Journal of Real-Time Image Processing (2022) 19:233–245
https://doi.org/10.1007/s11554-021-01180-1

ORIGINAL RESEARCH PAPER

Dynamic programming with adaptive and self-adjusting penalty for real-time accurate stereo matching

Mohamed Hallek[1] · **Hamdi Boukamcha**[1] · **Abdellatif Mtibaa**[2] · **Mohamed Atri**[3]

Received: 21 March 2021 / Accepted: 26 October 2021 / Published online: 8 November 2021

Abstract
Dense disparity map extraction is one of the most active research areas in computer vision. It tries to recover three-dimensional information from a stereo image pair. A large variety of algorithms has been developed to solve stereo matching problems. This paper proposes a new stereo matching algorithm, capable of generating the disparity map in real-time and with high accuracy. A novel stereo matching approach is based on per-pixel difference adjustment for the absolute differences, gradient matching and rank transform. The selected cost metrics are aggregated using guided filter. The disparity calculation is performed using dynamic programming with self-adjusting and adaptive penalties to improve disparity map accuracy. Our approach exploits mean-shift image segmentation and refinement technique to reach higher accuracy. In addition, a parallel high-performance graphics hardware based on Compute Unified Device Architecture is used to implement this method. Our algorithm runs at 36 frames per second on 640×480 video with 64 disparity levels. Over 707 million disparity evaluations per second (MDE/s) are achieved in our current implementation. In terms of accuracy and runtime, our algorithm ranks the third place on Middlebury stereo benchmark in quarter resolution up to the submitting.

Keywords Absolute difference · Rank transform · Guided filter · Dynamic programming · Penalty parameter · CUDA

1 Introduction

In recent years, various evolutions have been made in the field of image processing and computer vision. Stereo is one of the most popular topics in computer vision. It is fundamental for many applications such as video surveillance, robotic surgery, obstacle detection and autonomous vehicles. The identification of the correct matching pixels between two rectified images and the disparity map extraction are the principal purposes of the stereo matching step. Although, the development of stereo matching algorithm requires the understanding of some problems such as occlusion, noise and repetitive texture. A wide variety of algorithms have been developed to solve these problems using global optimization approaches, Patch-based image synthesis methods and convolutional neural network [1–3].

This paper focuses on the stereo matching process and proposes an accurate cost matching function. Similar to the combined cost measurements developed in Refs. [4–6], the proposed cost function is based on the Absolute Difference (AD), the Gradient Matching (GM) and the Rank Transform (RT). At first, we introduce the means-shift and the stereo images are initially segmented. The resulting cost volume is aggregated using Guided Filter (GF). Thanks to its edge preservation property and its fast implementation, GF allow us to generate a smooth disparity map, as discussed in Refs. [7, 8].

In addition, this work proposes an adaptive version of Dynamic Programming (DP) for disparity calculation and optimization. To further improve the accuracy, this paper presents a DP with self-adjusting and adaptive penalty values instead of single penalty implemented in Refs. [9–11]. Finally, the stereo process is finished by a refinement technique in the goal to remove noise and correct outliers.

In the recent years, the real-time has become a reality through the complexity reduction of the calculation and the use of graphics hardware. In our work, we present a

✉ Mohamed Hallek
hallekmohamed@yahoo.fr

1 Faculty of Sciences of Monastir, Monastir, Tunisia

2 National Engineering School of Monastir, Monastir, Tunisia

3 College of Computer Science, King Khalid University, Abha, Saudi Arabia

GPU (Graphics Processing Units) implementation based on CUDA (Compute Unified Device Architecture) language to implement the suggested method. In this paper, a fast and accurate stereo matching algorithm is proposed and the most significant contributions are as follows:

- The proposed method acquires a final cost volume using three types of cost metrics: AD, GM and RT. In addition, this algorithm exploits mean-shift segmentation in the cost calculation step. All these functions are employed to obtain accurate results.
- This approach proposes a dynamic programming with self adjusting and adaptive penalties. This method integrates the inter-scanline and intra-scanline consistency constraints to increase the matching accuracy and outperforms other optimization approaches.
- This work exploits the graphics hardware to achieve real-time performance. The proposed algorithm is evaluated on Middlebury stereo benchmark. It ranks among the fast and accurate methods in Middlebury evaluation table.

This paper is organized as follows: After reviewing the related work in Sect. 2, our stereo matching method is detailed in Sect. 3. While the GPU CUDA implementation of our algorithm is presented in Sect. 4. In Sect. 5, we evaluate our approach on Middlebury dataset and we report experimental results. Finally, some conclusions are given in Sect. 6.

2 Related works

According to the taxonomy [12], stereo algorithms generally perform four basic steps: cost calculation, cost aggregation, disparity calculation and disparity refinement. The stereo studies have developed many approaches to realize each stage. The most common matching costs include absolute difference or block matching calculated over fixed or adaptive windows [1, 2, 5, 13]. The failure on discontinuities areas and sensitivity to the lighting variation are the main limitation of these techniques. The image gradient magnitude is also taken to calculate the similarity between the stereo pair [1]. A simple comparison of light intensities over windows is not always enough. Therefore, the use of non-parametric transformations as Census Transform (CT) and RT is desired. According to the works [1, 14, 15], CT and RT perform comparably to correlation and difference metrics and they are more robust against the lighting variations. Nevertheless, the robustness of these transforms is kept with a relatively large kernel size.

All of these matching costs have strengths and weaknesses. So, some studies have been devoted to combining the strengths of multiple methods to achieve better performance. Mei et al. [16] combined AD with CT for computing the initial matching cost. While the cost volume in Ref. [7] is obtained by the combination of AD and GM. In the work of Hamzah et al. [6], the global cost matching is based on the combination of three cost functions: AD, GM and CT. While Kordelas et al. [17] use RGB information, CT and Scale-Invariant Feature Transform in cost calculation step. These well-combined cost measurements achieve their goals and perform better than the single one. We adopt this combined cost strategy in this work. To obtain our global cost matching, we combine AD, GM and RT. In addition, the means-shift segmentation is used in cost calculation step to reduce the effect of lighting variations as implemented in Ref. [17].

Moreover, numerous techniques for cost aggregation step are used. Most aggregation functions can be classified into window-based method [9, 18], segment-tree-based method [19] and filter-based method [7, 10, 20]. A summation of the matching cost is calculated for each pixel over the support region to remove the possible influence of noise. Yoon and Kweon [13] use an adaptive weights between the neighboring pixel and the center pixel in a fixed window. But this method is prone to difficulties within textureless regions. Edge-preserving filters as Bilateral Filter (BF) and GF are utilized in this step. BF gives acceptable results but it is computationally expensive as implemented in Refs. [10, 21]. To solve the computational limitation of BF, Rhemann et al. [7] adopted the GF into cost aggregation. An adaptive GF is presented by Zhu et al. [20]. Also, an iterative GF is implemented in Ref. [6]. Recently, Wu et al. [22] propose fusing adaptive support weights, where the cost is aggregated using segment-tree-based method and GF. But this method is identified by its computational complexity. As realized in many real-time algorithms [8, 23], we exploit GF to aggregate our global cost matching.

The disparity calculation for local methods is performed using Winner Take-All (WTA) algorithm. Besides from local methods, efficient global stereo methods are also studied. Among the optimization approach, the dynamic programming is employed in many real-time stereo works. In the works of Wang et al. [10], GPU implementation of BF and DP achieves 192 fps with 340×240 input images and 16 disparity levels. While, Hallek et al. [9] present a stereo algorithm based on Fourier descriptor and DP whose implementation achieves about 93 MDE/s.

