Algorithm-level Approximation for Fast (or not) Embedded Stereovision Algorithm

Justine Bonnot, Karol Desnos, Daniel Ménard

Algorithm-level Approximation for Fast (or not) Embedded Stereovision Algorithm

Justine Bonnot, Karol Desnos, Daniel Menard
Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164
Rennes, France
Email: firstname.lastname@insa-rennes.fr

***Abstract*—Because of the growing concern towards the energy consumption of embedded devices, the quality of an application is now considered as a new tunable parameter during the implementation phase. Approximations are then deliberately introduced to gain performance. Nevertheless, when implementing an approximate computing technique, quality deteriorations may appear. In order to check that the application Quality of Service is still met despite the induced approximations, several metrics can be used. The proposed method introduces an algorithm-level approximate computing method in a stereovision algorithm. The proposed algorithm-level approximation aims at reducing the computational load in a stereo matching algorithm that outputs a depth map from two rectified images. Based on a smart loop perforation technique, this method offers an interesting quality/complexity trade-off. However, when comparing the obtained results to a more basic approximation technique, the results show that the quality/computation time trade-off is strongly dependent on the metric used. Our paper presents the impact of the choice of the quality metric on the results of the proposed approximate computing technique.**

I. Introduction

The technological progress in microelectronics as well as the Internet of Things era require that embedded applications integrate more and more sophisticated processing. Especially, most embedded applications incorporate data-oriented processing with mathematical computations. The complexity of embedded applications is increasing even though the energy constraints they must fulfill are more and more drastic. The design challenge is to provide a real-time implementation of these applications without sacrificing energy consumption. To reduce energy consumption in an application, numerous approximate computing techniques have been developed. Approximate computing takes advantage of the fact that strict accuracy is generally not needed in image and signal processing applications. The accurate implementation of applications is slightly modified to tolerate a mastered and acceptable error in order to save energy and/or computation time.

In this paper, the approximation of a computer vision algorithm, the stereo matching algorithm is under consideration. The targeted stereo matching algorithm is built with basic blocks massively used in computer vision algorithms as in [1]. The stereo matching algorithm is used to extract a 3D information from two 2D images taken by two rectified cameras spaced by a small distance. This 3D information is represented in a depth map, also called disparity map. The disparity map, whose size is identical to the size of the input images, represents the horizontal distance between the position of a pixel in the first image, and its position in the second image. The challenge, when introducing approximate computing in the stereo matching algorithm, is to lighten the computational load of the algorithm to be able to embed it in widespread platforms, as for instance digital signal processors. Several techniques have been proposed to reduce the complexity of the stereo matching algorithm: 1) The accuracy can be controlled with the data format used to implement the algorithm [8]. 2) The input images can be downsampled, thus reducing the volume of computations but also drastically the output quality. To evaluate the quality of the stereo matching algorithms, two metrics can be used: 1) The Middlebury metric [10] is considered as the reference metric to evaluate the quality of the obtained depth maps. 2) The structural similarity metric [12] is proposed to render the human visual perception of the degradation better.

In this paper, we propose a method to reduce the computational load of the stereo matching algorithm using an algorithm-level approximation technique. Approximate computing is indeed interesting for the stereo matching algorithm since the end-user of the algorithm can be a human or a neural network. These end-users are both error-resilient since the perception of a human user limitates the required accuracy, and a neural network learns from the approximate output and compensates the induced errors. The quality of the proposed approximation technique has been evaluated with the reference Middlebury metric and the structural similarity metric. Nevertheless, despite an interesting quality/complexity trade-off, the results obtained on the quality/computation time trade-off are strongly dependent on the metric used. The proposed article presents the algorithm-level approximation technique as well as the obtained results on the complexity and on the computation time.

The paper is organized as follows. First, the related work is detailed in Section II. The proposed approximation method for computing the depth map is detailed in Section III. Finally, the experiment results in comparison with the reference stereo matching algorithm are given in Section IV.

