An automatized method to parameterize embedded stereo matching algorithms

Judicaël Menant, Guillaume Gautier, Muriel Pressigout, Luce Morin, Jean Francois Nezan

An automatized method to parameterize embedded stereo matching algorithms

Judicaël Menant

UBL, INSA , IETR UMR 6164, Rennes, France

Guillaume Gautier

UBL, INSA, IETR UMR 6164, Rennes, France

Muriel Pressigout

UBL, INSA, IETR UMR 6164, Rennes, France

Luce Morin

UBL, INSA, IETR UMR 6164, Rennes, France

Jean-François Nezan

UBL, INSA, IETR UMR 6164, Rennes, France

Abstract

Many applications rely on 3D information as a depth map. Stereo Matching algorithms reconstruct a depth map from a pair of stereoscopic images. Stereo Matching algorithms are computationally intensive, that is why implementing efficient stereo matching algorithms on embedded systems is very challenging for real-time applications.

Indeed, like many vision algorithms, stereo matching algorithms have to set a lot of parameters and thresholds to work efficiently. When optimizing a stereo-matching algorithm, or changing algorithms parts, all those parameters have to be set manually. Finding the most efficient solution for a stereo-matching algorithm on a specific platform then becomes troublesome.

This paper proposes an automatized method to find the optimal parameters of a dense stereo matching algorithm by learning from ground truth on a database in order to compare it with respect to any other alternative.

Finally, for the C6678 platform, a map of the best compromise between quality and execution time is obtained, with execution times that are between 42 ms and 382 ms and output errors that are between 6% and 9.8%.

Keywords: embedded vision, DSP, stereo matching

Introduction

Embedded vision is the merging of two technologies corresponding to embedded systems and computer vision. An embedded system is any microprocessor-based system that is not a general-purpose computer [1]. Our work is about smart implementation of computer vision algorithms in modern embedded systems to provide them with stereo perception.

Stereo perception aims to add depth information to the color images of a scene. There are two ways to obtain Red Green Blue Depth (RGBD) information onto embedded systems: either active devices such as Kinect or stereo matching algorithms that compute depth information from two or more images. Active systems[2] emit signales on the observed scene, as for example an infra-red grid for the Kinect sensor, or laser beams for structured-lights. The disparity map is then deduced from this sensed-back grid. Those devices are limited to indoor use. For example the kinect sensor is limited by its 5 meter range and its sensitivity to infrared interferences. This paper focuses on binocular stereo vision algorithms to bypass these limitations.

Stereo matching aims to create 3D measurements from two 2D images, generating a disparity map which is inversely proportional to the depth of any object to the acquisition system [3]. Disparity maps are used in a wide range of scenarios where depth must be computed (3D TV, free-view point video,...). Stereomatching algorithms can be divided into two main classes: the dense one and the sparse one. Sparse stereo matching algorithms con-

Email addresses: jmenant@insa-rennes.fr (Judicaël Menant), ggautier@insa-rennes.fr (Guillaume Gautier), mpressig@insa-rennes.fr (Muriel Pressigout), lmorin@insa-rennes.fr (Luce Morin), jnezan@insa-rennes.fr (Jean-François Nezan)

sider only a set of interest points whereas dense stereo matching algorithms deal with all pixels. In this paper we consider only dense stereo matching algorithms as they are a lot more computionnaly expensive and used in many 3D applications such as autonomous cars, drone navigation and transitional view reconstruction.

Most existing implementations of dense stereo matching algorithms are carried out on desktop Graphical Processor Unit (GPU), leading to poor energy efficiency. On top of that the GPUs used for the implementations are different so that it is very difficult to really compare the complexity of the algorithms. Energy-efficient embedded platforms are now available. They are ideal for widespread integration into everyday objects. The C6678 platform is a recent 8-core Digital Signal Processor (DSP) platform at the state of the art in the field. The C6678 is clocked at 1 GHz with a standard 10W power consumption. The C6678 platform targets video processing applications, and has large computing capabilities, up to 320 Giga Multiply ACcumulate per Second (GMACS). Moreover, one of the main problematic with stereo matching algorithms is the high rate of memory transfer. The C6678 platform has a cache that can be configured as local memory, which allows to stretch the limits and avoid the memory wall. All those characteristics make the C6678 a good platform for stereo matching applications. Let note that the C6678 being a typical DSP platform, the results can be extrapolated to any DSP platform and Central Processor Unit (CPU).

However, the architecture of embedded systems is significantly different to the architecture of desktop systems. The challenge is therefore to find and adapt algorithms and implementations that can fully exploit the powerful computational capabilities of such an architecture. First of all, to be efficiently implemented on embedded systems, algorithms must be ported to fixed point implementation. In previous works [4], we adapt some state-of-the-art algorithms. However it was difficult to compare them since they rely on parameters that need to be tuned.

The goal of this paper is to provide a setup that enables to compare different dense stereo matching algorithm configurations with several trade-offs between the output quality (*ie.* a good estimation of the depth map) and the latency on the C6678 platform. The presented algorithms are the state-of-the-art algorithms that fit efficiently on a DSP platform.

Indeed, to get the best compromise between the quality and the latency, many configurations must be tested. Each configuration has parameters that can be divided in two classes. Some of these parameters affect both the execution time and the output quality, and the others ones affect only the quality. Currently state-of-the-art algorithms have empirically set those parameters. This paper proposes to set by a non-supervised automatized method the parameters of stereo matching algorithms, based on a trichotomic search for the ones that affect only the quality and not the latency.

The next section introduces the stereo matching techniques and all algorithms used in the comparison part. Secondly, the configurations considered in this paper are enumerated. Then the method used to automatized the configuration of those algorithms is detailed. Finally, the results will be exposed followed by a conclusion.

1. Stereo matching techniques

Our work focuses on dense stereo matching algorithms, as exposed previously in the Introduction.

In order to retrieve depth information from two stereoscopic images, dense stereo matching algorithms find the pixel-wise correspondence between those two images for each pixel. The correspondence is defined by the disparity: the displacement vector of the pixel between the two images. When the system is rectified, this disparity is then a scalar value corresponding to the horizontal 2D motion between corresponding pixels in left and right images (see figure 1). All disparities values of an image are stored in a disparity map. The disparity map provides a disparity value for each pixel. The bigger the disparity, the closer the object from the two cameras. As the disparity is inversely proportionnal to depth, it may also be named (improperly) depth map.

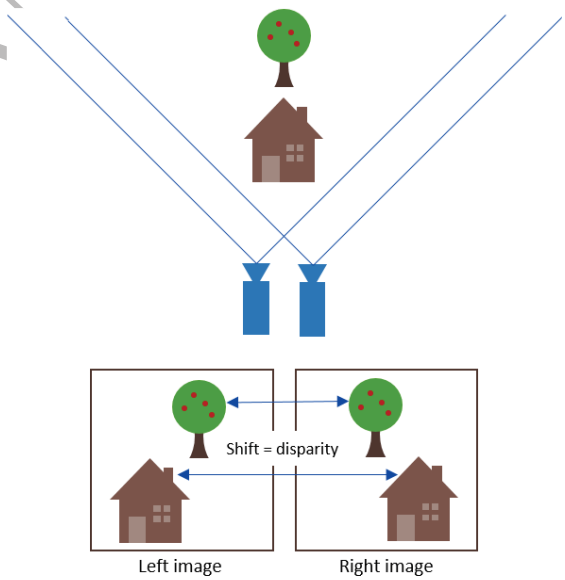

Figure 1: Disparity in stereoscopic images

Dense stereo matching algorithms are mainly divided into three classes, *local* and *global* and *semi-global* methods [5]. The disparity value computed with a local method depends only on colorimetric values of pixels within a finite window. It is thus very sensitive to noise. Semi-global stereo matching algorithms add a constraint of smoothness of the output disparity map. Global approaches aim to optimize the estimated depth map by minimizing a cost function computed on the whole image. As memory constraints of embedded systems do not allow the use of global

algorithms, global approaches will not be considered in this paper.

