Optimal Algorithms for Stereo Correspondence Estimation

By

Md. Abdul Mannan Mondal

Registration No.	:	74 (Original)
Session	:	2010-2011(Original)
Registration No.	:	159 (Re-registration)
Session	:	2016-2017

Supervised by
Professor Dr. Md. Haider Ali

Submitted to the Department of Computer Science and Engineering of the Faculty of Engineering and Technology in the University of Dhaka for partial fulfillment of the requirements of the degree of Doctor of Philosophy (Ph.D.)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERISTY OF DHAKA, BANGLADESH
September 2023

Approval

As the candidate’s supervisor, I have approved this dissertation for submission.

Name: Prof. Dr. Md. Haider Ali

Signature:
………………………………………

Declaration

We declare that this thesis entitled "**Optimal Algorithms for Stereo Correspondence Estimation**" presented in it are performed by me under the supervision of Prof. Dr. Md. Haider Ali, Department of Computer science and Engineering, University of Dhaka, Dhaka-1000. We have confirmed that the full part of the work is done during this Ph.D. research work in the University of Dhaka. We have also declared that any part of this thesis has not previously been submitted for an award of any degree or other qualification. We have discussed related published works of others with appropriate references and this thesis work is done entirely by us and our contributions and enhancements from other works are clearly stated.

Candidate:

(Md. Abdul Mannan Mondal)

Signature:
…………………………………………

Supervisor:

(Prof. Dr. Md. Haider Ali)

Signature:
…………………………………………

List of Publications

[1] Md. Abdul Mannan Mondal and Mohammad Haider Ali, "**Self-Guided Stereo Correspondence Estimation Algorithm**," *International Journal of Image and Graphics (IJIG),* Vol. 21, No. 3, pp. 21500281-215002826, January 2021, © World Scientific Publishing Company, USA. Available at: https://www.worldscientific.com/worldscinet/ijig , https://doi.org/10.1142/S0219467821500285

[2] Md. Abdul Mannan Mondal and Mohammad Haider Ali,**"Disparity of Stereo Images by Self-Adaptive Algorithm,"** *International Journal of Advanced Computer Science and Applications(IJACSA)* Vol. 11, No. 5, pp. 441-454, May 2020. DOI: 10.14569/IJACSA.2020.0110558

Available at: https://thesai.org/Publications/ViewPaper?Volume=11&Issue=5&Code=IJACSA&SerialNo=58

[3] Md. Abdul Mannan Mondal and Mohammad Haider Ali, "**Stereo Correspondence Estimation by Two Dimensional Real Time Spiral Search Algorithm,**" *International Journal of Engineering and Advanced Technology(IJEAT)* Vol.9, No. 5, pp.96-103, June 2020. DOI: 10.35940/ijeat.D8592.069520.

Available at: https://www.ijeat.org/wp-content/uploads/papers/v9i5/D8592049420.pdf

[4] Md. Abdul Mannan Mondal and Md. Haider Ali**, "Disparity Estimation by a Real Time Approximation Algorithm,**" *International Journal of Image* Processing *(IJIP),* vol. 10 Issue: 03, pp: 126-134, July 2016. Available at: http://www.cscjournals.org/library/manuscriptinfo.php?mc=IJIP-1076

[5] Md. Abdul Mannan Mondal and Md. Haider Ali, **"Performance Review of the Stereo Matching Algorithms,"** *American Journal of Computer Science and Information Engineering,* vol.04, Issue: 01, pp: 7-15, June 2017. Available at: http://www.aascit.org/journal/ajcsie

[6] Md. Abdul Mannan Mondal and Md. Haider Ali ,**"On Stereo Correspondence Estimation: A Spiral Search Algorithm,"** in *Proceeding of International Conference on Signal and Information Processing (ICSIP),* December 2010, pp: 204-207. Available at: https://doi.org/10.1117/12.913526

