x}, n)$ is the vector of a previously calculated block. This is depicted in Fig. 3.3. The spatial and temporal candidates are taken from position S and T, respectively. C denotes the current block. The update vector $\vec{u}$ is chosen randomly from update set **U**:

$$\mathbf{U} = \left\{ \begin{array}{llll} \frac{1}{4}\vec{x}_u, & \frac{1}{2}\vec{x}_u, & \vec{x}_u, & 2\vec{x}_u, \\ -\frac{1}{4}\vec{x}_u, & -\frac{1}{2}\vec{x}_u, & -\vec{x}_u, & -2\vec{x}_u, \\ \vec{y}_u, & -\vec{y}_u & & \end{array} \right\}, \tag{3.2}$$

where $\vec{x}_u = (1,0)^T$ and $\vec{y}_u = (0,1)^T$. According to [15], integer accuracy in vertical disparity is sufficient. Sub-pixel accuracy in the vertical direction does not add more accuracy in the horizontal direction.

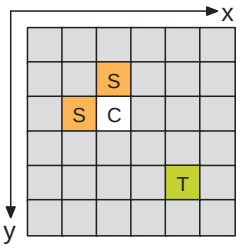


Figure 3.3: Relative position of candidates in 3DRS (excluding updated candidate vectors).

3.2.2 *Experiments*

The 3DRS algorithm is mainly used in estimation of motion between successive frames. However, stereo signals have different characteristics, compared to mono-view signals. Therefore, using this algorithm to estimate screen disparity, requires several adjustments to suit disparity estimation on stereo signals. Several experiments are performed to determine the optimal parameters for the 3DRS-based disparity estimator.

Remove Vertical Updates

EXPERIMENT The update set is changed, such that it contains no vertical updates. The update set **U** is limited to:

$$\mathbf{U} = \left\{ \begin{array}{cccc} \frac{1}{4}\vec{x}_u, & \frac{1}{2}\vec{x}_u, & \vec{x}_u, & 2\vec{x}_u, \\ -\frac{1}{4}\vec{x}_u, & -\frac{1}{2}\vec{x}_u, & -\vec{x}_u, & -2\vec{x}_u \end{array} \right\}. \tag{3.3}$$

MOTIVATION Stereo signals are captured by two cameras. Since the cameras are only horizontally offset and vertically aligned, disparity vectors should have pure horizontal components only.

RESULT The disparity vector field shows inconsistency in the temporal direction. The vectors are also inaccurate. Since only horizontal updates are applied, the algorithm tries to find the best match on the same frame-line. The quality of disparity vector field is deteriorated significantly. Therefore, the assumption on vertically alignment of cameras is not valid. Adding a few vertical updates to the set, led to accurate vectors and much less inconsistency in the vector field. The recommendation is to leave in the vertical updates. According to [15], this can be limited to integer updates only.

Remove Zero Vector Candidate

EXPERIMENT The zero-vector candidate is removed from the candidate set.

MOTIVATION In motion estimation, the zero vector candidate has a special meaning. This candidate contributes to achieve true motion vectors in static scenes, consider a newsreader sitting behind a desk. However, in disparity estimation, this zero vector has no special meaning anymore. This vector is even likely to occur than any other vector.

RESULT Tests have shown that result of the disparity estimator has increased quality. The zero vector candidate can introduce errors in the disparity vector field. It mainly occurs in homogeneous areas. Removing this candidate improves the result. This is also recommended by [15].

Static scenes have also static disparity vectors. To extend the candidate set with a candidate that is similar to the zero-vector candidate, the zero-vector *temporal* candidate could be selected. However, this is a temporal vector, which depends on all previous calculations.

Add More Horizontal Updates

EXPERIMENT The update set is extended with several more horizontal updates. I.e. the update set is modified to:

$$\mathbf{U} = \left\{ \begin{array}{cccc} \frac{1}{4}\vec{x}_u, & \frac{1}{2}\vec{x}_u, & \vec{x}_u, & 2\vec{x}_u, \\ -\frac{1}{4}\vec{x}_u, & -\frac{1}{2}\vec{x}_u, & -\vec{x}_u, & -2\vec{x}_u, \\ \vec{y}_u, & -\vec{y}_u & & \\ \frac{1}{4}\vec{x}_u, & \frac{1}{2}\vec{x}_u, & \vec{x}_u, & 2\vec{x}_u, \\ -\frac{1}{4}\vec{x}_u, & -\frac{1}{2}\vec{x}_u, & -\vec{x}_u, & -2\vec{x}_u \end{array} \right\}. \tag{3.4}$$

Two out of eighteen vectors are vertical updates (compared to two out of ten). Therefore, the distribution of updates has changed.

The candidate set is also extended with two more updated spatial candidates.

MOTIVATION More candidate vectors are updated with a horizontal update. Consequently, the convergence speed of vector should increase. Fewer vertical updates are applied, but are essential for camera misalignments. Furthermore, adding two more updated candidate vectors to the candidate set should increase the convergence speed.

RESULT Extending the candidate set and update set significantly increases the convergence speed. However, adding candidates to the candidate set is more computationally expensive. For each block in the matching image, two more blocks in the matched image are evaluated.

Adding even more updated spatial candidates did not increase the convergence speed significantly.

Add Occlusion Correction

EXPERIMENT Occlusion detection and correction is added, prior to disparity estimation of the next frame.

MOTIVATION Stereo frames are taken from two camera positions. Since these images are taken from different perspectives, objects closer to a viewpoint may occlude (parts of) other objects in the other viewpoint. Therefore, certain areas are only visible in one of the two images. These areas are called occlusion areas. The 3DRS algorithm tends to assign disparity vectors belonging to the foreground object to occlusion areas. However, occlusion areas are part of the background and should have corresponding disparity vectors.

RESULT The implementation of occlusion detection is based on Left-Right Checking (LRC), proposed in [11]. This implies that disparity vectors are estimated in two directions. Disparity vectors are estimated from the left to the right image, and in the opposite direction. The implementation of dis-occlusion detection is based on replacing the foreground vector with the nearest background vector on the same line. These steps are performed on the result of the disparity estimator. As a post-processing step on the disparity vector field, occlusion handling increases the accuracy of disparity vectors near edges of objects. Consequently, this increases the quality of depth image. However, it has not been evaluated extensively and needs further investigation (i.e. which blocks are considered to be occluded?).

3.2.3 *Results*

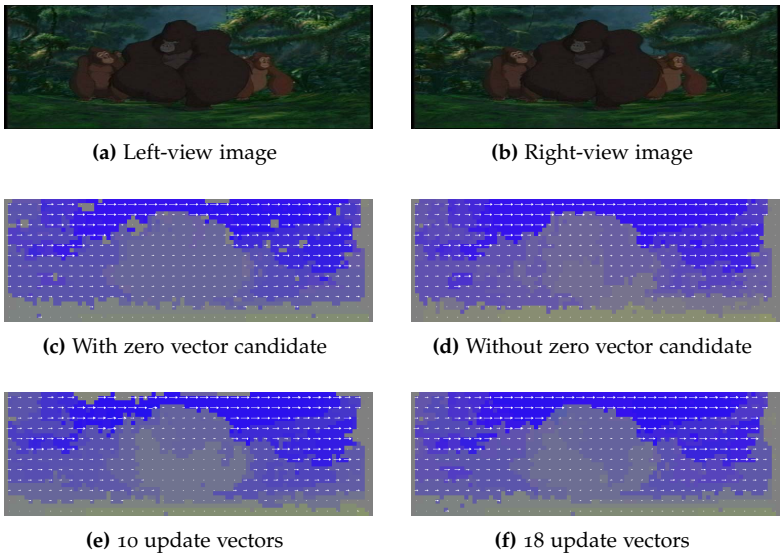
Fig. 3.4 shows results of several experiments conducted previously. The top two images are the input images of the disparity estimator.

The outcome of the estimator is shown in bottom four images. Disparity vectors are shown in color, as well as white arrows. The color intensity is a measure for the disparity vector length. The images are

mostly blue, which means a disparity vector is pointing towards right. However, on the bottom of these images, yellow color is visible. This color means a disparity vector towards left.

The difference between including a zero vector candidate or not, is shown in the middle two images (Fig. 3.4(c) and Fig. 3.4(d) respectively). If the disparity estimator also evaluates the zero vector candidate, errors are introduced in the disparity vector field. This results in holes in the vector field. This supports the conducted experiments in [15].

Adding more horizontal updates gives also little differences in the disparity vector field. Convergence is increased by adding updated candidates. The images shown at the bottom (Fig. 3.4(e) and Fig. 3.4(f)), have clear differences. However, using different content makes this difference less visible.



(a) Left-view image

(b) Right-view image

(c) With zero vector candidate

(d) Without zero vector candidate

(e) 10 update vectors

(f) 18 update vectors

Figure 3.4: Input frames (a), (b) and results of several experiments (c), (d), (e) and (f) with various 3DRS parameters.

3.2.4 *Discussion*

The 3DRS algorithm creates a small set of disparity vectors. Each vector consists of a horizontal component and a vertical component.

First, vertical updates are necessary to assure much more reliable disparity vectors. Second, vertical accuracy is limited to integer accuracy. This means, that no interpolation in the vertical direction is needed to determine the sub-pixel values. Next, adding more updates and adding two more candidates increased the convergence speed significantly. However, adding even more candidates had little influence.

The characteristics of stereo-view content are very different from mono-view content for determining point correspondences. Several experiments show that 3DRS algorithm can be adopted from motion estimation to disparity estimation. However, to determine the optimal parameters for the 3DRS-based disparity estimator, more tests are needed (i.e. is quarter pixel accuracy sufficient?).

3.3 FROM DISPARITY VECTORS TO DEPTH IMAGE

3.3.1 *Horizontal disparity as depth*

The disparity estimator produces disparity vectors, which are composed of horizontal and vertical components. However, the horizontal vector component only is the determining factor for the screen disparity.

Due to possible misalignment of stereo cameras, disparity vectors can have arbitrary disparity components, both horizontal and vertical. The vertical component is expected to be small. However, for disparity (or depth), only the horizontal component is considered. The vertical component is necessary for the 3DRS-based disparity estimator to perform reliable.

To store the disparity in a depth image, sub-pixel accuracy should be kept. Assuming a vector range of $\pm$ 80 pixels with quarter pixel accuracy, the disparity can be described using at least 10 bits (including sign bit). Consequently, a 10 bit depth image is needed. In [2], the disparity range is automatically adjusted to full 8 bit range of depth image. However, this also adjusts the depth information originally contained in the input content.

Consider estimating disparities with quarter-pixel accuracy. To maintain the accuracy within the depth image and to store it as an integer value, disparity values are multiplied by 4. Each disparity vector $\vec{d} = (d_x, d_y)^T$ is converted to a disparity value D_v according to Eq. (3.5):

$$D_v\left(\vec{d}\right) = \frac{1}{accuracy} d_x + offset\,, \tag{3.5}$$

where $accuracy$ is 0.25 (for quarter-pixel accuracy) and $offset$ is equal to 512 (for unsigned 10-bit depth images). Subtracting an offset of 512 and a division by 4 is needed to retrieve the stored disparity.

3.3.2 *Depth processing*

The resolution of the depth image is much lower than the reference image. For each 8x8 block of pixels in the image, one disparity value is calculated. However, a frame-resolution depth image is required. Therefore, image up-sampling is needed.

The depth information is related to the reference image. This information determines the perceived depth position of each pixel (or object). Therefore, the objects in the depth image should align with the corresponding objects in the reference image. In [18], a depth up-sampling method is introduced, which consists of cross bilateral filters and cross bilateral up-samplers.

