High-Quality Real-Time Hardware Stereo Matching Based on Guided Image Filtering

Christos Ttofis and Theocharis Theocharides
KIOS Research Center for Intelligent Systems and Networks
Department of ECE, University of Cyprus
{ttolis.christos, ttheocharides}@ucy.ac.cy

Abstract— Stereo matching is a vital task in several emerging embedded vision applications requiring high-quality depth computation and real-time frame-rate. Although several stereo matching dedicated-hardware systems have been proposed in recent years, only few of them focus on balancing accuracy and speed. This paper proposes a hardware-based stereo matching architecture that aims to provide high accuracy and concurrently high performance in embedded vision applications. The proposed architecture integrates a compact and efficient design of the recently proposed guided image filter; an edge-preserving filter that reduces the hardware complexity of the implemented stereo algorithm, while at the same time maintains high-quality results. A prototype of the architecture has been implemented on a Kintex-7 FPGA board, achieving 60 fps for 720p resolution images. Moreover, the proposed design delivers leading accuracy when compared to state-of-the-art hardware implementations.

Keywords—Stereo Matching; Embedded Systems; FPGAs;

I. INTRODUCTION

Stereo matching, the task of matching the images taken from a stereo camera and extracting the depth of objects in a scene [1], is commonly employed in embedded vision applications such as intelligence surveillance, autonomous vehicles and mobile robots [1]. Such applications need to satisfy real-time processing speed, high matching accuracy and low-power consumption constraints. The matching algorithm and the implementation platform are both factors that play a significant role in satisfying the requirements of an embedded stereo matching system. Global stereo matching algorithms produce very accurate results [2], but rely on the high-end hardware resources of multi-core CPUs and/or GPU platforms to achieve real-time processing. Such platforms therefore appear to be unsuitable for the realization of stand-alone stereo matching systems, and also consume excessive power, which is not desirable in battery-powered mobile and embedded devices. In contrast, local algorithms can be greatly benefited by the use of parallel structures implemented on either FPGAs or ASICs, providing the necessary computational power and energy efficiency for embedded vision applications [3].

To this end, several real-time stereo matching hardware systems have been developed during the past decade. However, the majority of them have implemented local algorithms that rely on standard block-based aggregation with a fixed support window [3]. While these algorithms can achieve very high frame rates when implemented in hardware [3], they lead to low matching accuracy [4]. As such, a few attempts have been made recently to implement dedicated hardware architectures of more accurate algorithms, such as Semi Global Matching (SGM) [5], [6] and Adaptive Support Weight (ADSW) [7], [8], [9]. For the past few years, hardware implementations based on SGM and ADSW algorithms have become the preferred solution towards higher matching accuracy in embedded vision applications. However, existing implementations have made several modifications and simplifications to adapt the algorithms for real-time processing, resulting in noticeable quality reduction compared to the original software algorithms. In addition, their high memory and hardware demands might limit their scalability to higher resolution images. Recently, the idea to utilize the Guided Image Filter (GIF) [10] in local ADSW stereo matching algorithms has been proposed to reduce the complexity of the cost aggregation step. Such software implementations have yielded promising results [11].

Motivated by the results of the software implementation presented in [11], this paper proposes a fully pipelined, parallel and scalable stereo matching hardware architecture that integrates the recently proposed GIF. There are two main novel contributions in this paper. First, it presents a new and efficient hardware design of the GIF (that can be potentially adopted in other uses of the filter). Second, it explores and concurrently discusses the hardware design parameters and optimizations involved in integrating the GIF hardware architecture in the cost aggregation step of ADSW-based hardware stereo matching systems. The latter reduces the overall hardware complexity of cost aggregation, which in turn allows real-time stereo matching of high definition images (HD), as well as improvements of the overall matching accuracy, thanks to the edge-preserving property of the GIF. The paper furthermore presents a complete prototype of the proposed GIF-based stereo matching architecture on a Kintex-7 FPGA board. The prototype was built to support 720p@60Hz HD video sequences (captured from a custom-built stereoscopic camera system) and various disparity ranges. Additionally, qualitative and quantitative evaluation, based on Middlebury benchmark image sets [2], shows that the developed FPGA prototype provides competitive accuracy compared to existing state-of-the-art stereo matching hardware systems.

