Stereo vision methods: from development to the evaluation of disparity maps

Gabriel da Silva Vieira[1*§], Fabrizzio Alphonsus A.M.N. Soares[2*], Gustavo T. Laureano[3*],
Naiane M. de Sousa[4§], Jehymison G. A. Oliveira[5§], Rafael T. Parreira[6*], Júlio C. Ferreira[7§], Ronaldo M. Costa.[8*]
*Federal University of Goiás, *Pixelab Laboratory*. Goiânia - GO, Brazil.
{fabrizzio[2], gustavo[3], ronaldocosta[8]}@inf.ufg.br
§Federal Institute Goiano, *Computer Vision Laboratory*. Urutaí - GO, Brazil.
{gabriel.vieira[1], julio.ferreira[7]}@ifgoiano.edu.br, {naimsousa1[4], jehymes.gil[5], rafaeltp3[6]}@gmail.com

***Abstract*—The challenge in building disparity maps is to determine the best method to calculate consistency among points and use it to approximate the differences between the views. Fortunately, the number of methods to construct disparity maps have increased in the last years. On the other hand, a lot of work needs to be done to evaluate these methods by using different arguments. Besides, it is not clear enough how to build stereo vision algorithms with concepts as reuse and scalability. Thus, we propose a software design architecture to be applied in stereo vision systems. Furthermore, we have implemented disparity methods to perform an evaluation which objective was to determine what cost function better fits in each method analyzed. We conclude that MLMH method with SSD cost function is a good choice to be applied in the scenes we have selected, according to the statistical analysis performed.**

***Index Terms*—stereo vision; software design; disparity methods; disparity costs.**

I. Introduction

Computer vision uses different techniques to give a computer the capability to see the world. To reach this objective mathematical models are proposed to represent the way we see and understand things around us. A specific field of study examines this problem by the use of camera sensors. Usually, a stereo rig is prefered to simulate the human vision. This area is known as stereo vision and it generally uses two bidimensional images to reconstruct the depth coordinate. When this approach is used it is called binocular vision, a kind of specialization from stereo vision.

However, other methodologies use more than two images to achieve the objetive to estimate 3D points. Techniques like these use a multiview or n-view approach and they consider multiple images from a scene to approximate from what we see.

These two types of stereo vision approaches have common processes. They perform a series of steps which are often started with camera calibration, passing througth rectification images, stereo correspondence, triangulation and among others. One of them is very important and it has been investigated by many researchers, which is the disparity calculation.

The disparity term was first introduced in the literature about human vision than in computer science. It was used to describe the diference between features captured by both eyes [1]. In computer vision it is often treated as inverse depth and it means that if two correspoding points (one in each image) has a big disparity between them, then the depth point will be small (closer to the camera), similarly, a small disparity will give a big depth (i.e., a point will be further away from the camera).

The disparity calculation allows us to construct a map which contains the difference between corresponding points. This disparity map, as it is known, shows the disparity for all points which have been considered. Thus, dense disparity corresponding points can be calculate and it can be plotted as an image map. The challenge is to determine the best method to calculate consistency among points and use it to approximate the differences between the views shown in the two, or more, images.

In disparity map processing, the number of calculations increases according to the number of pixels per image. This phenomenon causes the matching problem to be computationally expensive. To reduce this complexity, otimization techniques like *integral images* (or summed area table) and *box-filters* (or moving average) are used to make this process a reasonable time consumption [2]. These techniques are important especially in a correlation matching computation, or area-based matching costs.

Parallel processing is another way to deal with the disparity computational complexity. Improvements in hardware technology have made important advances in the stereo vision field. Thus, real time applications have become more common such as autonomous driving, autonomous robotic navigation and more interactive 3D games. Because of these improvements, the implementation of disparity maps in hardware has grown substantially in recent years [3] [4].

In addition to it, the number of methods to construct disparity maps increases because of the emergence of frameworks to evaluate stereo algorithms, such as the Middlebury platform [1] which has become a standard benchmark for performance evaluation and comparison to other algorithms. Probably, this platform is one of the most famous web frameworks that provides stereo imagery with ground truth, evaluation software, and a rank which shows the most effective algorithms.