Bilateral filters

A bilateral filter is an edge-preserving smoothing filter. Bilateral filters have a wide range of applications: tone management, exposure correction, noise reduction, etc [17]. A bilateral filter is defined as:

$$I'_p = \frac{\sum_{q \in S} w_{p,q} \cdot I_q}{\sum_{q \in S} w_{p,q}}\,, \tag{3.6}$$

where

$$w_{p,q} = s(\|p - q\|)r(I_p - I_q) \ .$$

A bilateral filter is a non-linear filter which not only utilizes the domain properties, but also range properties. An output image I'_p is a weighted average of pixels at position q from input image I_q in a neighborhood S of position p. The weight calculation depends on a spatial distance function s and intensity range function r. The weights defined by the range function decreases with increasing intensity differences. In this way, edges are preserved. The weights defined by the spatial distance function decreases with increasing distance (i.e. Gaussian filter).

A bilateral filter has two kinds of weights. These weights are determined by the spatial and range functions and are based on *one* image. In a *cross* bilateral filter (CBF) the weights come from *two* kinds of images.

To up-sample the depth image, cross bilateral up-sampler is used. For each pixel in the low resolution depth image, 4 new depth values are calculated. Thus each repetition doubles the resolution. The contribution of a depth value is determined by the content of the reference image. This procedure is depicted in Fig. 3.5. The small squares in the

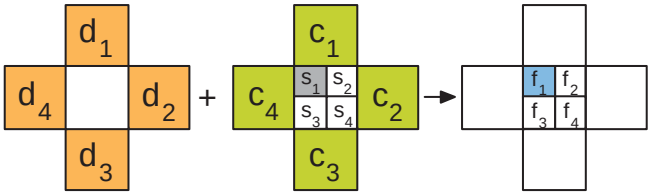


Figure 3.5: Cross bilateral up-sampling procedure.

centers have twice the resolution of the big squares. The low resolution depth image is shown on the left. The reference image is shown in the middle, for which both resolutions are needed. On the right, up-sampled depth image is produced, according to the following equation:

$$f_p = \frac{\sum_{q \in S} w_{p,q} \cdot D_q}{\sum_{q \in S} w_{p,q}} \ , \tag{3.7}$$

where

$$w_{p,q} = f(|s_p - c_q|) \ .$$

For example, f_1 is a weighted average of depth values d_1 through d_4. The weights are defined by the differences between s_1 and the corresponding c_q. The weight is large with small differences, and is small with large differences.

Up-sampling procedures

To increase the resolution, this cross bilateral filtering technique is used to filter and up-sample the depth image. The edges in a depth image should match the edges in the reference image. Using a cross-bilateral

filter, the edges in the reference image are preserved in the depth image. It smooths the pixels in the depth image, but it limits the smoothing to 'similar regions' in the reference image. This means that the weights of the range function are not based on the depth image, but on the reference image.

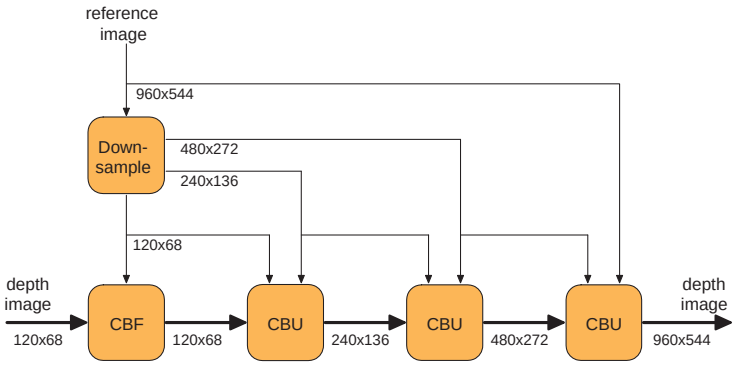


Figure 3.6: Depth processing chain, proposed in [18]

[18] proposes a step-wise approach to up-sample the depth image. Each step performs up-sampling by a factor of 2 both horizontal and vertical. This is depicted in Fig. 3.6. It is composed of a cross bilateral filter (CBF) and three cross bilateral up-samplers (CBU). Different filter configurations produces different results. For each filter or up-sampler an aperture can be specified.

The last stage in the up-sampling process is very computational intensive. The number of addressed pixels is 64 times higher than the image before up-sampling. To reduce the complexity, this stage can be replaced by a median-based block erosion stage.

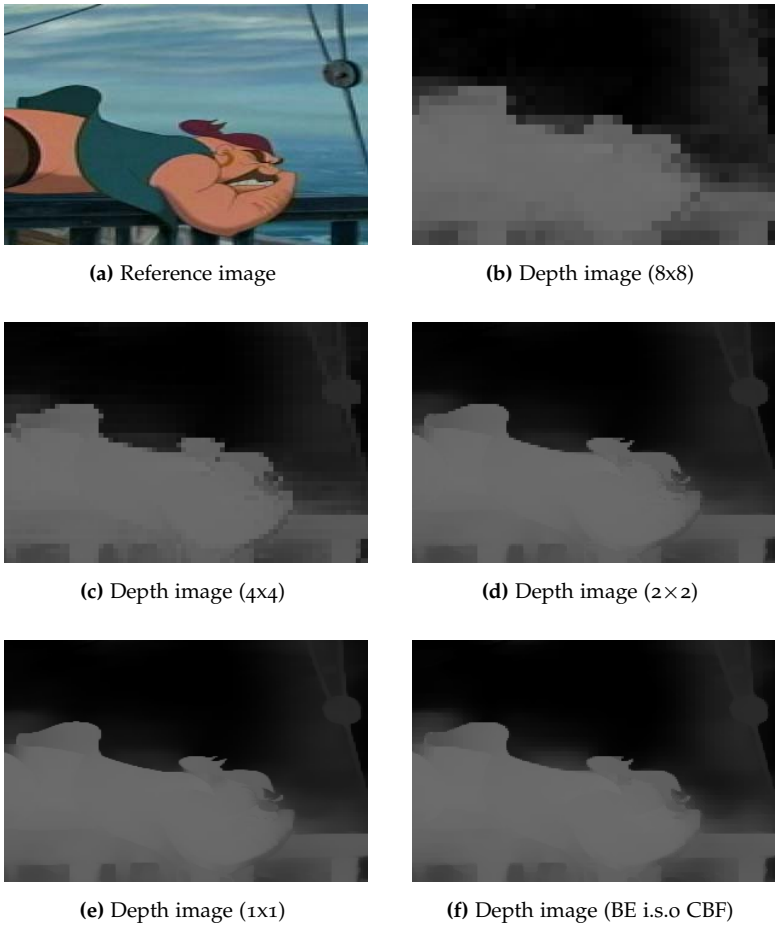
Depending on the quality of the low resolution disparity image, each CBF or CBU can be assigned different apertures. The aperture affects the size of the kernel of each filter [18]. To improve the quality of inaccurate disparity images, an extra dilation filter can be added after the first CBF or CBU. A dilation filter grows objects within the disparity image and is defined as:

$$D'_{p,q} = \max_{i,j=-1}^{1} \left[D_{p+i,q+j} \right] , \tag{3.8}$$

where D is the depth image and p, q determines the position within the disparity image. Consider a foreground object in a disparity image with an inaccuracy (i.e. a hole in the edge). The dilation filter causes the hole to be filled with the disparity values of the foreground object. However, the remaining of the object has grown to much now. To reduce the introduced error, the following up-samplers try to correct the disparity values.

3.3.3 *Results*

Fig. 3.7 shows the intermediate and final results of the depth up-sampling procedure.



(a) Reference image

(b) Depth image (8x8)

(c) Depth image (4x4)

(d) Depth image (2×2)

(e) Depth image (1x1)

(f) Depth image (BE i.s.o CBF)

Figure 3.7: Intermediate and final results of depth up-sampling procedure.

The quality impact of replacing cross bilateral up-sampling process in the last stage by a median-based block erosion is very low. The difference is shown in Fig. 3.7(e) and Fig. 3.7(f).

3.3.4 *Discussion*

The depth up-sampling procedure proposed by [18] produces very sharp depth images and is well suited for this algorithm. These images are also aligned with the reference images. The quality of the result depends on the quality of the disparity vector field, resulting from the 3DRS-based disparity estimator.

However, in certain situations, this up-sampling procedure produces artifacts, when the difference in luminance values between foreground and background objects is small. The depth value of the foreground object is then also projected into the background object. The range function has weights defined for luminance differences. These weights can be adjusted to suit different types of content.

3.4 VIEW SYNTHESIS

A view synthesis algorithm uses a reference image with a corresponding disparity image to synthesize an output image. This algorithm maps every pixel in the reference image to a new location according to its disparity value. This type of algorithm is commonly called a Depth-Image-Based Rendering (DIBR) algorithm.

A general *parallactic disparity mapping* formula is defined in [1]. This formula is used to synthesize images from a reference image. Consider the image captured by the left camera in Fig. 3.1 on page 17 and a corresponding disparity image is provided. Then, the projected points in the synthesized (right) image are determined by the mapping formula in Eq. (3.9). This formula is defined as:

$$s_1 = s_0 + \gamma\,(-d_1 + \delta)\ , \tag{3.9}$$

where

- γ is the depth gain,
- δ is the screen plane offset,
- d_1 is the disparity value.

Both γ and δ are control parameters. The γ parameter controls the range of observed depths. The δ parameter controls which part of the image will appear on the screen plane. However, in this solution these parameters have a fixed value, namely γ is equal to one and δ is equal to the offset value of Eq. (3.5).

Since only a reference with disparity image is available, two output images are synthesized to create stereoscopic vision. These two images resemble a viewing position horizontally left to the reference image (left-view image), and a viewing position horizontally right to the reference image (right-view image). Mapping of pixels only takes place in horizontal direction, there is no vertical displacement.

3.4.1 *Disparities to Perceived Depth*

Fig. 3.8 shows the relation between screen disparities and perceived depth. It also shows the relation between two points on the screen plane. If a point Q is perceived behind the screen plane, the projected point in the right-view image appears on right-hand-side of the projected point in the left-view image. If a point Q is perceived in front of the screen plane, the projected point in the right-view image appears on the left-hand-side of the projected point in the left-view image.

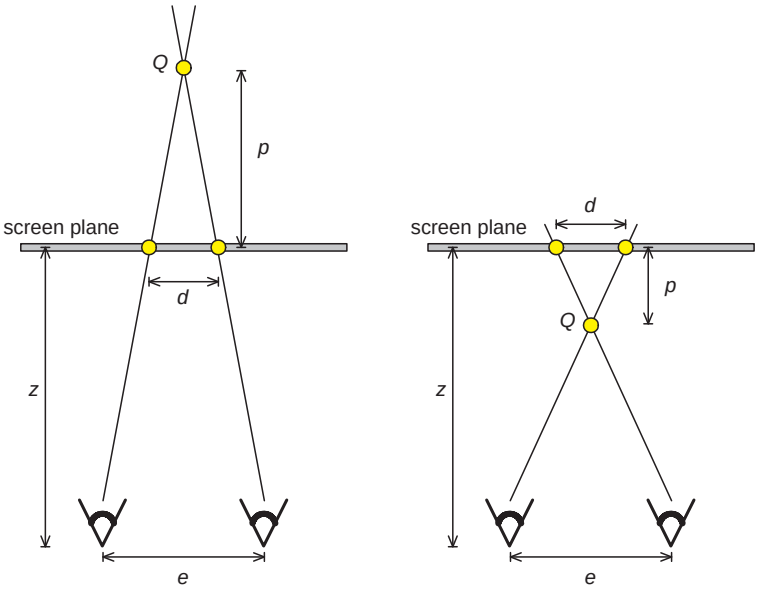
The perceived depth of point Q depends on the screen disparity d, the eye separation e, and on the viewing distance z. This relation is according to two formulas in [13] and are defined as:

$$p = \frac{z}{\frac{e}{|d|} - 1} \tag{3.10}$$

for perceived depth behind the screen plane, and

$$p = \frac{z}{\frac{e}{|d|} + 1} \tag{3.11}$$

for perceived depth in front of the screen plane. Where



(a) Point Q perceived behind the screen plane

(b) Point Q perceived in front of the screen plane

Figure 3.8: Screen disparities related to perceived depth.

p is the perceived depth,

z is the viewing distance,

e is the eye separation,

d is the screen disparity.

