Computational Visual Media
https://doi.org/10.1007/s41095-019-0133-4 Vol. 5, No. 1, March 2019, 59–71

Research Article

Real-time stereo matching on CUDA using Fourier descriptors and dynamic programming

Mohamed Hallek [1] (✉), **Fethi Smach**[2], **and Mohamed Atri**[1]

Abstract Computation of stereoscopic depth and disparity map extraction are dynamic research topics. A large variety of algorithms has been developed, among which we cite feature matching, moment extraction, and image representation using descriptors to determine a disparity map. This paper proposes a new method for stereo matching based on Fourier descriptors. The robustness of these descriptors under photometric and geometric transformations provides a better representation of a template or a local region in the image. In our work, we specifically use generalized Fourier descriptors to compute a robust cost function. Then, a box filter is applied for cost aggregation to enforce a smoothness constraint between neighboring pixels. Optimization and disparity calculation are done using dynamic programming, with a cost based on similarity between generalized Fourier descriptors using Euclidean distance. This local cost function is used to optimize correspondences. Our stereo matching algorithm is evaluated using the Middlebury stereo benchmark; our approach has been implemented on parallel high-performance graphics hardware using CUDA to accelerate our algorithm, giving a real-time implementation.

Keywords generalized Fourier descriptors; stereo matching; dynamic programming; CUDA

1 Introduction

Due to technological advances and improvements in digital cameras, stereo vision is an important research area. However, dense correspondences and 3D reconstitution are key problems for computer vision researchers. Provision of an efficient algorithm for the reconstitution of 3D information from a stereo image pair of the same scene taken from distinct viewpoints is the main objective of stereo systems. The stereo system must follow three basic steps: calibration, correspondence, and reconstruction.

In this work, we focus on the correspondence step. The main aims of the stereo matching algorithm are to correctly identify corresponding pixels in the rectified stereo images and fill the disparity map [1, 2]. Stereo matching algorithms must overcome various problems, the most commonly encountered ones being noise, occlusion, and repetitive textures. Also the researcher must respect various constraints including epipolar geometry, ordering constraints, and smoothness. Many stereo matching algorithms have been developed to solve the correspondence problem using patch-based image synthesis methods [3, 4]. Analysing state of the art algorithms, stereo matching methods may be divided into local and global categories [5, 6]. The most popular local methods are based on block matching and feature matching [7, 8]. Generally, these methods involve an analysis of local light intensities around each pixel or some regions in the image. However all pixels in the image are involved in global methods, such as graph cut, belief propagation, and dynamic programming [5].

In 2002, Scharstein and Szeliski [1] defined a taxonomy to categorize dense correspondence algorithms. It shows that most existing stereo matching methods contain four steps:

- Cost function calculation: a matching process for each pixel at all possible disparity levels.
- Cost aggregation: aggregating the cost over the support region.

1 Faculty of Sciences of Monastir, 5000 Monastir, Tunisia. E-mail: hallekmohamed@yahoo.fr (✉).

2 Technologies et services de l'information, Paris, France.

Manuscript received: 2018-11-05; accepted: 2019-01-27

- Disparity computation and optimization: selecting the disparity value that optimizes the function and filling the disparity map.
- Disparity refinement: post-processing to improve the results.

Different techniques are used to realize each step. For example, block matching and box filters provide the most popular techniques for cost calculation and cost aggregation respectively. Usually block matching is done on gray images or on the intensity channel of color images. In this work, we use a mathematical transformation before calculating the cost function. Here, the first step is done by calculating generalized Fourier descriptors then finding the Euclidean distance between descriptors. Next, we apply a boxer filter to aggregate the cost function. The optimization and disparity computation is done using dynamic programming while the last step is performed using a stereo consistency check and median filter to improve the final disparity map.

Many stereo matching algorithms, especially the global methods, are computationally expensive. For this reason, many research works are interested in runtime reduction and real-time implementation. In this paper, we present a new approach for stereo image matching based on generalized Fourier descriptors. This approach is detailed in Section 3. In Section 4, we evaluate our algorithm and we give some experimental results. In Section 5, we present a CUDA-based real-time implementation of our approach on a GPU. Finally, conclusions are presented in Section 6.

2 Related work

As already indicated, the stereo matching process is realized by following four fundamental steps. It starts by defining a cost function and calculating the volume cost for each pixel at all disparity levels, then aggregating the matching cost. Next the energy function is optimized and the disparity map filled. Finally the obtained disparity map is postprocessed. In the literature, there are several techniques to achieve each step. The most common cost functions are absolute difference and block matching. The two functions are characterized by linear computational complexity, simplicity of implementation, and fast runtime. However, the limitations of these techniques are their failure in discontinuous areas and theirs sensitivity to the window size used. A simple comparison of light intensities is not always enough, hence the use of mathematical transformations such as census or rank transforms is required. These non-parametric transformations provide standard metrics and are more robust to radiometric distortion and occlusion [9]. In our work, we use a mathematical transformation based on the Fourier transform to extract robust descriptors. Similarity of descriptors provides our cost function. We note that many stereo matching methods are based on feature extraction and point of interest detection. In this context we can mention various descriptors such as the scale-invariant feature transform (SIFT) [10], and speeded-up robust features (SURF) [11]. Zernike moments are also used for the determination of corresponding points [12]. Generally, a mathematical transformation is calculated and robust descriptors are extracted to define an efficient cost function.

For cost aggregation, many techniques are employed. The simple solution is the use of linear image filters such as a box or Gaussian filter. Edge-preserving filters such as the bilateral filter and guided filter can also be good solutions, but they are computationally expensive. In our work, we adopt a simple box filter for cost aggregation.

In the disparity computation, we note that winner takes-all is the most common solution. This step can be improved using semi-global or global optimization algorithms such as graph cuts [13], belief propagation [14], or dynamic programming [15, 16]. Disparity refinement is done using the same approach as in Mattoccia et al. [17] and Kordelas et al. [18] to detect occlusions and depth borders. In this step, three consecutive processes are applied: invalid disparity detection, fill-in of invalid disparity values, and median filtering.

Many works consider acceleration of these computationally intensive algorithms. Different architectures are used to achieve real-time performance. One is based on field-programmable gate arrays (FPGAs) [19]. A second alternative is based on graphics hardware using the CUDA language, and is used in many real-time algorithms such as the work of Kowalczuk et al. [20] and Congote et al. [16]. Different real-time algorithms focus on reducing the complexity associated with cost calculation, at the expense of reduced accuracy. In our work we exploit NVIDIA's GeForce GTX960 computing capabilities to produce an accurate disparity map.

3 GFD for stereo matching

Figure 1 presents a block diagram of our stereo matching algorithm. The cost function is based on similarity of generalized Fourier descriptors, denoted SGFD. The cost is aggregated over square windows using a box filter. Then we use dynamic programming for energy optimization and filling the disparity map. At this stage, the obtained disparity map contains some invalid or unwanted pixels, so a post-processing step is required. A left–right consistency check allows us to detect these invalid pixels. Then we perform a fill-in process to replace the invalid pixels with valid minimum values. The refinement step includes median filtering to remove noise and enforce smoothness between neighboring pixels.