We use Middlebury's dataset to reach our objective which is to evaluate disparity methods by using different types of cost functions. We start by selecting methods to be applied and we are concerned with preparing a software design with an architecture prepared to grow, that is scalable.

In previous work, [5], we made a literature review about the approaches, applications and challenges in the construction of depth maps. We revisited these papers but at th

0-7695-6357-0/17/$31.00 ©2017 IEEE
DOI 10.1109/WVC.2017.00030

interested in methods to construct disparity maps. Considering how the quantity of methods is surprisingly high, we have defined a strategy to select them based on a cross reference technique. The idea was to select those papers which provided important observations and somehow have influenced new trends in different papers. Furthermore, we have considered techniques that have been implemented only on a standard CPU, without any other additional processing hardware, and those which use only two images.

The outline of the paper is as follows. In Section II, we first discuss the related works and present techniques they used to construct disparity maps. In Section III, we discuss our architecture software design and present its potential to be enlarged to accommodate a growing amount of work. Finally, we present experimental results and conclusions in Section IV and V, respectively.

II. Related Work

Most stereo vision disparity map algorithms have been implemented using multistage techniques [3]. The taxonomy proposed by Scharstein and Szeliski, [1], is a commonly used framework for this development. This taxonomy was based on the observation that stereo algorithms generally perform four steps: (1) matching cost computation, (2) cost aggregation, (3) disparity selection and (4) disparity refinement. Moreover, a pre-processing stage mainly to compensate for photometric distortions and a post-processing steps to improve the results are sometimes deployed [2].

Stereo correspondence algorithms have a way of measuring the similarity of a refecence image and a target one. Similarity means that pixels that correspond to the same scene point have similar or ideally the same values in the images [6]. A matching cost algorithm, or photo-consistency measure, may use a pixel-based or an area-based function to verify the similarity between images. In the first case, a comparison is done pixel-to-pixel while in the second a support window is used to make a correlation process using the neighborhood of a point fixed in the central position. For both, a common disparity selection otimization is the Winner Takes All (WTA) technique.

Typical pixel-based matching cost include absolute intensity differences (AD) and squared intensity differences (SD). In the case of area-based, Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), Normalized Cross-Correlation (NCC), Rank Transform (RT), Census Transform (CT), Birchfield and Tomasi [7] sampling insensitive pixel dissimilarity (BTSAD and BTSSD), among others, are frequently used.

Area-based algorithms are classified as a local approach. This is because they consider only local information and therefore have a low computational complexity and a short run time [3], as the *fixed windows* approach. Unfortunately, they run into problems. If the support window is small it leads to noisy results, but if it is too large, pixels at different depths near depth discontinuities lead to overblown foreground objects, the *fattening effect* [8]. In spite of their limitations, area-based algorithms are widely adopted in practice and they are the most frequently used algorithms for real applications [2].

On the other hand, the global approach redefines depth estimation as an energy minimization problem which its objective is to find a disparity function that minimizes global energy. A global method can be optimized by algorithms such as graph cuts, belief propagation, cooperative optimization, dynamic programming or scanline optimization [9] [2]. In contrast, global methods frequently use the pixel-based approach to make the disparity map calculation, and so they skip the aggregation step [1].

A matching cost is computed for all disparities under consideration. Thus, for each pixel of the reference image we need to compute the cost between the patch (a rectangular portion of an image) centered at this reference pixel, and all the patches of the target image. It is the same for a pixel-based approach, in this case we can consider a patch as 1×1 support window.

This understanding reveals that a naive evaluation of the matching cost for a set of M relative offsets, using square patches of size r^2, takes $O(Mr^2)$ operations for each patch in the reference image. Thus, for an image with N pixels, the computation of all the matching costs require $O(MNr^2)$ operations [10]. Facciolo et al., [10], have described how the *integral image* can be applied to compute the matching costs with $O(MN)$ operations, regardless of the window size.

Another way to deal with this circumstance is to apply the *box filtering* technique. Its main advantage is speeding computer programs that use support windows by the reuse of previous computation. In this sense, it is also independent of the windows size. Moving average is a common filter that employs this technique, mainly because it is easy to understand and to use [11].