II. Related Works

A. Approximate computing techniques

Three action levels exist when implementing an approximate computing technique: the hardware, the data and the computation level, as illustrated in Figure 1. Among the three action levels illustrated in Figure 1, several approximation techniques have been applied on the proposed stereovision algorithm in the literature.

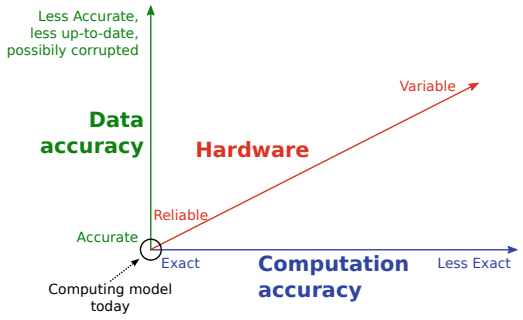


Fig. 1: Different action levels of approximate computing

To begin with, the format of the data transiting in the stereo matching algorithm can be modified with fixed-point coding, reducing the dynamic range as well as the output precision but offering a faster implementation than with floating-point data. In [8], the stereo matching algorithm has been converted to fixed-point to speed-up the computations. Then, the amount of data to process can also be reduced, downsampling the input images. The complexity of the algorithm is then reduced, as its computation time.

Approximations in the stereo matching algorithm can also be done implementing approximate operators, as presented in [2]. Approximate operators are implemented slightly modifying the accurate structure of the original operator to save energy.

In [8], an intricate mathematical function has also been replaced by piecewise linear functions. Different techniques of approximations on the computations can be used: for a hardware implementation, bi or multipartite tables [4] can be used, and for software function evaluation, a polynomial approximation can be implemented [3]. To reduce the computational volume at the algorithm level, loop perforation can be used. Only a subset of the iterations of a loop is executed, as proposed in [11]. In [9], an algorithm-level technique is applied to trade off the performance of the stereo matching application and its output quality of service. In the spirit of loop perforation, the proposed method is an approximation at the algorithm-level. A subset of the iterations of the loop is not executed to introduce a mastered error adjusted with the choice of this subset. The introduced error is then used to trade-off the complexity of the algorithm and the quality of the output depth map.

B. The reference stereo matching algorithm

The stereo matching algorithm has been mostly designed and studied on Desktop Graphics Processing Units (GPUs) and has consequently not been optimized for embedded systems. The high complexity of the stereo matching algorithm limits its embeddability. Indeed, if no special care is taken during the prototyping phase of an algorithm, the high precision of the produced results is often obtained at the expense of high latency, memory storage or energy consumption. However, embedded systems are interesting targets for computer vision applications. Real-time and power consumption constraints driving the design of embedded systems increase the need to adapt the stereo matching algorithm before embedding it. Besides, depth information becomes necessary on such platforms with the development of general public consoles using 3D information, as for instance the Kinect peripheral [6]. The proposed stereo matching algorithm aims at answering the problems of the Kinect peripheral. The Kinect peripheral uses an infra-red grid to derive the depth map, and is consequently limited to indoor usage as well as sensitive to infra-red interferences.

The stereo matching algorithm mimics the human visual system. It outputs a depth map from two rectified input images. The two rectified images correspond to the images seen from the left and right eye respectively, in the human visual system. In the stereo matching algorithm, a pixel in the first image is selected, and its matching pixel in the second image is searched. The pixel searched corresponds to the same physical point as the pixel in the first image in the captured scene. Then, the horizontal distance between those two pixels is computed: this is the disparity of this pixel. The disparity of a pixel corresponds to its depth in the input images, and is found minimizing a cost function. The cost function to minimize computes the cost of matching a pixel in the first image to a pixel in the second image. The principle of the stereo matching algorithm is illustrated in Figure 2 extracted from [5].