3.1 Representation of GFD

In pattern recognition, the Fourier transform has been used for many years to extract a set of invariants. In 2002, generic Fourier descriptors [21] were applied to grayscale images. Color descriptors called generalized Fourier descriptors (GFDs) were defined in 2008 which can be applied to both grayscale and color images [22]. These descriptors are given in Eq. (1), when (r,θ) are polar coordinates of the input point M and $\tilde{f}(r,\theta)$ is the Fourier transform of the function f at the point $M(r,\theta)$.

$$D_f(r) = \int_0^{2\pi} \parallel \tilde{f}(r,\theta)^2 \parallel \mathrm{d}\theta \tag{1}$$

These invariants are functions of a single variable (radius) which makes them simple to calculate. In an image, the integral in Eq. (1) becomes a discreet sum that gives us a set of values forming a vector. All descriptors must satisfy some invariance and robustness properties. For GFDs, these properties are well respected. The theoretical properties of GFDs are detailed in the work of Smach et al. [22] and we note their invariance under motion, change of scale, and reflection. In practice, GFDs are obtained for color images using the flowing steps:

- decompose the color image into three channels (red, green, and blue);
- calculate the Fourier transform and its square modulus for each channel;
- vary the radius r and compute the sum of the pixels located along each ray.

The final descriptors concatenate the descriptors for each channel. These descriptors give a complete and robust description of the image which can be used for color object recognition and image classification. Smach et al. [22] evaluated the performance of the GFD on several standard and personal databases. The results obtained using GFD and support vector machines (SVMs) for classification indicated the robustness of these descriptors. GFDs outperform various families of invariants, such as Zernike moments. See Refs. [22, 23] for more details of GFDs.

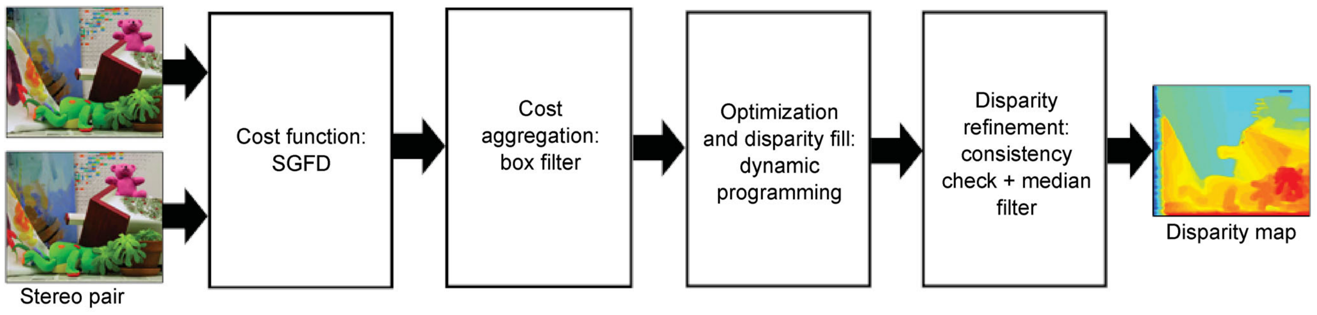
3.2 Local cost function

Invariance under geometric transformations, and robustness to noise and lighting changes are important properties of GFDs which allow us to use GFDs for stereo matching. A full and easily accessible description can characterize a region in the reference image and identify it in a target image.

In stereo matching, the cost function is the main step and it differs from one algorithm to another. Our energy function, denoted SGFD, is defined by similarity between GFDs: for a stereo pair we take the left image as the reference image and characterize some region in this image by the left color descriptors, $GFD_{\mathrm{c}}^{\mathrm{l}}$. Then, we calculate the descriptors for the candidate region in the right image, $GFD_{\mathrm{c}}^{\mathrm{r}}$. The two descriptors can be expressed as below:

$$GFD_{\mathrm{c}}^{\mathrm{l}} = [a_0,\ldots,a_N]' \tag{2}$$

$$GFD_{\mathrm{c}}^{\mathrm{r}} = [b_0,\ldots,b_N]' \tag{3}$$



Stereo pair
Cost function: SGFD
Cost aggregation: box filter
Optimization and disparity fill: dynamic programming
Disparity refinement: consistency check + median filter
Disparity map

Fig. 1 Block diagram of our approach.

where a_i and b_i are the components of the left and right color descriptors and N is the radius of the window. After computing these descriptors, we compute their similarity $SGFD_{\mathrm{c}}$ using their Euclidean distance:

$$SGFD_{\mathrm{c}} = \mathrm{dist}(GFD_{\mathrm{c}}^{\mathrm{l}}, GFD_{\mathrm{c}}^{\mathrm{r}}) = \sqrt{\sum_i (a_i - b_i)^2} \tag{4}$$

Many algorithms combine the matching costs for colors and gradients in order to increase the robustness against radiometric differences. Yang et al. [24] and Hosni et al. [25] combined absolute differences of color and gradient using a parameter α. In a similar way, our matching cost combines the Euclidean distance for color descriptors ($SGFD_{\mathrm{c}}$) and the Euclidean distance for gradient descriptors ($SGFD_{\mathrm{g}}$). The global cost function for a pixel p at the disparity value d is denoted by $SGFD(p, d)$:

$$SGFD(p, d) = \alpha SGFD_{\mathrm{c}}(p, d) + (1-\alpha) SGFD_{\mathrm{g}}(p, d) \tag{5}$$

The parameter α is used to balance the color and gradient terms as in Yang et al. [24]. In the above, we need to calculate $SGFD_{\mathrm{g}}(p, d)$ based on the gradient. This is done using the following steps:

- Calculate the gradient values in horizontal and vertical directions for the left and right images (I_{l}, I_{r}). These values, denoted G_x,G_y, are given by Eqs. (6) and (7).
- Calculate the gradient magnitude G for both images, as given by Eq. (8).
- Calculate the generalized Fourier descriptors for reference and target images.
- Compute the Euclidean distance between descriptors.

The necessary equations are given below, where $A = [1\ 0\ -1]$ and I is either the left or right image. Here $*$ denotes the convolution operation, and A^{T} is the transpose of A.

$$G_x = A * I \tag{6}$$

$$G_y = A^{\mathrm{T}} * I \tag{7}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{8}$$

After calculating $SGFD_{\mathrm{g}}(p, d)$, this cost function is aggregated using a simple technique. By applying a box filter, we aggregate the matching cost over a square window ω. The aggregated cost for a pixel p at disparity value d is given by

$$CA(p, d) = \omega(p, d) * SGFD(p, d) \tag{9}$$

following Scharstein and Szeliski [1].