Any dense stereo matching algorithm can be divided into three main parts :

- *Cost construction* measures the similarity of two pixels considering the colorimetric values of those pixels in a finite neighbourhood.
- *Cost aggregation* is optional. It refines costs at output of the cost construction step by adding consistency between disparity values.
- *Disparity selection* deduces the final disparity value for a given pixel from the costs produced by the previous steps.

This paper studies five different configurations of stereo matching algorithms extending the algorithms presented in [6, 7]. Compared with [7], new algorithms and configurations have been implemented and evaluated : One Dimension Belief Propagation (BP-1D) is implemented with a census cost ; Semi Global Matching (SGM) disparity selection proposed by Heiko Hirshmüller [8] has been implemented on the C6678 platform ; a new configuration based on the combination of the Bilateral Filter Aggregation (BFA) as cost aggregation and BP-1D as disparity selection is proposed and evaluated. Those configurations have been added since they are expected presenting an interesting trade-off between execution time and output quality.

The goal of this paper is to provide a help to choose the best compromise between execution time and quality in an embedded environment.

Indeed, each part of stereo matching algorithm, like most of image processing algorithms, has a set of parameters and constants (thresholds ...) that have to be set. This paper provides a fully automatised method for finding the set of optimal parameters of stereo matching algorithms.

This section presents all algorithms used as cost construction (census), cost aggregation (BFA) and disparity selection (Winner Takes it All (WTA), SGM, BP-1D).

In this paper the following notations will be used :

- I_b and I_m are the input images. I_b is the base image, and I_m is for the matching image. When disparities are computed, the output disparity map is mapped on I_b. In this paper I_b is the left image and I_m is the right image : the disparity is therefore positive. I_b and I_m are gray scale images.
- A disparity level is noted d. It ranges from 0 to N_{disp}, where N_{disp} is the number of disparity level considered.
- Pixel p coordinates are noted $p = (x, y)$ and a pixel p' shifted by disparity level d is noted $p' = (x - d, y)$.

1.1. Cost construction : Census

As presented previously, the cost construction is the first step in the stereo-matching process.

In our previous work, several cost construction approaches have been considered [4]. Census cost has always given the best trade-off between execution time and quality when implemented onto VLIW DSP cores. Moreover, the time/quality trade-off can be tuned by changing the neighbourhood window's size. Therefore, all algorithms in this paper use a Census cost construction, and we thus first introduce this method.

Census cost construction is based on pixel-wise Census signature, computed on the pixel neighbourhood, defined by a $N \times N$ window where N is odd.

A pixel Census signature $cen(p)$ is obtained by comparing the pixel p to its $N^2 - 1$ neighbours. It is composed of $N^2 - 1$ bits for each pixel, noted $cen(p)[k]$ where k is the bit index. $cen(p)[k]$ is defined by Equation (1). An example of Census signature computation for a 3×3 census is given in Figure 2.

$$cen(p)[k] = \begin{cases} 0 & \text{if } I(p) > I(p_k) \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

where p_k is the k^{th} neighbour of pixel p.

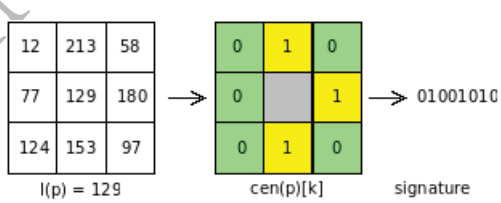

Figure 2: 3x3 census example

We define cen_b and cen_m, the Census signature obtained respectively from I_b and I_m grey level images. The matching process comes to find the same signature through the Census cost. The Census cost between a pixel p in I_b and a pixel p' in image I_m is defined by the difference between the signatures of two pixels :

$$C_{CEN}(p, p') = \sum_{k=1}^{N^2-1} cen_b(p)[k] \oplus cen_m(p')[k] \quad (2)$$

where $\oplus$ is the *exclusive or* operator. Its value is 1 if the two boolean operands are equal and 0 if they are different.

The size N of the Census window is a parameter of the Census cost computation. In this paper, three different values are tested : 3, 5 and 7. When the census window's size is superior to 7, we have observed experimentally for that types of images that the output quality reduces. Therefore a census size superior to 7 is not considered in this paper.

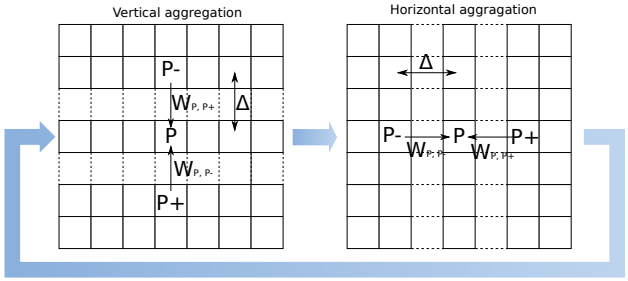

Figure 3: BFA iterations

1.2. Cost aggregation : BFA

The cost aggregation is the second and optional step of the stereo matching algorithm. The output of the first step, the census cost construction, is fundamentally noisy but in average is a good matching cost. To reduce this noise, the cost aggregation step performs smoothing on areas with a similar colour in the input image. The algorithm chosen in our paper for this part is BFA. It has been originally proposed by Mei [9, 10]. The BFA algorithm considers each disparity level independently. This is a key point regarding implementation, parallelization and memory footprint.

The cost aggregation algorithm's structure is similar to a bilateral filter. As shown in Figure 3, at each step the cost is refined according to Equation (3).

$$E_{i+1}(p) = \frac{W(p,p_+)E_i(p_+) + E_i(p) + W(p,p_-)E_i(p_-)}{W(p,p_+) + 1 + W(p,p_-)} \quad (3)$$

where E_i is the current cost map to be refined E_0 being the output of cost construction and pixels p_+ and p_- have a position relative to pixel p such as :

- $p_+ = p + \Delta_i$
- $p_- = p - \Delta_i$

Δ_i is a 1D offset : aggregation is computed alternatively for horizontal and vertical directions :

- When i is odd, it is a vertical aggregation and the offset Δ_i is vertical.
- When i is even, it is a horizontal aggregation and the offset Δ_i is horizontal.

At each iteration the parameter Δ_i grows as shown in Equation (4), thus further pixels p_+ and p_- are used for smoothing p. The influence range is limited by the modulo in Equation (4), here with $D_{max} = 33$ [11] ie $\Delta_i \in [0, 32[$.

$$\Delta_i = floor(i/2)^2 \bmod D_{max} \quad (4)$$

Weights W in equation (3) are defined by equation (5):

$$W(p_1,p_2) = \frac{thr - min(thr, sim(p1,p2))}{thr}.(1 - \Delta.C_d) \quad (5)$$

where thr is a threshold and $sim(p_1,p_2)$ is a measure of colour similarity in the neighbourhood such as:

$$sim(p_1,p_2) = \sum_{col\in(r,g,b)} |I_m col(p_1) - I_m col(p_2)| \quad (6)$$

where $col \in (r,g,b)$ are respectively the red, green and blue component of I_m and C_d is the distance weighting factor as presented in [9].