In the following: Sections II & III provide background and related work, while Section IV presents the proposed hardware implementation. Section V shows results and comparison with related work. Finally, Section VI concludes the paper.

II. BACKGROUND ON STEREO MATCHING

A. Overview & Classification of Stereo Matching Algorithms

Stereo matching is a technique aimed at inferring depth information of a scene from a pair of stereo images (usually called reference and target images) [3]. The depth is determined by locating corresponding pixels in the stereo

This work was co-funded by the Republic of Cyprus through the Research Promotion Foundation and the EUREKA Organization under the Eurostars Programme (Project: E! 5527 RUNNER).
images. Given that the input images are rectified [12], the correspondence of a pixel at coordinate (x, y) of the reference image, can only be found at the same vertical coordinate y, and within a maximum horizontal bound, called disparity range \( D \), in the target image. The disparity is then computed as the location difference of corresponding pixels in both images. The disparities of all pixels form a disparity image, or disparity map, from which depth information can be extracted.

According to [2], stereo matching algorithms mostly follow four steps: 1) matching cost computation, 2) cost aggregation, 3) disparity computation/optimization and, 4) disparity refinement. Moreover, [2] classifies stereo matching algorithms into two broad categories: global and local. Global algorithms are usually formulated as an energy minimization problem, which is solved with techniques such as Dynamic Programming, Graph Cuts and Belief Propagation. Such methods produce very accurate results at the expense of high computational complexity and memory needs. The Semi-Global Matching (SGM) [3], [6] methods renounce part of the complexity directly dependent on the support window size. In contrast, local algorithms determine the disparity associated with a minimum cost function (see [2] for a review) at each pixel by performing block matching and winner-takes-all optimization. Hence, they have lower computational complexity and memory requirements compared to global and SGM methods. Early local algorithms relied on simple aggregation strategies that perform block matching by using either a fixed (typically square) window, or multiple windows with different sizes. However, these approaches are prone to matching errors at depth discontinuity regions; they blindly aggregate pixels belonging to different disparities due to the use of a fixed window (shape and/or size) [4]. Among local algorithms, the most recent adaptive support weight (ADSW) methods are currently the most accurate. They work by assigning different weights to the pixels in the support window based on their color/ proximity distances to the central pixel. In this way, they aggregate only those pixels that lie at the same disparity, leading to very good quality at depth borders [4].

Despite their good quality results, ADSW algorithms cannot take advantage of the “integral image” or “sliding window” techniques, as the adaptive weights have to be recomputed at every pixel. This makes the cost aggregation’s hardware complexity directly dependent on the support window size.

**B. Stereo Matching using Guided Image Filtering**

The recently proposed Guided Image Filter (GIF) [10] has been employed in [11] to reduce the complexity of the cost aggregation step in ADSW methods, leading to a high-quality fast and simple algorithm (Fig. 1), with the following steps:

1) **Cost Volume Construction.** This step calculates a matching cost for each pixel \( p \) at all possible disparities. The output is a three-dimensional structure consisting of \( D \) cost images (Stereo Cost Volume - SCV). Each cost is computed as the truncated absolute difference of colors and gradients, a metric that exhibits good robustness to illumination changes.

\[
M(p,d) = \sum_{i=1}^{3} |l_{i_{left}}(p) - l_{i_{right}}(p - d)| \quad (1)
\]

\[
G(p,d) = \nabla_x(l_{i_{left}}(p)) - \nabla_x(l_{i_{right}}(p - d)) \quad (2)
\]

\[
C(p,d) = a \cdot \min(T_c,M(p,d)) + (1 - a) \cdot \min(T_g,G(p,d)) \quad (3)
\]