3.3 Optimization and disparity fill approach

After cost matching calculation and use of a box filter for the aggregation step, the third step in Scharstein and Szeliski's taxonomy is performed. The aim of this step is to optimize the cost and fill the disparity map. The most popular method is the winner takes-all (WTA) technique, which selects the minimal aggregated corresponding value for each pixel:

$$d = \mathrm{argmin}_{d \in d_{\mathrm{r}}} CA(p, d) \tag{10}$$

where d_{r} defines the set of allowed discrete disparity levels in an image. The use of WTA reduces computational complexity but can produce unmatched pixels and invalid disparity values at the image border and occluded regions.

Once the global approach has been developed, a variety of algorithms can be used to find the correctly matching points. These algorithms make explicit smoothness assumptions and optimize a global cost function that combines matching cost and smoothness cost as detailed in Ref. [1]. This global cost function is defined by

$$E(d) = E_{\mathrm{data}}(d) + \lambda E_{\mathrm{smooth}}(d) \tag{11}$$

Dynamic programming (DP) is a popular optimization approach. Generally, the aim of DP is to solve a global problem by dividing it into smaller sub-problems whose solution can easily be obtained. The global solution is the concatenation of the solutions of all sub-problems. This optimization approach was introduced for stereo vision by Ohta and Kanade [26]. DP exploits ordering and the smoothness constraints to optimize the matching cost between two scanlines. This technique is based on two stages: a step for constructing the cost matrix for all pixels at all possible disparity levels and a step in which pairs of corresponding pixels are selected by searching for the optimal path. Let the aggregated cost function be denoted by $CA(x, y, d)$ where x and y represent the position of a pixel p. The matrix extracted from a volume cost $CA(x, y, d)$ for a fixed line is denoted $M_{\mathrm{h}}(x, d)$. The dimensions of M_{h} are $W \times D_{\max}$ where W and $D_{\max}$ represent the width of the image and the maximal disparity. In our work, we use the DP approach developed by Congote et al. [16]. Each matrix M_{h} that represents the matching between two scanlines is updated according to

$$\begin{aligned} M_{\mathrm{h}}(x, d) = & CA(x, d) + \\ & \min(\lambda + M_{\mathrm{h}}(x-1, d-1), \\ & M_{\mathrm{h}}(x-1, d), \lambda + M_{\mathrm{h}}(x, d+1)) \end{aligned} \tag{12}$$

We note that λ represents a penalty for change in disparity values between neighboring pixels. After calculating M_{h}, we compute the disparity value of the corresponding line using the algorithm for the optimal path.

3.4 Disparity refinement

By this stage, the cost function has been calculated and aggregated. Dynamic programming is also applied to fill disparity map. The obtained disparity map contains unmatched pixels due to occlusion and repetitive textures. Thus, a postprocessing step is required. This step involves three consecutive processes: invalid disparity detection, fill-in of invalid disparity values, and weighted median filtering. Disparity refinement starts by detection of invalid disparity values and unmatched pixels in the depth map. This stage is done using a left–right consistency checking process, comparing the left disparity map to the right disparity map. Inconsistent pixels between the two disparity maps are marked as having invalid disparities. In our work, we use the same approach defined by Mattoccia et al. [17] and Kordelas et al. [18].

Disparity values are marked as invalid if they do not satisfy the condition below:

$$|D_{\mathrm{LR}}(p) - D_{\mathrm{RL}}(p - D_{\mathrm{LR}}(p))| \leqslant 1 \tag{13}$$

where D_{LR} and D_{RL} represent the left reference disparity and right disparity map respectively.

The next step for disparity refinement is fill-in of invalid disparity values. The disparity of each unmatched pixel is replaced with the nearest valid disparity. Knowing that, the used valid value is located in the same line or in the starting line. This process is by

$$d(p) = \begin{cases} d(p-i), & \text{if } d(p-i) \leqslant d(p+j) \\ d(p+j), & \text{otherwise} \end{cases} \tag{14}$$

where the disparity value at the location of p is defined by $d(p)$ and $(p-i)$ indicates the location of the first valid disparity on the left side while $(p+j)$ is the location of the first valid disparity on the right side.

We finish the refinement step with median filtering in order to reduce noise and enforce smoothness between neighboring pixels.

4 Experimental results and analysis

To evaluate our stereo matching approach it is necessary to use standard databases. We confronted our algorithm with some stereo matching problems involving occlusion and discontinuous regions. The average number of bad pixels in these regions indicates the accuracy of our stereo matching method, and is given by

$$b_p = \sum_x |D_x - GT_x| > \delta \tag{15}$$

where D_x is the obtained disparity map, GT_x is the ground truth, and δ presents a disparity tolerance.

To evaluate our approach, we start by evaluating our proposed local function: SGFD can be compared with other local energy functions. We study the effect of the window size and give the evaluation errors. The test was done by changing the window size from (3×3) to (25×25) and the errors were calculated in the non-occluded regions. The images used were ArlL and Teddy from the Middlebury dataset. Results are shows in Fig. 2.

The first column in Fig. 2 shows the left image of ArtL and the percentage of bad pixels in non-occluded areas (curves (a) and (b)). The second column presents the Teddy image and the corresponding errors (curves (c) and (d)). The results obtained for all cost functions were calculated without performing any post treatment and errors were calculated using Eq. (15) with $\delta = 1$. We compare SGFD, ZSAD (zero mean sum of absolute differences), and NCC (normalized cross correlation) in curves (a) and (c) for ArtL and Teddy images respectively. Curves (b) and (d) compare SGFD, ENCC (enhanced normalized cross correlation) [27], and ZNCC (zero mean normalized cross correlation) for the two stereo pairs. For the images used, we note that the lowest error in non-occluded areas is obtained by SGFD using a small window size (3×3) or (5×5). In addition, this error always remains lower compared than the errors given by other functions for a large window. Thus, these curves indicate that SGFD is more accurate than other local cost functions and more robust to window size variation.

After evaluating the local function, we evaluated our stereo matching algorithm on the Middlebury dataset. This database is used as a de facto standard for comparing stereo matching methods and ranking them according their performance. We started by using version 2 of this database, denoted MV2; it contains only 4 stereo pairs (Tsukuba, Venus, teddy, cones). Results are provided in Tables 1 and 2. We denote the average errors in non-occluded regions