[7] Md.Abdul Mannan Mondal and Md. Haider Ali ,**"Disparity Estimation by Reverse Fuzzyfication,"** in *Proceeding of International Conference on Signal and Information Processing (ICSIP),* December 2010. pp: 234-237. DOI : 10.1117/12.913533

ABSTRACT

Stereo correspondence has attained a position of overwhelming dominance in Computer Vision for long days for determining three-dimensional depth information of objects using a pair of left and right images from a stereo camera system. In this thesis we propose four novel ideas for improving the efficiency and accuracy of stereo correspondence estimation in stereo vision. First idea presents a ***"Real Time Approximation (RTA)"*** algorithm for computing the disparity of the stereo image sequences. The algorithm has been organized to make it dedicated for real time-applications. To do this, the original image is scaled down and obtained highest speed to compute the stereo correspondences. The second idea is a searching algorithm titled ***"Two Dimensional Real Time Spiral Search Algorithm (2DRTSSA)"*** to compute the stereo correspondence two dimensionally. The 2DRTSSA thus increases the speed and accuracy over the existing state-of-the-art methods of one dimensional and left-right searching strategy. The third idea is a new and significant searching method, is explored by the name ***"Self-Adaptive Algorithm (SAA)"*** for computing stereo correspondence or disparity of stereo image. According to the SAA method, stereo matching search range can be selected dynamically until finding the best match. The searching speed is almost doubled by reducing the search range half of its original, by dividing the searching range into two regions. First one is $-d_{max}$ to 0 and second one is 0 to $+d_{max}$. To determine the correspondence of a pixel of the reference image (left image), the window costs of the right image are computed either for $-d_{max}$ to 0 region or for 0 to $+d_{max}$ region depending on the result of previous matching. The speed and accuracy are further improved by introducing the fourth idea entitled ***"Self-Guided Stereo Correspondence (SGSC) Estimation"*** algorithm. The SGSC algorithm is directed by photometric properties of the candidate-pixels. Searching performance is slightly improved by utilizing this photometric property of the candidate-pixels as well as by implanting the pioneer threshold technique. These two key techniques reduced the computational costs with further improvement of accuracy. The achievements of the SGSC method are testified on *Middlebury standard stereo datasets* of 2001, 2003, 2006 and *Middlebury latest Optical Flow Datasets*. Moreover, the newly invented algorithms ***RTA***, ***2DRTSSA, SAA and SGSC*** have been justified on real images which are acquisitioned in our laboratory in complex environment. The overall performances of all algorithms are satisfactory in case of real stereo images. Finally, the proposed methods are compared with present state-of-the-art methods and our 2DRTSSA, SAA and SGSC outperforms the latest methods in terms of speed, visualization of hidden ground truth, 3D reconstruction and accuracy.

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my revered supervisor Dr. Md. Haider Ali, Professor, Dept. of Computer Science and Engineering, University of Dhaka for his constant inspiration, encouragement, supervision and understanding throughout the research. I would like to express immense gratitude to Dr. Md. Hasanuzzaman, Professor, Dept. of Computer Science and Engineering, University of Dhaka for his valuable suggestions and constant inspiration. I am grateful to Dr. Saifuddin Md. Tareek, Professor and Chairman, Dept. of Computer Science and Engineering, University of Dhaka for his pragmatic suggestions and immense sense of encouragement and helps during this work. I am grateful to Dr. Shabbir Ahmed and Dr. Md. Mustafizur Rahman, Professors, Dept. of Computer Science and Engineering, University of Dhaka. I would like to give thanks to all of my honorable teachers of the Dept. of Computer Science and Engineering in the University of Dhaka, for their invaluable comments and suggestions. I am also indebted to all the staffs of the Dept. of Computer Science and Engineering, University of Dhaka for their help. I would also like to present my humble thanks to Middlebury Vision Benchmark for providing the standard stereo images for this research work. Warm thanks to all my friends, who were with me all the time for their effective appreciations regarding my research.