Veksler [12], in her *variable windows* algorithm, has used the integral image to efficiently search the space of windows to find a useful one. This algorithm works by exploring all square windows between some minimum and maximum size for each image row. Thus, it deals with the problem of varying window size by empirical tests, a common problem in fixed window algorithms.

Although Veskler has developed a new window cost, her method requires many user-specified parameters which demands a lot of experimentation. In a different way to deal with the problem, Gerrits and Bekaert [8] has proposed a *segmentation-based* algorithm. It assumes that depth discontinuities occur across color discontinuities and based on that it uses the *mean shift* algorithm to segment the reference image. Then, during the aggregation step, all pixels inside of a window which are out of a segment are considered as outliers. This algorithm uses the moving average in its implementation and like Veskler's method it has many parameters to deal.

Scharstein and Szeliski, [1], have shown a simple way to reach important results even with the use of fixed windows. The *shiftable windows* technique starts with a simple aggregation step and it uses this output to take the minimum matching score across all points in the same neighborhood from a

centered point. This technique recovers object boundaries quite accurately but it fails in textureless areas.

Although the area-based approach is quite used, we need to deal with the problem of selecting the best window size and shape. It is not easy to deduce because each scene demands a different window, or each part of a scene could demand a different one. Thus, it is usually done empirically in both fixed window and adaptive window methods.

Cox et al., [13], have proposed a method to avoid the adaptive windowing problem of area-based correlation methods. In the *maximum likelihood, minimum horizontal* (MLMH) stereo algorithm the total number of horizontal and vertical discontinuties along and between adjacent lines are minimized to preserve border areas. Bobick and Intille, [14] , have followed this idea to propose the *large occlusion* stereo algorithm which deals with the presence of significant occlusion. Both methods use the pixel-based approach and the dynamic programming (DP) to find matches and occlusions simultaneously. Though, they suffer from the *streaking effect*.

III. Software Design

We propose a software design architecture to be used in stereo vision computer programs. This architecture is based on the taxonomy proposed by Scharstein and Szeliski [1] which contains four steps as shown in Fig. 1. We have used this taxonomy to build a flexible and modular software structure that can be extended to aggregate more functions.

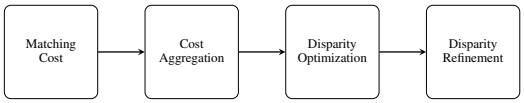


Fig. 1: A taxonomy for stereo vision systems [1].

In our software design, the input images are obtained from a passive camera sensor and we have assumed that these images have passed through a rectification process. Thus, it reduces the search space to one dimension (along the epipolar lines) between two images. As shown in Fig. 2, the region of pixels within a patch in the left image is used to find the closest matching patch in the right image. The disparity is just the horizontal distance between the centers of the patches.

(a) Left Image

(b) Right Image

Fig. 2: A search in epipolar lines. The patch in the Left Image is the origin block and the patch in the Right Image (more left) is the closest matching block.

The main data structure in this architecture is the Disparity Space Image (DSI), a 3D matrix which contains the cost of matching each pixel of the reference and target images. That is, each element $C(x, y, d)$ of the DSI represents the cost of the correspondence between $I_r(x_r, y)$ and $I_t(x_t+d, y)$. In this formulation, (x, y, d) represents the disparity map coordinates, where (x, y) are the coordinates of the pixel of interest and d is the disparity value. Typically, in the matching process, I_r is used as the reference image and I_t represents the target image.

Fig. 3 shows the construction of a DSI. The target image is shifted one by one pixel until the max disparity value. Thus, the disparity cost is stored in a 3D structure and each cost can be retrived by the coordinates from the DSI ($C(x, y, d)$). Considering that the goal of a stereo correspondence algorithm is to produce a univalued function in disparity space, we can use the DSI to find the best correspondences. For example, for each pixel, the corresponding disparity value with the minimum cost is assigned to that pixel (WTA optimization).

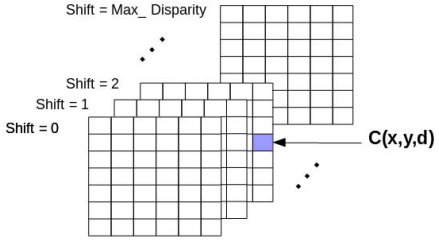


Fig. 3: Disparity Space Image representation.