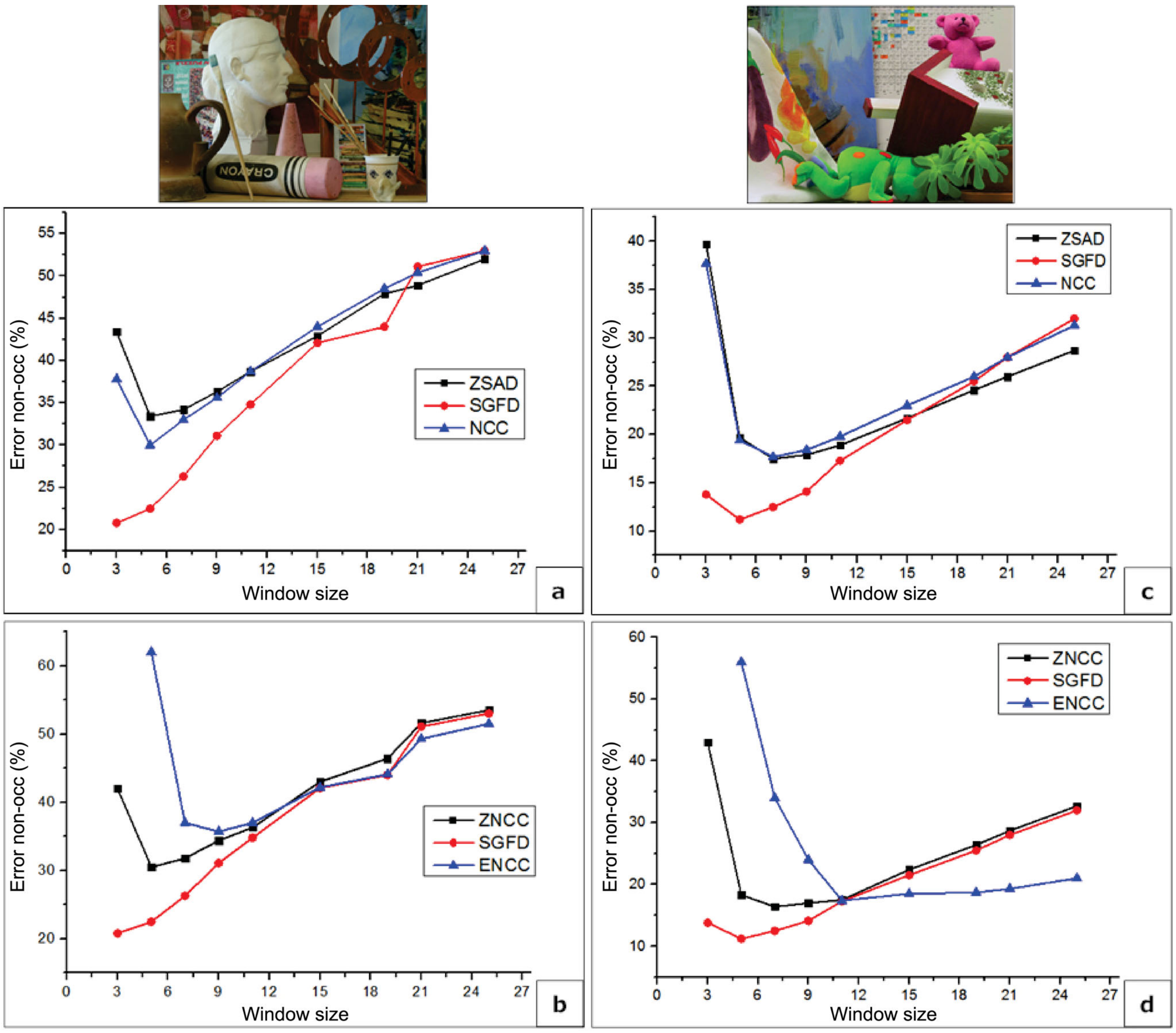


Fig. 2 Effect of window size variation for SGFD and other cost functions.

Table 1 Comparison of methods on MV2 using $\delta = 1$

Method	Threshold $\delta = 1$			
	Avg non-occ	Avg all	Avg disc	Avg
Our approach	4.08	7.16	13.7	8.31
OptimizedDP [28]	4.25	9.04	13.2	8.83
AdaptAggrDP [29]	3.85	8.42	12.91	8.40
LCVB-DEM [30]	5.59	9.20	19.27	11.60
SGMDDW [31]	3.89	8.95	15.22	9.36

Table 2 Comparison of methods on MV2 using $\delta = 2$

Method	Threshold $\delta = 2$			
	Avg non-occ	Avg all	Avg disc	Avg
Our approach	2.52	5.70	8.11	5.44
OptimizedDP [28]	2.83	6.87	7.37	6.62
AdaptAggrDP [29]	2.40	5.96	8.32	5.56
LCVB-DEM [30]	4.18	7.42	14.49	8.69
SGMDDW [31]	2.67	7.03	10.95	6.89

and the average of absolute errors by Avg non-occ and Avg all respectively, while the average of depth discontinuity errors are denoted Avg disc. The metric Avg indicates the average of the bad pixel error. These measures are used as the main metrics to evaluate the accuracy of all stereo matching methods. We note that the calculation of these parameters is performed using Eq. (15). To evaluate our approach, we determined the disparity maps for all images in MV2. Then we calculated the four metrics using $\delta = 1$ and $\delta = 2$ as indicated respectively in Tables 1 and 2. These two tables show that our approach gives the lowest average errors for both thresholds, and confirm that our proposed approach outperforms the optimized dynamic programming method of Salmen et al. [28], and the approach of Wang et al. [29] based on joint bilateral filtering and dynamic programming. In addition, our approach is more accurate than other recent works given in the evaluation table for MV2 (`http://vision.middlebury.edu/stereo/eval`).

Furthermore, we evaluated our stereo matching algorithm on version 3 of the Middlebury benchmark (MV3). This dataset contains 30 stereo pairs: 15 pairs for training and 15 pairs for testing. The images in this database have resolution up to 750×500 and a maximal disparity that can reach 200. Evaluation results for MV3 are shown in Table 3. It presents the percentage of bad pixels in non-occluded regions and in all regions (nonocc, all). The errors are calculated for the two regions using thresholds equal to 2 and 4 (Bad2, Bad4). The average absolute error is denoted Avgerr and indicated in the last column. Table 3 summarizes the evaluation results on the training dataset. The obtained results are detailed in Tables 4 and 5, which respectively show the error for each stereo pair on non occluded and all regions.

Table 3 Comparison of methods on MV3

Method	Bad2		Bad4		Avgerr	
	nonocc	all	nonocc	all	nonocc	all
Our approach	33.4	41.4	20.3	29.5	9.76	17.6
SPS [32]	21.1	28.4	16.0	23.1	10.4	16.6
DoGGuided [33]	37.0	44.0	22.2	30.5	12.0	22.3
BSM [34]	37.1	44.8	23.4	32.6	13.4	23.5
ICSG [35]	37.7	43.3	32.0	37.5	21.3	26.9
SGBM1 [36]	27.4	34.5	22.0	29.1	11.3	18.9

The evaluation on MV3 indicates that our proposed approach outperforms other algorithms such as DoGGuided [33] that use a guided filter based on the response of the difference of Gaussian, binary stereo matching (BSM) [34] and other recent approaches. In addition our stereo matching algorithm is more accurate than ICSG [35] and semi global matching (SGBM1) [36]. Further details of the evaluation on MV3 are provided by disparity maps displayed in Fig. 3. This figure presents respectively the left images and ground truths (GT) for the stereo pairs used in the second and the third columns; the disparity maps in the fourth column are calculated using the proposed approach. Further columns show the disparity maps obtained with other stereo matching algorithms. Errors in both regions (all and non-occluded) for these disparity maps are given Tables 4 and 5.

Stereo matching methods are classified according to different measures, essentially based on the disparity map quality and the execution time. Global methods are computationally expensive. Their complexity is $O(N^N)$ operations per scanline, for scanlines of N pixels. The use of dynamic programming can reduce this complexity to $O(N^2)$ but this does not offer a fast-running implementation. A software implementation of our method has a long runtime and does not meet realtime needs.