.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CA	Cellular Automata
CWT	Continuous Wavelet Transform
DP	Dynamic Programming
DSI	Disparity Space Image
2DRTSSA	Two Dimensional Real Time Spiral Search Algorithm
DSG	Deep Self-Guided
EGF	Edge-aware Geodesic Filter
EM	Energy Minimization
FAB	Fast Area Based
FA	Fast Algorithm
FAS	Fusing Adaptive Support
GT	Ground Truth
HD	Hierarchical Disparity
NURBS	Non-Uniform Rational B-Splines
NCT	Normalized Correlation Technique
PSM	Pyramid Stereo Matching
RTA	Real Time Approximation
SAD	Sum of Absolute Difference
SSD	Sum of Square Difference
SMP	Single Matching Phase
SAA	Self-Adaptive Algorithm
SGSC	Self- Guided Stereo Correspondence
TSA	Two Stage Approximation
TF	Tree Filtering
WC/W_c	Window Cost

Chapter 1

Introduction

1.1 Disparity or Stereo Correspondence Estimation and Rudiments

Stereo correspondence or disparity is one of the most important research areas in computer vision. It has been found in many applications e.g., industrial inspection for 3D objects, 3D sensing, 3D growth monitoring, Z-keying, novel view synthesis, image-based rendering, autonomous vehicles and robotics. The applications of stereo correspondence are found in medical-biomedical and/or bioengineering fields. It can be also used in surveillance, transportation for traffic scene analysis, digital photogrammetry, remote sensing, 3D database for urban and town planning, stereo lithography, stereo sculpting, on-line shopping and many more. Dense depth measurements obtained from disparity are required in applications such as teleconferencing, robot navigation and control, exploration and modeling of unstructured environments, virtual reality etc.

The reference pixel $\mathbf{P}_L(x_L, y_L)$ of left image as shown in **Fig. 1.1** should be matched in the same co-ordinate position of the right image. But practically, the matching pixel $\mathbf{P}_R(x_R, y_R)$ in the right image is found in earlier or later position due to noise or different camera-position. Usually the deviation occurs along the x axis. The difference in the coordinate of the corresponding pixels from left and right images is known as disparity, which is inversely proportional to the distance of the object from the camera [1].

Disparity, $$d = x_L - x_R = \frac{Bf}{Z} \qquad (1.1)$$

Where f is the focal length of the cameras, B is baseline distance between two identical cameras and Z is the distance from the object $P(x, y, z)$ to the camera. By knowing the parameters of right side of equation (1.1), we can also measure the disparity d. This is the main rudiments of stereo corresponding or disparity.

For of a pixel in the left image, its correspondence has to be searched in the right image based on epipolar line for 3D scene reconstruction. Stereo correspondence or disparity is conventionally determined from the pixels of the matching windows by using Sum of Square Differences (SSD), Sum of Absolute Differences (SAD) or Normalized Correlation Techniques (NCT). Another rudiment is the cost aggression process using SSD, SAD or NCT. The aggregation of the window cost using SSD, SAD or NCT functions, leads to the score of most of the stereo vision methods, which can be mathematically expressed as follows-

$$SSD(x,y,d) = \sum_{i=1}^{Wx}\sum_{j=1}^{Wy}\left[fL(x+i,y+j) - fR(x+i+d,y+j)\right]^2 \tag{1.2}$$

$$SAD(x,y,d) = \sum_{i=1}^{Wx}\sum_{j=1}^{Wy}\left| fL(x+i,y+j) - fR(x+i+d,y+j) \right| \tag{1.3}$$

$$\text{Correlation method NCT } (x,y,d) = \frac{\sum_{i=1}^{Wx}\sum_{j=1}^{Wy} fL(x+i,y+j)\, fR(x+i+d,y+j)}{\sqrt{\sum_{i=1}^{Wx}\sum_{j=1}^{Wy} fL^2(x+i,y+j)\sum_{i=1}^{Wx}\sum_{j=1}^{Wy} fR^2(x+i+d,y+j)}} \tag{1.4}$$