We propose an architecture with six subsystems: Disparity Cost, Cost Aggregation, Block Matching, Disparity Selection, Subpixel Block Matching and Disparity Methods.

- *Disparity Cost* - is a subsystem that contains the criterions to measure the similarity between two pixels (or patches), such as NCC and SSD.
- *Cost Aggregation* - is used for efficiently performing Block Matching and its input is the output form Disparity Cost. Two components can be part of it, the *integral image* and *box filter* techniques can be used to speed up the aggregation process.
- *Block Matching* - computes the matching costs between patches of two images and it considers the minimum and maximum possible disparities. It uses as input the output from Cost Aggregation subsystem and its output is the DSI structure.
- *Disparity Selection* - is responsible to determine which discrete set of disparities from DSI best represents the scene surface. WTA and DP are representatives of this subsystem and their outputs are the disparity map.
- *Subpixel Block Matching* - extends the Block Matching for integer offsets to consider subpixel disparities which provides an easy way to get better results with little additional computation. It can be used instead of Block Matching subsystem.

- *Disparity Methods* - can incorporate a variety of proposals that seek to construct disparity maps. Local and global methods, pixel and area based approaches, can be easily accommodated to it. Thus, we can reuse the previous components in a fashionable way and we can make scalable stereo vision programs.

Fig. 4 shows a component diagram[1] which represents the proposed software design. This design is composed of six independent and interchangeable modules. The *Block Matching* subsystem allows to make patches of different sizes by exposing an interface to be used. The *Disparity Methods* subsystem uses the required interface provided by *Block Matching* or by *Subpixel Block Matching*. The *Disparity Cost* subsystem exposes an interface which allows to select different types of costs, such as SAD and SSD. After the disparity cost calculation we can aggregate the cost by using an interface provided by the *Cost Aggregation* subsystem. Finally, the outputs from the *Block Matching* and *Subpixel Block Matching* subsystems are the DSI, then we can use the interface provided by *Disparity Selection* subsystem to construct the dispartity map.

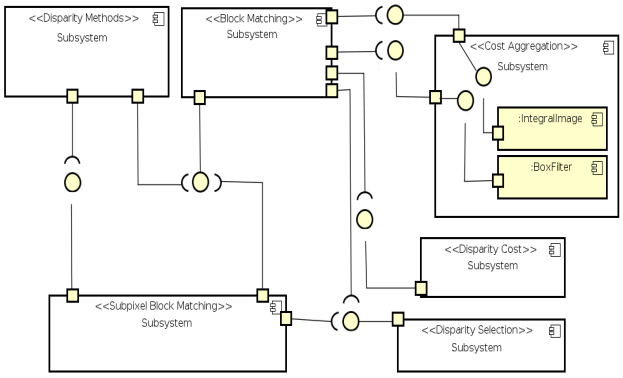


Fig. 4: Software design components.

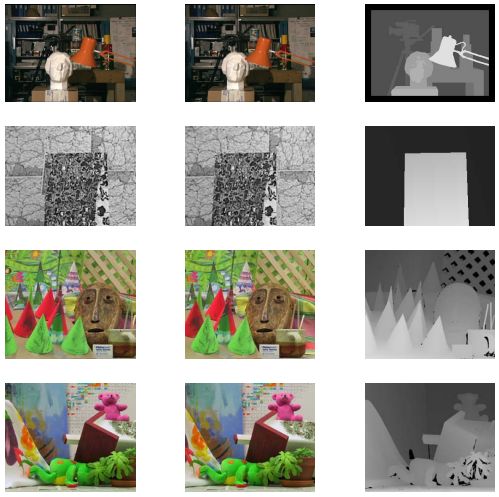
IV. Experimentation and Results

We implemented 10 matching costs and 6 methods that use these costs. We have considered all possible combinations to analyse how each cost and methods deal with the disparity calculation. To select them, we performed a literature review with three constraints: (1) only methods that work on a single stereo image pair, (2) only methods that work on a standard CPU and (3) only methods that the running time is reasonable to be processed, no more than a few seconds.