Table 3.1 shows screen disparities in pixels, assuming a 46 inch display with an effective resolution of 960×540 pixels. The distance between a viewer and screen plane is set to the recommended viewing distance of $2 \times Width$.

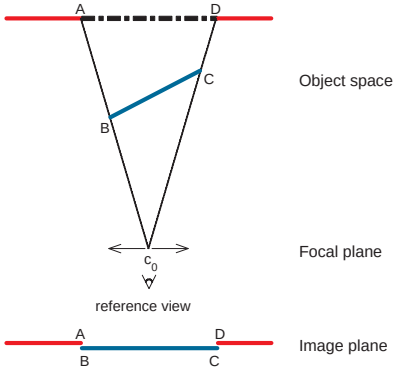
SCREEN SIZE (16:9)	RECOMMENDED VIEWING DISTANCE (cm)	DESIRED DEPTH (cm)	REQUIRED DISPARITY (mm)	REQUIRED DISPARITY (pixels)
46"	204 cm	50 cm	12.6 mm	11.9 px
		100 cm	21.1 mm	19.9 px
		204 cm	32.0 mm	30.2 px
		102 cm	21.3 mm	20.1 px

Table 3.1: Screen disparities for display with horizontal resolution of 960 pixels, interpupillary distance 64 mm.

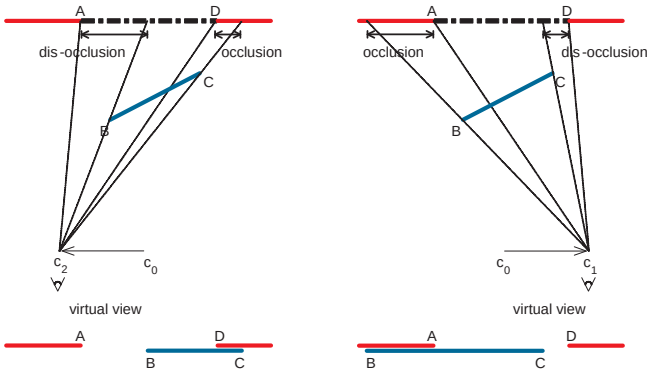
If one wants to perceive a point 50 cm behind the screen plane, the screen disparity in pixels should be equal to 11.9 pixels.

3.4.2 *Visibility*

Consider a set of objects in an object space. Two points in the space are visible to each other, if the line segment between those two does not intersect any object. This is depicted in Fig. 3.9. The object space is



(a) Reference-view position



(b) Left-view and Right-view position

Figure 3.9: Visibility problem with occlusion and dis-occlusion areas.

shown from the top, viewpoints are in the focal plane. Fig. 3.9(a) shows viewpoint c_0 of the reference image. If the viewpoint is changed from c_0 to c_1, occlusion and dis-occlusion areas become apparent. These are indicated on the right side in Fig. 3.9(b). The viewpoint changes to the right, but line segment BC moves to the left (shown in the image plane, bottom of the image). Next to that, the same line segment BC is also stretched, which is called a magnification. However, changing the viewpoint from c_0 to c_2, occlusion and dis-occlusion areas are on the opposite sides. Line segment BC has become shorter, which is called a minification.

The direction of change in the viewpoint determines which areas will be occluded and dis-occluded, determines which directions each object will move and determines whether objects stretch (magnifications) or shrink (minifications). The larger the depth difference between points A and B and points C and D, the larger the occlusion and dis-occlusion areas will be, respectively.

Occlusion areas

Consider the right-view position in Fig. 3.9(b). A part of the area on the left of point A is covered. This part is occluded by a foreground object. In order to correctly synthesize these images, only those pixels of objects should be displayed, which are only visible from the viewpoint.

Dis-occlusion areas

Consider the right-view position in Fig. 3.9(b). A part of the area on the left of point D, which is not visible in the reference view, becomes visible. This part is dis-occluded and are also referred to as holes.

A view synthesis algorithm using a reference image and disparity image, has to cope with occlusion and dis-occlusion areas. Discontinuities in disparity values causes these areas to occur. Occlusion areas can be handled automatically, by making sure that the occluded background is never displayed in the output images. Dis-occlusion areas become visible in synthesized views. Ideally, these areas are filled with true background information. However, this information is not available. Therefore, an in-painting method is required.

3.4.3 *Architecture of Existing Algorithm*

The algorithm is composed of three steps:

- *Warping and occlusion handling.* Pixels are mapped, or warped, to a new location. In stereo content, only horizontal displacement takes place. With proper warping, occlusion areas can be handled automatically.
- *Re-sampling.* Warped pixel values are not always perfectly aligned with the pixel grid of the output image. Therefore, a re-sampling procedure is needed. This algorithm calculates the contribution of each input pixel to several output pixels. This referred to as splatting.
- *Dis-occlusion handling.* During pixel warping, dis-occlusion areas exist, and need to be filled using an in-painting method. This algorithm uses background extrapolation. The closest background value is copied into each dis-occlusion area.

These steps are depicted in Fig. 3.10. Depth information is shown at the top. A disparity value equal to the offset δ has a perceived depth equal to the viewing distance. Consequently, no displacement of corresponding pixels is involved. The disparity of the foreground object leads to a displacement of pixels, shown by diagonal arrows during warping. The displacement is relative to the spatial position of the input pixel. See Eq. (3.9).

An output image is produced line-by-line. First, all pixels of one line are warped and re-sampled. Subsequently, dis-occlusion areas are filled. This is repeated for each line. The algorithm produces two output images: one image for the left-view position and one image for the right-view position. Therefore, producing lines is interweaved. After producing a line for left-view image, the same line for right-view image is produced.

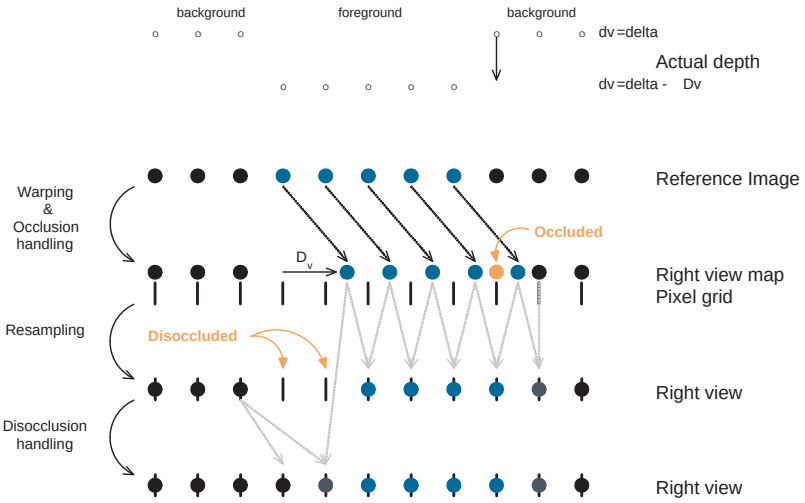


Figure 3.10: Architecture of view synthesis algorithm.

Warping and Re-sampling

During pixel warping stage, the displaced pixel may not exactly match the output pixel grid. Therefore, a re-sampling procedure is needed. Two re-sampling procedures have been evaluated:

- *Nearest neighborhood.* The closest output pixel is chosen to assign the pixel value. Although this is a very cheap procedure to do re-sampling, it produces visible artifacts in the temporal direction.
- *Splatting with integral weights.* A pixel value on the output grid is calculated as a sum of weighted input samples. These weights are defined by the intersection of the reconstruction filter area and the pre-filter area. The reconstruction filter area is defined by the midpoints surrounding the corresponding warped pixel. The pre-filter area is centered on the output grid pixel. This is depicted in Fig. 3.11.

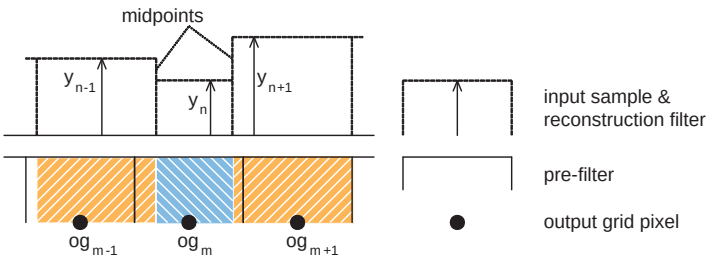


Figure 3.11: Splatting (re-sampling) with integral weights.

Warping pixels and re-sampling is considered to be one step. Each time a pixel is warped, it is immediately re-sampled to the output image.

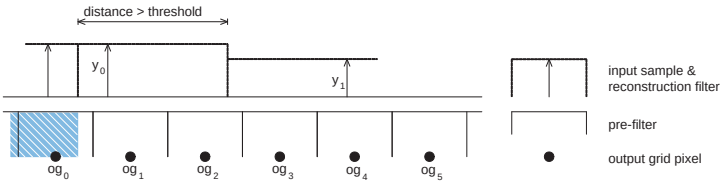
The output pixel value is a sum of weighted input pixel values. For example, input sample y_{n-1} contributes mostly to output pixel on grid position og_{m-1}, but also contributes to output pixel og_m, although with a small area.

Occlusion and Dis-occlusion Handling

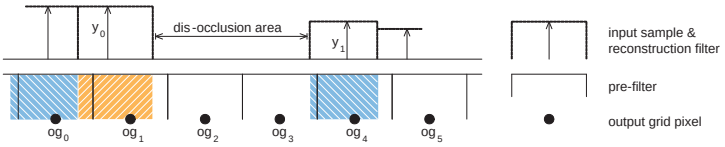
In case of producing the right-view image, output pixels are produced from right-to-left. The left-view image is produced from left-to-right. This direction is important in determining (dis-)occlusion areas.

Consider Fig. 3.9(b) and producing the right-view image in position c_2. Dis-occlusion areas are on the right-hand-side of foreground objects, where occlusion areas occur on the left-hand-side. In order to determine whether occlusion or dis-occlusion occurs during warping, the last warped position is recorded. If the next pixel (which is left to previous pixel) is not beyond the last warped position, occlusion occurred (thus belongs to background). Re-sampling of this pixel should be avoided. Therefore, the new warped position is adjusted to the last warped position.

If the new warped position is not occluded, the midpoints determine the areas for the contribution to output pixels. In case the distance between midpoints becomes too large, dis-occlusion occurs. See Fig. 3.12(a). Therefore, the midpoints are adjusted before splatting.



(a) Mapping pixel y_1 and detecting dis-occlusion



(b) Warping pixels y_0 and y_1 after midpoint adjustments and mapping next pixel

Figure 3.12: Dis-occlusion handling during mapping, warping.

After splatting, midpoints are adjusted again and information about dis-occlusion area is added to a list for deferred processing. This is depicted in 3.12(b).

The difference between two successive midpoints is a measure for dis-occlusion detection. If the difference is larger than 1 pixel, dis-occlusion is detected. However, this would lead to many small dis-occlusion areas, where small magnifications occur. By increasing the threshold from 1 to 1.5 pixel difference [3] (internal document), the number of dis-occlusion areas decreased significantly. This also affects the re-sampling procedure: one input sample can now affect at most three output samples. Therefore, the re-sampling procedure is modified. The contribution to the third output pixel is also calculated.

Background extrapolation is used to fill dis-occlusion areas. The closest background pixel to the foreground object is used to fill in this area. As with re-sampling of warped pixels, contributions to each output pixel in dis-occlusion areas are calculated and assigned.

3.4.4 *Picture Quality*

To address the picture quality of this algorithm, the reference image and its disparity image is depicted. See Fig. 3.13, which are used by [23].

(a) Reference image **(b)** Disparity image

Figure 3.13: Image plus depth frames from ballet sequence.