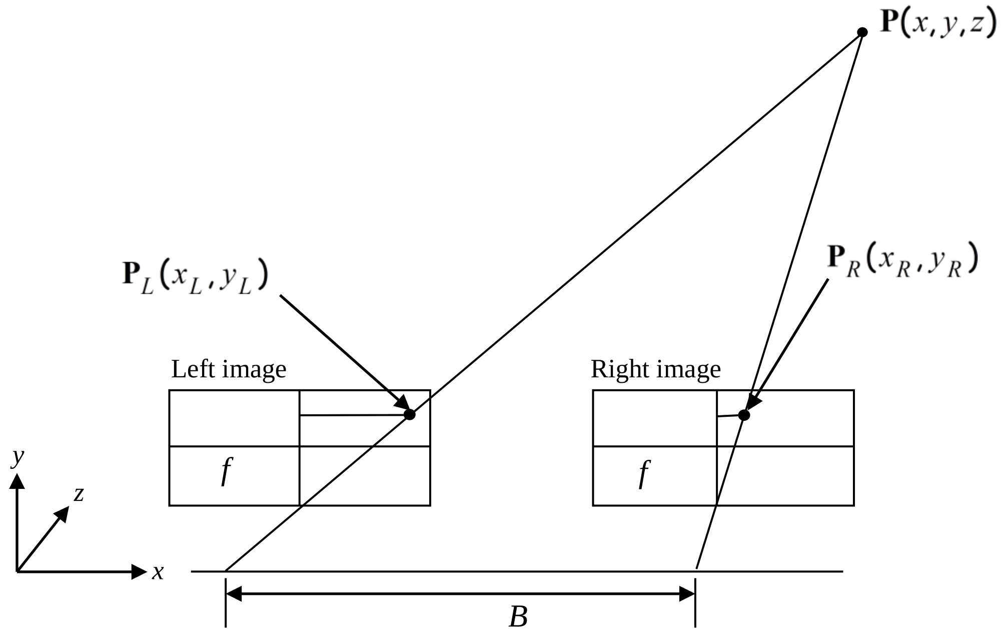


Fig. 1.1 Disparity in a stereo pair of images.

where *fL, fR* are the intensity values in left and right image, (*x, y*) are the pixel's coordinates, *d* is the disparity value under consideration and *W* is the window cost of masking region[9].

Window-based stereo correspondence estimation technique is widely used due to its efficiency and ease of implementation. However, there is a well-known problem in the selection of an appropriate size and shape of window [2-3]. If the window is small and does not cover enough intensity variation, it gives inaccurate result due to low signal to noise ratio. On the other hand, if window is large, it includes a region where the disparity varies or discontinuity of disparity happens, then the result becomes erroneous due to different projective distortions in the left and right images. Pixels those are close to a disparity discontinuity require windows of different shapes to avoid crossing the discontinuity. Therefore, different pixels in an image require windows of different shapes and sizes. To overcome this problem, many researchers proposed adaptive window techniques using windows of different shapes and sizes [4-7]. In adaptive window technique, it requires comparing the window costs for different window sizes and shapes, so the computation time is relatively higher than that of fixed window-based technique. For example, in [6] and [7] the authors used a direct search over several window shapes to find that one, which gives the best window cost. Beside gray scale stereo images, the use of color stereo images brings a substantial gain in accuracy with the expense of computation time [8].

New approaches are introduced every year. But none of them are still now perfect for stereo matching algorithm in real sense. Some special applications, like autonomous vehicle and robot navigation, virtual reality and stereo image coding in 3D-TV, require a very fast estimation of dense stereo correspondence. It was aimed that the pruning technique was useful in such situations for speedy determination of dense disparity [9].