2) **Cost Volume Filtering.** This step utilizes the GIF to smooth each slice of the SCV. Due to its edge-preserving protery, the GIF leads to good accuracy at depth discontinuities. Typically, the filtered cost value at \( p \) and disparity \( d \) is a weighted average of the pixels in the same slice of the SCV, and is expressed as in (4).

\[
q(p,d) = \sum_{i,j} W_{i,j}(I)C(p,d) \quad (4)
\]

The GIF generally uses a guidance image \( I \) to filter a guided image \( f \). In our case, the guidance image is the grayscale reference image, while the guided image is a slice \((x, y)\) of the SCV. The filter weights are defined as in (5), where \( \mu_r \) and \( \sigma_r \) are the mean and the variance of \( f \) in a squared window \( w_0 \) with dimensions \( r \times r \), centered at pixel \( k \). \( |w| \) is the number of pixels in the window and \( \epsilon \) is a smoothness parameter. An inherent advantage of the GIF is that the weights can be computed with some linear operations (see [10]), which can be decomposed into a series of mean filters with windows radius \( r \). This leads to an efficient algorithm (pseudocode given in Fig. 2).

\[
W_{i,j} = \frac{1}{|w|^2} \sum_{k \in (i,j) \in w} \left( 1 + \frac{(l_i - \mu_k)(l_j - \mu_k)}{\sigma_k^2 + \epsilon} \right) \quad (5)
\]

3) **Disparity Selection.** Once the SCV slices are filtered, the best disparity for pixel \( p \) is chosen through a simple Winner-Takes-All (WTA) minimization approach using (6).

\[
d_p = \arg \min_{d \in D} q(p,d) \quad (6)
\]

4) **Disparity Refinement.** A left/right consistency check (L-R check) is performed. Hence, the disparity map, \( D_{LR} \), using the right check as reference is also computed. The L-R check marks disparities as invalid if the disparity \( D_{LR}(p) \) and its corresponding disparity of \( D_{RL}(p) \) differ by more than 1 pixel. Invalid pixels are then filled with the minimum disparity between their closest consistent pixels in the left and right direction. Weighted and typical median filtering are applied next to smooth the filled regions and remove spikes.

![Fig. 1. Major Steps of the GIF-based Stereo Matching Algorithm.](image)

![Fig. 2. Pseudocode of the Guided Image Filter [10].](image)
III. RELATED WORK

In recent years, a fair amount of work has been carried out on real-time hardware implementations of local stereo matching algorithms (e.g. [13]); a thorough review is presented in [3]. The majority of these implementations have adopted simple fixed support and multiple window methods, therefore trading accuracy for speed. High matching accuracy though is of foremost importance in many of today’s embedded vision applications. As such, a few attempts have been made recently directed towards improving the matching accuracy, either by combining different stereo algorithms together, or by implementing modified versions of SGM and ADSW algorithms. The hardware implementation in [14] performs a modified version of the Census transform in both the intensity and gradient images, in combination with the SAD correlation metric. An FPGA implementation of a stereo algorithm based on the neural network and Disparity Space Image (DSI) data structure is introduced in [15]. The real-time FPGA-based stereo matching design presented in [16] combines the mini-Census transform and the Cross-based cost aggregation. SGM-based stereo matching systems have been introduced in [5], [6] and implemented on FPGAs and a hybrid FPGA/RISC architecture, respectively. The technical details/parameters of the different implementations are summarized in Table I.

The works that are closest to ours in terms of the matching algorithm are the works in [7], [8], [9], which implement ADSW-based algorithms. [7] proposed the VLSI design of a hardware-friendly ADSW algorithm that adopted the mini-Census transform to improve the accuracy and robustness to radiometric distortion. [8] proposed the implementation of a complete stereo vision system, which incorporates an ADSW algorithm and also integrates pre- and post-processing units. Finally, a hardware-oriented stereo matching system based on the adaptive Census transform is presented in [9]. The aforementioned high-quality ADSW-based systems follow a similar algorithm-to-hardware mapping methodology. That is, a complex, but accurate, algorithm is adapted for dedicated hardware implementation through a series of algorithmic modifications/approximations. In most cases, however, these implementations scarify part of the accuracy, quality reduction compared to the original implementation of the ADSW approach in [4] is ~ 4-5%.