Traditional DP optimizes the matching cost on a scanline by scanline without enforcing inter-line consistency constraint. To address this limitation, Veksler [24] introduces the tree structure and optimize the global cost function defined on a 2D. An important parameter is usually used for DP named smoothness penalty. This parameter is assigned empirically and defines the penalty of the change in the disparity value between neighboring pixels [9, 11]. In similar works based on Semi Global Matching (SGM), some studies evaluate the performance of the penalty functions as realized in Ref. [25]. Recently, Karkalou et al. [26] present SGM with a Self-Adjusting Penalties (SGM-SAP) witch outperforms the classic SGM version [27]. In the goal to increase the matching accuracy, an improved DP is developed. As implemented in SGM-SAP work, we present a generic algorithm based on self-adjusting penalty parameter for DP.

The runtime is considered as an important metric for evaluating the stereo algorithms. Many works are interested in runtime reduction. To reach real-time performance, the Field Programmable Gate Array (FPGA) [28, 29] or the GPU implementation [9, 18, 30] are used. While many real-time approaches focus on reducing the complexity associated with matching and aggregation costs, at the expense of decreased matching accuracy. In this paper, we use NVIDIA GeForce RTX 2070 and we exploit the RTX 2070s computing capabilities to obtain a highly accurate results that maintains real-time performance.

3 Matching algorithm

Similar to a basic stereo matching development, the proposed approach involves the same steps in the taxonomy [12]. Figure 1 shows the block diagram of the proposed algorithm and the main steps are detailed in the following sections.

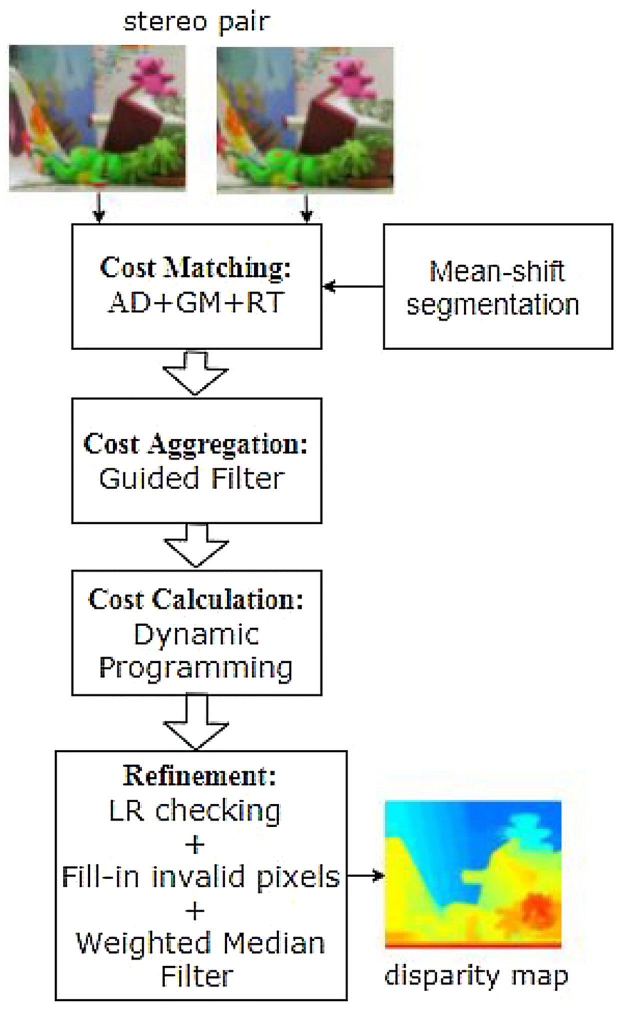
stereo pair
Cost Matching: AD+GM+RT
Mean-shift segmentation
Cost Aggregation: Guided Filter
Cost Calculation: Dynamic Programming
Refinement: LR checking + Fill-in invalid pixels + Weighted Median Filter
disparity map

Fig. 1 Diagram block of our approach

3.1 Mean-shift segmentation

The first step in the proposed method is to decompose the stereo pair image into regions of homogeneous color or gray-scale. The mean-shift segmentation is one of popular methods applied to image segmentation. It is exploited by numerous stereo matching algorithms [17, 31] in various ways, since it assists to acquire disparity map with a high accuracy. In this work, left and right images are initially segmented into non-overlapping regions using a state of the art mean-shift segmentation, which relies on color and edge information. More details about means-shift segmentation are found in Ref. [32]. The main advantage of the mean-shift segmentation is based on the fact that edge information is incorporated as well. In segmented image, pixels that belong to the same mean-shift segment have an individual label and their mean color value is computed. Figure 2 shows the left image of "Teddy" stereo pair and its mean-sift segmentation map.

The segmented maps of the left and right images are calculated once and then they are used in the following steps.

3.2 Cost matching function

The new proposed cost function contains three terms. It is the combination of three per-pixel difference measurements with adjustment element: Absolute Difference (AD) for color, Gradient Matching (GM) and Rank Transform (RT). Each term will be more detailed below:

The absolute difference is detailed and implemented in Ref. [5]. This cost function has taken into account the difference intensity for three channels. It is characterized by its rapidity and its simplicity during the implementation. The limitation of this cost function matching is the high distortions in low textured regions. The work presented in this paper proposes a method which is able to minimize

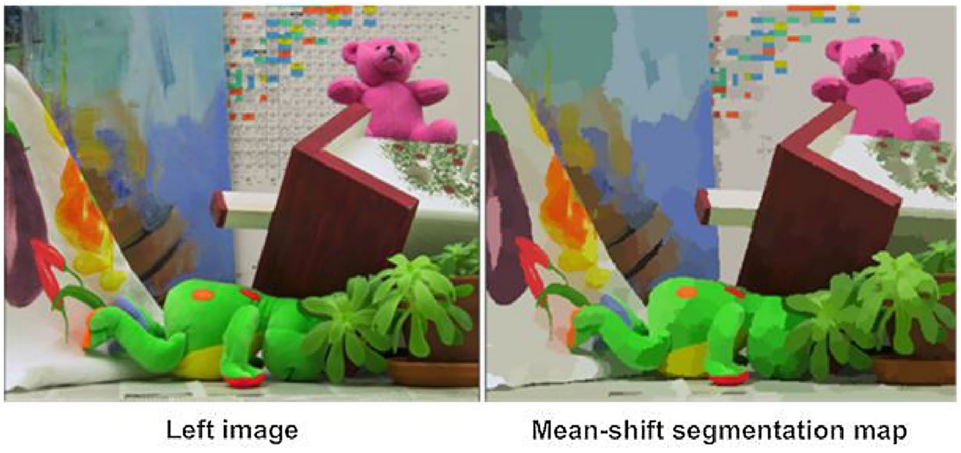

Fig. 2 Mean-shift segmentation "Teddy" image

these errors. The new proposed cost function introduces a coefficient α to balance the absolute difference and gradient matching cost. The color absolute difference is noted $AD(p, d)$ in Eq. (1). It presents the intensity difference between two color pixels at left and right images denoted I_l and I_r, respectively.

$$AD(p,d) = \frac{1}{3}\left(\left|I_l(p)^r - I_r^r(p-d)\right| + \left|I_l^g(p) - I_r^g(p-d)\right| + \left|I_l^b(p) - I_r^b(p-d)\right|\right) \quad (1)$$