To accelerate our stereo matching system we propose an implementation based on graphics

Table 4 Performance comparison of quantitative evaluation results based on nonocc error from MV3

Method	Avg	Adiron	ArtL	Jadepl	Motor	MotorE	Piano	PianoL	Pipes	Playrm	Playt	PlaytP	Recyc	Shelvs	Teddy	Vintg
Our method	9.76	6.29	8.40	41.4	6.08	5.95	4.74	8.64	13.6	8.68	12.0	4.78	3.90	10.0	3.73	6.96
SPS [32]	10.4	3.57	5.34	22.8	3.11	3.15	9.34	22.9	6.78	12.5	9.70	7.64	6.27	22.3	1.52	52.6
DoGGuided [33]	12.0	15.2	9.57	27.1	5.64	8.31	8.09	32.4	9.67	14.0	24.5	5.32	5.56	16.2	4.15	15.0
BSM [34]	13.4	7.27	11.4	30.5	6.67	6.52	10.8	32.1	10.5	12.5	24.4	12.8	7.42	16.4	4.88	32.8
ICSG [35]	21.3	24.0	6.93	54.2	12.0	10.4	15.6	29.3	18.4	24.7	26.7	10.7	17.7	23.6	7.73	72.9
SGBM1 [36]	11.3	18.3	7.45	15.7	3.48	29.1	6.51	38.4	5.37	12.8	13.5	3.24	3.44	15.1	3.00	11.1

Table 5 Performance comparison of quantitative evaluation results based on all error from MV3

Method	Avg	Adiron	ArtL	Jadepl	Motor	MotorE	Piano	PianoL	Pipes	Playrm	Playt	PlaytP	Recyc	Shelvs	Teddy	Vintg
Our method	17.6	10.3	22.0	58.8	13.1	12.9	8.66	12.4	24.7	23.5	18.1	11.0	7.13	12.5	11.4	12.6
SPS [32]	16.6	6.51	15.2	40.0	8.35	8.45	12.0	25.0	16.1	25.2	15.7	12.4	8.81	23.7	8.01	53.7
DoGGuided [33]	22.3	20.1	28.0	56.5	13.8	16.8	13.4	37.3	23.8	30.3	30.8	13.0	9.13	19.0	13.4	23.6
BSM [34]	23.5	12.7	28.7	58.7	14.8	14.7	16.0	35.8	24.5	29.4	13.0	20.2	12.1	19.2	14.3	39.3
ICSG [35]	26.9	26.2	17.3	72.9	17.1	14.7	18.8	31.8	28.0	37.4	30.3	12.5	19.0	24.2	11.6	73.7
SGBM1 [36]	18.9	21.1	17.8	38.7	11.0	36.4	11.6	40.0	13.6	25.4	20.0	8.74	5.97	17.6	10.7	18.3

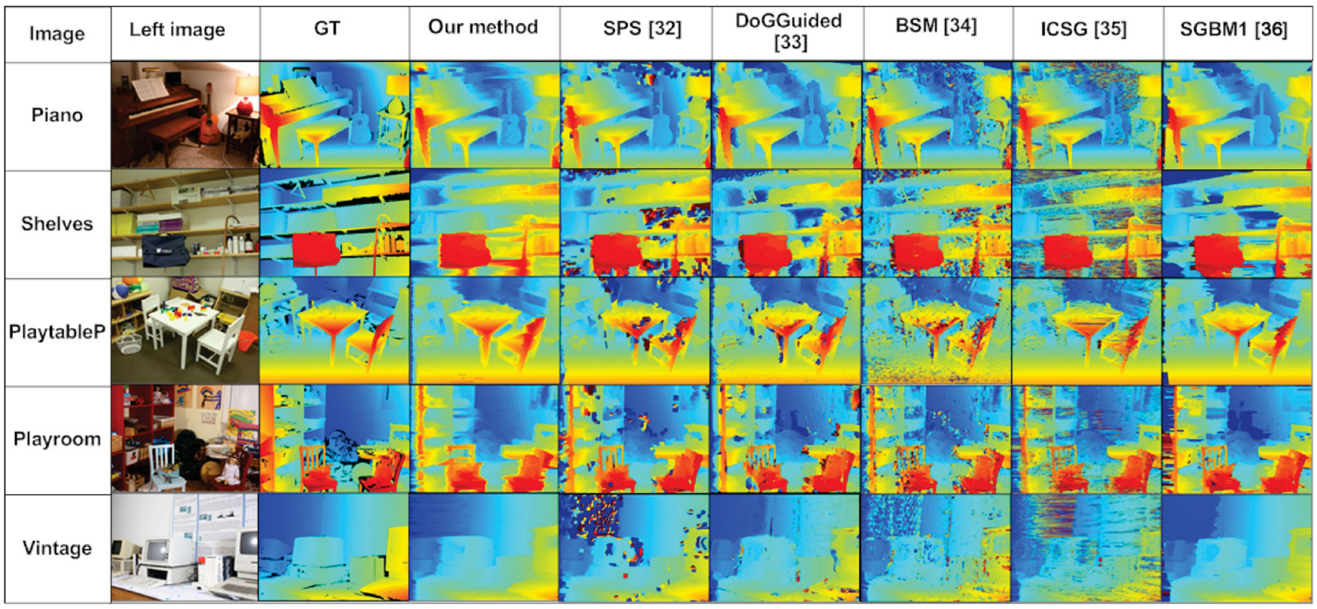


Fig. 3 Disparity maps for the proposed algorithm and other methods on MV3.

hardware as detailed in the following section.

5 CUDA implementation

Realtime stereo matching has become a reality. Until very recently, all realtime implementations made use of GPUs or FPGAs. Our method is based on the calculation of generalized Fourier descriptors. We mentioned previously that the GFD calculation is done for each channel separately, then the final descriptor is calculated by concatenating the results. We can use parallelism to compute descriptors for the three channels simultaneously and consequently reduce the time required to get the final descriptors. On the other hand, dynamic programming allows us to optimize matching between two scan-lines. A CPU implementation performs these steps successively, which is why we seek an appropriate environment for simultaneous processing. We believe that GPU implementation can provide an efficient solution.

5.1 Approach

In a few years, GPUs have become powerful tools for massively parallel intensive computing. They are currently used for several applications including image processing. These applications exploit classical image processing methods implemented on a GPU, typically using a specific language, CUDA, defined by NVIDIA in 2007. There are many predefined functions and libraries for image processing using CUDA language. As our descriptors are based on the Fourier transform, we can exploit the CUFFT library for fast Fourier transform calculation. Our stereo model relies on CUDA for parallel processing, result visualization, and reduction of data transfer costs between CPU memory and GPU memory. This model contains 4 steps:

- **Loading input images**: transfering the stereo pairs from the CPU to GPU memory (host to device).
- **Thread allocation**: fixing the number of threads for the calculation grid so that each thread can perform processing on a pixel template.
- **Parallel processing with CUDA**: executing kernel stereo functions N times using the N threads created in the previous step.
- **Presentation of results**: transfering results from the GPU to CPU memory (device to host).