In contrast, the proposed stereo matching architecture implements the ADSW aggregation step in a different way; by smoothing the SCV with the GIF, this type of filter can have an efficient dedicated hardware design, as the basic operation involved is the mean filter with windows of radius r. The mean intensity of pixels over rectangular windows in the image can be implemented in a fast way using the integral image technique. However, this technique requires huge amount of memory, especially for high-resolution images. Therefore, we instead followed a variant of the approach in [17], to implement a custom mean filter design that consumes compact hardware resources. The main idea is to maintain a sum for each column in the image to be filtered. Each column sum accumulates 2r+1 pixels, while the window sum is computed by adding 2r+1 adjacent column sums. While filtering the image, the window sum is updated using the two-step approach illustrated in Fig. 3 (a). When the window is moved to the right from one pixel to the next, the column sum to the right of the window is yet to be computed for the current row, so it is centered one row above. Therefore, the first step consists of updating the column sum to the right of the window, by subtracting its topmost old pixel and adding one new pixel below it. The second step moves the window to the right and updates the window sum by subtracting its leftmost column sum (old column sum), and adding the updated column sum computed in step 1 (new column sum).

The mean filtering process is implemented in hardware with simple arithmetic operations (addition, subtraction and fixed-point multiplication) and a series of read/write operations to a memory buffer (stores the column sums) using the architecture shown in Fig. 3 (b). The mean filter architecture receives the new pixel and the old pixel, and outputs the mean corresponding to the window being filtered. Initially, the column sum yet to be updated is read from the column sum memory (its size depends on the image width), and once updated (after adding and subtracting the new and old pixels, respectively), it is written to the memory at the same address (read operation performed one clock earlier to maintain pipeline consistency). The window sum is computed by adding and subtracting the updated and old column sums, respectively, from the content of window sum register. However, we avoid fetching the old column sum from the memory, as we aimed to make the architecture flexible for both ASICs and FPGAs.

![Image of mean filtering process and hardware architectures](image-url)
supporting dual-ported BRAMs (2 ports already used to update the new column sum). Access to the old column sums is instead obtained through a shift register (queue) with size 2r+1 (an old column sum at cycle t is a delayed version of the new column sum at cycle t-2r-1). The final mean value is computed by multiplying the window sum with 1/(2r+1)^2.

The architecture of the GIF is depicted in Fig. 3 (c). It receives two pixels from the reference image (used as guidance image) and two from the slice of the SCV to be smoothed. The architecture consists of four mean filters that compute the mean of 2r+1 pixels ( BRAM outputs) by introducing one cycle delay through a shift register (queue). The remaining values and the new pixel costs can be implemented not only using shift registers, but also using FIFO/BRAM memories. Thus, the buffer types can be selected as per the design effort and application demands.

B. Proposed GIF-based Stereo Matcher (GIF-SM)

The simplicity of the GIF architecture makes the stereo matching process independent of the match window size. Therefore, stereo matching can now rely on pixel-based operations. In particular, only two pixels (the new pixel and old pixel) need to be processed every clock cycle. Hence, the hardware complexity can be reduced or a higher level of parallelism can be exploited to enable higher frame rates. Moreover, the memory buffers that store the RGB and gradient values need to be stored). It was found that by only computing the mean over the x direction (using an accumulator and a FIFO queue) rather than on a rectangular window, it reduces the quality by less than 0.5 %, but it also eliminates the need to store the aforementioned values in BRAMs or shift registers.