Knowing that p is color pixel at location (x, y) in the reference image and d presents the disparity value. While r, g and b are the three color channels. The color absolute difference is thresholded using τ_{AD}. The threshold τ_{AD} defines the truncated value as used by Refs. [5, 6] to increase the robustness against the illumination changes. In Eq. (2), we give the condition for the use of AD:

$$AD(p,d) = \begin{cases} \tau_{AD} & \text{if } AD(p,d) \geq \tau_{AD} \\ AD(p,d) & \text{otherwise} \end{cases} \quad (2)$$

Meanwhile, to determine the gradient matching cost the components of the gradient from each image are initially computed. The gradient value in horizontal direction G_x and vertical direction G_y are calculated. Fundamentally, G_x and G_y are given by Eqs. (3) and (4), where $A = [1\ 0\ -1]$ and A^T is the transpose of A.

$$G_x = A * I \quad (3)$$

$$G_y = A^T * I \quad (4)$$

The image I is either the left or right stereo image, while $*$ defines the convolution operation. The gradient magnitude G is given by Eq. (5):

$$G = \sqrt{G_x^2 + G_y^2} \quad (5)$$

The modulus of gradient operator in Eq. (5) is applied for left and right images to give respectively G_l and G_r. The absolute difference between gradient images for a pixel p is denoted $GM(p, d)$, where d is the disparity level. The gradient difference is given by Eq. (6):

$$GM(p,d) = \left|G_l(p) - G_r(p-d)\right| \quad (6)$$

The final gradient cost is given by Eq. (7), where τ_{GM} is the threshold or truncated value of GM.

$$GM(p,d) = \begin{cases} \tau_{GM} & \text{if } GM(p,d) \geq \tau_{GM} \\ GM(p,d) & \text{otherwise} \end{cases} \quad (7)$$

Now, the per-pixel difference adjustment is applied for color and gradient distances. The costs $AD(p, d)$ and $GM(p, d)$ are combined to obtain $\text{cost}_{cg}(p, d)$. This function is presented by Eq. (8). The term α is used to balance color and gradient measures as realized in Refs. [5, 6].

$$\text{cost}_{cg}(p,d) = \alpha AD(p,d) + (1-\alpha)GM(p,d) \quad (8)$$

In this paper, we try to make our cost matching function more robust against illumination variations. A non-parametric local transform is added to the initial cost function $\text{cost}_{cg}(p, d)$. This work adopts the rank transform, one of the old functions used on stereo matching. It is introduced by Ref. [15] and given by Eq. (9), where W represents square window.

$$RT(p) = \sum_{p' \in W} \zeta\left(I(p), I(p')\right) \quad (9)$$

The intensity value of the central pixel is noted by $I(p)$. While $I(p')$ is the intensity of the neighboring pixel and ζ represents a comparison operator. The rank transform produces the sum of comparison between intensity of central pixel and the intensity of neighboring within a small window. This comparison gives 1 if $I(p) > I(p')$ and 0 otherwise. In stereo matching, the rank transform is calculated for left and right images to get RT_l and RT_r. The absolute difference between RT_l and RT_r is computed to obtain the rank distance as presented in Eq. (10). The final cost based on rank information $\text{cost}_{rt}(p,d)$ is given by Eq. (11), where τ_{RT} is the threshold value of $cost_{rt}$.

$$\text{cost}_{rt}(p,d) = \left|RT_l(p) - RT_r(p-d)\right| \tag{10}$$

$$\text{cost}_{rt}(p,d) = \begin{cases} \tau_{RT} & \text{if } \text{cost}_{rt}(p,d) \geq \tau_{RT} \\ \text{cost}_{rt}(p,d) & \text{otherwise} \end{cases} \tag{11}$$

For integrated similarity matching cost, we apply a robust cost function including three measurements. The exponential function has the advantage of mapping the values of a measure in the range of [0, 1] and the final matching cost function is given by Eq. (12).

$$C(p,d) = 2 - \exp(-\text{cost}_{cg}(p,d)) - \exp(-\text{cost}_{rt}(p,d)) \tag{12}$$

The used measurements with different ranges are initially scaled into the same range and then they are combined. The global cost matching $C(p, d)$ uses normalized cost function as realized in Refs. [6, 17].

3.3 Cost aggregation

The cost aggregation is a basic step for stereo matching problem. This step allows us to minimize the matching uncertainties and reduce noise. In this paper, we employ guided filter to aggregate the initial cost volume $C(p, d)$ for many reasons:

- GF has a linear execution time which is independent of the filter size and basically depends on the image size.
- GF has an edge preservation property and its application for cost volume gives a good quality results.
- Based on box filter a real-time implementation of GF can be achieved.

As its name indicates, the guided filter used guidance image. The initial left or right images are used as reference or guidance image during filtering process. The kernel function of GF for the image I is defined by Eq. (13), where w indicates to the pixel number in the support window w_k. While k denotes the central pixel, q represents the set of the neighboring pixels in the used window. Knowing that ϵ is a smoothness parameter, μ and σ define respectively the mean and variance of the intensity values on the reference image.

$$GF_{p,q}(I) = \frac{1}{\left|W^2\right|} \sum_{p \in w_k} \left(1 + \frac{(I_p - \mu_k)(I_q - \mu_k)}{\sigma_k^2 + \epsilon}\right) \tag{13}$$

The result of filtering process gives us the aggregated cost function $CA(p, d)$ as presented in Eq. (14):

$$CA(p,d) = GF_{p,q}(I)M(p,d) \tag{14}$$

The image $M(p, d)$ is the input filtering image. While $GF_{p,q}(I)$ is the weight of guided filter using left image as guidance image.

3.4 Disparity computation and optimization

For global methods, the disparity calculation is performed using energy optimization approaches. These methods are based on the smoothness assumption, meaning that the scene is locally smooth except for object boundaries and the neighboring pixels should have very similar disparities. Thus, the optimization approaches make explicit the smoothness assumptions and minimize the global matching function that contains data and smoothness terms as indicated in Eq. (15). The term $E_{\text{data}}(d)$ is the matching cost and E_{smooth} encodes the smooth assumption.

$$E(d) = E_{\text{data}}(d) + E_{\text{smooth}}(d) \tag{15}$$

The dynamic programming algorithm can efficiently handle this class of problems. It exploits the smoothness and the ordering constraints to optimize matching cost between two scanlines. DP is applied on the aggregated cost volume $CA(x, y, d)$, where x and y present the coordinates of the pixel p. DP contains two main steps: a step for constructing the cost matrix $M_h(x,d)$ for each line y and a step in which pairs of corresponding pixels are selected by searching the optimal path. The M_h dimensions are $W \times D_{\text{max}}$, where W and D_{max} represent the image width and the maximal disparity. After M_h calculation, the best path can be found by back-tracking. The optimal path assigns a disparity value for each pixel in the scanline y. To generate the full disparity map the DP process is repeated over all the scanlines. The calculation of M_h is given by Eq. (16), where λ presents the penalty of the change in the disparity value between neighboring pixels.

$$\begin{aligned} M_h(x,d) = CA(x,d) + \min(&\lambda + M_h(x-1,d-1), \\ &M_h(x-1,d), \lambda + M_h(x,d+1)) \end{aligned} \tag{16}$$

The penalty parameter λ is the only variable used during the application of DP. It has an important influence on the disparity map accuracy. A small value of λ can produce a noise and discontinuities regions on disparity map while a big value of λ can remove the object boundary and gives poor results in areas that contain discontinuities. Figure 3 shows