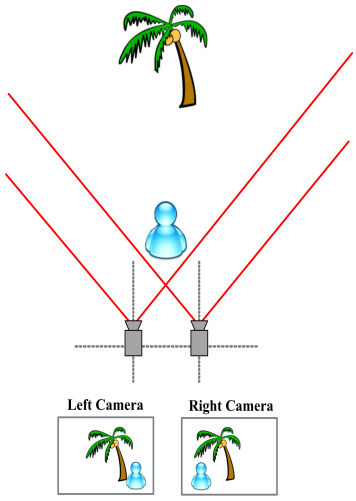


Fig. 2: Principle of the stereo matching algorithm

Two categories of algorithms have been proposed in the literature, depending on the technique used to minimize the cost function. The minimization technique can be global or local. Local methods optimize the computed cost of matching a pixel in the first image with a pixel in the second image using the neighboring pixels of the pixel under consideration. Global methods optimize the computed cost over the whole image. Intuitively, the matching cost will be lower for pixels with similar colors and layout of neighboring pixels (similar gradient, colors, edges). Local methods provide a lower quality compared to global methods but are more efficient in terms of computation time. The chosen reference stereo matching algorithm uses a local minimization method to better suit the targeted platform.

The main steps of the stereo matching algorithm are detailed below and represented in Figure 3:

- **Cost Construction**: computation of the cost of match-

ing a pixel in the left image with a pixel in the right image for a given disparity (distance between those pixels).

- **Cost Aggregation**: refinement of the cost maps obtained from the cost computation step.
- **Cost Minimization**: selection of the disparity leading to the minimum cost. This step computes the *argmin* of the cost function over the different possible disparity levels.

Let N_p and N_d be the total number of pixels in the image and the number of tested disparity levels respectively. This computationally intensive algorithm studies, for a pair of pixels, all the disparity possibilities in order to create the disparity map. In other words, the cost construction, aggregation and minimization functions are applied $N_p * N_d$ times. An exhaustive search of the minimum of the cost function is processed. This costly exhaustive search appears to be a challenge when it comes to embedding the stereo matching algorithm.

III. Proposed method

A. The approximate stereo matching algorithm

The proposed method skips computations of the cost function on several disparity levels, as it has been proposed in [7] with image perforation. In this contribution, a technique similar to loop perforation is applied to image pipelines. The authors proposed to skip the computations of costly images to the benefit of a speed-up in the algorithm computation time. In the case of the stereo matching algorithm, to avoid the exhaustive search of the disparity level minimizing the cost function, a study of the cost function is proposed. The cost function computes the cost for each disparity level d, of matching a pixel of coordinates (x, y) in the first image to a pixel of coordinates $(x + d, y)$ in the second image. The cost function computes the similarity between those two pixels, in terms of intensity and local texture. In terms of complexity, the computation of the matching cost for a given pixel and a given disparity needs the computation of $2+28*N$ multiplications, where N is the number of iterations to refine the cost computation.

A study of the minimized cost function allows selecting a subset of the disparity levels, avoiding to test them all.

For each pixel, the function $Cost = f(d)$ where d is the disparity level, is studied and an example of this function is represented in Figure 4. The cost function possesses the quasi-unimodality property, i.e. the following property is quasi ensured:

Definition III-A.1. *A function f, defined on the interval I, is unimodal on I if it has a unique minimum x_0 on I, it is strictly decreasing on $]-\infty; x_0]$ and strictly increasing on $[x_0; \infty[$.*

The cost functions have been characterized of being quasi-unimodals since they possess a global minimum but also small jumps. The strict monotonicity is then not guaranteed. For each pixel, the cost function is quasi strictly monotonic on each side of its minimum. Consequently, a search space decimation technique can be implemented to find the minimum of the function without testing all the disparity levels. Iteratively refining the search space around local minima converges towards the global minimum in case of a unimodal function, if the number of iterations is big enough. If the search space is well derived, the output quality is acceptable. The challenge of the search space decimation technique is to encompass the global minimum in the decimated space. To ensure this property, the proposed solution aims at exploring the more disparity levels in the first iteration, so as to ensure the convergence towards the global minimum and not a local one.