The Two-Stage Approximation (TSA) method showed in such situations for speedy determination of dense disparity [10]. But its accuracy as well as visual quality of dense map was very poor. At present the researcher trying to pursue real-time execution speed and better accuracy.

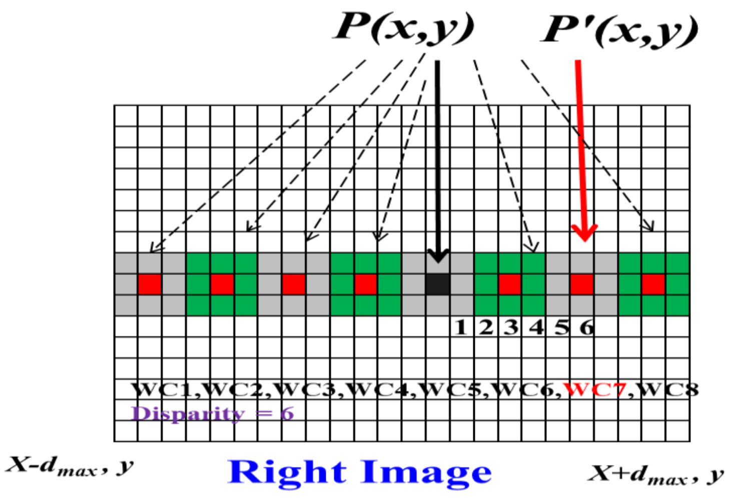


Fig. 1.2 Graphical illustration of window cost calculation.

To determine the correspondence of a pixel in the left image using equation (1.2), (1.3) or (1.4) is computed window costs (WC) for all candidate-pixels in the right image within the search range. The computational window costs are presented by **Fig. 1.2** within the search range from $-d_{\max}$ to $+d_{\max}$ along the x axis. Only eight window costs (WC1…WC8) are shown within the specified search range for simplicity and better understanding, practically it is more than this. The pixel in the right image that gives the best window cost is considered as the corresponding pixel in the left image. Suppose the reference pixel $P(x, y)$ in the left image is matched with the pixel $P'(x, y)$ in right image. Therefore, the minimum window cost is finalized at WC7. Since the difference from original position to matching position is six (6) pixels, therefore the disparity is 6.

A constraint in the stereo matching is that the corresponding pixels should be close in color or intensity [9-10]. Based on this constraint, we proposed and implemented four new methods in this research work. All the methods are original, where it is not necessary to compute the window costs of all candidate-pixels in the right image within the search range. A problem with window-based matching is that, the window size must be large enough to include enough intensity variation for matching but small enough to avoid the effects of projective distortion.

In this chapter, comparative performance analysis of existing stereo matching algorithms is explored in details. All stereo matching algorithms are classified into two categories [11-12]. First one is named as local method while the second one is global method. The algorithms taken consideration in literature are analyzed by frame-rate, accuracy and disparity range. Experimental results applied on different image sizes and different image sets (Tsukuba Stereo pair, Sawtooth stereo pair, Map Stereo pair and Venus Stereo pair) are also presented. Some neural network and automata based latest algorithms are analyzed. Besides these, some algorithms are not fallen into above mentioned categories are also discussed in details within the literature review section. Critical analyses of recent related works are discussed in last portion of literature review section.

1.2 Literature Review

Many researchers worked on a dense two-frame stereo in many ways. They try to optimize the dense disparity in locally or globally on a stereo pair. So, the dense matching algorithms are divided into local and global ones [11]. The best classifications have been presented by Scharstein and Szeliski [12] and many new methods have been proposed here. Local methods are also known as area based stereo matching that can perform better frame-rate compare to global methods. According to this, disparity is being calculated at a point in a fixed window. Global methods are also known as intensity or energy based stereo matching that can perform better accuracy compare to local methods. In this method, the global cost function is reduced to minimum as possible. This cost function synthesizes image data and smoothness terms. Because of increasing the computational power, some algorithms that results dense map became very popular in the recent decade. That is why dense disparity is more interested research area than spare disparity results.