The architecture of the GIF is depicted in Fig. 3 (c). It uses only the gradients in x direction in the final SCV function, this work utilizes the gradients in y direction as well, as it was found to yield better quality results. Therefore, the GCCs implement the Sobel operator to calculate the gradient values in both directions using the architecture shown in Fig. 4(b). The architecture performs convolution of 3x3 windows (fetched from a scanline buffer) with the Sobel kernels using two convolution units (CONV), and normalizes the result in the range [0 255] (NORM units). The GCMMU is also responsible for buffering the computed gradients and input color values, and providing them in a synchronized way to the CVCU. It stores 2r + 1 lines of both RGB and gradient data in BRAMs working in read-first mode; this allows access on both the new and old pixels. The new pixels (BRAM inputs) are synchronized with the old pixels (BRAM outputs) by introducing one cycle delay through register buffers. Finally, the RGB and gradient values of the most recently fetched old and new pixels in the target image (right) are stored in serial-in parallel-load shift registers. This allows the CVCU to exploit disparity level parallelism.

The CVCU employs a cascade of Cost Computation Units (CCUs) that calculate the pixel-wise costs between the new and old pixels in the left image and their 2dref corresponding pixels in the right image. The architecture of a CCU is shown in Fig.4(c). It consists of absolute difference units, adders and comparators that calculate the truncated color and gradient costs, which are then summed up to compute the overall cost. However, prior to the summation, the truncated color and gradient costs are multiplied by constant values, in order to balance their influence in the overall cost. The constants are selected to be powers of 2 to replace multiplication with shifting. The final pixel costs calculated by using the left image as reference are stored in cost memory buffers, and are reused for the computation of the right disparity map by the CVFDSU.
memories, to smooth the SCVs corresponding to the left and right disparity maps, respectively. Smoothing is performed using the architecture of the GIF shown in Fig. 3 (c). The CVFDSU also incorporates 2 WTA units (composed of comparators organized in tree structures) that select the disparities with the minimum costs.

The DRU implements the L-R check and filling approach (Section II) through a set of comparators and priority encoders that locate the nearest valid disparities in the left and right direction of the invalid pixels. The adaptive median filter is implemented by extending the approach in [18], which presents a median filter architecture based on cumulative histograms, to the case of adaptive weights. The histogram-based median filter architecture shown in Fig. 4(d) has been designed in a parallel manner, allowing the computation of the median value for a window of size $m \times m$ in a single clock cycle. Its major hardware units include ROM memories, adder-trees, comparators and a priority encoder that locates the median value among the totality of the bins. We used binary weights generated by image segmentation (“one” if a disparity value lies in the same segment with the central pixel of the window, “zero” otherwise), instead of using the adaptive weights from [17]. This approach is not only hardware friendly, but also preserves object borders since pixels lying in the same segment are more likely to lie at the same disparity. A simple method that partitions the disparity image into segments based on thresholding was utilized. The binary weights are used as the enable signals on the ROMs. The same structure of the architecture in Fig. 4(d) is also used for implementing a typical median filter (spike removal) by setting all enable signals to 1.

V. FPGA IMPLEMENTATION RESULTS

A. Experimental Platform and Methodology

The proposed system architecture has been implemented on the Inrevium’s Kintex-7 FPGA Display Kit [19], which is equipped with a Xilinx Kintex-7 FPGA (XC7K325T-FFG900) [20]. We used a custom-built stereo camera system, which was directly interfaced to the FPGA board through capturing FMC daughter cards [19]. The entire setup was built on 720p@60Hz HD video support. The captured stereo video sequences were rectified (through a custom hardware stereo image rectification unit that follows the architecture presented in [12]), and used as input data to the system architecture shown in Fig. 4(a). We configured the system architecture to receive stereo video sequences synchronized with the HDMI pixel sampling clock. The refined disparity maps were also synchronized with the pixel clock, and directed to an HDMI-compatible monitor. The system diagram of the prototype FPGA implementation is given in Fig. 5 (a). The different system parameters were set to constant values: \( \{r, \epsilon, T_s, T_g\} = \{3, 0, 7, 2\} \). These values were found empirically and were used throughout our experiments.