The proposed search space decimation technique is modeled by a tree structure $\mathcal{T}$ representing at each level of the tree the number of tested disparity levels. This tree structure represents the number of iterations to converge towards the solution (depth of the tree N_l), as well as the number of tested disparity levels in each iteration of the approximate algorithm (number of children at each level of the tree $\mathcal{T}[i]$). In Figure 4, the exhaustive search of the disparity of the considered pixel leads to test 60 disparity levels. The disparity level leading to minimum cost is $d = 33$. With the proposed approximate algorithm, an example of the tree structure used on this cost function is represented in Figure 5. This cost function is extracted from the image Teddy [10].

The depth N_l of the proposed tree structure is 3. To ensure that this tree structure encompasses the disparity level leading to the minimum cost, the more disparity levels are tested in the first level. The cost computation and cost minimization steps are applied on subsets of the disparity levels to test. Once the disparity leading to the minimum cost has been found for each pixel, the neighboring disparity levels are tested in the following levels of the tree.

In **level** 0, the first iteration aims at testing a maximum of disparity levels to be close enough to the global minimum of the cost function. In the proposed example, $\mathcal{T}[0] = 15$, thus, the cost is computed on 15 disparity levels, for the considered pixel. The disparity leading to the minimum cost is equal to 34. The minimum will be searched in the following levels next to the disparity level 34. In **level** 1, the obtained result is refined. In the proposed example, $\mathcal{T}[1] = 2$, thus, the cost is computed on the 2 disparity levels surrounding the disparity level selected in the previous level (disparity levels 32 and 36) and the disparity level leading to the minimum cost is selected, $d = 32$. The minimum will be searched in the following levels next to $d = 32$. In **level** 2, $\mathcal{T}[2] = 2$. The cost is computed on the 2 disparity levels surrounding the disparity level selected in the previous level (disparity levels 31 and 33) and the disparity level leading to the minimum cost is selected as being the disparity of the considered pixel, $d = 33$.

If the search space has been decimated encompassing the minimum of the cost function, the disparity obtained at the end of the approximate algorithm is the disparity level minimizing the cost function.

The tree $\mathcal{T}$ storing the decimated search space, models a trade-off between the computation time of the approximate stereo matching algorithm and the quality of the output depth map. Indeed, the complexity of the algorithm, which impacts the computation time, is reduced with the total number of tested disparity levels, since the cost computation and cost minimization functions composing the stereo matching algo-

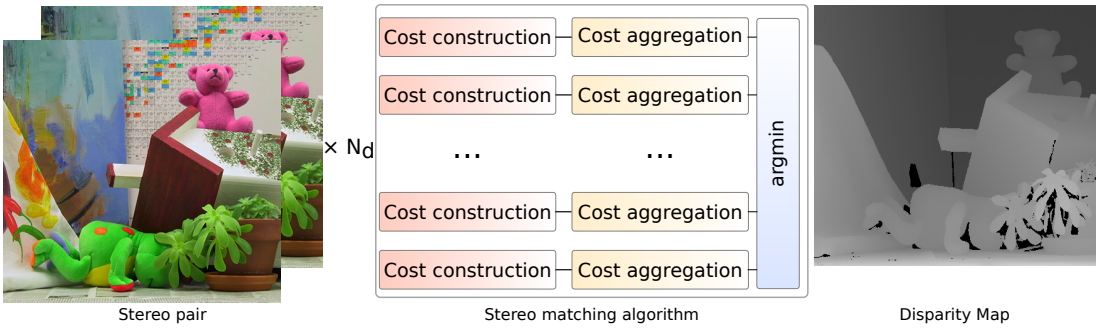


Fig. 3: Illustration of the reference stereo matching algorithm: exhaustive test of all the disparity levels