To conclude the BFA presentation, the input cost of the BFA algorithm is refined by Equation (3). The number of iteration is fixed, typically five vertical and horizontal iterations [11]. This latter sets the trade-off between the latency and the output quality, and is discussed in section 4.

The interest of a cost aggregation step is to enhance the cost map provided by the cost construction to the disparity selection. This step is even necessary when the cost construction produces noisy cost maps such as the census. The three disparity selection processes considered in this paper are presented in the next parts of this section.

1.3. Disparity selection : WTA

The disparity selection is the third and final step of the stereo matching algorithm after the cost construction (here the census) and an optional cost aggregation (here the BFA). The goal of a disparity selection algorithm is to select the disparity that minimizes its input cost map. The output of disparity selection is a dense integer disparity map providing a disparity value for each pixel in the right image I_m. We will present three possibilities to perform this step.

The first one, the WTA, is also the simplest disparity selection presented in this paper. For each pixel, the minimum disparity $Disp(p)$ is selected. The Equation (7) expresses the selection of the smallest disparity with the argmin operator.

$$Disp(p) = \underset{d\in[0,N_{disp}]}{\operatorname{argmin}} \; E_{d,Nit}(p) \quad (7)$$

The WTA is the most simple disparity selection algorithm and therefore sensitive to noise. The next parts of this section exposes more complex, and thus more robust and time consuming disparity selection algorithms.

1.4. Disparity selection : SGM

The SGM algorithm is the second choice to perform disparity selection. The SGM selects the disparity that minimizes the cost at its input whilst at the same time maximizing the smoothness of the disparity map.

The SGM algorithm [12] is one of the best semi-global methods [5]. It resolves a Hidden Markov Model (HMM) with a belief propagation. The SGM aggregates several belief propagation along several directions (Figure 4). It is important to notice that the aggregation of the several directions has nothing in common with the cost aggregation step of stereo matching.

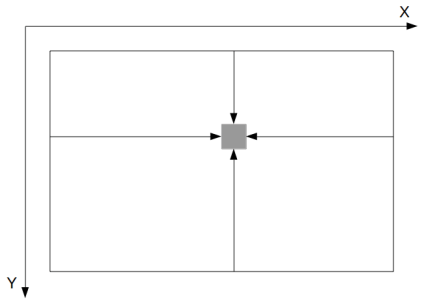

Figure 4: aggregation paths

Equation 8 defines SGM cost propagation along one direction r. P_1 and P_2 are penalties for small and large discontinuities. $C(p,d)$ is the cost at pixel p and disparity d. $L_r(p,d)$ is the cost propagated along direction r for disparity d at pixel p. p_{-r} is the previous pixel before p along direction r.

$$\begin{aligned} L_r(p,d) &= C(p,d) - \min_k(L_r(p_{-r},k)) \\ &+ \min\Big(L_r(p_{-r},d), L_r(p_{-r},d-1)+P_1, \\ &L_r(p_{-r},d+1)+P_1, \min_i(L_r(p_{-r},i))+P_2\Big) \end{aligned} \quad (8)$$

Costs for disparity d propagated along all r directions are finally aggregated according to equation (9):

$$S(p,d) = \sum_r L_r(p,d) \quad (9)$$

The output disparity map is obtained by minimising the aggregated cost of Equation (9) as shown in Equation (10):

$$Disp(p) = \underset{d\in[0,N_{disp}]}{\operatorname{argmin}} S(p,d) \quad (10)$$

1.5. Disparity Selection : BP-1D

The BP-1D algorithm is very close to the SGM algorithm. Indeed, both are based on HMM and use believe propagation to solve the problem. BP-1D propagates costs only horizontally, but adds logic to manage occulted areas by adding three possible states per disparity pixel (present on both image, only in the base image or only in the matching image).

The BP-1D uses a cyclopean view as reference instead of the left image. A cyclopean view is the view that would be obtained from a hypothetical third camera located exactly between the left and the right camera. The notations I_{left} and I_{right} are used for left and right image.

The process relies on a Graph of Profile Variants (GPV) as in Figure 5. Each node $u_x(d)$ of the GPV is related to one pixel in the left image $I_{left}(x_1)$ and one pixel in the right image $I_{right}(x_2)$ such as $x = (x_1+x_2)/2$ and $d = x_1 - x_2$. Each node has three possible states s : s can be equal to B if it is a binocularly visible point (i.e visible in both right and left pictures), or MR (resp. ML) if it is a right (resp. left) monocularly visible point. A Profile Variant is a line in the Graph of Profile Variants (GPV) giving a disparity d for each value of x as given in example in Figure 5. Due to ordering and visibility constraints, the transitions in the nodes v_x of the GPV are constrained as illustrated Figure 6. The stages v_x for $x = 1, ..., n$ of the HMM corresponds to a column of the Cyclopean view (Figure 5) where x is the position in the line. Each stage v_x has $m = N_{disp} * 3$ possible hidden states $u_x(d,s) \in U$ with $d \in [0, N_{disp} - 1]$ the disparity evaluated and s the state of the node.

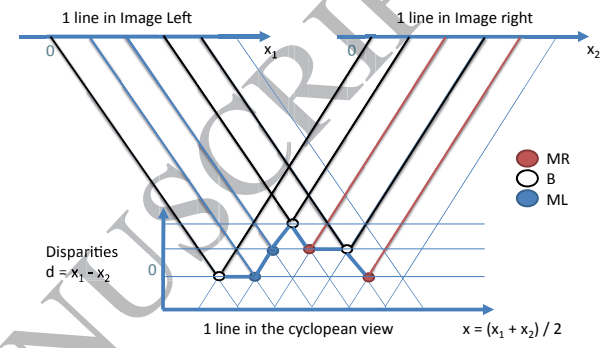

Figure 5: Graph of Profile Variants

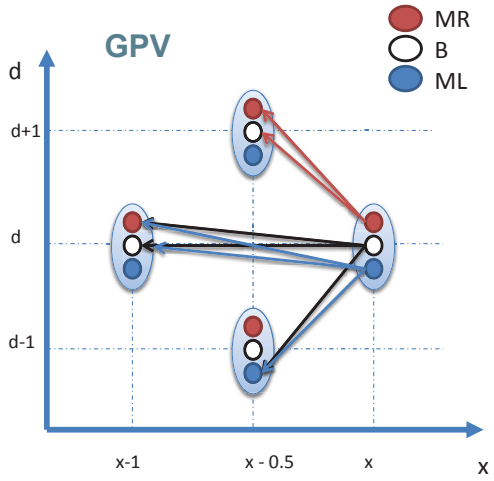

Figure 6: possible transitions in the GPV

BP-1D aims at finding the best Profile Variant in the GPV, i.e the Profile Variant minimizing an energy function E^* of the HMM as set in Equation (11). For a well-posed energy minimization problems, BP-1D gives the unique optimizer $\mathbf{u^*}$. However, as the stereo matching is an ill-posed problem, BP-1D gives one of the several possible solutions.

$$E^* = \sum_{x=1}^{n} E(v_x) \tag{11}$$

with

$$E(v_x) = \min_{d,s} E(u_x(d,s)) \tag{12}$$

The disparity selected for each node v_x is the one minimising energy $E(v_x)$:

$$D(x) = \underset{d \in [0, N_{disp}-1[}{\operatorname{argmin}} E(u_x(d,s)) \tag{13}$$

The energy $E(v_x)$ is the sum of several costs enabling to take into account the neighbourhood of the node in the HMM : F is the forward message, ϕ the backward message and φ a cost associated to each node.

$$E(u_x(d,s)) = F_x(d,s) + \phi_x(d,s) + \varphi_x(d) \tag{14}$$

The cost $\varphi_x(d)$ is the input cost provided by the cost construction or the cost aggregation.