The most visible artifacts are the horizontal streaks. In Fig. 3.14 these artifacts are shown. These streaks are the consequence of dis-occlusion handling. The edge of the foreground object in the disparity image matches the edge in the reference image. However, edges in the reference image are not infinitely sharp and are in the discrete domain. Therefore, dis-occlusion area is filled using mostly foreground object pixels!

(a) Highlighted area one **(b)** Highlighted area two

Figure 3.14: Artifacts in synthesized views.

This artifact is reduced by transforming the disparity image prior to synthesizing the two views. The transformation consists of a horizontal dilation filter, followed by a horizontal blur filter. The consequence is, that foreground objects in the disparity image grow, and therefore, covers the corresponding objects in the reference image. The blur filter causes the dis-occlusion area to be distributed over multiple smaller dis-occlusion areas. Therefore, dis-occlusions are filled with multiple background pixels. The quality of the synthesized views is significantly improved. See Fig. 3.15.

The dilated disparity value D_d at position $\vec{x} = (x, y)^T$, is defined as:

$$D_d(\vec{x}) = \max_{i=-3}^{3} D\left((x+i, y)^T\right), \tag{3.12}$$

(a) Highlighted area one

(b) Highlighted area two

Figure 3.15: Pre-processing depth image improves quality of synthesized views.

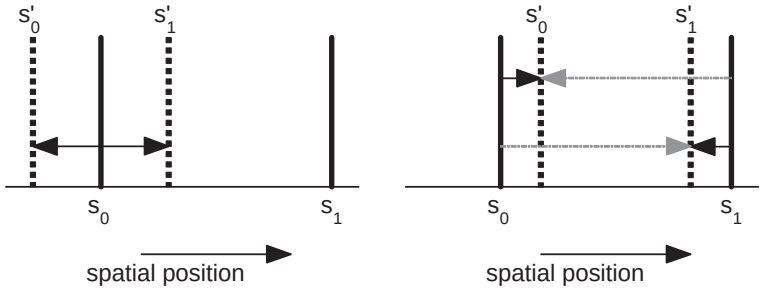
where $D(\vec{x})$ is the original disparity value. The horizontal blur filter is a 1×3 box-filter defined as:

$$h_{hor.LPF} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}. \tag{3.13}$$

The in-painting method (background extrapolation) assumes a homogeneous area surrounding the foreground object. In cases where the background is not homogeneous, visible artifacts occur like one is depicted in Fig. 3.15(b).

3.4.5 *Input perspective versus output perspective*

The intermediate format consists of a reference image and a depth image. The reference image is either the left view or the right view from the stereo signal. From a perspective point of view, this may not be the best choice. The view synthesis algorithm produces two new views based on the information in the intermediate format. One view on the left side, s_0', and one view on the right side, s_1', is produced, w.r.t. the reference image. This is depicted in Fig. 3.16(a). The perspective



(a) Images on positions s_0' and s_1' are created solely on image plus depth from position s_0

(b) Images plus depth available for both positions s_0 and s_1 to create images on positions s_0' and s_1'

Figure 3.16: Example of spatial positions of left and right perspectives, which changes with image-plus-depth view synthesis.

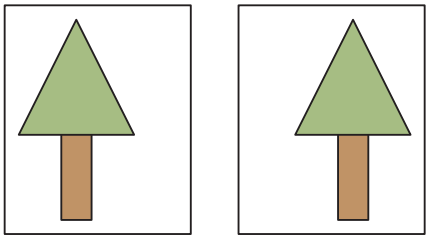
has changed by producing left-view and right-view images using the intermediate format.

In order to maintain the perspective of the input material, one could synthesize the reference view that is in the exact middle of s_0 and

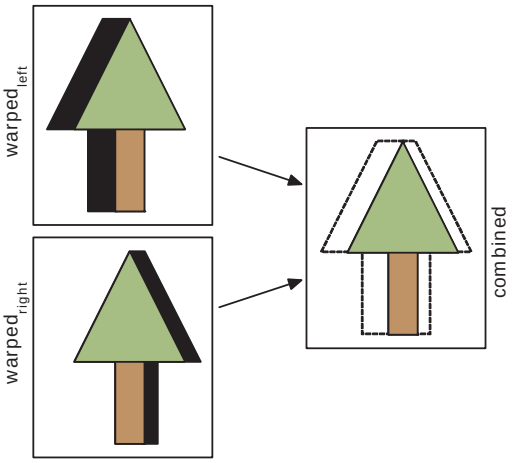
s_1. Furthermore, the disparity image should match this synthesized reference image.

Another way to maintain the perspective, is to generate one view with the left-view image as reference image and generate the other view with the right-view image as the reference image. This is depicted in Fig. 3.16(b). The black arrows indicate the spatial distance from each view position. In comparison to Fig. 3.16(a), the spatial distance of both views to the reference image becomes smaller. Consequently, occlusion and dis-occlusion areas also become smaller, and therefore, fewer artifacts are visible.

To improve picture quality, both input images can be used to generate each output image. This is also depicted in Fig. 3.16(b) with the gray arrows. See Fig. 3.17 for an illustration. The images of left-view and



(a) Left-view image **(b)** Right-view image



(c) Combine two warped images, dis-occlusion areas reduced

Figure 3.17: Synthesize view from warped left-image and warped right-image to improve picture quality in dis-occlusion areas.

right-view positions are warped to the desired view-position. Consequently, the foreground object (tree) is warped to exact same position in both output images. The dis-occlusion areas are on different sides. Therefore, dis-occlusion areas (i.e. from left-view warped image) can be filled with information of the other warped image (i.e. from right-view warped image).

This only holds when the new spatial positions are between the existing view positions. This method requires two disparity images, one for each view.

4 ARCHITECTURE MAPPING

Previous chapter defined a complete processing chain for a stereo signal conversion algorithm. [15] and [18] show that complexity analysis has been performed on 3DRS-based disparity estimator and joint-bilateral depth up-scaling, respectively. The complexity of existing view generation algorithm (also referred to as Integral Weights algorithm) has not been analyzed yet. This chapter elaborates on this existing view synthesis algorithm, which will be mapped onto a specific high-end video processing platform, completed with complexity analysis.

4.1 TARGET PLATFORM

The existing view synthesis algorithm is mapped onto a high-end video processing platform. This platform consists of multiple TriMedia TM3282 media-processors with co-processor support.

The TriMedia TM3282 media-processor has a Very Long Instruction Word (VLIW) architecture. This processor can also access the Motion Estimation and Motion Compensation 1 (MEMC1) co-processor. Both access main memory via the memory interface. See Fig. 4.1. Table 4.1 gives an overview of the main architectural features of the TM3282 processor.

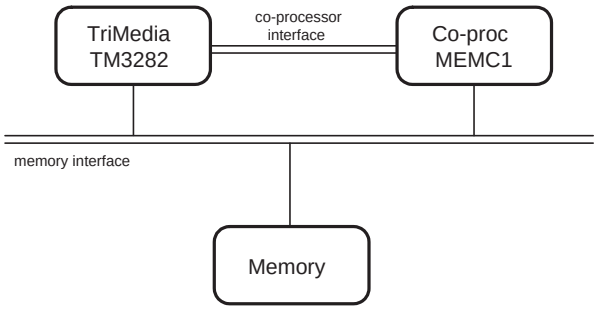


Figure 4.1: Schematic overview of high-end video processing platform.

Next to the RISC-like operations, the instruction set also contains custom operations that can improve performance. These custom operations include operations with subword parallelism (i.e. dual-operations or quad-operations) and 'two-slot' operations. This last type of operations uses two neighboring issue slots and therefore, are specified with at most 4 source registers and 2 destination registers. These custom operations are extensively used and almost all the operations can be individually guarded. The media-processor also contains two cache memories, one for data and one for instructions.

The TM3282 is connected to the MEMC1 co-processor through a separate co-processor interface. The co-processor provides efficient

ARCHITECTURAL FEATURE	QUANTITY
Architecture	8 issue slot VLIW guarded RISC-like operations pipelined processor design
Address width	32 bits
Data width	32 bits
Register file	128 general purpose registers
SIMD capabilities	1×32-bit, 2×16-bit, 4×8-bit
Instruction cache	32 Kbyte, 128-byte lines 8-way set-associative LRU replacement policy
Data cache	32 Kbyte, 128-byte lines 4-way set-associative LRU replacement policy

Table 4.1: Architectural features of TM3282 media-processor.

access to two-dimensional video images and has a cache-based design. Both are connected to main memory via the memory bus. This enables the TM3282 processor to offload work to the co-processor and become more a 'control' processor. The co-processor has a fixed instruction set which comprises of operations for different types of applications, specifically for motion estimation and compensation. Next to that, also specific operations related to H.264 codec are available.

The algorithm is mapped onto this media-processor. Currently, it is unsure whether or not the co-processor can help to enhance the execution of the view synthesis algorithm. Next to mapping and complexity analysis, usage of co-processor is evaluated.

4.2 MAPPING ALGORITHM TO PLATFORM

In previous chapter, the details of the existing view synthesis algorithm are explained. This algorithm is mapped onto the TriMedia TM3282 media-processor and several implementation aspects are discussed.

4.2.1 *Implementation Aspects*

Fixed-point Notation

A full HD frame has a resolution of 1920×1080 pixels. To address each pixel in a single line, 12 bits are required. To specify sub-pixel positions, 4 bits are used. This is also denoted as 12.4 notation, indicating how many bits before and after decimal sign are used. Consequently, to position a pixel at a sub-pixel position, 16 bits are required. Hence, two positions can be stored in a 32-bit word. Several macros are defined to

convert values to and from this 12.4 fixed-point notation. These macros are implemented as shift operations.

Midpoint Calculation

The area of the reconstruction filter intersected with area of the pre-filter determines the contribution of input pixels to output pixels (see Fig. 3.11 on page 32). To define the area of the reconstruction filter, midpoints are calculated. In turn, these midpoints depend on the warped pixels, which are defined as sum of input pixel position and its disparity value. Pseudo code is given in Listing 4.1.

```
//store previous warped position
prv_WarpedPos = nxt_WarpedPos;
//calculate new warped position
nxt_WarpedPos = (cur_PixelPos << 4) - DepthValue[cur_PixelPos];
```

Listing 4.1: Calculating new warped pixel position

Midpoints are calculated as sum of current warped position and previously warped position and divided by two, see pseudo code in Listing 4.2.

```
//store previous midpoint
prv_MidPoint = nxt_MidPoint;
//calculate new midpoint
nxt_MidPoint = prv_WarpedPos + nxt_WarpedPos) >> 1;
```

Listing 4.2: Calculating new midpoint

Re-sampling

Re-sampling is performed by splatting input pixels to the output pixel grid at warped positions. Since magnifications may occur and midpoints determine the reconstruction filter area, more than one output pixels may be affected. Therefore, these pixels are identified first.

The area of the warped pixel surrounded by two midpoints (reconstruction filter area), can be divided over a maximum of three output pixels. The area that intersects with pre-filter area of each output pixel, can vary between 0 (no intersection) to 16 (intersected area equals pre-filter area). Remember that 4 bits are used to determine sub-pixel positions. For each affected output pixel, corresponding intersections are calculated.

The contribution to each output pixel is calculated by the product of input sample and each corresponding area. Pseudo code is given in Listing 4.3.

```
//calculate contributions to output pixels
output[pos1] = output[pos1] + ((cur_Y * area1) >> 4);
output[pos2] =                (cur_Y * area2) >> 4;
output[pos3] =                (cur_Y * area3) >> 4;
```

Listing 4.3: Calculating new midpoint

Corresponding chrominance sample is not re-sampled. This sample is directly stored at position closest to the warped position of luminance sample. (i.e. nearest neighborhood).

Occlusion Handling

Consider producing an image in the left-view position, each line is produced from left to right. Foreground objects will be shifted more towards the right than background objects. Occlusion areas occur on the right side of foreground objects. For each warped pixel, its position is checked against the previously warped pixel position. If the new warped pixel does not go beyond the previous position, occlusion occurs. Therefore, this warped pixel should not contribute to the output image. The new midpoint should be set equal to the previous warped pixel position.