1.2.1 Local Methods

Local methods provide good results and show speedy performance. Disparity has been calculated from color stereo images [8]. Sum of Absolute Difference (SAD) technique is used for RGB color image and a fast median filter uses to result in [8]. Its scanning frame-rate is 20 *fps* for 160 × 120 image size. The method is suitable for real-time application. A Fast Area Based (FAB) stereo matching algorithm has been introduced by L. D. Stefano et al. [13]. As it is a unidirectional searching, it is also referred as Single Matching Phase (SMP). Based on uniqueness constraint, it rejects previous matches as soon as better result is detected. It also uses SAD technique for error function, but any technique could be used. This method results a dense disparity map in real-time. It performs 39.59 *fps* frame-rate for 320 × 240 image size and 16 disparity levels and the root mean square (r.m.s.) error for Tsukuba pair is 5.77.

Shaped based stereo matching is reported in [14], where shape of the target is depicted by the algorithm. It demonstrates the importance of the horizontal and vertical slanted surfaces. The authors propose the replacement of the standard uniqueness constrain referred to pixels with a uniqueness constraint referred to line segments along a scanline. In this method interval matching is performed instead of pixel matching. Matching factor is performed based on the absolute intensity difference and the stretching factor is obtained. The object is also achieved by minimum segmentation. The experimental results show that 1.77%, 0.61%,3.00% and 7.63% errors for the Tsukuba , Sawtooth, Venus and Map stereo pair respectively. The execution speed of the algorithm varies from 1 to 5 *seconds* on 2.4 GHz processor.

Almost real-time performance method is reported in [15] presented by Yoon et al. It uses SAD method and a left-right consistency check. This method is able to find out the errors in the problematic regions are reduced using different sized correlation windows. Accordingly, a median filter is used in order to interpolate the results. The algorithm can process 7 *fps* for 320 × 240 pixels images and 32 disparity levels. The result has been justified by using an Intel Pentium 4 at 2.66 GHz Processor.

Table 1.1: Comparative study of local algorithms of earlier trends.

Author's Name	**Published Year**	**Method**	**Frame-rate (in *fps*)**	**Image Size**	**Disparity Levels**	**Computational Platform**
Muhlmann et al.[8]	2002	SAD	20	160×120	N/A	Processor:P3 Speed:800MHz RAM: 512 MB
Di Stefano et al.[13]	2004	SAD	39.59	320×240	16	Processor:P3 Speed:800MHz RAM: 512 MB
Yoon et al.[15]	2005	SAD	7	320 ×240	32	Intel Pentium 4 2.66GHz
Ogale and Aloimonos[21]	2005	SAD	1	384 ×288	16	Processor:P3 Speed:2.4 GHz
Binaghi et al.[26]	2004	ZNCC	0.024	284× 216	30	Processor:P3 Speed:300MHz
Yoon and Kweon[27]	2006	SAD	0.016	384× 288	16	AMD 2700+
Zach, Karner and Bischof[28]	2004	SAD	50	256 × 256	88	ATI Radeon 9700 Pro
Mordohai and Medioni[48]	2006	NCC	0.002	384 ×288	20	Intel Pentium 2.8MHz

The use of Cellular Automata (CA) is presented in [16]. This work presents architecture for real-time extraction of disparity maps. The proposed method can process 1Mpixels image pairs at more than 40 *fps*. The key idea behind the algorithm relies on matching pixels of each scan-line using a one-dimensional window and the SAD matching cost. According to the method a pre-processing mean filtering step and a post-processing CA based filtering one are employed. CA's are models of physical systems, where space and time are discrete and interactions are local. They can easily handle complicated boundary and initial conditions. In CA analysis, physical processes and systems are described by a cell array and a local rule, which defines the new state of a cell depending on the states of its neighbors [17].