We start by fixing the number of threads and blocks and loading left and right images into device memory. All processes of our stereo matching algorithm are performed with specific functions, or kernels, that are executed in parallel by multiple threads. For our method, the organization of the kernel functions is presented in Fig. 1. The first step in our algorithm is the calculation of the cost volume $V(x, y, d)$ where x, y indicate the position of the pixel p and d is the disparity value. This volume is obtained for all pixels and for all possible disparity values. It is obtained by matching pixel p to $\bar{p}$ at position $(x+d, y)$ using SGFD defined by Eq. (5). Therefore, the aim of the first kernel function is to calculate $V(x, y, d)$ based on our local cost function. The SGFD is built from Fourier transforms and calculated using CUFFT. This library implements several FFT algorithms for varying types of input and output including C2C (complex input to complex output), R2C (real input to complex output), and C2R (complex input to real output). CUFFT offers highly optimized algorithms to calculate the FFT for different dimensions: 1D, 2D, and 3D. In our approach, we use FFT 2D to compute the FFT of a square window, following Haythem et al. [37].

After the descriptors are obtained to characterize this region, the Euclidean distance is employed to determine the matching cost as indicated in Algorithm 1. We start by loading the input images, fix the window size, and extract the templates from the original images $(I_\mathrm{l}, I_\mathrm{r})$ and gradients $(G_\mathrm{l}, G_\mathrm{r})$. Next, we calculate the generalized Fourier descriptors (GFD) for all templates. We obtain four descriptors: $GFD^\mathrm{l}_\mathrm{c}$ for the left window, denoted *Tmp_left*, $GFD^\mathrm{r}_\mathrm{c}$ for the right window, denoted *Tmp_right*, and $GFD^\mathrm{l}_\mathrm{g}$, $GFD^\mathrm{r}_\mathrm{g}$ for gradient left and right windows denoted respectively by *Tmp_grad_l*, and *Tmp_grad_r*. The Euclidean distance is computed between the descriptors and the final cost function is determined. The aggregated cost volume $CA(p, d)$ in Eq. (9) is easy to calculate using a box filter to average the cost. The next kernel function is dedicated for optimization and calculation of the disparity. The goal of this kernel is defined in Eq. (12). In our work, we follow Congote et al. [16], where the dynamic programming kernel is well described. Before transferring the results to host memory, a last kernel performs postprocessing. The goal of this function is to improve the disparity map by detection of invalid disparity values, to fill them in, and apply a median filter. We start by simple comparison between left disparity and right disparity to identify the unwanted pixels according the condition in Eq. (13). We then replace invalid pixels with valid values from the left or right side as indicated in Eq. (14). Finally a simple median filter is used to reduce the noise and impose smoothing between neighboring pixels.

Algorithm 1 Cost function at disparity value d

```
**Input:** left image $I_\mathrm{l}$, right image $I_\mathrm{r}$, gradient image left $G_\mathrm{l}$, gradient image right $G_\mathrm{r}$, parameter $\alpha$.
**Output:** cost function $SGFD_\mathrm{c}(x, y, d)$.
**for** every pixel $p(x, y)$ **do**
    **for** $p \leftarrow y - w/2$ *to* $y + w/2$ **do**
        **for** $q \leftarrow x - w/2$ *to* $x + w/2$ **do**
            $Tmp\_left\,[p.\omega + q] \leftarrow I_\mathrm{l}\,[y.width + x]$ ;
            $Tmp\_right\,[p.\omega + q] \leftarrow I_\mathrm{r}\,[y.width + x + d]$ ;
            $Tmp\_grad\_l\,[p.\omega + q] \leftarrow G_\mathrm{l}\,[y.width + x]$;
            $Tmp\_gradt\_r\,[p.\omega + q] \leftarrow G_\mathrm{r}\,[y.width + x + d]$;
        **end for**
    **end for**
    $SGFD_\mathrm{c} \leftarrow \mathrm{dist}(GFD(Tmp\_left), GFD(Tmp\_right))$ ;
    $SGFD_\mathrm{g} \leftarrow \mathrm{dist}(GFD(Tmp\_grad\_l), GFD(Tmp\_grad\_r))$;
    $SGFD(p, d) \leftarrow \alpha SGFD_c(p, d) + (1 - \alpha) SGFD_g(p, d)$;
**end for**
```

5.2 Implementation results

The computational complexity of our stereo method and its execution time distribution are now discussed. In practice, the graphics card available was an NVIDIA GeForce GTX960 with Maxwell architecture. It has 1024 CUDA cores running at 1.2 GHz. It is connected to an Intel Core i7-3770M based CPU with a clock speed of 3.4 GHz. We tested our stereo matching implementation on images with resolution 320×240 pixels and 32 disparity levels. Our implementation gives us an execution time of 26.2 ms, with the steps of cost calculation and aggregation taking 76% of the overall runtime. Optimization and disparity filling processes take 18% of the total processing time and the refinement kernel takes 6% of the total runtime.

In order to compare our stereo matching method with other real-time algorithms, we used the same stereo pairs with a resolution of 320×240 pixels and 32 disparity levels. We evaluated the performance of the algorithms based on three important metrics: the number of millions of disparity computations performed per second (MDE/s), the number of the frames per second (FPS), and the average percentage of bad pixel errors across all test images. Results using accuracy and runtime metrics are indicated in Table 6, which provides a comparison between our proposed algorithm and other real-time stereo matching methods. Our implementation achieves 38 frames per second, and is more accurate and faster than DCBGrid [38], and RealTimeGPU [39] based on adaptive cost aggregation and dynamic programming. ReliabilityDP [40], using reliability based dynamic programming, produces less accurate results and is slower than our proposed algorithm. Moreover, our approach gives us almost the same accuracy as that

Table 6 Comparison of accuracy and speed for real-time methods

Method	MDE/s	FPS	Avg % bad pixels
Our approach	93.3	38	8.31
DCBGrid [38]	25.1	10	10.9
RealTimeGPU [39]	52.8	21	9.82
ESAW [41]	194.8	79	8.20
ReliabilityDP [40]	20.0	8	10.70

Table 7 Accuracy and speed for MV3

Method	Avg nonocc	Avg all	Time (s)	Time/MP	Time/GD
Our approach	9.76	17.6	0.27	0.86	10.7
SPS [32]	10.4	16.6	22.1	4.33	14.2
DoGGuided [33]	12.0	22.3	439	1371	9999
BSM [34]	13.4	23.5	196	623	8063
ICSG [35]	21.3	26.9	160	31.9	105
SGBM1 [36]	11.3	18.3	14.3	2.79	8.64

obtained using ESAW [41], although this last method is faster.

Evaluation of our approach on the Middlebury MV3 database requires the calculation of disparity maps for all stereo pairs. We present some results in Fig. 4. The first line indicates the left images of each stereo pair and the second line shows the ground truths for each stereo pair. The last line presents the disparity maps obtained using our approach.