Since the current pixel is warped and previous warped pixel will be re-sampled, the new midpoint is adjusted to the maximum of the two. Pseudo code is given in Listing 4.4.

```
//keep extent
nxt_MidPoint = MAX(nxt_MidPoint, prv_WarpedPos);
```

Listing 4.4: Calculating new midpoint

This assures that the previous pixel is correctly re-sampled up to its warped position. If subsequent warped pixels are also occluded, the MAX-operator keeps the midpoint on the exact same position. Consequently, the area between current and previous midpoints is zero. Hence, no contribution is made to the output image. Warping and re-sampling continues without actually detecting occlusions.

Dis-occlusion Handling

Dis-occlusion is detected when the distance between two adjacent midpoints is larger than a threshold (see Fig. 3.12 on page 33). This threshold is set to 1.5 pixel. Whenever a dis-occlusion is detected, the new calculated midpoint is adjusted according to Listing 4.5.

```
//adjust midpoint
nxt_MidPoint = prv_MidPoint + 16;
```

Listing 4.5: Calculating new midpoint

This value added is equal to exactly one output pixel width (in fixed-point notation).

After input sample is re-sampled to the output image, midpoint is adjusted again according to Listing 4.6.

```
//adjust midpoint
nxt_MidPoint = nxt_WarpedPos - 8;
```

Listing 4.6: Calculating new midpoint

The area that is skipped by the splatting procedure is dis-occlusion area. Starting and ending positions of the dis-occlusion areas are stored in a separate list. Furthermore, the corresponding input location is stored to determine the value for dis-occlusion handling. This list is used after mapping and re-sampling, to fill dis-occlusion areas using background extrapolation.

4.2.2 *Data-cache Behavior*

The data cache provides a smal region of fast memory between the processor and the external memory. It retains data that has been accessed recently, so it can be accessed faster the next time it is needed. The TM3282 media-processor contains a data cache of 64kb with 4-way set associative architecture and cache lines of 128 bytes. This means that a single cache line can be placed at one of 4 positions within the cache-set. Each *way* has 16kb (64kb/4way) of cache, each cache line is 128 bytes. Hence, there are 128 cache-sets.

Which set is chosen depends on the load address. Since input data is sequentially stored in memory, data will also be stored sequentially in cache, i.e. running cyclicly through the cache.

Cache requirement

Data is mapped into the cache. Table 4.2 shows the data cache requirement for this algorithm. This table shows that the total data cache

DESCRIPTION	CURRENT	FUTURE	CACHE SET
Reference Image (Y)	1×960	1×960	0
Reference Image (UV)	1×960	1×960	0
Depth (Y)	1×960	1×960	32
Output Image R (Y)	1×960	1×960	96
Output Image R (UV)	1×960	1×960	96
Output Image L (Y)	1×960	1×960	64
Output Image L (UV)	1×960	1×960	64
Hole	2×960	0	0
Total	9×960	7×960	

Table 4.2: Cache requirement of Integral Weights algorithm.

requires almost 9kb for the current line. Another 7kb cache is required for the upcoming line.

With proper aligning of data into the cache, a reduction of data cache stall cycles can be achieved. This means, data is read only once into cache and output is written only once into main memory. The last column shows the alignment of data in the cache (i.e. data of depth image should start at address ending on 0x1000).

Pre-fetching

Typically, when retrieving data in video processing algorithms, nearby data will be needed soon. Hence, when data is retrieved, retrieve also data nearby. This is known as pre-fetching. Each time a load occurs and the load address falls within a pre-defined region, the processor checks whether *nearby data* is already in the cache. Whenever that is not the case, a pre-fetch request is issued. Nearby data is located at the current load address plus *stride*. The stride is a fixed value.

This algorithm produces output images line by line. Except for dis-occlusion handling, input data is read sequentially and output data is written sequentially. This depends upon discontinuities in the depth image.

By default pre-fetching is enabled for one region. This region covers the entire memory space. Therefore, input data is in cache by the time it is needed. Due to proper cache alignment, data remains in cache uninterrupted until the line is processed.

Pre-allocation

Writing data to memory cannot be done without first allocating a cache line. For example, if one wants to update bytes in memory, the entire cache line needs to be copied to the cache. After updating the cache, this line is ready to be written back to memory. This is known as allocate-on-write-miss policy. As pre-fetching helps loading nearby data that will be needed soon, pre-allocation helps in allocating nearby data that will be (over)written soon.

By default, pre-allocation is also enabled for the entire memory space. Therefore, data is pre-allocated sequentially for the output images. However, producing the right-view image occurs from right to left. The addresses are sequentially decreasing within each line. Consequently, pre-allocation has no advantage and write-misses occur on the first access of every cache line. With a region-based cache architecture, pre-allocation can still be used. To assure proper pre-allocation, an extra pre-allocating region is defined with a *negative stride*. Hence, the processor pre-allocates cache lines with decreasing memory addresses, which reduces write misses significantly.

4.2.3 *Complexity Analysis*

To know whether this algorithm will execute properly on a TriMedia processor, the analysis of complexity is necessary. The two most important aspects are: *instruction cycle count* and *bandwidth usage*. The following assumptions are made:

- Resolution of reference image, depth image and output images are all 960x540.
- The color space of reference image is YUV422[1], and depth image is only Y component.
- The color space of the output images is also in YUV422.
- Frame-rate is 60 frames per second.
- Memory accesses have delays of 100 cycles.
- Cache size of TM3282 processor is set to 64kb.

The analysis of complexity of this algorithm is performed using the TriMedia Compilation and Simulation System (TCS). This system enables the developer to create highly optimized multimedia applications in C (and C++) programming language. It provides a suite of system software tools to compile and debug media applications, analyze and optimize performance, and simulate execution on a TriMedia processor.

1 For each two horizontal luminance (Y) samples, there is only one chrominance (U and V) sample.

Expected Bandwidth Usage

A lower bound can be calculated for the bandwidth usage, see Table 4.3 (denoted in MBytes per second). As depicted in Fig. 4.1 on page 37, memory is an external part which can be accessed by multiple parties. In the ideal case, one wants to read the input data (reference and depth image) exactly only once to memory. Furthermore, the output data (left and right frame) should be written once to memory.

DESCRIPTION			1 FPS	60 FPS
Reference image 2× 960×540		0.99		
Depth image 1× 960×540		0.49		
	subtotal		1.48	89.0
Left view 2× 960×540		0.99		
Right view 2× 960×540		0.99		
	subtotal		1.98	118.7
		Total	3.46	207.6

Table 4.3: Lower bound on bandwidth usage.

Analysis

The complexity numbers are extracted using the cycle-accurate machine-level simulator and a trace analysis tool.

Table 4.4 below shows the processor load requirement. The cycle

DESCRIPTION	CYCLES	INSTR.	STALLS	I$	D$	ILP
Integral Weights algorithm	671.0	578.2	92.8	0.9	91.8	6.20
• Mapping and Re-sampling	628.2	539.4	88.9	0.6	88.2	6.49
• Dis-occlusion handling	33.1	32.4	0.7	0.2	0.5	2.60

Table 4.4: Instruction cycles of Integral Weights algorithm.

count is composed of instruction cycles and stall cycles. The latter part is again divided into instruction cache stall cycles and data cache stall cycles. All numbers in this table are expressed in MHz, which are obtained by the product of cycle count per frame and the frame rate. The last column shows the Instruction Level Parallelism (ILP), which is an average of executed instructions per cycle. As the TM3282 media processor has 8 issue slots, the maximum ILP achievable is 8.

Next to the processing requirements, bandwidth usage is also important. Bandwidth usage is detailed in Table 4.5. The total bandwidth usage is composed of read and write bandwidth usage. Read bandwidth can be divided into instruction read and data read bandwidth. All numbers in this table are expressed in MBytes/second, which are obtained by the product of bandwidth count per frame and the frame rate.

DESCRIPTION	BANDW.	WRITES	READS	INSTR.	DATA
Integral Weights algorithm	212.0	117.9	49.1	4.9	89.2

Table 4.5: Bandwidth usage of Integral Weights algorithm.

4.2.4 *Discussion*

Assuming a TriMedia TM3282 media processor has 500 MHz of processing capacity, the complexity analysis shows that this algorithm will not run real-time.

By evaluating the possibilities of the co-processor and the required computations, the co-processor cannot be used efficiently to enhance the execution of this algorithm. The instruction set is fixed and the only possible advantage could be loading and storing of, respectively, input and output images. Therefore, it is not used and left out in this solution.

In order to run this algorithm real-time on this platform, complexity needs to be reduced. There are several ways to accomplish this. The remainder of this chapter will elaborate on techniques to reduce complexity of the view synthesis algorithm.

4.3 COMPLEXITY REDUCTIONS

There are several techniques to reduce complexity. The first technique is to generate views on a block basis. This means that each block will have one single depth value. The second technique is to come up with a different algorithm, which has comparable picture quality. A third technique is to integrate the co-processor and extend its instruction set.

Reducing computational complexity by changing algorithmic aspects also affects picture quality. Therefore, evaluating the picture quality is another important aspect of reducing complexity. This leads to tradeoffs between (reduced) computational complexity and picture quality.

4.3.1 *Block-based Algorithm*

Approach

In a block-based algorithm, each block of pixel data is taken as unity data. Each pixel within a block is processed equally. In the view synthesis algorithm, this means that each pixel within a block has the same depth value. The use of this assumption can greatly reduce the complexity.

The instruction set of the TM3282 processor has custom operations, which support subword parallelism. These are quad and dual operations and operate on four 8-bit values and two 16-bit values respectively. Therefore, processing with a block-based approach on 4×4 blocks and 2×2 blocks are proposed.

In order to quickly evaluate the performance of this block-based algorithm, prototyping is used. This view synthesis algorithm generates

a new left-side view and a new right-side view based on the reference image. Prototyping functionality reflects the largest part of an algorithm, from which performance numbers can be extracted with a certain accuracy.

Benefits

The benefits of generating views using a block-based approach are summarized below:

- *Fewer warped positions to calculate.* New warped position needs to be calculated once per block. If the new position is known for one pixel within a block, then the position of every other pixel is also determined.
- *Depth image is smaller.* Only one depth value per block saves a factor 4 (2×2 blocks) and a factor 16 (4×4 blocks) in bandwidth for the depth image.
- *Equal depth value within a block.* In contrary to the pixel-based approach, each pixel within a block has the same depth value. Each pixel within a block is also equidistant located, which simplifies the re-sampling procedure. Except for contributions of the outer pixels, pixels contribute only to two output pixels. Each block consists of multiple lines, and the weights calculated for one line also applies to all other lines within the block.

Picture Quality Impact

Without having the actual block-based view synthesis algorithm, the pixel-based (existing algorithm) is used to simulate and evaluate block-based processing and its picture quality. Each pixel within a block of pixels, i.e. 4×4 pixels, is assigned a same depth value.

To identify the picture quality impacts of block-based processing, a dilation filter like Eq. (3.12) and a blur filter like Eq. (3.13) are not used. The artifacts are expected to be similar with the pixel-based algorithm.

Two major artifacts that degrade picture quality significantly. These artifacts are staircasing and streaks in occlusion and dis-occlusion areas, respectively. These artifacts are shown in Fig. 4.2 and are explained thereafter.