B. Disparity Map Quality Evaluation

We used the Middlebury benchmark dataset [2] to evaluate the quality of the resulting disparity maps. We measured the percentage of bad matching pixels relative to the ground truth disparity maps of four pairs of test images in the dataset. The benchmark disparity maps and those produced by the proposed system architecture are shown in Fig. 5 (b). Quantitative evaluation results are listed in Table I (columns 2-5), which also provides a comparison with related work. As it can be observed, the proposed architecture generates high quality results, even at regions with low texture and close to object boundaries; this is evidenced by obtaining the lowest error rates among related hardware implementations at disc regions. Most importantly, among the implementations for which an overall percentage of bad matching pixels is provided, the proposed system obtains the lowest error rate. Besides the Middlebury benchmark images, the proposed architecture is able to deal with real-world scenes as well, producing detailed and accurate disparity maps with clean object boundaries (see Fig. 5 (c)).

We have also investigated how the resulting disparity maps are compared with those generated by the original GIF-based algorithm presented in [11]. As it can be observed from Table I, the distance in quality between the work in [11] and the proposed architecture is only 1.14% on average. Moreover, the algorithm in [11] implements a color version of the GIF, which in general yields better accuracy. Therefore, the distance in quality can be further reduced if the proposed GIF-SM is extended to support color guidance images as well. Finally, it is worth noting that the matching accuracy of the proposed GIF-based stereo matcher approaches the quality of the original ADSW algorithm in [4] (6.69% vs. 6.67%), while the ADSW dedicated hardware implementations in [8], [9] exhibit a quality reduction compared to [4] of ~4.84% and ~4.68%, respectively. This evidences the superiority of the proposed hardware-based GIF-SM in maintaining the matching accuracy of the original ADSW algorithm, thanks to the integration of the GIF.

C. Processing Speed

We measured the processing speed of the proposed GIF-SM in frames processed per second (fps) and in Million Disparity Estimations per second (MDE/s), a metric that also takes into account the number of pixels and the disparity range \( MDE/s = M \cdot N \cdot D \cdot fps \). The FPGA prototype of the architecture is able to process 720p (1280x720) HD video at 60 fps, with a pixel clock rate of ~74.25MHz. When considering the post place-and-route frequency (103 MHz) provided by the Xilinx ISE Design tool, the maximum throughput of the architecture is expected to reach ~83 fps. In general, the architecture presents good scalability with respect to the frame rate and image size, as it is intensively pipelined and...
AMBROSCHI D.

Applications requiring fast and accurate stereo matching. Conclusively, the obtained speed/quality results indicate that (w/o the DRU) for different values of MEMORIES. The required DSPs are currently limiting resources observed that the hardware implementation complexity of the GIF architecture to support color guidance images. We also targeted high-quality disparity map estimation. Only the work outperforming all dedicated hardware implementations fast implementations when considering the MDE/s metric, not competitive in terms of quality, mainly due to the use of a hardware design of the filter was illustrated, which was then utilized to develop a parallel and scalable ADSW stereo matcher. The GIF design reduces the hardware complexity of the ADSW cost aggregation, enabling real-time frame-rates on HD images. Moreover, the GIF’s inherent edge-preserving nature allows for improved matching accuracy when compared to existing state-of-the-art hardware stereo matching systems. As an immediate follow up of this work, we intend to extend the GIF architecture to support color guidance images. We also plan to perform a detailed design space exploration of the stereo matcher hardware resources with respect to the various system parameters (radius kernel, image size, etc).