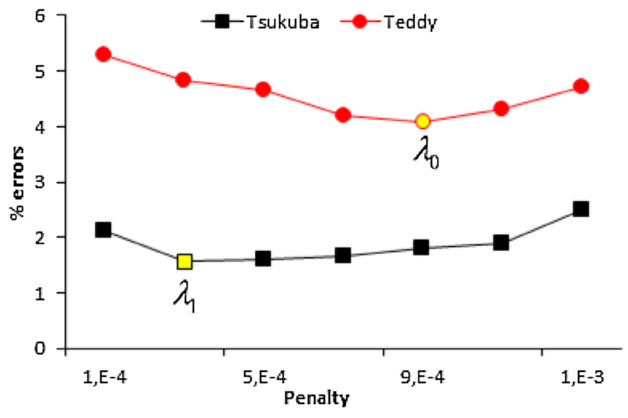

Fig. 3 The penalty effect on the disparity map accuracy

the effect of penalty value on results quality. The test is done based on two images of Middlebury dataset knowing that the costs are scaled in the range of [0,1]. From Fig. 3, one may notice that the disparity map accuracy depends the penalty value. Also, this figure indicates that minimum errors for the two images are obtained using different penalty values. The suitable penalties for "Teddy" and "Tsukuba" are λ_0 and λ_1 as marked in the curves. Therefore, the suitable penalty is not the same for all images in the same dataset.

DP-based algorithms use single and constant value of penalty. This parameter is empirically determined as realized in Refs. [11, 24, 33]. It is fixed once and then used for all images in the dataset. In a similar works, the algorithm of Ref. [25] evaluates the performance of different penalty functions. Besides, Karkalou et al. [26] define SGM algorithm with self adjusting penalties, where the penalty values are extracted from Disparity Space Image (DSI). On the other hand, the algorithm of DP on tree [24] indicates that the smoothness penalty for assigning disparities d_p and d_q to the pixels p and q is inversely proportional to the absolute value of the intensity difference $|I(p) - I(q)|$. Moreover, the penalty parameter is employed to penalize abrupt disparity changes when a depth discontinuity is unlikely. In the same stereo pair there is always a lighting variations and discontinuity areas according to image content. So, an adaptive penalty which varies from one line to another is required. The idea is to find a function which produces a vector of penalties for each stereo pair. Based on the works [24–26], the proposed penalty function is shown in Eq. (19). It gives a set of penalties by varying y from 1 to H. Where H is the number of lines in the image. In Eq. (17), we define $S_{\min}$ from the aggregated cost volume $CA(x, y, d)$, where 1 and $D_{\max}$ present respectively the minimum and the maximum disparity values. Equation (18) calculates the distance between a pixel of interest p and its neighbours q. I_d measures the discontinuity in the gradient image I.

$$S_{\min}(x, y) = \min_{d=1:D_{\max}} (CA(x, y, d)) \tag{17}$$

$$I_d(p) = |I(p) - I(q)| \tag{18}$$

$$\lambda_v(y) = \frac{(\sum_x S_{\min}(y, x))/W}{\max\limits_{x=1:W} (I_d(x, y))} \tag{19}$$

For each line, the average value of the columns in $S_{\min}$ is devised by the maximum value per all columns in the image I_d. W is the image width used to normalize the result. From one line to another the penalty varies smoothly according to the costs and the discontinuities of the processed line. During DP process, for each scanline y we calculate the matrix M_h in Eq. (16) using $\lambda_v(y)$ instead of constant λ. Then, the optimal path is determined to assign the disparity for each position in this scanline.

3.5 Disparity refinement

At this stage, the obtained disparity map contains noise and unmatched pixels due to repetitive texture and occlusion. Therefore, post treatment step is required to improve the results. This step concatenates several consecutive processes which are invalid disparity detection, fill-in invalid disparity value and Weighted Median Filter (WMF). The first process for disparity refinement is the detection of invalid disparity and unmatched pixels in the disparity map. This stage is performed using Left-Right (LR) consistency checking process as implemented in Refs. [5, 9, 17]. The main of this process is to detect problematic areas, especially outliers in occluded regions and depth discontinuities. The LR consistency checking process is done from left reference disparity map that coincides with the right disparity map. The set of inconsistent pixels between the two disparity maps are marked as invalid disparity. LR consistency checking process is calculated based on the Eq. (20). where D_{LR} represents the left reference disparity map. It is obtained using left image as a reference image and the matching process is performed from the left to the right image. While D_{RL} is the right side disparity map, calculated using right image as a reference image.

$$\left|D_{LR}(p) - D_{RL}(p - D_{LR}(p))\right| \leq 1 \tag{20}$$

After unmatched pixels detection, the fill-in invalid disparity process is necessary. The invalid pixel on the disparity map is replaced with a nearest valid value. Knowing that, this valid value exists on the same line or on the starting line. The disparity map $d(p)$ is filled according to the conditions indicated in the Eq. (21). In this equation, $d(p - i)$ and $d(p + j)$ represent the location of the first valid disparity on the left and right side respectively.

$$d(p) = \begin{cases} d(p-i) & \text{if } d(p-i) \le (d(p+j) \\ d(p+j) & \text{otherwise} \end{cases} \quad (21)$$

The next step is the filtering process based on weighted median filter. WMF is used in different works for disparity refinement as Refs. [6, 34]. This filter is performed to remove the remaining noise and for smooths the hole-filled disparity map. The principle of the weighted median filter is to replace the central pixel over window by a weighted median value of its neighbors. Over square window, the weight of each pixel can be determined using both color and spatial distances between this pixel and neighboring pixels. The weighted filter median performs edge-preserving filters many times based on the weighted histograms. In this work, we based on the fast implementation of the weighted median filter [35].

4 Implementation on graphics hardware

In recent years, the graphic processors have become powerful tools for massively parallel intensive computing. Generally, they are employed for several applications including image processing and computer vision. In November 2007, NVIDIA Corporation introduced CUDA, which is a parallel computing architecture and a complementary application programming interface that enables fast development of massively parallel applications. The GPU CUDA implementation is characterized by faster processing compared to power CPU. We adopt the predefined functions and existing libraries on CUDA to obtain an efficient description for each stereo matching step and consequently high-performance implementation for complete algorithm.

On the other hand, the cost aggregation step using filtering process is computationally expensive and many works focus on reducing the complexity associated with this step as Refs. [8, 10, 18]. The GPU implementations of these algorithms achieve their goals and yield real-time performance. In addition, our approach is based on DP to calculate the matching between two scanlines. DP's scanline-independent computational structure allows us to easily take advantage of the manycore architectures of GPUs. For these reasons, we estimated that GPU implementation can be an efficient solution. Our stereo model relied on CUDA language for parallel treatment contains four fundamental steps:

- Loading input images: Transfer the stereo pair images from the CPU to the GPU memory (Host to Device).
- Thread Allocation: Fix the threads number of the calculation grid such that each thread can perform its processing on a pixel template.
- Parallel processing: Realize parallel processing provided by kernel functions. These functions are executed N times using the N threads defined in the previous step.
- Presentation of the results: Transfer the result from the device to the host memory (Device to Host) and visualize the obtained disparity map.

After fixing necessary threads and blocks number and loading the stereo pair image in the memory of the device, all processes of our approach are performed with kernel functions. These functions are executed in parallel by multiple threads. In the proposed stereo matching algorithm, the organization of the kernel functions is indicated in Fig. 4. This figure presents our algorithm implementation on the GPU while explaining the transactions between CPU, GPU and vice versa.