The literature review gave us some of the widely used stereo matching costs and important methods that have influenced different approaches to construct disparity maps. Because of that, methods that we selected are not new but we used them to observe their behavior in the scenes we have selected.

We selected four image pairs from Middlebury web site: *Tsukuba*, *Map*, *Cones* and *Teddy*, as shown in Fig. 5, respectively. These images are rectified and because of that the parallax movement of a point occurs only along a line. Each image pair has a ground truth which holds the disparities between the reference and target images. Thus, it can be used to evaluate the accuracy of stereo vision methods.

Tables I and II show the disparity methods and disparity cost we have selected, respectively. Methods which have been discussed in Section II were implemented and details of the disparity costs we selected can be found in [10]. We used the software design discussed in Section III to implement them all.



(a) Left Image (b) Right Image (c) Gr. Truth

Fig. 5: Image pairs with Ground Truth.

TABLE I: Disparity Methods

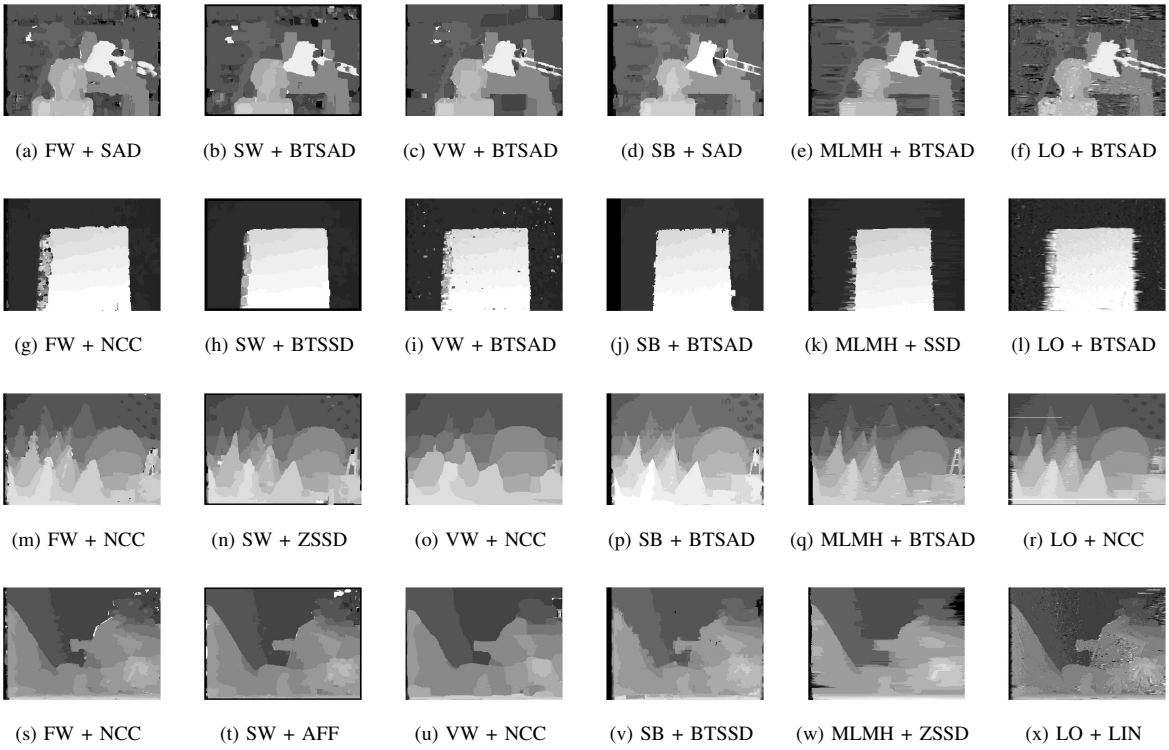
Disparity Method	Algorithm Category	Area/Pixel Based	Optimization	Author
Fixed Windows (FW)	Local	Area	WTA	–
Shiftable Windows (SW)	Local	Area	WTA	[1]
Variable Windows (VW)	Local	Area	DP	[12]
Segmentation Based (SB)	Local	Area	WTA	[8]
MLMH	Global	Pixel	DP	[13]
Large Occlusion (LO)	Global	Pixel	DP	[14]

TABLE II: Disparity Costs

Initials	Description
SAD	Sum of Absolute Differences
SSD	Sum of Squared Differences
ZSAD	Zero-mean SAD
ZSSD	Zero-mean SSD
SSDNorm	SSD Normalized
NCC	Normalized Cross Correlation
AFF	Affine Similarity Measure
LIN	Variant of the AFF cost
BTSAD	Birchfield and Tomasi SAD
BTSSD	Birchfield and Tomasi SSD

The experimentation was made in three steps. In the first step, the images were transformed to grayscale level and the input arguments were defined. We decided to use the same

[1]It is a diagram from UML - Unified Modeling Language.