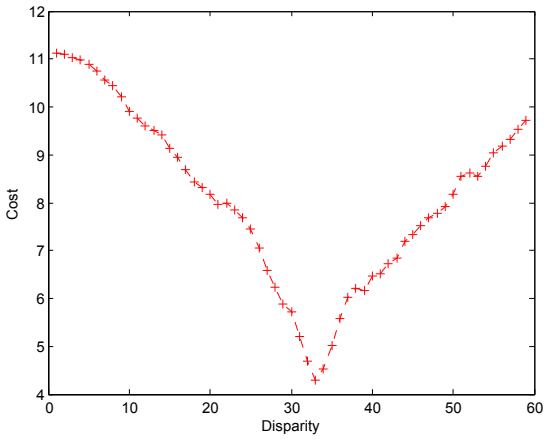


Fig. 4: Cost against the disparity level for a given pixel

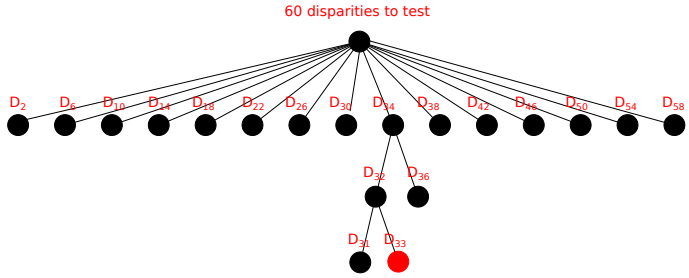


Fig. 5: Tree Structure $\mathcal{T}$ obtained for a given pixel for Teddy ($N_l = 3$)

rithm are called a smaller number of times. On the exhaustive search, for the image Teddy, the three functions are called $N_d \times N_p$ times. With the proposed method, the number of tested disparity levels for each pixel is equal to the sum of numbers of disparity levels tested at each level of the tree $\mathcal{T}$. With the tree structure proposed in 5, the number of tested disparity levels for each pixel is reduced to $\sum_{i=1}^{3} \mathcal{T}[i] = 19$.

The global complexity of the algorithm is reduced along with the output quality.

B. *Selection of the tree structure*

The proposed method is based on the selection of the tree structure $\mathcal{T}$ modeling the search space. To select $\mathcal{T}$, a minimization problem under constraints is solved. The tree of depth 1 has the highest computational complexity since every disparity level is tested. The tree of depth 1 gives the quality reference, outputting a depth map equivalent to the depth map given by the reference algorithm. The tree of maximum depth, which is the number of factors in the prime factorization of the number of disparity levels to test, tests the least disparity levels. If ϵ is the maximum acceptable degradation compared with the reference output, the problem to solve is the following:

Let $\mathcal{T}$ be the tradeoff tree, $\mathcal{T}[i]$ be the number of disparity levels tested at level i, N_l be the number of levels in $\mathcal{T}$, N_d be the maximum number of disparity levels and $\mathcal{Q}$ the output quality compared to the output reference:

$$\begin{aligned} \underset{i}{\text{minimize}} \quad & \sum_{i} \mathcal{T}[i] \\ \text{subject to} \quad & \mathcal{Q} \leq \epsilon \\ & \prod_{i=1}^{N_l} \mathcal{T}[i] = N_d \end{aligned}$$

The second constraint of the optimization problem expresses the fact that no overlapping is tolerated when testing the disparity levels. The deeper the tree, the lower the output quality. Indeed, if the global minimum of the cost function has to be reached in a fixed number of iterations, N_l, the more disparity levels are tested in the first iteration, the more chances to reach the global minimum there are. The tree that tests the most disparity levels in the first iteration is the least deep. On the proposed example, the maximum depth of the tree $\mathcal{T}$ is 4.

The second constraint, $\prod_{i=1}^{N_l} \mathcal{T}[i] = N_d$, can be relaxed so as to increase the output quality while testing a lower number of disparity levels. Two other parameters can be adjusted to derive the different disparity levels to test: the spacing between each tested disparity level at each level of the tree, represented in column $\mathcal{S}$ of Table I, as well as the values used to compute the first disparity level at each level of the tree, from the first disparity of the previous level of the tree, represented in column $\mathcal{F}$ of Table I. The different tested tree structures are represented in Table I. For the tree structure illustrated