Taking into account the possible transitions as illustrated in Figure 6, the forward message $F_x(d,s)$ is computed in a forward pass using the following equations :

$$F_x(d,ML) = F_x(d,B) = \min \begin{cases} F_{x-0.5}(d-1,ML)) + \varphi_0; \\ F_{x-1}(d,B)) + \varphi_{x-1}(d); \\ F_{x-1}(d,MR)) + \varphi_0; \end{cases} \tag{15}$$

$$F_x(d,MR) = \min \begin{cases} F_{x-0.5}(d+1,B)) + \varphi_{x-0.5}(d+1); \\ F_{x-0.5}(d+1,MR)) + \varphi_0; \end{cases} \tag{16}$$

The backward message $\phi_x(d,s)$ is computed in a backward pass using the following equations :

$$\phi_x(d,ML) = \min \begin{cases} \phi_{x+0.5}(d+1,ML)) + \varphi_0; \\ \phi_{x+0.5}(d+1,B)) + \varphi_{x+0.5}(d+1); \end{cases} \tag{17}$$

$$\phi_x(d,B) = \phi_x(d,MR) = \min \begin{cases} \phi_{x+1}(d,ML)) + \varphi_0; \\ \phi_{x+1}(d,B)) + \varphi_{x+1}(d); \\ \phi_{x+0.5}(d-1,MR)) + \varphi_0; \end{cases} \tag{18}$$

Only a few data from the input is required for the computation of the disparity and there is no data dependency between two lines. All the processors can thus access the data to compute and store the result in a local memory.

2. Considered configuration

In this section, five different algorithms are selected and exposed. In all configurations, the census cost is used. This choice has been made because it offers a good trade-off between time execution and quality output [4]. Moreover this trade-off can be changed by tuning the size of the considered windows from 3x3 to 7x7. Above 7x7 the memory consumption is too high and reduces significantly the execution time mainly because of the memory wall.

All different configurations are noted C1, C2, C3, C4 and C5 in this paper. The five next sections expose those different configurations.

2.1. C1 : Census + BFA + WTA

This configuration uses the most simple disparity selection. Because the census provides noisy cost maps, they need to be smoothed with the BFA cost aggregation step.

The BFA is the most computational block of this configuration.

2.2. C2 : Census + SGM

The C2 configuration uses a smart disparity selection, the SGM. The SGM algorithm resolves hidden Markov chain based on belief propagation algorithm and is thus robust to the noisy cost map provided by the census cost construction. The cost aggregation step may therefore be skipped. The main drawback is that SGM is the most complex disparity selection block of this paper.

2.3. C3 : Census + BP-1D

The BP-1D disparity selection uses a belief propagation to resolve a HMM like SGM but in one dimension. Thus its robustness is good and allows it to be used on a noisy census cost construction without any cost aggregation. The BP-1D implementation is very fast thanks to its cyclopean view that makes memory accesses cache friendly.

2.4. C4 : Census + BFA + BP-1D

In this configuration, the census cost construction is refined with BFA and the disparity is selected with BP-1D. Thus BP-1D has a better disparity map as input than in the previous case. This may compensate the fact that BP-1D considers only one direction.

2.5. C5 : Census + BFA + SGM

This configuration should provide the best quality by using a BFA cost aggregation with the SGM disparity selection. The fact that the census is noisy is corrected by the BFA cost aggregation. As a consequence, the SGM disparity selection has as input better quality cost maps.

2.6. Summary

To sum up this section, there are three types of configurations that are presented :

- The configuration that has a robust cost aggregation and a simple disparity selection: C1.
- Configurations that have a robust disparity selection and no cost aggregation: C2 and C3.
- Configurations that cumulate both a robust cost aggregation and a robust disparity selection : C4 and C5.

Table 1: Parameters that do not affect the latency.

Algorithm	Parameters
C1	Threshold of similarity : thr in (5) Limiting distance factor : $Dmax$ in (4) Distance coefficient : C_d in (5)
C2	Small discontinuity penalty : $P1$ in (8) Large discontinuity penalty : $P2$ in (8)
C3	BP-1D penalty : φ_0 in (16)
C4	Threshold of similarity : thr in (5) Limiting distance factor : $Dmax$ in (4) Distance coefficient : C_d in (5) BP-1D penalty : φ_0 in (16)
C5	Threshold of similarity : thr in (5) Limiting distance factor : $Dmax$ in (4) Distance coefficient : C_d in (5) Small discontinuity penalty : $P1$ in (8) Large discontinuity penalty : $P2$ in (8)

Table 2: Parameters that affect the latency.

Algorithm	Parameters
C1	Census size (3x3 to 7x7) Number if iterations (2 to 7)
C2	Census size (3x3 to 7x7) Number of paths (2 to 16)
C3	Census size (3x3 to 7x7)
C4	Census size (3x3 to 7x7) Number if iterations (2 to 7)
C5	Census size (3x3 to 7x7) Number if iterations (2 to 7) Number of paths (2 to 16)

To conclude this section, the parameters required in the considered stereo matching processes and affecting only output quality are exposed in Table 1. They will be set with the methods described in section 3. In Table 2 all the parameters that set a trade-off between quality and execution time are exposed. For those latter parameters all possible solutions will be explored in order to keep the best possible trade off between the quality and the latency.

3. Parameter estimation

For the five configurations exposed in the previous section, the optimal value of all parameters has been found thanks to the method proposed in this section.

An objective criterion is required in order to minimize the output error, that is to say maximize the output quality, without supervision. A well known criterion in literature is the number of bad pixels compared to the ground truth. The values of this criterion is given in Equation (19). It is used in a tool proposed by Middlebury university [5]. This tool takes as input the disparity map to test, a ground truth disparity map and provides metrics. This tool expects the left image to be the reference image. Middlebury university also provides a set of stereoscopic images with the associate ground truth.

$$Error = \frac{1}{N} \sum_{p \in (x,y)} (|D_T(p) - D_E(p)| > \delta) \tag{19}$$

where D_t is the ground truth, D_E the evaluated disparity map, N the number of pixels, and δ a threshold fixed to 1.0 [5].

First, the method to set a parameter that affects only the output quality will be presented, then the overall process of optimization that considers all parameters will be introduced.

3.1. Trichotomic approach

For all the parameters that do not affect execution time, their output quality is a convex function of their value. The goal is to find the minimum of this function. To do so, an iterative approach is used.