Initially, we start with cost calculation kernel which returns the cost volume $V(x, y, d)$ computed by Eq. (12). The first step is to allocate memories for the input images and output cost volume V. The used images are segmented left and right images (I_l, I_r), the two gradient images (G_l, G_r) and the rank transform of the two images (RT_l, RT_r). Then, we calculate the color distance and gradient matching noted respectively by $C0$ and $C1$. These two variables are performed by Eqs. (2) and (4) using the threshold τ_{AD} and τ_{GM}. The variable $C3$ is associated with rank distance, it is obtained by applying absolute difference between RT_l and RT_r as indicated in Eq. (11). The result of the combination

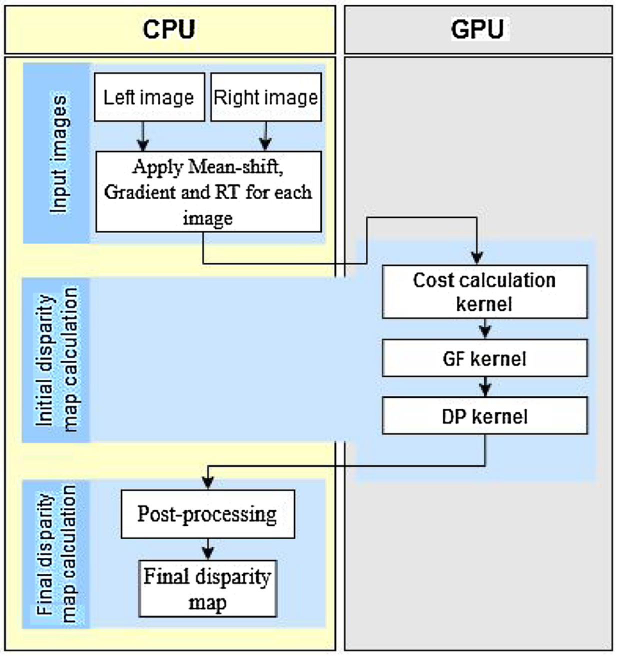

Fig. 4 Partition of our stereo matching algorithm

of these variables gives us the matching cost of the pixel p and it is stored in the output V.

To treat all pixels, the grid size is $(\frac{W}{BX}, \frac{H}{BX})$, where H and W are the height and width of the image. The size of the block of threads is $(BX \times BX)$, where BX is equal to 32. For each disparity value d, the cost function of each pixel p at location (x, y) in reference image to pixel $\overline{p}$ at location $(x - d, y)$ in target image is computed. Instead of processing pixel by pixel, this kernel performs the matching cost for all pixels in reference image to all pixels shifted by d in target image. Details of the cost calculation kernel are given by Algorithm 1 using left image as reference images.

Algorithm 1: Matching cost function.

```
Input  : $I_l, I_r, G_l, G_r, RT_l, RT_r, \tau_{AD}, \tau_{GM}, \tau_{GM}, \tau_{RT}, \alpha$
Output: $V$
$j \leftarrow threadIdx.y + blockIdx.y \times blockDim.y$
$i \leftarrow threadIdx.x + blockIdx.x \times blockDim.x$
for $d = 0$ to $d < d_{max}$ do
    if $(j > H$ or $i > W)$ then
        return ;
        $y \leftarrow i + j \times cols$;
    else
        $C0(x) \leftarrow \mathtt{min}(\mathtt{abs}(I_l(x) - I_r(x-d)), \tau_{AD})$;
        $C1(x) \leftarrow \mathtt{min}(\mathtt{abs}(G_l(x) - G_r(x-d)), \tau_{GM})$;
        $C2(x) \leftarrow \mathtt{min}(\mathtt{abs}(RT_l(x) - RT_r(x-d)), \tau_{RT})$;
        $V(x + d \times W \times H) \leftarrow$
          $2 - exp(-(\alpha.C0 + (1-\alpha).C1)) - exp(-C3)$;
    end
    end
end
_syncthreads();
```

The second kernel function is dedicated to perform the cost aggregation step. It is realized using GF. This filter is already described in Sect. 3.3. Generally, GF uses a guidance image I to filter a guided image f. In our work, the guidance image is the grayscale left image, while the guided image is a slice (x, y) of the cost volume. Theoretically, GF weights are given by Eq. (13). An inherent advantage of GF indicated in Ref. [36] is that the weights can be computed with some linear operations, which can be decomposed into a series of box filters with windows radius r. GF can be computed in $O(N)$ time, where N is the number of image pixels. Moreover, we do not need to implement the box filtering manually since two libraries OpenCV of Intel and NPP of Nvidia already have offered. To aggregate our cost matching, we start by extract $P(x, y)$ from $V(x, y, d)$ at fixed disparity d. Then, we follow the steps of the filtering process detailed by Algorithm 2 to obtain filtered image img_out. We note that the function $boxf$ is box filter, ϵ is a smoothness parameter and r presents a guided filter radius.

Algorithm 2: Guided filter for cost aggregation.

```
Input  : $I_l, P, r, \varepsilon$
Output: $img\_out$
$mean\_I_l \leftarrow \mathtt{boxf}(I_l, r)$
$mean\_P \leftarrow \mathtt{boxf}(P, r)$
$mean\_I_lP \leftarrow \mathtt{boxf}(I_l. \times P, r)$
$cov\_I_lP \leftarrow mean\_I_lP - mean\_I_l. \times mean\_P$
$mean\_I_lI_l \leftarrow \mathtt{boxf}(I_l. \times I_l, r)/N$
$var\_I_l \leftarrow mean\_I_lI_l - mean\_I_l. \times mean\_I_l$
$a \leftarrow cov\_I_lP/(var\_I_l + \varepsilon)$
$b \leftarrow mean\_P - a. \times mean\_I_l$
$mean\_a \leftarrow \mathtt{boxf}(a, r)$
$mean\_b \leftarrow \mathtt{boxf}(b, r)$
$img\_out \leftarrow mean\_a. \times I_l + mean\_b$
```

The next kernel function is devoted to disparity computation and optimization based on DP with automatic estimation of adaptive penalties. This optimization approach is characterized by its parallelism and the matching of each line can be calculated independently. The main of DP kernel function is to compute matrix cost M_h and find the minimum cost path. In our approach, we start by calculating the vector of penalties λ_v as given by Eq. (19). The matrix M_h for scanline y is computed by replacing λ with $\lambda_v(y)$ in Eq. (16). Inspired by the works [9, 10, 37], for each scanline y, we construct M_h and also the matrix A which contains the minimum cost index. The two matrices have $D_{\max} \times W$ entries, where $D_{\max}$ is the disparity range and W is the image width. We traverse M_h from left to right position by updating the entries in M_h and A. After the rightmost column is filled, the optimum path can be extracted from A. The DP process for a scanline y is given by Algorithm 3.

In this work, we adopt the kernel developed by Ref. [11] to calculate each scanline disparity. The inputs of DP kernel are the aggregated cost volume V and the vector of penalties λ_v. The size size of the block of threads is $(1 \times BX)$.