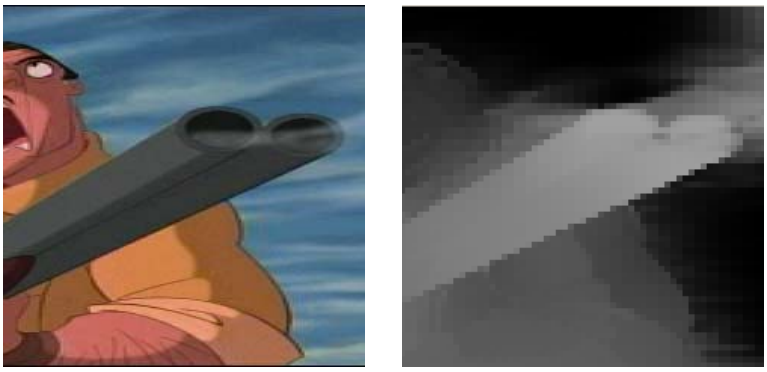
Edges in a depth image divides objects into foreground and background objects. In block-based view synthesis, one depth value warps a block of 4×4 pixels (or 2×2). The edges in a depth image should correspond to edges in the reference image. Due to the granularity of blocks, edges of an object in the reference image are also divided into foreground and background parts. See Fig. 4.3.

The following observations are made:

- Small parts of the background become part of the foreground object.
- Small parts of the foreground object become part of the background.

The artifacts visible in Fig. 4.2 occur, since the depth image divides foreground from background in an incorrect way.

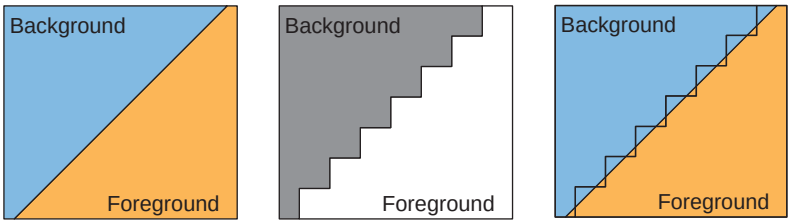
Consider producing a left-view image based on the content depicted in Fig. 4.3. Foreground objects move more towards right than back-



(a) Reference image **(b)** Depth image

(c) Left view image **(d)** Right view image

Figure 4.2: Artifacts in block-based view synthesis, blocks of 4×4 pixels.



(a) Reference image **(b)** Depth image **(c)** Overlay image

Figure 4.3: High-resolution image is divided into foreground and background parts.

ground objects. Therefore, *dis-occlusion* occurs on the left side of foreground objects. The parts that are regarded by the depth image as background, but belong to the foreground object are warped according to background depth value. Since dis-occlusion handling involves background extrapolation, certain dis-occlusion areas are filled with foreground-regarded-as-background parts. This produces streaks.

Consider producing a right-view image based on content depicted in Fig. 4.3. Foreground objects move more towards left than background objects. Therefore, *occlusion* occurs on the left side of foreground objects. The parts that are regarded by depth image as background, but belong to foreground object are not visible anymore after warping. These

parts are occluded and produces staircasing. A solution to reduce these artifacts is proposed below.

These artifacts are removed by applying a dilation filter with 3×3 kernel on the depth image. Dilation causes foreground objects to extend in area, therefore fully covering the object in the reference image. The dilation filter is defined as:

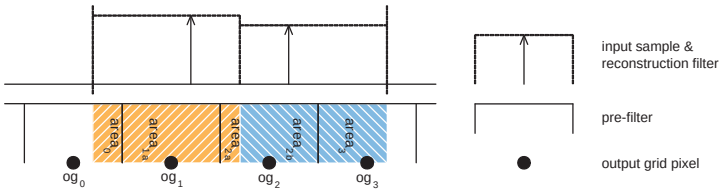
$$D'_{p,q} = \max_{i,j=-1}^{1} \left[D_{p+i,q+j} \right] , \tag{4.1}$$

where D is the depth image and p, q determines the position within the image. It should be noted that this filter is applied on a 4×4 (or 2×2) lower-resolution depth image. This filter is similar to a dilation filter with a 4×4 (or 2×2) kernel on full-resolution depth image.

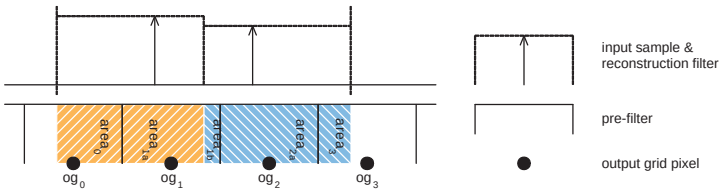
The streaks and staircases are removed, but a new artifact is introduced: a halo-artifact. Because the depth image is "grown", more background is regarded as foreground object. Therefore, more background is warped along with foreground objects. This artifact is much less visible than streaks and staircases.

Re-sampling of 2×2 Blocks

Each time a warped position is calculated for a block of 2×2 pixels, this group will be splatted. Consider Fig. 4.4, where the distance between midpoints is maximal. In other words, this is the maximum affected area, without dis-occlusion being detected.



(a) Pixels are mapped to the right of nearest pixel grid position



(b) Pixels are mapped to the left of nearest pixel grid position

Figure 4.4: Mapping and re-sampling of next two pixels.

The dis-occlusion threshold is equal to 1.5 pixel. Since two adjacent pixels are warped together, the midpoint between these pixels is 0.5 pixel. This leaves a maximum of 1.0 pixel on each side of both pixels, without detection of dis-occlusion.

In Fig. 4.4 two situations can occur. Situation in Fig. 4.4(a) occurs, if the first luminance pixel is mapped on the right-hand-side of the corresponding output pixel. Otherwise, the situation in Fig. 4.4(b) occurs. To generalize the re-sampling procedure for both situations, areas are assigned, which will contribute to the output pixel. The pseudo code in Listing 4.7, is used to determine the contributions to each output pixel.

In each of the situations, either $area_{1b}$ or $area_{2a}$ (see Fig. 4.4) will be equal to zero. Therefore, no contribution will be made.

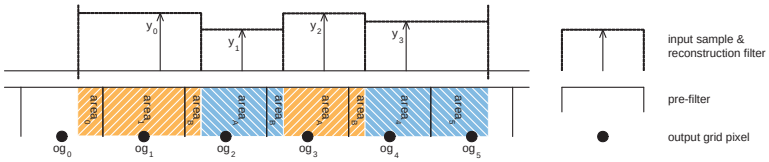
```
//splat two adjacent luminance pixels
og0 = og0 + ((area0  * y0)               ) >> 4;
og1 = og1 + ((area1a * y0) + (area1b * y1)) >> 4;
og2 =       ((area2a * y0) + (area2b * y1)) >> 4;
og3 =       (                (area3  * y1)) >> 4;
```

Listing 4.7: Re-sample two adjacent pixels

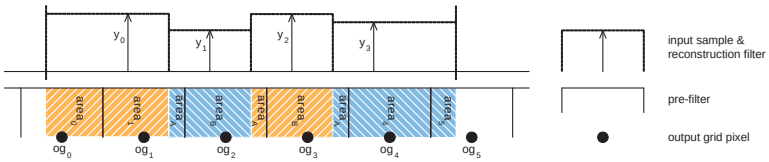
This is repeated for each line within a block. The coefficients remain equal, only the luminance samples vary per line.

Re-sampling of 4×4 Blocks

Each time a warped position is calculated for a block of 4×4 pixels, this group will be splatted also. Consider Fig. 4.5, where the distance between the first and second midpoint and fourth and last midpoint is maximal. In other words, this is the maximum affected area, without dis-occlusion being detected.



(a) Pixels are mapped to the right of nearest pixel grid position



(b) Pixels are mapped to the left of nearest pixel grid position

Figure 4.5: Mapping and re-sampling of next four pixels.

The dis-occlusion threshold is equal to 1.5 pixel. Since four adjacent pixels are warped together, all midpoints between all four pixels are on 0.5 pixel distance. This leaves a maximum of 1.0 pixel on each side of group of pixels, without detection of dis-occlusion.

Furthermore, here two situations can occur, see Fig. 4.5. Situation in Fig. 4.5(a) occurs, if the first luminance pixel is mapped on the right-hand-side of the corresponding output pixel. Otherwise, the situation in Fig. 4.5(b) occurs. The re-sampling procedure cannot be generalized and a distinction between both situations is made. The pseudo code in Listing 4.8, is used to determine the contributions to each output pixel. Each four warped pixels are equidistant located. Therefore, midpoints of these pixels are also equidistant. Consequently, areas defined by inner two input pixels, have equal sized areas. These areas are denoted by areaA and areaB.

These calculations are repeated for each line within a block. The coefficients remain equal, only the luminance samples vary per line.

```
//splat four adjacent luminance pixels
if(situation a) {
    og0 = og0 + ((area0 * y0)              )>>4;
    og1 = og1 + ((area1 * y0)              )>>4;
    og2 =       ((areaB * y0) + (areaA * y1))>>4;
    og3 =       ((areaB * y1) + (areaA * y2))>>4;
    og4 =       ((areaB * y2) + (area4 * y3))>>4;
    og5 =       ((area5 * y3)              )>>4;
} else { //situation b
    og0 = og0 + ((area0 * y0)              )>>4;
    og1 = og1 + ((area1 * y0) + (areaA * y1))>>4;
    og2 =       ((areaB * y1) + (areaA * y2))>>4;
    og3 =       ((areaB * y2) + (areaA * y3))>>4;
    og4 =       ((area4 * y3)              )>>4;
    og5 =       ((area5 * y3)              )>>4;
}
```

Listing 4.8: Re-sample two adjacent pixels

4.3.2 *Processor Optimizations*

Re-sampling of 2×2 Blocks

Consider re-sampling of a single block of 2x2 pixels, and the pseudo code in Listing 4.7. Calculating these four output pixels for one line is mainly in the form of

$$((a * b) + (c * d)) >> 4$$

One specific custom operation, SUPER_USCALEMIXUI_QUAD8, is capable of performing four of these sum-of-products at once (with shifting and clipping), namely $((a * b) + (c * d)) >> 6$. To rectify the shifting operator, each coefficient is multiplied by factor 4. Since coefficient values are between 0 and 16, coefficients will never overflow.

The coefficients (or areas) are first calculated, followed by packing into two 32-bit words.

Both input samples are extrapolated to serve as input for all four calculations. This is implemented using SUPER_MERGEL_QUAD8 operation.

Because the first two output samples need to be accumulated to possibly previously calculated output pixels, an add-operation is added: DSPUADD_QUAD8. This operation computes four separate sums of four 8-bit pairs. The results are also clipped.

All these steps are depicted in Fig. 4.6.

These three operations are also executed for the second line of 2 pixels in the block. The coefficients remain unchanged. Operation DSPUADD_QUAD8 leaves room for two more add-operations. However, to incorporate this, several more instructions are needed to select, pack and store affected values.

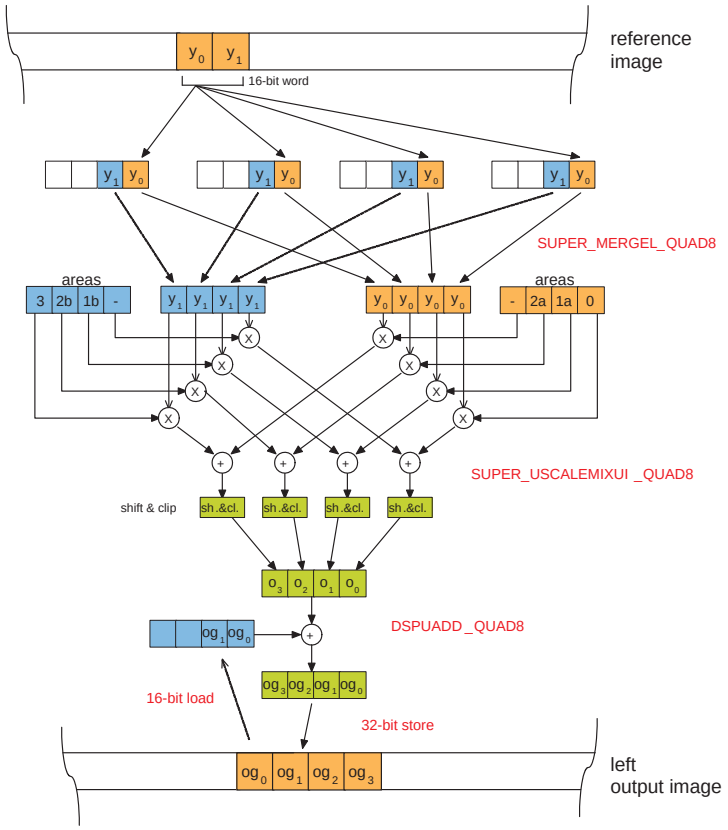


Figure 4.6: Optimized re-sampling procedure for 2×2 blocks.