	$\mathcal{T}$	$\mathcal{S}$	$\mathcal{F}$	Complexity
Depth 1	$\{60\}$	$\{1\}$	$\{0\}$	60
Depth 2	$\{30, 2\}$	$\{2, 1\}$	$\{1, -1\}$	32
	$\{20, 3\}$	$\{3, 1\}$	$\{1, -1\}$	23
Depth 3	$\{15, 2, 2\}$	$\{3, 2, 1\}$	$\{3, -2, -1\}$	19
	$\{16, 2, 2\}$	$\{3, 2, 1\}$	$\{3, -2, -1\}$	20
	$\{17, 2, 2\}$	$\{3, 2, 1\}$	$\{3, -2, -1\}$	21
	$\{18, 2, 2\}$	$\{3, 2, 1\}$	$\{3, -2, -1\}$	22
Depth 4	$\{5, 3, 2, 2\}$	$\{12, 4, 2, 1\}$	$\{9, -6, -2, -1\}$	12

TABLE I: Different tested tree structures

in Figure 5, the tested configuration of $\mathcal{T}$ is $\mathcal{T} = \{15, 2, 2\}$, $\mathcal{S} = \{3, 2, 1\}$ and $\mathcal{F} = \{3, -2, -1\}$.

In Table I, the complexity is the number of tested disparity levels on each pixel. The reference stereo matching algorithm computes the cost construction, aggregation and minimization on $N_d = 60$ disparity levels. Reducing the number of computations of these three functions should reduce the computation time of the stereo matching algorithm along with the output disparity map quality.

IV. Experimental Results

The proposed method provides a trade-off between the quality of the output depth map and the complexity/computation time of the stereo matching algorithm. The proposed approximate algorithm is compared to the reference stereo matching algorithm as well as to a downsampling method with two quality metrics. The first quality metric is the reference metric on the stereo matching algorithm, the Middlebury metric, and the second one is more representative of the real perceived quality, the structural similarity metric.

The downsampling method consists of applying the reference stereo matching algorithm to a pair of images of resolution divided by 4. The proposed method has been tested on the 2003 dataset of the Middlebury database [10] (two images in full resolution, Teddy and Cones) and the outputs of the different methods are compared to the Ground Truth given by the Middlebury database and obtained thanks to laser measurements of the depth of the pixels.

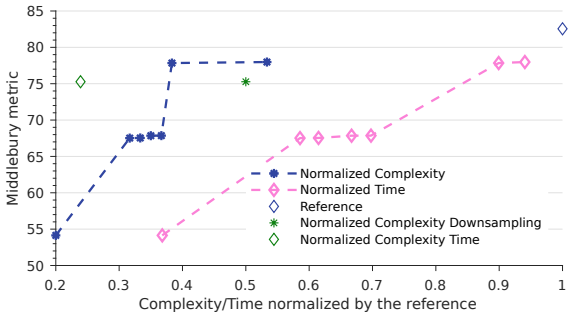
The results have been obtained on an Odroid XU3 board [1] possessing a Cortex A15 processor working at 1.8 GHz. The Odroid XU3 board is heavily used in the embedded systems domain, since it is highly energy-efficient while offering an important computational capacity. This target allows being in the real conditions of embedding the stereo matching algorithm.

A. Results with the Middlebury metric

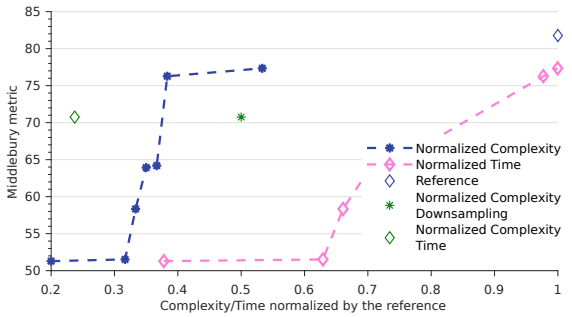
The first quality metric is proposed along with the Middlebury database. This quality metric compares the output of the chosen stereo matching algorithm to the Ground Truth obtained with laser measurements. This metric outputs a percentage of good pixels, corresponding to the percentage of pixels sharing the same disparity level in both images.