Algorithm 3: DP calculation and optimal path extraction.

```
Input  : $V$, $\lambda_v(y)$
Output: $M_h$, $A$
for $d = 0$ to $d < D_{max}$ do
    $M_h(d,0) \leftarrow V(0,y,d)$
end
for $x = 1$ to $x < W$ do
    $c_{min0}(d,0) \leftarrow V(x,y,d) + M_h(d,x-1)$;
    $c_{min1}(d,0) \leftarrow V(x,y,d) + M_h(d-1,x-1) + \lambda_v(y)$;
    $c_{min2}(d,0) \leftarrow M_h(d+1,x-1) + \lambda_v(y)$;
    Uptade $M_h$ based on Equation (16);
    if ($M_h(d,x) == c_{min0}$) then $A(d,x) \leftarrow (d,x-1)$;
    if ($M_h(d,x) == c_{min1}$) then $A(d,x) \leftarrow (d-1,x-1)$;
    if ($M_h(d,x) == c_{min2}$) then $A(d,x) \leftarrow (d+1,x)$;
end
```

5 Experimental results

The computational complexity of our stereo matching approach and the runtime are discussed in the following. For the experiment, The used graphics card is NVIDIA GeForce RTX 2070, equipped with 2304 CUDA cores running at 1.4 GHz. It is connected with Intel Core i7-3770M based CPU with a clock speed 3.4 GHz. Our algorithm is evaluated on versions 2 and 3 of Middlebury benchmark are respectively represented by MV2 and MV3. This dataset is used as facto standard for comparing and ranking the stereo matching algorithms according to their performance. For Middlebury benchmark, the stereo methods are assessed by the average of bad pixels given by Eq. (22), where D_x is the obtained disparity map, GT_x is the ground truth and δ presents the disparity tolerance.

$$b_p = \sum_x |D_x - GT_x| > \delta \tag{22}$$

In this section, we start by evaluating the performance of our approach based on DP with self-adjusting and adaptive penalties and compare it with other optimization methods. Figure 5 shows the disparity maps of some images on Middlebury benchmark using our approach, WTA, dynamic programming with constant penalty value (DP) and SGM. While Fig. 6 indicates the percentage of the errors in the non-occluded regions.

We note that WTA generally gives the biggest errors. This implies that the use of an optimization approach (DP or SGM) improves the results. Furthermore, our proposed approach gives us the smallest errors for all images and outperforms DP and SGM.

In MV2, the stereo algorithms are ranked according to their accuracy using different values of the disparity tolerance. This dataset contains only 4 stereo pairs (Tsukuba, Venus, Teddy and cones) with different disparity levels. We calculate the average of the errors for all stereo pairs on MV2 (Avg). We note the average errors in non-occluded region by Avg Non-occ. Avg All and Avg Disc denote the average of absolute errors and the average of depth discontinuities errors respectively. These measures are computed for a threshold $\delta = 1$. The evaluation results on MV2 are shown in Table 1.

This table confirms that our method outperforms recent methods ranked in the evaluation table of MV2. Among these methods, we find the method of two step global

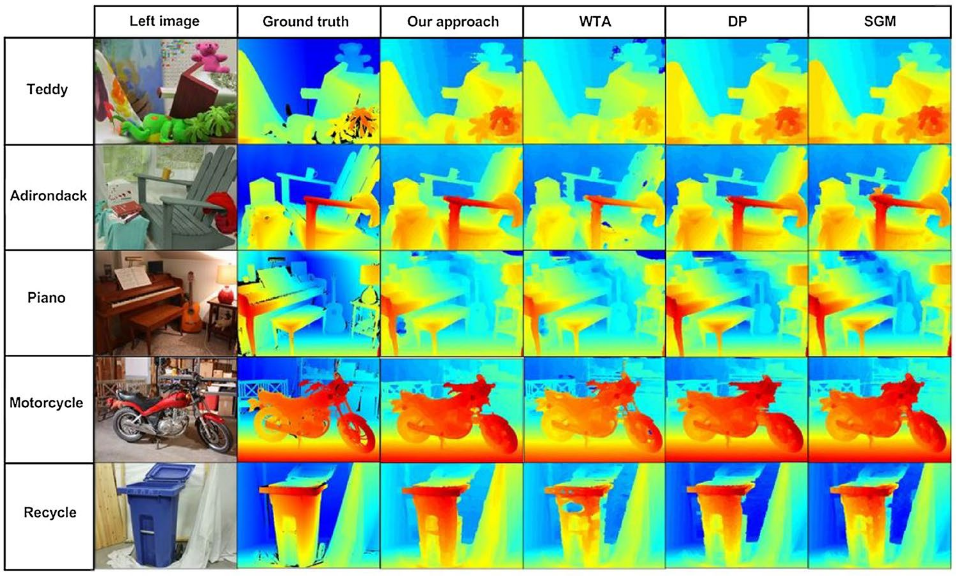

Fig. 5 Obtained disparity maps using our approach and other optimization techniques

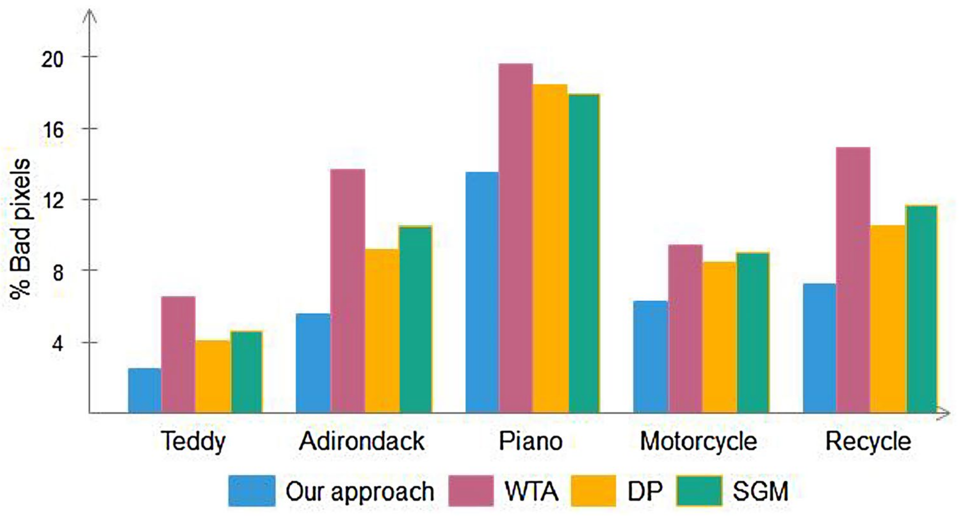

Fig. 6 Percentage of the errors on some images of Middlebury dataset

Table 1 Quantitative evaluation of our method on MV2

Method	Threshold $\delta = 1$			
	Avg non_occ	Avg all	Avg disc	Avg
Our approach	1.80	5.00	6.20	4.33
CostFilter [7]	2.64	5.57	8.42	5.55
IDR [18]	3.24	6.36	8.99	6.20
AdaptiveGF [23]	2.2	5.36	7.23	4.98
OptimizedDP [37]	4.25	9.04	13.2	8.83
TSGO [38]	2.06	3.9	6.2	4.06
CCRADAR [39]	2.17	4.91	7.17	4.75

Table 2 Accuracy and speed on MV3

Method	Avg nonocc	Avg all	Time (s)	Time/MP	Time/GD
Our approach	5.24	8.74	0.04	0.13	1.61
MSMD-ROB [3]	6.46	9.20	0.74	2.37	32
IGF [6]	6.39	9.49	79.2	261	3253
IDR [18]	5.33	8.57	0.34	0.27	0.82
FASW [22]	5.12	8.47	51.8	161	1962
MBM [30]	6.28	10.1	0.01	0.01	0.02
LS-ELAS [40]	9.66	12.9	2.47	0.50	1.86
DDL [41]	5.18	8.49	70.9	222	2597
ISM [42]	6.45	9.65	15.1	46.3	550
ReS2tAC [43]	7.55	13.9	0.13	0.41	5.61
FADNet [44]	6.01	6.16	0.60	0.49	1.69

optimization TSGO [38], CostFilter [7] based on AD and GM costs aggregated by guided filter and the method that use adaptive guided filtering [23]. Moreover, the proposed method improves upon the average percentage of bad pixels by 4.5% and 1.87% using the optimized DP [37] and IDR [18] respectively.