Re-sampling of 4×4 Blocks

Now consider re-sampling of a single block of 4×4 pixels, and see pseudo code in Listing 4.8). Two situations can occur, in both cases six output pixels are calculated per line, and are mainly in the form of

$$((a * b) + (c * d)) >> 4$$

As before, custom operation SUPER_USCALEMIXUI_QUAD8 is used. This operation is capable of performing four of these sum-of-products at once (with shifting and clipping), namely $((a * b) + (c * d)) >> 6$. To rectify the shifting operator, each coefficient is multiplied by factor 4. Since coefficient values are between 0 and 16, coefficients will never overflow.

To warp four adjacent pixels at a time, six sum-of-products need to be calculated. Over two lines of a block, there are twelve sums to calculate. This will fit into three of these custom operations. The remaining two sum-of-products are accumulated with output pixels. Hence, combining these two from two lines is an obvious choice. In fact, it will save two DSPUADD_QUAD8 operations.

Depending on the position of the warped pixels (situation A or B, coefficients are packed, as well as luminance pixels. Followed by a SUPER_USCALEMIXUI_QUAD8 operation and a 32-bit store. The optimization of calculating og_2 through og_5 is straightforward and is executed for each line within the block. See Fig. 4.7.

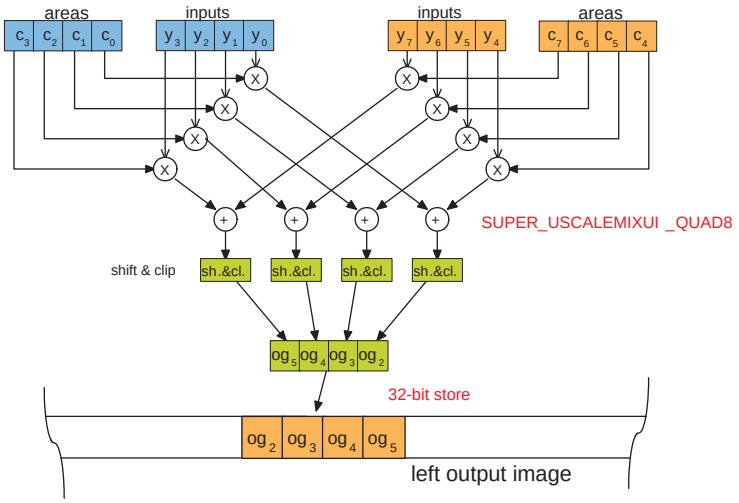


Figure 4.7: First part of optimized re-sampling procedure.

The optimization of remaining two accumulated sum-of-products per line is more complicated. Consider Fig. 4.8, two luminance samples of two lines are merged into two 32-bit words using SUPER_MERGEL_QUAD8 operation. The coefficients of y_0 are equal in both situations, unlike the coefficients of y_1. Only in situation B, contribution of y_1 is taken into account. Hence, the integration of coefficient area_A. In all other cases the coefficients are set to zero. Subsequently, these results are accumulated with output samples of the corresponding two lines. The operation DSPUADD_QUAD8 is executed after the output samples are loaded and packed. Finally, the result is stored back into memory. This part is also executed for each first two pixels of line three and four within each block.

Data-cache Behavior

The requirements for data cache increases with the size of the blocks. More input data needs to be present, more output data is produced simultaneously. The requirement is shown in Table 4.6.

Except for the depth image, pre-fetching is set up differently than before. Consider processing 2×2 blocks. Each line of blocks is processed sequentially. Ideally, the next line of blocks should be pre-fetched. This can be achieved by enabling a pre-fetch region for the reference image (Y and UV) with a *pre-fetch stride* equal to 2×960 (2 lines). This approach is similarly applied for the 4×4 block-based approach, but with a pre-fetch stride of 4×960 (4 lines). The default pre-fetch region suffices for pre-fetching depth image.

Pre-allocation is not changed. The right-view image is still produced from right to left. Negative pre-allocate stride assures pre-allocation of cache lines with decreasing addresses.

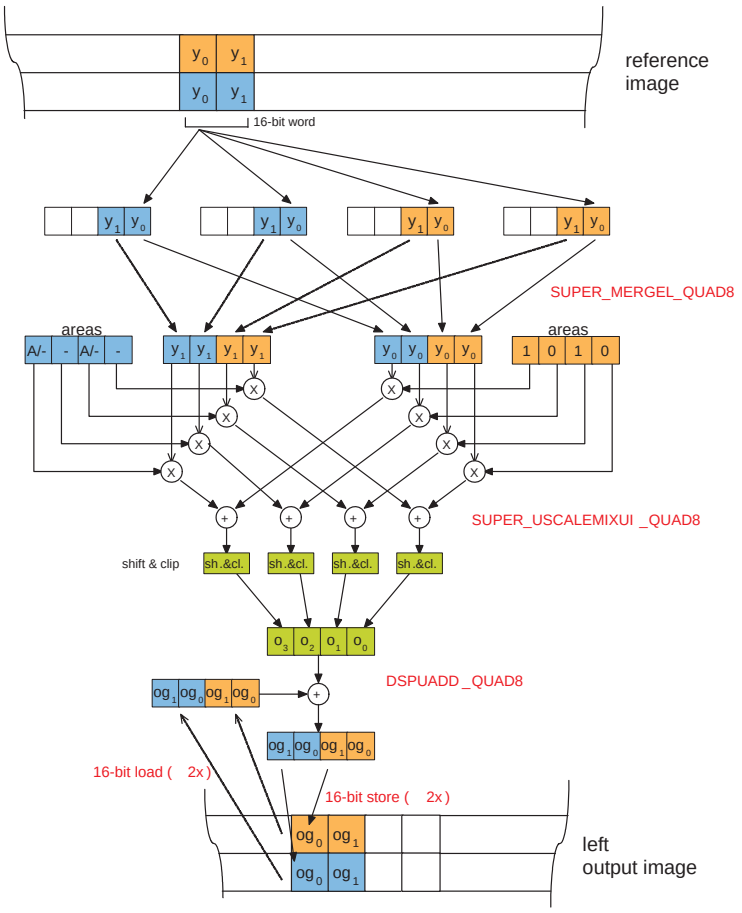


Figure 4.8: Second part of optimized re-sampling procedure.

DESCRIPTION	CURRENT	FUTURE	CURRENT	FUTURE
Ref. image (Y)	2×960	2×960	4×960	4×960
Ref. image (UV)	2×960	2×960	4×960	4×960
Depth (Y)	0.5×960	0.5×960	0.25×960	0.25×960
Outp. image R (Y)	2×960	2×960	4×960	4×960
Outp. image R (UV)	2×960	2×960	4×960	4×960
Outp. image L (Y)	2×960	2×960	4×960	4×960
Outp. image L (UV)	2×960	2×960	4×960	4×960
Hole	2×960	0	2×960	0
Total	14.5×960	12.5×960	26.25×960	24.25×960

Table 4.6: Cache requirement of Integral Weights algorithm for 2×2 and 4×4 blocks.

4.3.3 *Complexity Analysis*

These algorithms are also analyzed on the complexity. The assumptions stated previously hold also for the block-based approach. There are only two exceptions:

- Resolution of depth image is 480×270 (for 2×2 block-based)
- Resolution of depth image is 240×135 (for 4×4 block-based)

Expected Bandwidth Usage

The depth images have a lower resolution. Therefore, the lower bound for expected bandwidth usage is also lower, in comparison to the pixel-based algorithm. See Table 4.7 for details. The differences depicted in

DESCRIPTION			1 FPS	60 FPS
Reference image 2× 960×540		0.99		
Depth image 1× 960×540		0.49		
(pixel-based)	subtotal		1.48	89.0
Reference image 2× 960×540		0.99		
Depth image 1× 480×270		0.13		
(2×2 block-based)	subtotal		1.13	66.7
Reference image 2× 960×540		0.99		
Depth image 1× 240×135		0.03		
(4×4 block-based)	subtotal		1.02	61.2

Table 4.7: Lower bound on bandwidth usage.

this table are only for input data. Bandwidth usage on output images has not changed, as the sizes of the output images remain unchanged (see Table 4.3 on page 43).

Analysis

The complexity numbers are extracted using the cycle-accurate simulator and a trace analysis tool.

Table 4.8 shows the processor load requirement for both block-based approaches. The cycle count is composed of instruction cycles and stall

DESCRIPTION	CYCLES	INSTR.	STALLS	I$	D$	ILP
Integral Weights (2×2)	377.0	324.9	52.1	2.1	50.0	4.62
• Mapping and Resampling	267.0	247.2	22.8	0.9	21.9	6.16
• Dis-occlusion handling	68.8	67.5	1.2	1.2	0.03	1.40
Integral Weights (4×4)	247.6	168.6	79.1	2.0	77.1	5.00
• Mapping and Resampling	147.3	99.9	47.4	0.8	46.6	7.55
• Dis-occlusion handling	61.2	59.0	2.2	1.1	1.1	1.95

Table 4.8: Instruction cycles of block-based Integral Weights algorithm.

cycles. The latter part is again divided into instruction cache stall cycles and data cache stall cycles. All numbers in this table are expressed in MHz, which are obtained by the product of cycle count per frame and the frame rate. The last column shows the Instruction Level Parallelism (ILP), which is an average of executed instructions per cycle. As the TM3282 media processor has 8 issue slots, the maximum ILP achievable is 8.

Next to the processing requirements, bandwidth usage is also important. In Table 4.9 is bandwidth usage detailed for both approaches.

DESCRIPTION	BANDW.	WRITES	READS	INSTR.	DATA
Integral Weights (2×2)	187.7	118.5	69.3	2.2	67.0
Integral Weights (4×4)	184.1	120.1	63.9	2.1	61.8

Table 4.9: Bandwidth usage of block-based Integral Weights algorithm.

4.3.4 *Alternative Algorithm*

This chapter described the implementation and optimization of the existing view synthesis algorithm. Together with the complexity reductions based on a block-based approach, the re-sampling procedure takes a major part of all computations needed. Calculating midpoints, coefficient (or areas) and sum-of-products contributes to a complex algorithm. Each output pixel is an accumulation of multiple fractions of input pixels.

A new algorithm is developed with reduced complexity, while maintaining the picture quality level. This algorithm is detailed and also analyzed on the complexity in Chapter 5.

5 ALTERNATIVE VIEW SYNTHESIS

The contents of this chapter is restricted.

6 CONCLUSIONS AND FUTURE WORK

This thesis has presented two signal conversion algorithms for three-dimensional television (3D-TV) signals. 3D-TV is considered to be the next logical step in the history of television. After transitions from analog to digital and SD to HD television, the step from 2D to 3D signals adds a third dimension to the viewing experience. Methods to produce 3D movies have become available in the film industry and digital cinemas are already prepared to display 3D signals. In the consumer market, display panels with 3D capabilities are increasingly becoming available on the market, but the quantities are still small.

Problem description

3D-TV signals come in many flavors (i.e. anaglyphs, stereo signal, etc.). A stereo signal (or two-view signal) is composed of two signals. These two signals are in full color and are placed side-by-side. Each signal is intended for one eye only. However, an anaglyph signal has two views combined in a single signal. Each perceived view for one eye has reduced color information, because each eye obtains only color-filtered information. For our study, we have considered both types of input signals: full-color stereo signal and an anaglyph signal. Both possible input signals have been converted, such that they can be displayed on full-color stereo display. The displayed signal is perceived by the user employing polarized glasses. The problem statement is now detailed in two ways. First, for anaglyph images, we have explored a conversion of the color shifted information towards a disparity oriented shift suitable for feeding the left and right stereo channel. Second, for stereo images, we have studied a conversion into another stereo signal with which the user can control the depth perception of the 3D signal.