VI. CONCLUSION AND FUTURE WORK

This paper presented a high-quality real-time hardware stereo matcher based on the GIF. A compact and efficient hardware design of the filter was illustrated, which was then utilized to develop a parallel and scalable ADSW stereo matcher. The GIF design reduces the hardware complexity of the ADSW cost aggregation, enabling real-time frame-rates on HD images. Moreover, the GIF’s inherent edge-preserving nature allows for improved matching accuracy when compared to existing state-of-the-art hardware stereo matching systems. As an immediate follow up of this work, we intend to extend the GIF architecture to support color guidance images. We also

TABLE I. QUALITY AND SPEED COMPARISON WITH RELATED WORK

<table>
<thead>
<tr>
<th>Work</th>
<th>Average Error Rates</th>
<th>Nonocc</th>
<th>All</th>
<th>Disc</th>
<th>Overall</th>
<th>Image Size</th>
<th>D</th>
<th>Speed (fps)</th>
<th>MDE (10%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baha [15]</td>
<td>7.16</td>
<td>11.5</td>
<td>n.a.</td>
<td>n.a.</td>
<td>8.20</td>
<td>50x5735</td>
<td>50</td>
<td>46</td>
<td>388</td>
</tr>
<tr>
<td>Zhang [16]</td>
<td>4.41</td>
<td>7.41</td>
<td>12.8</td>
<td>8.20</td>
<td>102x678</td>
<td>64</td>
<td>60</td>
<td>5019%</td>
<td></td>
</tr>
<tr>
<td>Jin [13]</td>
<td>8.31</td>
<td>11.9</td>
<td>27.2</td>
<td>16.5</td>
<td>640x480</td>
<td>64</td>
<td>230</td>
<td>4522%</td>
<td></td>
</tr>
<tr>
<td>Ambrosch [14]</td>
<td>5.82</td>
<td>9.37</td>
<td>22.2</td>
<td>12.5</td>
<td>750x400</td>
<td>60</td>
<td>60</td>
<td>1080%</td>
<td></td>
</tr>
<tr>
<td>Chang [7]</td>
<td>n.a.</td>
<td>4.81</td>
<td>9.8</td>
<td>8.67</td>
<td>302x240</td>
<td>30</td>
<td>0.01</td>
<td>42.76%</td>
<td></td>
</tr>
<tr>
<td>Ding [8]</td>
<td>7.41</td>
<td>11.9</td>
<td>15.6</td>
<td>11.5</td>
<td>640x480</td>
<td>60</td>
<td>51</td>
<td>940%</td>
<td></td>
</tr>
<tr>
<td>Perri [9]</td>
<td>4.39</td>
<td>10.6</td>
<td>19.7</td>
<td>11.3</td>
<td>648x480</td>
<td>60</td>
<td>45</td>
<td>829%</td>
<td></td>
</tr>
<tr>
<td>Gehrig [5]</td>
<td>n.a.</td>
<td>8.13</td>
<td>n.a.</td>
<td>n.a.</td>
<td>340x240</td>
<td>30</td>
<td>27</td>
<td>118%</td>
<td></td>
</tr>
<tr>
<td>Banz [6]</td>
<td>8.43</td>
<td>n.a.</td>
<td>n.a.</td>
<td>n.a.</td>
<td>640x480</td>
<td>128</td>
<td>30</td>
<td>117%</td>
<td></td>
</tr>
<tr>
<td>ADSW [4]</td>
<td>3.48</td>
<td>8.53</td>
<td>9.8</td>
<td>8.67</td>
<td>302x240</td>
<td>30</td>
<td>0.01</td>
<td>42.76%</td>
<td></td>
</tr>
<tr>
<td>Hosni [7]</td>
<td>2.65</td>
<td>5.57</td>
<td>8.4</td>
<td>5.5</td>
<td>640x480</td>
<td>26</td>
<td>25</td>
<td>200%</td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>3.97</td>
<td>8.80</td>
<td>9.3</td>
<td>6.69</td>
<td>120x720</td>
<td>60</td>
<td>60</td>
<td>533%</td>
<td></td>
</tr>
</tbody>
</table>

All points except for occluded areas. * all points including half-occluded regions. ² only points along depth discontinuities. ³ all points except for occluded areas, ² all points including half-occluded regions, ¹ only points along depth discontinuities.

REFERENCES