Figures 6a and 6b show the trade-off between the quality of the output depth map measured with the Middlebury metric, and the complexity/computation time of the algorithm, for the images Teddy and Cones respectively. The theoretical

[1] http://www.hardkernel.com



(a) Teddy



(b) Cones

Fig. 6: Quality/complexity, computation time trade-off for the Middlebury metric.

results on the complexity reduction are compared with the obtained results on the computation time. Both curves are following the same trend, but the normalized computation time is shifted compared to the normalized complexity. Indeed, the reduction of the number of disparity levels to test requires the computation of the cost functions on a non-regular pattern, which generates an overhead in terms of computation time. This implementation overhead is not taken into account with the representation of the complexity. Besides, computing the cost function on the whole disparity map allows mutualizing the computations, thus saving computation time.

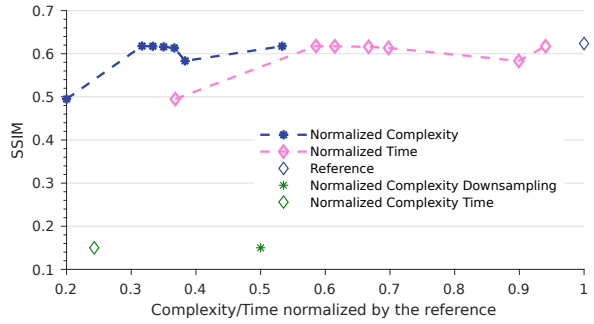
The reference algorithm, with no approximation is represented by the rightmost point ($N_l = 1$). The three functions are computing on 60 disparity levels for each pixel. Then, the points are evolving with the different tree configurations presented in Table I, showing an interesting group of points with a large complexity reduction while losing 15% of good pixels for Teddy, and 17% for Cones.

The downsampling technique is also represented. This method seems to give better results than the proposed approximate algorithm. For both images, to get a comparable quality of the output depth map, the downsampling technique takes half less time than the proposed approximation technique. According to the Middlebury metric, the proposed approximation technique possesses a non-negligible overhead to gain over a more simple technique as downsampling the input images.

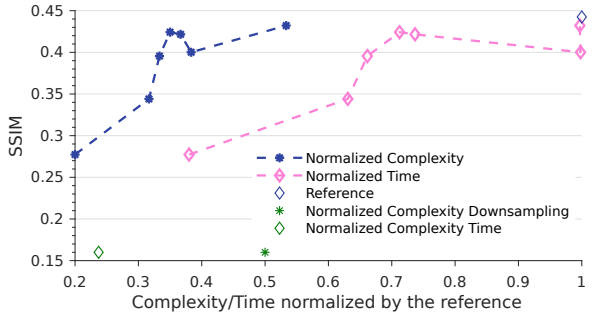
B. Results with SSIM metric

The degradation generated by the downsampling technique was negligible using the Middlebury metric. When visually comparing the output depth maps obtained with the downsampling technique, the quality seems to be strongly degraded

though. Along with approximation techniques allowing to benefit from the imperfection of the end-sensors of some algorithms, new error metrics have been developed to better take into account the real perceived quality. The structural similarity is a metric defined in [12] notably to better take into account the human perception when evaluating the degradation of an image signal. This metric is based on the assessment that the human eye mostly detects the image structural modifications. The computation of the Structural SIMilarity (SSIM) Index is applied when the reference image is known. This image is then considered as being of perfect quality. Then, the SSIM Index computation takes into account different parameters to compare them on the two images. To compare the reference image and the approximated one, the SSIM metric segments the images into blocks and compares statistical information between the corresponding blocks in both images. The results obtained with the SSIM metric are presented in Figures 7a and 7b for the images Teddy and Cones respectively.



(a) Teddy



(b) Cones

Fig. 7: Quality/complexity, computation time trade-off for the SSIM metric.