The test on MV3 allows us to place our algorithm in an evaluation table (`http://vision.middlebury.edu/stereo/eval3/`). From it, we extract the most important factors: average absolute errors in non-occluded regions (Avg nonocc), average absolute errors in all regions (Avg all), total time (time), time normalized by number of pixels (s/megapixels, denoted time/MP, and time normalized by number of disparity hypotheses (s/(gigapixels∗ndisp)) denoted time/GD. In Table 7, we compare our approach with other stereo matching algorithms in terms of accuracy and runtime metrics.

The results in Table 7 show that our approach gives average absolute errors in non-occluded regions equal to 9.76% and average absolute errors in all regions equal to 17.6%. The total execution time of our proposed algorithm is 0.27 s. These results indicate that our approach is more accurate and faster than DoGGuided [33], BSM [34], ICSG [35], and SGBM1 [36]. SPS [32] produces more accurate results over all regions (16.6%) but its results are less accurate in non-occluded regions (10.4%). Also, this method is slower than our stereo matching algorithm, with a global execution time of 22.1 s.

6 Conclusions

This paper presents a new cost function for stereo matching based on generalized Fourier descriptors. The cost function for the proposed stereo matching algorithm is Euclidean distance between Fourier descriptors applied to color and gradient images. Cost aggregation, disparity calculation, and result

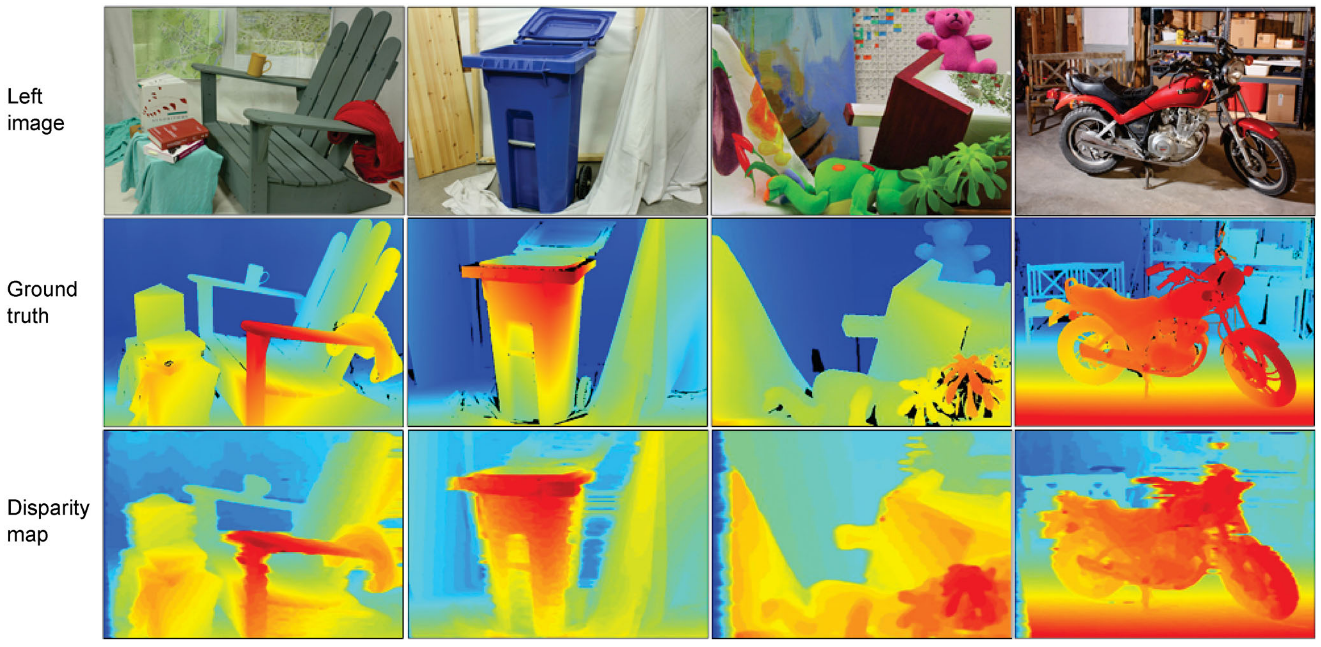


Fig. 4 Some disparity maps obtained for MV3.

refinement are performed respectively using a box filter, dynamic programming, and postprocessing. To evaluate our algorithm, we used the Middlebury stereo benchmark. The experimental results indicate that our proposed method outperforms many stereo matching algorithms including ones based on joint bilateral filtering and dynamic programming, semi global matching and optimized dynamic programming. Also, our proposed approach is more accurate than recent works involving binary stereo matching and stereo matching based on sampled photo-consistency computation.

Furthermore, we have presented an implementation of our approach on graphics hardware using CUDA. This implementation exploits the CUFFT library to compute the cost function and CUDA parallel computing architecture to implement the dynamic programming. Results show that this implementation can reach real-time performance, confirming that it outperforms many real-time algorithms in terms of accuracy and runtime metrics.

References

[1] Scharstein, D.; Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* Vol. 47, Nos. 1–3, 7–42, 2002.

[2] Wang, M.; Zhang, X.-J.; Liang, J.-B.; Zhang, S.-H.; Martin, R. R. Comfort-driven disparity adjustment for stereoscopic video. *Computational Visual Media* Vol. 2, No. 1, 3–17, 2016.

[3] Barnes, C.; Zhang, F.-L. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* Vol. 3, No. 1, 3–20, 2017.

[4] Zhang, F.-L.; Wang, J.; Shechtman, E.; Zhou, Z.-Y.; Shi, J.-X.; Hu, S.-M. PlenoPatch: Patch-based plenoptic image manipulation.*IEEE Transactions on Visualization and Computer Graphics* Vol. 23, No. 5, 1561–1573, 2017.

[5] Brown, M. Z.; Burschka, D.; Hager, G. D. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 25, No. 8, 993–1008, 2003.

[6] Hamzah, R. A.; Ibrahim, H. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors* Vol. 2016, 1–23, 2016.

[7] Bhat, D. N.; Nayar, S. K. Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 20, No. 4, 415–423, 1998.

[8] Lucas, B. D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, 674–679, 1981.

[9] Banks, J.; Corke, P. Quantitative evaluation of matching methods and validity measures for stereo vision. *The International Journal of Robotics Research* Vol. 20, No. 7, 512–532, 2001.

[10] Li, Z. Y.; Song, L. M.; Xi, J. T.; Guo, Q. H.; Zhu, X. J.; Chen, M. L. A stereo matching algorithm based on SIFT feature and homography matrix. *Optoelectronics Letters* Vol. 11, No. 5, 390–394, 2015.

[11] Saygili, G.; van der Maaten, L.; Hendriks, E. A. Improving segment based stereo matching using SURF key points. In: Proceedings of the 19th IEEE International Conference on Image Processing, 2973–2976, 2012.

[12] Gonidis, P.; Kotoulas, L.; Andreadis, I. A new hardware module for stereo matching using Zernike moments. In: Proceedings of the 3rd International Conference on Autonomic and Autonomous Systems, 33, 2007.