The database MV3 contains 30 pairs (15 for training and 15 pairs for test) with full, half and quarter resolution. The maximal disparity of quarter resolution (up to 750×500) can reach 200. In this dataset, the error is calculated using 2 masks: all and non-occluded. Several other metrics are added as total time per second (time), the average absolute errors in non-occluded regions (Avg nonocc), average absolute errors in all regions (Avg all). We compare our method against other relevant stereo algorithms listed in the evaluation table of MV3 and summarize the results in Table 2. The metric Time/MP presents the time normalized by number of pixels (*s*/*megapixels*). While Time/GD is the time normalized by number of disparity hypotheses (*s*/(*gigapixels* $*$ *ndisp*)).

Table 2 shows that our method takes 0.04 s and produces an average absolute errors in non-occluded regions equal to 5.24%. This table indicates that our approach is more accurate and faster than IGF [6], LS-ELAS [40], ISM [42] and MSMD-ROB [3]. Also, our method outperforms in terms of accuracy ReS2tAC [43] and the work of a multi-block matching on GPU [30]. But these algorithms are more faster than our approach. The proposed stereo matching framework generates slightly less accurate results than DDL [41] and FASW [22], but it is more fast than these works.

Another evaluation in terms of both speed and accuracy is made to compare the suggested method with recent real-time algorithms. We tested our approach on images with a resolution 640×480 pixels and 64 disparity levels. The used metrics are the average percentage of bad pixels, the number of the frames per second noted (FPS) and the number of millions of disparity per second (MDE/s). The results are shown in Table 3, where the second column contains the image size and maximal disparity value

Table 3 A comparison of accuracy and speed for real-time methods

Method	Size, D_{max}	GPU	MDE/s	FPS	% bad pixels
Our approach	(640×480),64	RTX 2070	707.7	36	4.33
Hosni [8]	(640×480),26	Quadro FX 5800	199.6	25	5.55
SGFD [9]	(320×240),32	GTX960	93.3	38	8.31
BF-DP [10]	(340×240),16	GTX 580	251	192	8.40
IDR [18]	(320×240),32	GTX 580	152.5	62	6.20
FastBilateral [21]	(320×240),32	Tesla C2070	50.6	21	7.31
RealTimeGPU [33]	(320×240),32	Radeon XL1800	52.8	21	9.82

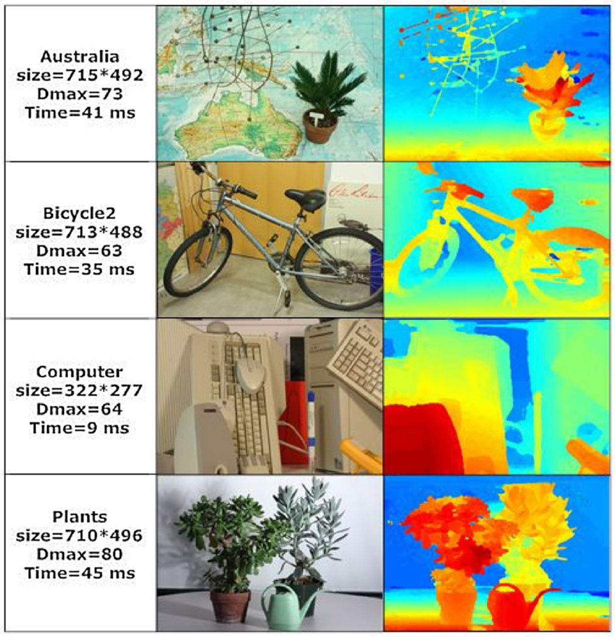

Fig. 7 Obtained disparity maps rand runtime on MV3 test

(D_{max}). The application of the proposed method on the used images takes 27.7 ms. So, our approach can achieve 36 frames per second. It is more accurate and faster than many DP-based algorithms as SGFD [9], BF-DP [10] and RealTimeGPU [33]. Table 3 indicates that our method outperforms Hosni [8] based on guided filter, IDR [18] and FastBilateral [21] in terms of both speed and accuracy.

In terms of speed, there are so fast algorithms as [45], this work runs at 40 FPS for 1436×992 pixels images with 145 disparity values. But its error rate on MV3 using threshold equal to 2 (Bad 2.0) is 32.82%. According to our evaluation, the same error rate is 27.1%. So, our work remains more accurate.

We evaluate our proposed method on MV3 testing dataset. Some obtained results are displayed in Fig. 7. Smooth disparity maps are produced as indicated in last column of the figure. The second column presents the left stereo image and the first column gives the image size, the maximal disparity value and the execution time (ms).

6 Conclusions

This paper presents a novel approach for real-time stereo matching with high accuracy. For disparity calculation step, this work defines a novel method of optimization aiming at the self-adjustment of adaptive penalty values for dynamic programming applicable for any image pair. This is obtained by the automatic estimation of the penalties vector through a simple process with low computational requirements, relying on the gradient images difference and the Disparity Space Image. The proposed stereo matching algorithm is evaluated on versions 2 and 3 of the Middlebury stereo benchmark. The results indicate that the suggested approach outperforms the classic version of dynamic programming, the algorithm of the two-step energy minimization, semi global matching and other existing methods.

Furthermore, we present an implementation of our method on graphics hardware using CUDA to reduce the execution time. We exploit CUDA parallel computing architecture to implement the proposed algorithm. The implementation results show that our approach can achieve real-time performance. In addition, they confirm that the proposed method outperforms many real-time algorithms in terms of accuracy and runtime.

Acknowledgements The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through research groups program under grant number GRP/337/42

References

1. Hamzah, R.A., Ibrahim, H.: Literature survey on stereo vision disparity map algorithms. J. Sens. **2016**, 1–23 (2016)
2. Hirschmuller, H., Scharstein, D.: Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans. Pattern Anal. Mach Intell. **31**(9), 1582–1599 (2009)
3. Lu, H., Xu, H., Zhang, L., Ma, Y., Zhao, Y.: Cascaded multi-scale and multi-dimension convolutional neural network for stereo