A window-based method is presented in [18] that use different support-weights. The support-weights of the pixels in a given support window are adjusted based on geometric proximity and color similarity to reduce the image ambiguity. The frame-rate for the Tsukuba image pair with a 35 × 35 pixels support window is about 0.016 *fps* on an AMD 2700+ processor. The error ratio is 1.29%, 0.97%, 0.99%, and 1.13% for the Tsukuba, Sawtooth, Venus and Map image sets respectively. The experimental results can be further improved through a left-right consistency checking. A novel method has been introduced in [26]. This method uses zero mean normalized cross correlation for matching, it also uses neural model that uses least-mean-square delta rule for training. Proper window size and shapes are selected by the neural network for each considering region. The results obtained by the network are better but the computational costs are not suitable for real-time applications.

1.2.2 Global Methods

In a global algorithm, the disparity of every single pixel is calculated by taking into consideration the whole image. Global optimization methodologies involve segmentation of the input images according to their colors. The accuracy of the global methods is very high but the computational costs are also high due to repetitive comparisons.

The research work presented in [18] based on unified framework that supports the fusion of any partial knowledge such as matching features and surfaces about disparities. Accordingly, it combines the results of edge, corner and dense stereo matching algorithm to act as a guide points to the standard dynamic programming method. The result obtained by fully automatic dense stereo system is up to four times faster and greater accuracy compared to that obtained by using dynamic programming. A method based on the Bayesian Estimation theory with a prior Markov Random Fields model for the assigned disparities is described in [19]. According to this method, the continuity, coherence, occlusion constraints and the adjacency principles are taken into consideration. The optimal estimator is computed using a Gauss-Markov random field model for the corresponding posterior marginal, which results in a diffusion process in the probability space. The results are accurate but the algorithm is not suitable for real-time applications, since it needs a few minutes to process a 256 × 255 stereo pair with up to 32 disparity levels, on an Intel Pentium III running at 450 MHz. Image color segmentation is reported in [20]. By this method disparity map is calculated using an adapting window-based technique. The segments are combined in larger layers iteratively. A global cost function is used to optimize the segments to layers. The quality of the disparity map is measured by warping the reference image to the second view and comparing it with the real image and calculating the color dissimilarity. For the 384 × 288 pixel of Tsukuba and the 434 × 383 pixel of Venus test set, the algorithm produces results at 0.05 *fps* frame-rate and needed 20 *seconds* to produce results. For the 450 × 375 pixel Teddy image pair, the running frame-rate decreased to 0.01 *fps* due to the increased scene complexity. Running speeds refer to an Intel Pentium 4, 2.0 GHz processor. The root mean square error was 0.73 for the Tsukuba, 0.31 for the Venus and 1.07 for the Teddy image pair.

Table 1.2: Comparative study of global algorithms of earlier trends.

Author's Name	**Published Year**	**Applied Method**	**Frame-rate (in *fps*)**	**Image Size**	**Disparity Levels**	**Computational Platform**
Gutierrez and Marroquin[19]	2004	Gauss-Markov random field	0.017	256×255	32	Processor:P3 Speed:450MHz
Bleyer and Gelautz[20]	2005	Global cost function	0.05	384×288	16	Processor:P4 Speed:2.0 GHz
Ogale and Aloimonos [21]	2005	Left-right diffusion process	0.5	384×288	16	Intel Pentium 4 Speed: 2 GHz
Veksler et al.[49]	2006	Graph cuts	1.04	434×383	20	Processor:P4 Speed:2.6 GHz
Hong and Chen[50]	2004	Graph cuts	0.33	384×288	16	Processor:P4 Speed:2.4 GHz
Kim et al.[51]	2005	DP	0.23	384×288	16	Intel Pentium 4 Speed: 2 GHz
Wang et al.[52]	2006	DP	43.5	320×240	16	3.0GHz CPU – ATI Radeon XL1800 GPU
Lei et al.[53]	2006	DP	0.1	384×288	16	1.4 GHz Intel Pentium M

An algorithm which is focused on achieving contrast invariant stereo matching [21]. It depends on multiple spatial frequency channels for local matching. The global solution is determined by a fast non-iterative left-right diffusion process. Occlusions are found by imposing the uniqueness constraint. The algorithm can perform significant changes in contrast between the two images and can handle noise in one of the frequency channels. The algorithm has been justified on standard image pairs and needs 2 to 4 *seconds* to process.