For the proposed method, the Ground Truth given by the Middlebury database is taken as being the reference image. The closer to 1 the SSIM Index is, the better the result. Consequently, compared to a downsampling technique, which offers a quicker result but a strong quality degradation (SSIM Index equal to 0.15 for Teddy and 0.16 for Cones), the proposed method keeps the SSIM Index close to the reference quality and is, in the case of the tree of depth 4, faster than the downsampling technique. The results of the proposed approximation form a plateau that gives the possibility to reduce the computation time by 50% while keeping almost the same output quality. With the SSIM quality metric, the quality loss on the output image is negligible for a relatively important computation time saving.

C. Analysis of the results

When implementing this approximation technique, it has been noted that the obtained results compared to a more basic downsampling method are strongly dependent on the chosen quality metric. Indeed, the downsampling technique does not save the contours and blurs the output disparity map. The reason why the Middlebury quality metric is more in favor of the downsampling technique is that if the selected disparity levels in the compared disparity maps differs from less than 3 levels for a given pixel, this pixel does not count as a wrong pixel. One may question the validity of such a metric that does not render the perceived quality.

V. Conclusion

The proposed algorithm-level approximate computing technique applied to the stereo matching application exploits the error-resilience property of this algorithm. This property allows testing fewer disparity levels per pixel using the quasi-unimodality property of the computed cost function to lighten the volume of computations. This cost is computed to select the disparity level leading to the minimum cost for each pixel. Avoiding to exhaustively test all the disparity levels for each pixel, the approximate algorithm has a reduced complexity and the computation time is reduced compared to the reference stereo matching algorithm. Nevertheless, compared to a basic approximation technique as downsampling the input images, the proposed contribution does not seem to be interesting using the reference metric. A metric more representative of the perceived quality is then strongly needed. The SSIM metric has then been chosen and has demonstrated the use of such an approximation technique, allowing to half the computation time of the stereo matching algorithm while having a negligible impact on the quality.

Acknowledgment

This project has received funding from the French Agence Nationale de la Recherche under grant ANR-15-CE25-0015 (ARTEFaCT project).

Acknowledgment

References

[1] Oliver Jakob Arndt, Daniel Becker, Christian Banz, and Holger Blume. Parallel implementation of real-time semi-global matching on embedded multi-core architectures. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, pages 56–63. IEEE, 2013.

[2] Justine Bonnot, Karol Desnos, Maxime Pelcat, and Daniel Menard. A fast and fuzzy functional simulator of inexact arithmetic operators for approximate computing systems. In *Proceedings of the 28th edition of the ACM Great Lakes Symposium on VLSI (GLSVLSI), accepted for publication*. ACM, 2018.

[3] Justine Bonnot, Erwan Nogues, and Daniel Menard. New non-uniform segmentation technique for software function evaluation. In *Application-specific Systems, Architectures and Processors (ASAP), 2016 IEEE 27th International Conference on*, pages 131–138. IEEE, 2016.

[4] F. de Dinechin and A. Tisserand. Multipartite table methods. *IEEE Transactions on Computers*, 54(3):319–330, March 2005.

[5] Maarten Dumont. *Real-Time View Interpolation for Eye Gaze Corrected Video Conferencing*. PhD thesis, transnationale Universiteit Limburg, 2015.

[6] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[7] Liming Lou, Paul Nguyen, Jason Lawrence, and Connelly Barnes. Image perforation: Automatically accelerating image pipelines by intelligently skipping samples. *ACM Transactions on Graphics (TOG)*, 35(5):153, 2016.

[8] Judicaël Menant, Muriel Pressigout, Luce Morin, and Jean-Francois Nezan. Optimized fixed point implementation of a local stereo matching algorithm onto c66x dsp. In *Design and Architectures for Signal and Image Processing (DASIP), 2014 Conference on*, pages 1–6. IEEE, 2014.

[9] Edoardo Paone, Davide Gadioli, Gianluca Palermo, Vittorio Zaccaria, and Cristina Silvano. Evaluating orthogonality between application auto-tuning and run-time resource management for adaptive opencl applications. In *Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference on*, pages 161–168. IEEE, 2014.

[10] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

[11] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 124–134. ACM, 2011.

[12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.