To start the minimization with respect to a parameter, a starting interval $[a_0, b_0]$ is set. At each iterative step n, the search interval $[a_n, b_n]$ is reduced. The search is trichotomic, *ie* the interval is cut in three parts at each iteration. The two cutting points c_n and d_n are given by Equation (20a) and Equation (20b). The new interval $[a_n, b_n]$ at step n is therefore defined by Equation (20c) and Equation (20d). $Q(p)$ is the quality obtained for a parameter value p.

$$c_n = a_{n-1} + \text{ceil}(\frac{b_{n-1} - a_{n-1}}{3}) \tag{20a}$$

$$d_n = b_{n-1} - \text{ceil}(\frac{b_{n-1} - a_{n-1}}{3}) \tag{20b}$$

$$a_n = \begin{cases} a_{n-1} & \text{when } Q(c_n) < Q(d_n) \\ c_n & \text{otherwise} \end{cases} \tag{20c}$$

$$b_n = \begin{cases} d_n & \text{when } Q(c_n) < Q(d_n) \\ b_{n-1} & \text{otherwise} \end{cases} \tag{20d}$$

The trichotomic search is stopped when the condition in Equation (21) is reached, that is to say when there are two or less points inside the considered interval.

$$b_n - a_n \leq 2 \tag{21}$$

The Figure 7 is an application of the trichotomic minimum finding. The considered parameter is the parameter $P1$ of SGM algorithm. In this example, several d_n and c_n values are used several times and do not need to be recomputed. For instance points 4 and 6 are used at least two times, and their value is therefore computed only once.

Some parameters such as the one studied in Figure 8 do not strictly evolve convexly. The minimization by trichotomy then provides a local minimum, that is not necessary the real minimum.

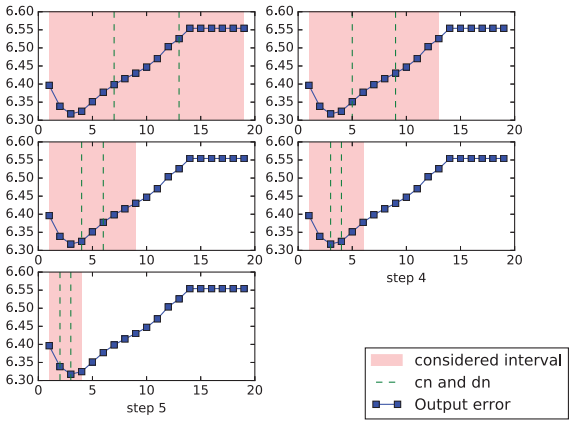

Figure 7: Example of trichotomic search on SGM $P1$ parameter.

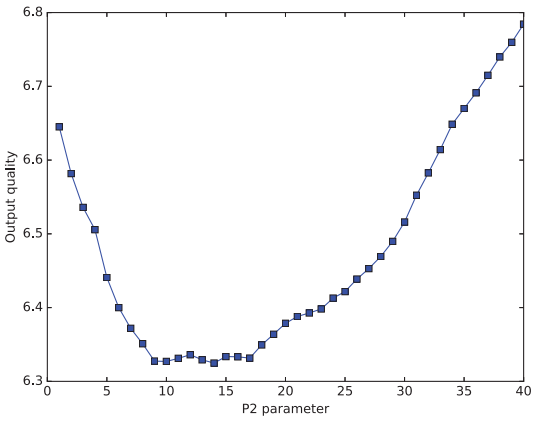

Figure 8: SGM $P2$ parameter variation.

In order to avoid a bad local minimum, an exhaustive search is used on an interval of fixed size around the minimum found with the trichotomic approach. Most of the time the quality for parameters around the point are already computed, thus the cost for an exhaustive search in a small interval is low. The size of the interval is set according to Table 4. While a minimum different from the previous one is found, a new exhaustive search is done around the new minimum.

The next part exposes how each parameter is estimated in order to minimize the error for the overall parameter set.

3.2. Overall minimization process

In order to minimize the overall set of parameters, all possibilities of parameters that affect both the latency and the quality have been tested. For each of those possibilities, the optimal value of parameters that affects the quality without affecting the execution time has been found thanks to the method proposed in the previous subsection. Each of those parameters is estimated while the other are fixed to their best known value such as it minimizes the output error. This approach reduces drastically the complexity compared to a global minimization method.

Table 3: Training images set

Image Name	Resolution	Disparity range
Teddy	450 x 375	59
Cones	470 x 375	59
Sawtooth	434 x 380	19
Tsukuba	384 x 288	15
Venus	434 x 383	19
Map	284 x 216	29

The order in which each parameter is minimized has an significant impact on the final results. For instance, changing one parameter of the first step of stereo matching algorithm, the cost construction, impacts its output which is the input of the next steps, the cost aggregation and the disparity selection. Then, changing a parameter in the first step, will impact the next steps, whereas changing a parameter in the last steps will not modify the output of the first one. This is why parameters are estimated according their order in the pipeline as shown in Figure 9.

Minimizing all parameters at once in the correct order is called a pass. When the parameter has strict ordering, for instance when there is only one parameter per block (construction, aggregation, selection) one pass is enough. When there are interdependent parameters (like $P1$ and $P2$) several passes are required. Experimentally it can be observed that two passes are enough. The third one is used to test the convergence criteria, that is to say, the two last passes converge to the same parameters set.

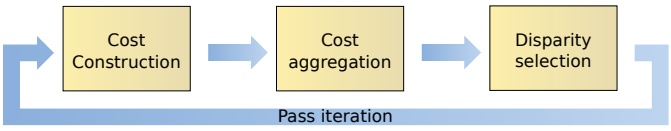

Figure 9: Parameters minimisation order.

3.3. Setup

The minimized criterion is the number of bad pixels given by Equation (19) averaged with six different images from Middlebury database given in Table 3. Using six images avoids to over specialized the set of parameters of the considered stereo matching algorithm to particularities of one image. The training set is therefore chosen such as the characteristics are various (grey level *vs* colour, disparity range, resolution). For example, *Map* is a grey level image, *Sawtooth* and *Venus* contains very simple plans. *Cones* and *Teddy* are complex stereo images with repetitive structures.

The exact configuration employed for those parameters is exposed in Table 4. Table 4 expresses for each parameter, the range in which parameter estimation is done, the first value used for this parameter when the estimation is started, and the size of the interval for the final exhaustive

Table 4: Parameter details

Parameter Name	range	starting point	Exhaustive search
Threshold of Similarity (BFA)	$[1; 128]$	20	3
Limiting distance factor (BFA)	$[2; NbIter^2]$	$NbIter^2 - 3$	$NbIter^2$
Distance coefficient (BFA)	$[1; 10]$	4	1
Small discontinuity penalty (SGM)	$[1; 75]$	10	2
Large discontinuity penalty (SGM)	$[1; 150]$	20	4
BP-1D penalty	$[1; 150]$	20	4

search. The distance factor for the BFA algorithm has an exhaustive search on the whole considered interval.

As exposed previously, the Middlebury tool needs a disparity map aligned on the left image. This is not a problem for most algorithms except BP-1D. Indeed, BP-1D uses a cyclopean view and thus does not use the left image as reference. BP-1D and SGM algorithms are almost equivalent when $P1 = P2 = \varphi_0$. Therefore the SGM algorithm is used instead of BP-1D by replacing $P1$ and $P2$ with φ_0 when comparing quality between all configurations using the Middlebury tool.

The next section exposes the results obtained by applying the minimizing method presented in this section. This method is applied to the five configurations presented in section 2 with all possible sets of parameters that affect both the quality and the execution time.