(a) FW + SAD (b) SW + BTSAD (c) VW + BTSAD (d) SB + SAD (e) MLMH + BTSAD (f) LO + BTSAD

(g) FW + NCC (h) SW + BTSSD (i) VW + BTSAD (j) SB + BTSAD (k) MLMH + SSD (l) LO + BTSAD

(m) FW + NCC (n) SW + ZSSD (o) VW + NCC (p) SB + BTSAD (q) MLMH + BTSAD (r) LO + NCC

(s) FW + NCC (t) SW + AFF (u) VW + NCC (v) SB + BTSSD (w) MLMH + ZSSD (x) LO + LIN

Fig. 6: Stereo vision methods with the lowest RMSE cost function.

arguments which were shown in the original papers when they were provided. In the second step, we put the programs to run. Then, the outputs were compared with the ground truths by measuring the Root Mean Squared Error (RMSE), as shown in Equation 1. In the third step, we performed a quantitative evaluation to find the cost function with minimum error to be applied in a specific method. To do that, we calculated the average between the RMSE from the four image pairs that we used. Furthermore, we have executed a statistical technique, ANOVA (Analysis of Variance), to find a cost function which best fits in all methods.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_i - \bar{c}_i)^2} \quad (1)$$

In Equation 1, $\bar{c}_i$ is the disparity map generated by stereo vision systems, c_i is the ground truth disparity map and N is the number of pixels in the image.

In order to get results with all cost functions and methods, we had to define appropriate test cases. Thus, in methods which are pixel-based we performed the correlation process by using a 3×3 fixed window. This approach was important to guarantee that cost functions like NCC and AFF kept working even in a pixel-based method. Besides that, we have not applied post-processing steps to improve the results. However, in MLMH algorithm we performed a *nearest interpolation* in the generated disparity maps. It was necessary to complete holes caused by occlusions, after that we got better results to be analyzed.

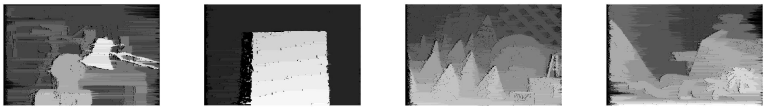
All methods were combined with all cost functions. Thus, each of the methods were run ten times for each of the four scenes we used. Considering that we have 6 methods, 10 cost function and 4 scenes, we made a script that runs 240 times.

Fig. 6 shows stereo vision methods combined with cost function that returned the lowest RMSE. We can observe that Fixed Windows with NCC is better to *Map*, *Cones* and *Teddy*, while SAD is better for *Tsukuba*. Besides, we can observe that the cost function most used in MLMH method is the BTSAD which means it should be a good choice to be applied in this method.

However, to confirm that, we performed another type of metric to analyze the cost function with less error that could be used on the methods that we implemented. Thus, we performed the average between the results which were brought by RMSE.

Table III shows these averages and the minimum errors are in boldface. We can see that NCC cost is the best for Fixed Windows and BTSAD is the best for Variable Windows and Segmentation-Based methods, while LIN is the best for Large Occlusion method. We can also observe that SSD cost is the best for MLMH instead of BTSAD. In addition, MLMH method provided the smallest mean error despite of being the oldest method which we have implemented.

Furthermore, we can observe that cost functions as ZSAD and SSDNorm at no point gave the smaller error. Because of



(a) Tsukuba (b) Map (c) Cones (d) Teddy

Fig. 7: Subpixel Block Matching for MLMH + SSD.

that, in an analytical sense, we can observe that these costs are not appropriate for scenes we have used.