With respect to anaglyph conversion, the objective is to determine an algorithm, which restores the missing color components from using colored representations to form full color stereo images. The color components are separated first. Since the missing color components can be taken from the available color information, the main problem can be translated into the problem of finding the screen disparities. If these disparities are determined, the missing color components are interpolated and combined with the existing color components.

For an input stereo signal, the emphasis is on a solution providing pleasant 3D viewing and perception by controlling the depth of the 3D signal. The objective is to determine a stereo signal conversion, which adjusts depth information contained in stereo signals. In order to adjust depth information, three sub-problems need to be solved. First, the screen disparities need to be determined. Second, the resulting disparities need to be aligned with the reference image. The last problem is determining the view synthesis.

Display panels are driven by high-end dedicated video processing platforms, employing advanced video processors, such as one or more TriMedia TM3282 media-processors. To this end, the computational

complexity of the view synthesis sub-algorithm is still unknown. The final objective is to determine its implementation and investigate options to reduce computational complexity. This imposes two important requirements:

- *Intermediate format.* A 3D-TV signal is converted to an intermediate format, to cope with different input and output signals. The intermediate format is defined as a reference image with corresponding depth image.
- *Real-time processing.* A frame rate of 60 frames per second achieves real-time processing. The frame resolution of 960×540 pixels is determined as the effective resolution of a 3D display panel.

Developed solutions

ANAGLYPH SIGNAL CONVERSION We developed an anaglyph conversion algorithm, which involves three steps. Firstly, the color information of both views contained in the anaglyph image are separated. Secondly, the missing color information of both views are restored. Finally, original and restored color information are combined for both views to form a full color stereo image. The first and last steps are trivial. However, we evaluated several methods to restore the missing color information. The recording of an anaglyph image can be depicted as sequential recording of the color components, with a camera movement between specific components.

The first solution involves the estimation of a 6-parameter affine motion model. Prior to estimation, the color components are equalized by histogram specification. Secondly, the affine motion model is estimated. Then, each color component is compensated for the motion, and missing color components are interpolated. However, results show that restored color components appear slanted. This is caused by the content of the image and the global affine motion model.

The second solution is based on a block-based motion estimator. 3-D recursive search is used as a disparity estimator to determine the apparent motion between the two views. This estimator assumes that the brightness constancy assumption is satisfied. However, the pixel intensities between color components are not similar. Consequently, this assumption does not hold. To still use 3DRS, we have evaluated two possible solutions. Either the color components are equalized, or a different similarity measure in 3DRS is defined. We have equalized the color components, either by histogram specification or filtering with a high-pass filter. The first method does not always equalize the color components sufficiently. Consequently, 3DRS finds incorrect disparity vectors. Filtering the high frequencies of the color components is susceptible to noise. However, 3DRS finds much better disparity vectors. We have implemented Mutual Information as the similarity measure in the 3DRS algorithm. Calculating Mutual Information involves determining the probability density function and the entropy of each pair of blocks. This is significantly more computationally complex than calculating SAD values of each pair of blocks. However, the disparity vector fields are similar or better, in comparison to the result of high-pass filtering.

STEREO SIGNAL CONVERSION We have composed a stereo signal conversion algorithm of existing sub-algorithms which involves three

steps. Firstly, a disparity estimator determines the screen disparity of the stereo input image. This results in a small set of disparity vectors. Secondly, the disparity vectors are converted to disparity values. This disparity image has a much lower resolution than the input image. Therefore, the disparity image is filtered and up-sampled to frame resolution using joint bilateral filtering techniques. However, the last up-sampler is replaced by a median-based block erosion filter, which reduces the computational complexity significantly, with a minor quality degradation. The resulting disparity image, together with one image of the stereo input image, forms the intermediate format. The last step synthesizes two new views to form a new stereo image. The view synthesis is based on the intermediate format (texture plus depth) only.

We have experimented with the 3DRS algorithm and modified its parameters to estimate screen disparity of stereo-view signals. We have extended the candidate set and update set, and removed the zero-vector candidate. We consider the horizontal vector component produced by 3DRS as the screen disparity.

We have used an existing view synthesis sub-algorithm, based on a texture plus depth representation. We have optimized the algorithm for a fixed parameter setting. For stereo conversion, optimizing the depth perception is performed by modifying the depth signal.

For implementation, we have mapped the view synthesis sub-algorithm to the TM3282 media-processor, and analyzed its computational complexity. Unfortunately, the real-time requirement of synthesizing 60 stereo images per second is not feasible (only 41 stereo images per second are achieved). Therefore, we have investigated options to reduce its complexity. The first option we explored, involved block-based processing. Each block of 2×2 or 4×4 pixels has only one disparity value, giving a frame rate of 79 and 119 synthesized stereo images per second, respectively. Next to the complexity reductions, we have also optimized the picture quality (artifact reduction) by employing a 3×3 dilation filter. Another method to reduce complexity, is to develop a new view synthesis algorithm. This new algorithm is proposed in an invention disclosure by [12]. We have studied and implemented this new proposal and achieved a frame rate of 96 stereo images per second, due to a much simpler design for the re-sampling technique. We have also evaluated a block-based approach as an alternative, resulting in 135 and 147 stereo images per second for 2×2 blocks and 4×4 blocks, respectively.

Conclusions

ANAGLYPH SIGNAL CONVERSION We have found that stereo color signals can be restored from anaglyph signals. The missing color components can be extracted from each opposite view. A global affine motion model is found insufficient for this conversion. Instead, 3DRS is well suited to find screen disparities, after transforming the individual color components with a high-pass filter. Unlike high-pass filtering, histogram specification is insufficiently capable of removing the possibly large intensity differences. As such, high-pass filtering prior to 3DRS leads to significantly more accurate results. As a metric for vector evaluation, Mutual Information gives somewhat better results than SAD. However, evaluating the Mutual Information increases the com-

putational complexity with a factor of five, even when using lookup tables and storing data in intermediate caches.

STEREO SIGNAL CONVERSION We have found that the perceived depth in stereo signals can be adjusted by manipulation of the depth signal. However, it is a complex algorithm and consists of three sub-algorithms. First, 3DRS is well suited for disparity estimation in stereo signals. Experimental results show that stereo signals have different characteristics in comparison to mono signals, in terms of differences in vector accuracy, candidate set and update set as parameters of this estimator. It produces robust and consistent disparity vectors. We have found that without vertical components in the 3DRS-based disparity estimator, this algorithm does not produce reliable and consistent disparity vectors. Screen disparity is measured as a horizontal component only. Second, the up-sampling procedure with cross-bilateral filters produces very sharp edges, which are aligned with the edges in the reference image. The edges in the low resolution depth image are propagated through up-sampling process. Similarly, the quality of 3DRS algorithm (vectors) has a direct influence on the quality of the depth signal. Third, we have compared two view synthesis algorithms. The basic algorithm shifts pixels to the desired position after which filtering is applied. The second algorithm is new and uses efficient re-sampling of data to interpolate the desired pixels. Experiments have shown that the novel algorithm is much more efficient than the existing algorithm, while producing a similar picture quality. We have implemented both algorithms and compared them using a TriMedia TM3282 processor to evaluate the computational complexity. These results are used to extrapolate the complexity for achieving real-time processing. Compared to the existing algorithm, the complexity of the new algorithm is reduced with a factor of 2.3 (41 vs. 96 frames per second). This reduction is mainly due to the efficiency of the re-sampling. Since the new algorithm appeared to be attractive, we have further experimented with complexity reduction techniques for that algorithm specifically. Instead of pixel-based processing, we have found that a block-based processing leads to a significant complexity reduction with a factor of 1.4 and 1.5 for blocks of 2×2 and 4×4 pixels, respectively. However, the subjective picture quality is affected by introducing a halo artifact due to the block size. The idea of block-based processing was also tested for the existing view synthesis algorithm and resulted in reductions of 1.9 and 2.9 for the above block sizes.

Future work

To obtain more reliable results for the near future, the 3DRS-based disparity estimator should be made more robust, requiring the implementation of a scene cut detection and occlusion detection and correction. Besides this, the estimator can be optimized for the use of Mutual Information as a similarity measure. With respect to the new view synthesis algorithm, the implementation of the dis-occlusion processing can be improved, as data can be partly overwritten in the dis-occlusion area.

BIBLIOGRAPHY

[1] R.-P.M. Berretty, F.J. Peters, and G.T.G. Volleberg. Real-time rendering for multiview autostereoscopic displays. volume 6055. SPIE, 2006.

[2] R.-P.M. Berretty, A.K. Riemens, and P.F. Machado. Real-time embedded system for stereo video processing for multiview displays. volume 6490, page 649014. SPIE, 2007.

[3] M. Ciacci. NXP internal document: Study notes, Jan 2009.

[4] Denmark ColorCode 3-D Center. Color code 3-d, May 2009. URL http://www.colorcode3d.com/.

[5] Rachel Cooper. What is stereo vision?, May 2009. URL http://www.vision3d.com/stereo.html.

[6] R. Dahyot, F. Vilariño, and G. Lacey. Improving the quality of color colonoscopy videos. *Journal of Image Video Processing*, 2008: 1–7, 2008.

[7] G. de Haan. *Digital Video Post Processing*. Philips Research Laboratory Eindhoven, 2006.

[8] G. de Haan and P.W.A.C. Biezen. Sub-pixel motion estimation with 3-D recursive search block-matching. *Signal Processing: Image Communication*, 6(3):229–239, June 1994.

[9] G. de Haan, P.W.A.C. Biezen, H. Huijgen, and O.A. Ojo. True-motion estimation with 3-D recursive search block matching. *Circuits and Systems for Video Technology, IEEE Transactions on*, 3(5): 368–379, 388, October 1993.

[10] G. Egnal. Mutual information as a stereo correspondence measure. Technical Report MS-CIS-00-20, University of Pennsylvania, 2000.

[11] G. Egnal and R.P. Wildes. Detecting binocular half-occlusions: Empirical comparisons of five approaches. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(8):1127–1133, Aug 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1023808.

[12] H. He, E. Bellers, and M. de Groot. Implementation 3DTV depth image based rendering (DIBR) by spatial domain over-sampling. NXP Invention Disclosure 81387446, 2009.

[13] N.S. Holliman. Three-dimensional display systems. In *Handbook of Optoelectronics*. J.P. Dakin and R.G.W. Brown, Taylor and Francis, 2006.

[14] J. Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, volume 2, pages 1033–1040, October 2003.

[15] P.F. Machado. Depth estimation for stereo content. Technical Report TN-2005-01168, Koninklijke Philips Electronics, December 2005.

[16] J.-M. Odobez, P. Bouthemy, and F. Spindler. Motion2D: A software to estimate 2D parametric motion models, May 2009. URL http://www.irisa.fr/vista/Motion2D/.

[17] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. A gentle introduction to bilateral filtering and its applications, May 2009. URL http://people.csail.mit.edu/sparis/bf_course/.

[18] A.K. Riemens, O.P. Gangwal, B. Barenbrug, and R.-P.M. Berretty. Multistep joint bilateral depth upsampling. In *Visual Communications and Image Processing 2009*, volume 7257, January 2009.

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

[20] J. Symanzik. Three-dimensional statistical graphics based on interactively animated anaglyphs. In *Proceedings of the Section on Statistical Graphics*, pages 71–76. American Statistical Association, 1993.

[21] R Wang. Image enhancement by contrast transform, May 2009. URL http://fourier.eng.hmc.edu/e161/lectures/contrast_transform/.

[22] P. Wimmer. Anaglyph methods comparison, May 2009. URL http://www.3dtv.at/Knowhow/AnaglyphComparison_en.aspx.

[23] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.