4. Results

This section exposes the results of the five considered configurations. As exposed in the previous section, the error criterion is computed with the Middlebury tool and is the average of bad pixels on the six images presented in Table 3. All execution time exposed in this section are for the *Sawtooth* image that has a resolution of 434 by 380 pixels and uses 19 disparity levels on a single core of the C6678 platform clocked at 1GHz.

The configuration C1 is composed of the census cost construct, a BFA cost aggregation and the WTA disparity selection. The Figure 10 exposes the results of configuration C1 for different census sizes and different BFA iterations. Each line corresponds to a census size, and each point is for a different number of iterations, from three iterations for the first point to eight iterations for the last point.

Figure 10 shows that increasing the number of iterations of the BFA algorithm above five does not increase significantly the output quality. For the C1 configuration the census 3x3 is not well suited for a good speed quality trade-off. If indeed the first point is the fastest, however with an error of 20.9 %, its output is almost not exploitable.

The configuration C2 uses a census cost construction with a SGM disparity selection. The Figure 11 exposes the quality and execution time of C2 for different census sizes and number of SGM paths. Each line corresponds to a census size, and each point is for a different number of SGM paths, two for the first point, four for the second point and eight for the last point. Increasing the census size above 5 in the configuration C2 does not increase its output quality.

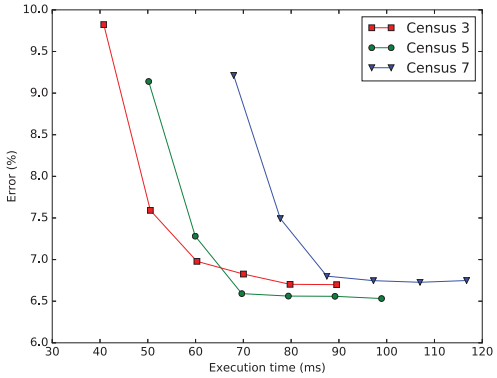

Figure 10: Speed and quality for C1

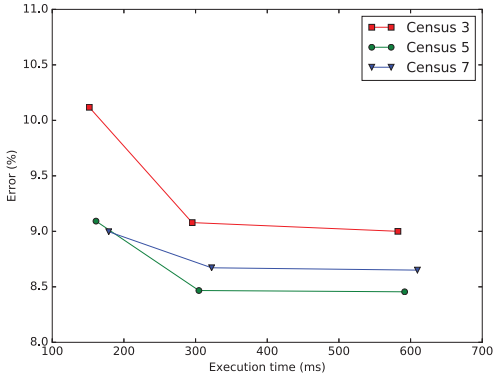

Figure 11: Speed and quality for C2

The C3 configuration has a census cost construction with a BP-1D cost construction, with only one parameter affecting both the quality and the latency, the census size. Thus, there are only three points (for three census sizes). The results are presented in Figure 12. As expected, the census 3x3 has the lowest quality, and the execution time for census 7x7 is only 20 ms more than the census 5x5, which may be unimportant when comparing to the order

of magnitude of execution time change due to the other parameters.

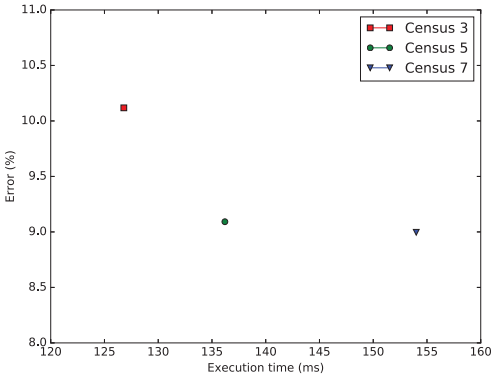

Figure 12: Speed and quality for C3

The configuration C4 is made of a census cost construction, BFA cost aggregation and BP-1D disparity selection. The Figure 13 exposes the quality and execution time of C4 for different census size and BFA aggregation. Each line corresponds to a census size, each point if for a different number of iterations, from three iterations for the first point to six iterations for the last point.

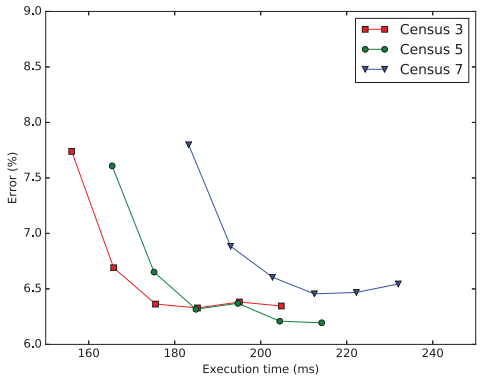

Figure 13: Speed and quality for C4

The configuration C5 is made of a census cost construction, BFA cost aggregation and BP-1D disparity selection. Figure 14 exposes the results for the configuration C5. There are three chunks of data that have very different execution time. Those chunks correspond to each possible number of paths in the SGM algorithm. The number of paths is directly annotated in the Figure. Dots with the same number of paths and the same window size are linked together, so each line represents the quality and execution time evolution according to the number of iterations of the BFA algorithm (from three to eight).

Figure 15 presents all possible variation in execution time and quality for each configuration. Each point of this figure is the execution time and quality of one configuration with a particular set of parameters that affect the execution time. All other parameters are optimized with the method explained above.

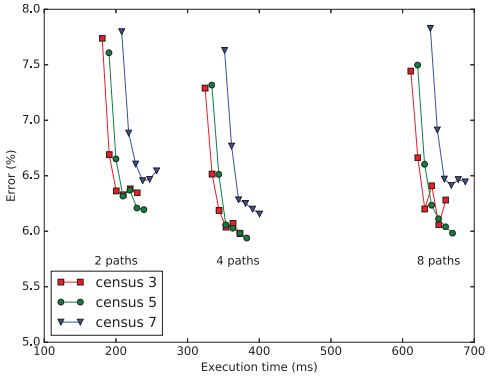

Figure 14: Speed and quality for C5

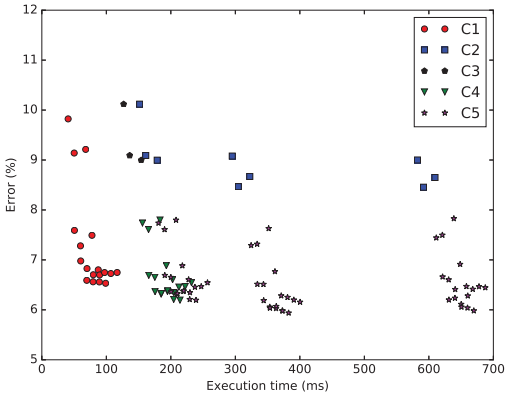

Figure 15: Speed and quality of different solutions

From all configurations, it can be observed that the census 3x3 has only an interest when the run time is crucial. Indeed increasing the census size to 5x5 is one of the best way to increase output quality. Moreover, the census 7x7 leads to a quality loss comparatively to census 5x5. So in most of case, the census 5x5 is one of the best compromise between quality and execution time.

Considering the SGM algorithm, increasing the number of paths is very computing expansive, however this is the only way to achieve the lowest error with the presented configuration.

Figure 16 presents the best solutions of Figure 15 for the *Sawtooth image*. The trade off between quality and latency evolves according to the Pareto principle.