TABLE III: Average between RMSE results

	FW	SW	VW	SB	MLMH	LO
SAD	3.2179	3.4870	3.0927	2.7236	3.5440	3.2506
SSD	3.2568	3.4265	3.3501	2.8045	**2.4158**	3.2841
ZSAD	3.4024	3.4791	4.1127	2.9200	3.0776	3.2518
ZSSD	3.2568	3.4542	3.4532	2.8045	2.7305	3.2842
SSDNorm	3.3117	3.4333	3.0361	2.8996	2.7239	6.6750
NCC	**3.0915**	3.5272	3.3735	7.3480	2.7368	3.7858
AFF	3.2972	3.4095	3.2721	6.2262	9.0788	3.8233
LIN	3.2974	3.4087	3.2033	2.9188	2.4858	**3.1268**
BTSAD	3.2001	3.3940	**2.9355**	**2.7096**	2.4316	3.2163
BTSSD	3.2209	**3.3449**	3.1981	2.8171	2.4407	3.2680

We applied ANOVA technique in two approaches. ANOVA *one way* was performed to observe what cost function could be better applied in a specific disparity method. ANOVA *two way* was performed to observe what combination between method and cost function could better fit to the scenes. For both, we considered a margin of error of 0.01 and because we have four scenes we considered four repetitions. With the results of ANOVA, we performed the Tukey's test to analyze possible representative groups.

Considering these statistical techniques we made some interpretations. First, the worst cost function to MLMH method was AFF and a group of costs formed by SSD, LIN, BTSAD and BTSSD appeared as the best costs for this method. Second, AFF has appeared over again as the worst cost function, as well as NCC, but at this time for Segmentation-Based method. Third, the data were insufficient to get the best combination that could be applied in all scenes. This interpretation is due to the fact that the average between methods is almost the same, so the tests did not efficiently separate groups.

Taking this results into account, we performed again the MLMH method combined with SSD cost to all scenes. At this time we considered a subpixel offset, that is, instead of moving the images one pixel at a time we moved each pixel $1/n$, where n was equal to 8. We did not apply either the *nearest interpolation* and aggregation process (except for *Map* that was applied a 3×3 fixed window). The result is shown in Fig. 7.

V. Conclusion

Stereo vision methods can be efficiently implemented with the use of techniques as *integral image* and *box filters*. But it is not enough to build an scalable architecture. Thus, components need to be modeled to guarantee reuse and the growing of the system. Based on a classical taxonomy, we have proposed a software design to be applied in the development of disparity maps. Moreover, we selected and implemented methods and cost functions that is commonly used in stereo vision field.

Considering these implementations and the quantitative evaluation that we performed, we can conclude that the MLMH is an important method to be applied in the scenes which we selected, specialy when it is used in combination with SSD cost function.

Acknowledgment

The authors would like to thank the Federal Institute Goiano for its financial support.

References

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[2] S. Mattoccia, "Stereo vision: Algorithms and applications," 2013. [Online]. Available: http://www.vision.deis.unibo.it/smatt/stereo.htm

[3] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, 2016.

[4] S. Mattoccia, "Stereo vision algorithms for fpgas," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 636–641.

[5] G. Vieira, F. Soares, R. Parreira, G. Laureano, and R. Costa, "Depth map production: approaches, challenges and applications," in *Proceedings of XII Workshop de Visão Computacional*, 2016, pp. 323–328.

[6] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[7] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401–406, 1998.

[8] M. Gerrits and P. Bekaert, "Local stereo matching with segmentation-based outlier rejection," in *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*. IEEE, 2006, pp. 66–66.

[9] B. Böggemann, "Stereo deph estimation from video sequence," 2010.

[10] G. Facciolo, N. Limare, and E. Meinhardt-Llopis, "Integral images for block matching," *Image Processing On Line*, vol. 4, pp. 344–369, 2014.

[11] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.

[12] O. Veksler, "Fast variable window for stereo correspondence using integral images," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–I.

[13] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs, "A maximum likelihood stereo algorithm," *Computer vision and image understanding*, vol. 63, no. 3, pp. 542–567, 1996.

[14] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *International Journal of Computer Vision*, vol. 33, no. 3, pp. 181–200, 1999.