matching. In: IEEE visual communications and image processing (VCIP) (2018)
4. Zhan, Y., Gu, Y., Huang, K., Zhang, C., Hu, K.: Accurate image-guided stereo matching with efficient matching cost and disparity refinement. IEEE Trans. Circuits Syst. Video Technol. **26**(9), 1632–1645 (2016)
5. Tan, P., Monasse, P.: Stereo disparity through cost aggregation with guided filter. Image Process Line **4**, 252–275 (2014)
6. Hamzah, R.A., Ibrahim, H., Hassan, A.H.A.: Stereo matching algorithm based on per pixel difference adjustment, iterative guided filter and graph segmentation. J. Vis. Commun. Image Represent. **42**, 145–160 (2017)
7. Rhemann, C., Hosni, A., Bleyer, M., Rother, C., Gelautz, M.: Fast cost-volume filtering for visual correspondence and beyond. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2011)
8. Hosni, A., Bleyer, M., Rhemann, C., Gelautz, M., Rother, C.: Real-time local stereo matching using guided image filtering. In: Proceedings of IEEE International Conference on Multimedia and Expo, pp. 1–6. Barcelona (2011)
9. Hallek, M., Smach, F., Atri, M.: Real-time stereo matching on CUDA using Fourier descriptors and dynamic programming. Comput. Vis. Media **5**(1), 59–71 (2019)
10. Wang, L., Yang, R.G., Gong, M.L., Liao, M.: Real-time stereo using approximated joint bilateral filtering and dynamic programming. J. Real Time Image Process. **9**(3), 447–461 (2014)
11. Congote, J., Barandiaran,J., Barandiaran, I., Ruiz, O.: Realtime dense stereo matching with dynamic programming in CUDA. In: Proceedings of the 19th Spanish Congress of Graphical Informatics, pp. 231–234 (2009)
12. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**, 7–42 (2002)
13. Yoon, K.J., Kweon, I.S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 650–656 (2006)
14. Hirschmuller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. Int. J. Comput. Vis. **47**, 229–246 (2002)
15. Zabih, R., Woodfill, J.: Non-parametric local transforms for computing visual correspondence. In: Proceedings of the Third European Conference on Computer Vision, Stockholm, pp. 151–158 (1994)
16. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Zhang, X.: On building an accurate stereo matching system on graphics hardware. In: GPUCV, Barcelona (2011)
17. Kordelas, G., Alexiadis, D., Daras, P., Izquierdo, E.: Enhanced disparity estimation in stereo images. Image Vis. Comput. **35**, 31–49 (2015)
18. Kowalczuk, J., Psota, E.T., Perez, L.C.: Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences. IEEE Trans. Circuits Syst. Video Technol. **23**(1), 94–104 (2013)
19. Yang, Q.: A non-local cost aggregation method for stereo matching. In: IEEE Computer Vision and Pattern Recognition, pp. 1402-1409 (2012)
20. Zhu, S., Wang, Z., Zhang, X., Li, Y.: Edge-preserving guided filtering based cost aggregation for stereo matching. J. Vis. Commun. Image Represent. **39**, 107–119 (2016)
21. Mattoccia, S., Viti, M., Ries, F.: Near real-time fast bilateral stereo on the GPU. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 136–143 (2011)
22. Wu, W., Zhu, H., Yu, S., Shi, J.: Stereo matching with fusing adaptive support weights. IEEE Access **7**, 61960–61974 (2019)
23. Yang, Q., Ji, P., Li, D., Yao, S., Zhang, M.: Fast stereo matching using adaptive guided filtering. Image Vis. Comput. **32**, 202–211 (2014)
24. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2005)
25. Stentoumis, C., Karkalou, E., Karras, G.: A review and evaluation of penalty functions for semi-global matching. In Proc. IEEE Int. Conf. Intelligent Computer Communication Processing, pp. 167–172 (2015)
26. Karkalou, E., Stentoumis, C., Karras, G.: Semi-global matching with self-adjusting penalties. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. **12**, 353–360 (2017)
27. Hirschmüller, H.: Stereo processing by semi-global matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008)
28. Wang, W., Yan, J., Xu, N., Wang, Y., Hsu, F.H.: Real-time high quality stereo vision system in FPGA. IEEE Trans. Circ. Syst. Video Technol. **25**(10), 1696–1708 (2015)
29. Cambuim, L., Oliveira, L., Barros, E., Ferreira, A.: An FPGA based real-time occlusion robust stereo vision system using semi-global matching. J. Real Time Image Proc. **30**, 1–22 (2019)
30. Chang, Q., Maruyama, T.: Real-time stereo vision system: a multi-block matching on GPU. IEEE Access **6**, 42030–42046 (2018)
31. Wang, Z.F., Zheng, Z.G.: A region based stereo matching algorithm using cooperative optimization. In: IEEE Comput. Soc. Conf. Comput. Vision Pattern Recogn, pp. 1–8 (2008)
32. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 603–619 (2002)
33. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission (2006)
34. Ma, Z., He, K., Wei, Y., Sun, J., Wu, E.: Constant time weighted median filtering for stereo matching and beyond. In: Proceedings of International Conference on Computer Vision (2013)
35. Zhang, Q., Xu, L., Jia, J.: 100+ times faster weighted median filter. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2014)
36. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE Trans. Pattern Anal. Mach. Intell. **35**(6), 1397–1409 (2013)
37. Salmen, J., Schlipsing, M., Edelbrunner, J., Hegemann, S., Lueke, S.: Real-time stereo vision: making more out of dynamic programming. In: Proceedings of International Conference on Computer Analysis of Images and Patterns (2009)
38. Mozerov, M.G., van de Weijer, J.: Accurate stereo matching by two-step energy minimization. IEEE Trans. Image Process. **24**(3), 1153–1163 (2015)
39. Jiao, J., Wang, R., Wang, W., Dong, S., Wang, Z., Gao, W.: Local stereo matching with improved matching cost and disparity refinement. IEEE MultiMed. **21**(4), 16–27 (2014)
40. Ait-Jellal, R., Lange, M., Wassermann, B., Schilling, A., Zell, A.: LS-ELAS: line segment based efficient large scale stereo matching. In: Proc. IEEE International Conference on Robotics and Automation (2017)
41. Yin, J., Zhu, H., Yuan, D., Xue, T.: Sparse representation over discriminative dictionary for stereo matching. Pattern Recogn. **71**, 278–289 (2017)
42. Hamzah, R.A., Kadmin, A.F., Hamid, M.S., Ghani, S.F.A., Ibrahim, H.: Improvement of stereo matching algorithm for 3D surface reconstruction. Signal Process. Image Commun. **65**, 165–172 (2018)
43. Boitumelo, R., Jonas, M., Martin, W., Stefan, H., Jurgen, B.: ReS2tAC-UAV-borne real-time SGM stereo optimized for embedded ARM and CUDA devices. MDPI Sensors **21**(11) (2021)

44. Wang, Q., Shi, S., Zheng, S., Zhao, K., Chu, X.: Fadnet: a fast and accurate network for disparity estimation. In: IEEE international conference on robotics and automation (ICRA), pp. 101–107 (2020)
45. Chang, Q., Maruyama, T.: Real-Time High-Quality Stereo Matching System on a GPU. In: IEEE 29th International Conference on Application-specific Systems, Architectures and Processors, pp. 1–8 (2018)

Mohamed Hallek is teacher researcher at the University of Monastir, Tunisia. He is member of the Laboratory of Electronics and Microelectronics, Faculty of Science of Monastir. Master in Micro and Nano Electronics obtained in 2012. His Fields of interest are real-time image processing, pattern recognition and stereo matching algorithms.

Hamdi Boukamcha Received his Ph.D. in 2017. He is an expert in image processing, Artificial intelligence and embedded system. Teacher researcher at the University of Monastir in Tunisia from 2012. Member of the Laboratory of Electronics and Microelectronics at the Faculty of Sciences Monastir, Founder of matlabrecognition-code.com. Currently, he is head of research and development at Kinestasia R & D laboratory in France.

Abdellatif Mtibaa received his Ph.D. degree in Electrical Engineering at the National School of Engineering of Tunis. Since 1990 he has been an Assistant Professor in Micro Electronics and Hardware Design with Electrical Department at the National School of Engineering of Monastir. Since 2007, he has been a professor at the Electrical Engineering Department at the ENIM. His research interests include highlevel synthesis, rapid prototyping and reconfigurable architectures for real-time multimedia applications.

Mohamed Atri is Professor in the Department of Computer Engineering at College of Computer Science, King Khalid University, Saudi Arabia. He received his Ph.D. Degree in Microelectronics from the University of Monastir, Tunisia in 2001 and the HDR degree in 2011. He is currently a member of the Laboratory of Electronics and Microelectronics, Faculty of Science of Monastir. His research interests include AI, Circuit and System Design, Image processing, Network Communication. He has authored and co-authored more than 200 publications including refereed journals, conference papers, and book chapters.