Figure 17 exposes the same results as Figure 16 but instead of using the average percentage of bad pixels for the six images used in the training set, the criterion used is the percentage of bad pixels of a particular image. The quality measure is not the same, because the complexity of images is different. However, conclusions remain the same: the C1, C4 and C5 configurations offer the best trades-off

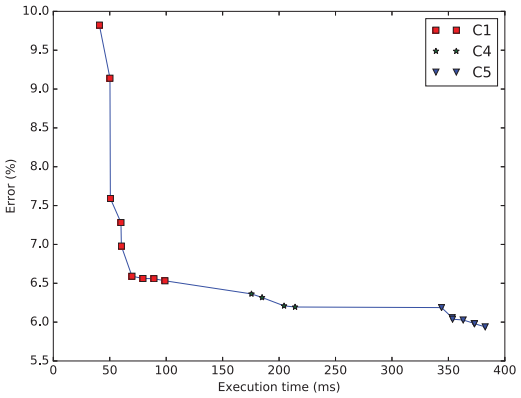

Figure 16: Latency and quality of the best solutions

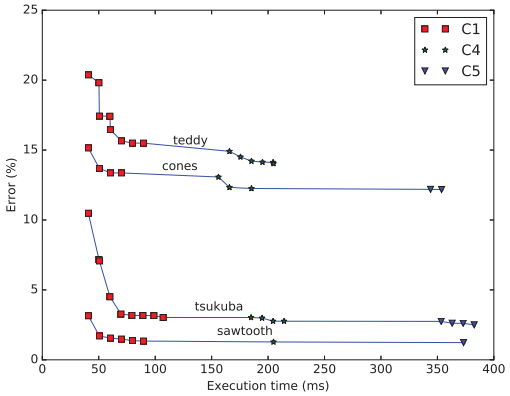

Figure 17: Best compromises for different images.

between latency and quality. This enforces the fact that changing the set of tested image will not change too much the final results.

Figure 18 shows the output quality of the *Aloe* image. This image was not in the learning database, and thus validates the fact that the stereo matching algorithm works for images not included in the database.

The Figure 19 exposes the different output disparity maps for the Teddy image of Middlebury database, with 60 disparity levels. Execution time are different since Figure 16 is related to the Sawtooth image of Middlebury database with a 434*380 resolution and 19 disparity levels. We can observe the same kind of results.

4.1. Discussion about latency and energy consumption

All latency results exposed in this section are for one core of the C6678 platform. The C1, C2 and C3 configurations have been parallelized in other works on the C6678 platform . Their latency speed-up is respectively 4.9, 8 and 7 when parallelized on eight cores [13]. Those results show that all the selected configurations scale well on the C6678 platform.

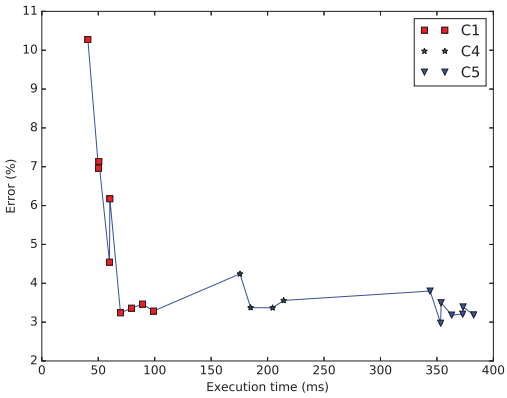

Figure 18: Output error for *Aloe* image that was not in the learning database.

Those results can be compared to a GPU platform. With an algorithm closed to the C1 configuration, but more suited for a GPU environment, it takes 50 ms to execute on a NVIDIA GT 540M[11] compared to the 11 ms on the C6678 platform when parallelized for a comparable output quality. Those performances have to be compared to the power consumption of each chip, the NVIDIA GT 540M consumes 35 Watts, and the C6678 consumes 10 Watts. That means, for the same application, the performance and power trade-off is respectively for the C6678 and the NVIDIA GT 540M of 9.1 FPS/Watt and 0.57 FPS/Watt.

Conclusion

Stereo Matching algorithm retrieves depth information from two stereoscopic images. Dense stereo matching algorithms are composed of three different steps : the cost construction, the cost aggregation and the disparity selection. Several algorithms for those three steps are available in literature. Those algorithms have parameters that must be set. Some of them affect the latency and the output quality, and the other ones affect only the output quality.

Every time a new combination of cost construction, cost aggregation and disparity selection is selected, all those parameters have to be optimized for this new configuration. Currently they are set empirically. This paper exposes an automatized non-supervised method to set the parameters that affect only the quality and not the latency. When parameters affect the latency and the quality, all the possibilities are explored and only the best quality versus latency trade-off is kept.

This paper also studies the latency versus quality trade-off obtained with several algorithms onto the C6678 Digital Signal Processor (DSP) platform. State-of-the-art stereo-matching algorithms have been introduced and combined to get five different configurations. They provide the best trade-off between latency versus quality for different use-cases. For all those configurations, parameters have been