[13] Altantawy, D. A.; Obbaya, M.; Kishk, S. A fast non-local based stereo matching algorithm using graph cuts. In: Proceedings of the 9th International Conference on Computer Engineering & Systems, 130–135, 2014.

[14] Yang, Q.; Wang, L.; Yang, R.; Stewénius, H.; Nistér, D. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 31, No. 3, 492–504, 2009.

[15] Veksler, O. Stereo correspondence by dynamic programming on a tree. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, 384–390, 2005.

[16] 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, 231–234, 2009.

[17] Mattoccia, S.; Tombari, F.; di Stefano, L. Stereo vision enabling precise border localization within a scanline optimization framework. In: *Computer Vision—ACCV 2007. Lecture Notes in Computer Science, Vol. 4844.* Yagi, Y.; Kang, S. B.; Kweon, I. S.; Zha, H. Eds. Springer Berlin Heidelberg, 517–527, 2007.

[18] Kordelas, G. A.; Alexiadis, D. S.; Daras, P.; Izquierdo, E. Content-based guided image filtering, weighted semi-global optimization, and efficient disparity refinement for fast and accurate disparity estimation. *IEEE Transactions on Multimedia* Vol. 18, No. 2, 155–170, 2016.

[19] Sabihuddin, S.; Islam, J.; MacLean, W. J. Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array. In: Proceedings of the Canadian Conference on Electrical and Computer Engineering, 1461–1466, 2008.

[20] 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 Transactions on Circuits and Systems for Video Technology* Vol. 23, No. 1, 94–104, 2013.

[21] Zhang, D. S.; Lu, G. J. Shape-based image retrieval using generic Fourier descriptor. *Signal Processing: Image Communication* Vol. 17, 825–848, 2002.

[22] Smach, F.; Lemaître, C.; Gauthier, J.-P.; Miteran, J.; Atri, M. Generalized Fourier descriptors with applications to objects recognition in SVM context. *Journal of Mathematical Imaging and Vision* Vol. 30, No. 1, 43–71, 2008.

[23] Smach, F.; Miteran, J.; Atri, M.; Dubois, J.; Abid, M.; Gauthier, J.-P. An FPGA-based accelerator for Fourier descriptors computing for color object recognition using SVM. *Journal of Real-Time Image Processing* Vol. 2, No. 4, 249–258, 2007.

[24] Yang, Q. Q.; Ji, P.; Li, D. X.; Yao, S. J.; Zhang, M. Fast stereo matching using adaptive guided filtering. *Image and Vision Computing* Vol. 32, No. 3, 202–211, 2014.

[25] Hosni, A.; Rhemann, C.; Bleyer, M.; Rother, C.; Gelautz, M. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 35, No. 2, 504–511, 2013.

[26] Ohta, Y.; Kanade, T. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 7, No. 2, 139–154, 1985.

[27] Psarakis, E. Z.; Evangelidis, G. D. An enhanced correlation-based method for stereo correspondence with subpixel accuracy. In: Proceedings of the 10th IEEE International Conference on Computer Vision, Vol. 1, 907–912, 2005.

[28] Salmen, J.; Schlipsing, M.; Edelbrunner, J.; Hegemann, S.; Lüke, S. Real-time stereo vision: Making more out of dynamic programming. In: *Computer Analysis of Images and Patterns. Lecture Notes in Computer Science, Vol. 5702.* Jiang, X.; Petkov, N. Eds. Springer Berlin Heidelberg, 1096–1103, 2009.

[29] Wang, L.; Yang, R. G.; Gong, M. L.; Liao, M. Real-time stereo using approximated joint bilateral filtering and dynamic programming. *Journal of Real-Time Image Processing* Vol. 9, No. 3, 447–461, 2014.

[30] Martins, J. A.; Rodrigues, J. M. F.; du Buf, H. Luminance, colour, viewpoint and border enhanced disparity energy model. *PLoS One* Vol. 10, No. 6, e0129908, 2015.

[31] Michael, M.; Salmen, J.; Stallkamp, J.; Schlipsing, M. Real-time stereo vision: Optimizing semi-global matching. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 1197–1202, 2013.

[32] LeGendre, C.; Batsos, K.; Mordohai, P. High-resolution stereo matching based on sampled photoconsistency computation. In: Proceedings of the British Machine Vision Conference, 2017.

[33] Kitagawa, M.; Shimizu, I.; Sara, R. High accuracy local stereo matching using DoG scale map. In: Proceedings of the 15th IAPR International Conference on Machine Vision Applications, 258–261, 2017.

[34] Zhang, K.; Li, J.; Li, Y.; Hu, W.; Sun, L.; Yang, S. Binary stereo matching. In: Proceedings of the 21st International Conference on Pattern Recognition, 356–359, 2012.

[35] Shahbazi, M.; Sohn, G.; Théau, J.; Ménard, P. Revisiting intrinsic curves for efficient dense stereo matching. In: Proceedings of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. III-3, 123–130, 2016.

[36] Hirschmuller, H. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 30, No. 2, 328–341, 2008.

[37] Haythem, B.; Mohamed, H.; Marwa, C.; Fatma, S. A. Fast generalized Fourier descriptor for object recognition of image using CUDA. In: Proceedings of the World Symposium on Computer Applications and Research, 1–5, 2014.

[38] Richardt, C.; Orr, D.; Davies, I.; Criminisi, A.; Dodgson, N. A. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In: *Computer Vision—ECCV 2010. Lecture Notes in Computer Science, Vol. 6313.* Daniilidis, K.; Maragos, P.; Paragios, N. Eds. Springer Berlin Heidelberg, 510–523, 2010.

[39] 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 the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission, 798–805, 2006.

[40] Gong, M.; Yang, Y.-H. Near real-time reliable stereo matching using programmable graphics hardware. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, 924–931, 2005.

[41] Yu, W.; Chen, T.; Franchetti, F.; Hoe, J. C. High performance stereo vision designed for massively data parallel platforms. *IEEE Transactions on Circuits and Systems for Video Technology* Vol. 20, No. 11, 1509–1519, 2010.

Mohamed Hallek was born in 1988. He is a Ph.D. student and member of the Laboratory of Electronics and Microelectronics at the Faculty of Sciences Monastir, Tunisia. He obtained his master degree in micro- and nano-electronics in 2012. His fields of interest are pattern recognition and stereo matching algorithms.

Fethi Smach was born in Tunisia in 1976. He received his master degree in computer science from the University of Sfax, Tunisia, in 2003. He is currently finishing his Ph.D. thesis at the University of Burgundy. His fields of interest are algorithms for pattern recognition, classification algorithms, and their real-time implementations.

Mohamed Atri was born in 1971. He received his Ph.D. degree in micro-electronics from the Science Faculty of Monastir, Tunisia, in 2001. He is currently a member of the Laboratory of Electronics and Micro-electronics and professor in the Faculty of Sciences Monastir, Tunisia. His research includes circuit and system design, image processing, network communication, IPs, and SoCs.