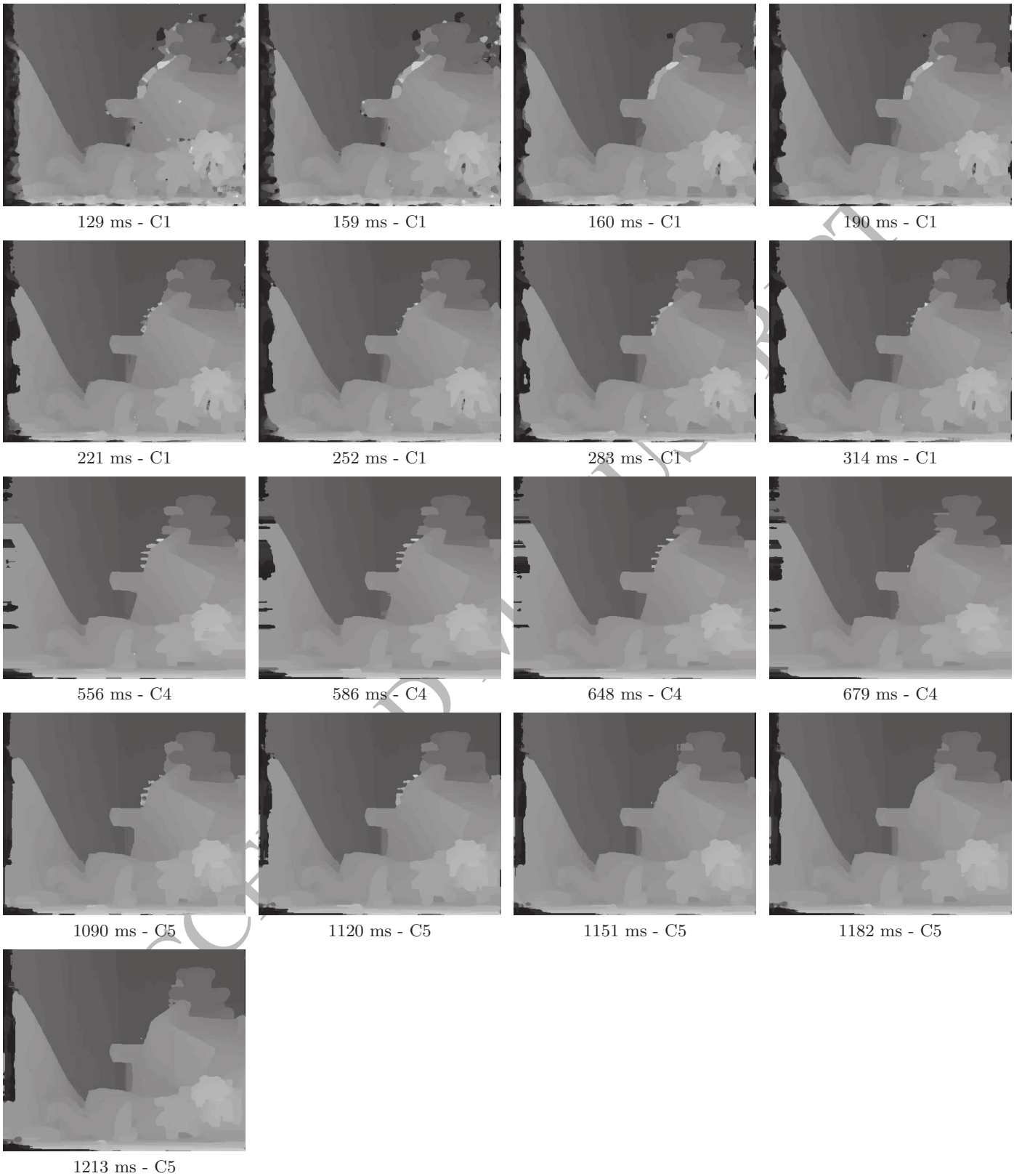

Figure 19: Disparity maps best trade-off for the Teddy image of Middlebury database, with 60 disparity levels.

set with a method proposed in section 3. This method uses a recursive approach to find the best point that minimizes the output error to avoid exhaustive search. This error is computed with a tool provided by Middlebury University [5].

This automatised setup enables to compare the stereomatching algorithms. The results exposed in section 4 shows that the most interesting configurations are C1, C4 and C5. Indeed, using a more sophisticated disparity selection adds more latency than adding a better cost aggregation.

All the trades-off between output quality and latency exposed in this paper are for the C6678 platforms. However those results can be easily extrapolated to any other DSP based platform and even most of Central Processor Unit (CPU) based platforms.

Future works will be dedicated to extend this contribution to many-core architectures. As it implies some very specific optimisations, the use and adaptation of prototyping tools as Preesm [14] to automatize the comparisons is very promising.

References

[1] E. V. Alliance, What is embedded vision., http://www.embedded-vision.com/what-is-embedded-vision.

[2] K. Khoshelham, S. O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors 12 (2) (2012) 1437–1454. doi:10.3390/s120201437.

[3] M. Kytö, M. Nuutinen, P. Oittinen, Method for measuring stereo camera depth accuracy based on stereoscopic vision, Vol. 7864, 2011, pp. 78640I–78640I–9. doi:10.1117/12.872015.

[4] J. Menant, G. Gautier, J.-F. Nezan, M. Pressigout, L. Morin, A comparison of cost construction methods onto a c6678 platform for stereo matching., in: conference on Design and Architecture for Signal and Image Processing, Rennes, 2016.

[5] R. S. Daniel Scharstein, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, International Journal of Computer Vision (47) (2002) 7–42.

[6] J. Menant, M. Pressigout, L. Morin, J.-F. Nezan, Optimized fixed point implementation of a local stereo matching algorithm onto C66x DSP, in: DASIP 2014, Madrid, Spain, 2014. doi:10.1109/DASIP.2014.7115636.
URL https://hal.archives-ouvertes.fr/hal-01101822

[7] A. Mercat, J.-F. Nezan, D. Menard, J. Zhang, Implementation of a stereo matching algorithm onto a manycore embedded system (2014) 1296–1299doi:10.1109/ISCAS.2014.6865380.

[8] H. Hirschmuller, D. Scharstein, Evaluation of stereo matching costs on images with radiometric differences, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (9) (2009) 1582–1599. doi:10.1109/TPAMI.2008.221.

[9] X. Mei, X. Sun, M. Zhou, On building an accurate stereo matching system on graphics hardware, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, 2011, pp. 467 –474.

[10] J. Zhang, J.-F. Nezan, M. Pelcat, J.-G. Cousin, Real-time gpu-based local stereo matching method, in: conference on Design and Architecture for Signal and Image Processing, Cagliary, 2013.

[11] J. Zhang, Prototyping methodology of image processing applications on heterogeneous parallel systems, Theses, INSA de Rennes (Dec. 2014).

[12] I. Ernst, H. Hirschmüller, Mutual information based semi-global stereo matching on the gpu, in: Proceedings of the 4th International Symposium on Advances in Visual Computing, ISVC 08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 228–239.

[13] J. Menant, M. Pressigout, L. Morin, J.-F. Nezan, A comparison of stereo matching algorithms on multi-core Digital Signal Processor platform, in: 3DIPM 2017, Burlingame, California USA, 2017.

[14] M. Pelcat, K. Desnos, J. Heulot, Welcome to preesm, http://preesm.sourceforge.net/website/ (2014).

Judicael Menant received the engineer degree in Electrical and Computer Engineering from the National Institute of Applied Sciences (INSA) in Rennes in 2012. He then worked for Renesas, a company that designs System On Chip for smart phones. Judicael Menant started his PhD at Institute of Electronics and Telecommunications of Rennes (IETR) on January 2014. His work focuses on Stereo Matching algorithm onto embedded systems.

Guillaume Gautier will graduate from the National Institute of Applied Sciences (INSA) in Rennes and the University of Strathclyde in Glasgow with an engineer degree in Electrical and Computer Engineering in 2017. He will start his PhD at Institute of Electronics and Telecommunications of Rennes (IETR) on October 2017.

Muriel Pressigout received the Ph.D. degree in 2006 with Prof. E. Marchand, from the University of Rennes 1, IRISA laboratory, Rennes, France. She is an assistant professor at INSA Rennes in the Department of Electrical and Computer Engineering since 2007. Her research activities are carried out in the VAADER team of IETR. Her current interests focus on computer vision for various applications, mainly 3D video and augmented reality. Muriel Pressigout has been involved in a French ANR project (PERSEE), R&D FUI project (RUBI3, APASH). She is currently involved in a R&D FUI project (PRISME) and a EU project (ADAPT).

Luce Morin is Full-Professor at the Department of Electrical and Computer Engineering (ECE) at the National Institute of Applied Sciences (INSA), Rennes, France. She leads the VAADER team in the IETR laboratory. She received the Ph.D degree from INPG, Grenoble, France in 1993. Her research activities deal with computer vision, 3-D reconstruction, image and video compression, representation and compression of multi-view videos. Luce Morin has published more than 70 scientific papers in international journals and conferences. She has been involved in different French ANR projects (V2NET, PERSEE), R&D projects (TELEGEO, FUTURIM@GE), R&D FUI project (RUBI3), EU project (NOE-SIMILAR).

Jean-François NEZAN is a Professor at the Department of Electrical and Computer Engineering at the National Institute of Applied Sciences (INSA) and the Institute of Electronics and Telecommunications of Rennes (IETR). He is coauthor or coeditor of more than 75 technical articles including 1 Book, 1 Book chapter, 16 publications in International Journals. He is involved in the French research society "GDR ISIS" and the European Network of Excellence HiPEAC. His research topic is the rapid prototyping of standard video compression on embedded architectures including signal processing systems, architectures, and software; hardware/software co-design; and fast prototyping tools.