Another algorithm that generates high quality results in real-time is reported in [22]. This algorithm is based on the minimization of a global energy function comprising of a data and a smoothness term. The propagation iteratively optimizes the smoothness and it achieves fast convergence by removing redundant computations. For real-time operation authors take advantage of the parallelism of graphics hardware. Experimental results indicate 16 *fps* processing frame-rate for 320 × 240 pixel self-recorded images with 16 disparity levels.

1.2.3 Other Methods

Besides the two above mentioned methods there are also some methods producing dense disparity maps. Continuous Wavelet Transform (CWT) reported in [23] can be placed in neither of previous categories. It makes use of the redundant information that results from the CWT. Using 1D orthogonal and bio-orthogonal wavelets as well as 2D orthogonal wavelet the maximum matching rate obtained is 88.22% for the Tsukuba pair.

An algorithm based on non-uniform rational B-splines (NURBS) curves presented in [24]. The curves replace the edges extracted with a wavelet-based method. The NURBS are projective invariant and so they reduce false matches due to distortion and image noise. Stereo matching is then obtained by estimating the similarity between projections of curves of an image and curves of another image. A 96.5% matching rate for a self-recorded image pair is reported for this method. Daniel Scharstein el at. [25] reported in High-Resolution Stereo Datasets with Sub-pixels Accurate Ground Truth to find high resolution thirty-three stereo datasets of static indoor scenes with highly accurate ground-truth disparities. The system includes novel techniques for efficient 2D sub-pixels correspondence search and self-calibration of cameras and projectors with modeling of lens distortion.

1.2.4 Related Works in Recent Trends

The matching costs related recent works are stated in [29] and [30]. The authors used bilateral filter to determine the cost aggregation and in order to reduce the computational cost, they also limit the label space. The work in [31] can be considered as a cost aggregation method by guided image filter. The average runtime [31] of the four standard Middlebury datasets (including Tsukuba, Venus, Teddy and Cones data sets) is 960 *milliseconds* reported in [34]. So the run time of single image pair like Tsukuba or Venus is about 240 *milliseconds* only. Disparity Space Image (DSI) structure and gradient information has been combined as a new technique is first time introduced by Nadia Baha and Slimane Larabi [32]. They used DSI technique with adaptive window-support. Another approach is introduced by themselves as DSI with refinement. The experimental results take time 0.2 *second* and 0.39 *second* respectively to process Tsukuba head image pair with different approach.

A new geodesic $O(1)$ filter is employed in [33] for the reliable disparity propagation. Such type of filter is very effective for the cost matching. As it is state-of-the-art method and the speed of this method has been justified on the Middlebury standard datasets, so we can compare this paper to our proposed methods. Xun Sun et al. [33] performed the experiment on PC equipped with a 3.0 GHz Intel Core i-5 CPU, 8 GB of memory and a Geforce GTX 580 graphics card. The processing time on Middlebury standard dataset is only 9 *milliseconds*.

A cost aggression has been adaptively estimated on a tree structure derived from the stereo image pair to preserve depth edges. This latest idea is launched by Q. Yang [34] in which shortest distances measure the similarity between two pixels on the tree. The average runtime of the four standard Middlebury datasets (including Tsukuba, Venus, Teddy and Cones data sets) is about 90 *milliseconds* using the tree filtering method. But he mentioned in same section that the runtime is about *7 milliseconds* on average on the Middlebury datasets. For identical comparison to our proposed methods we consider his second result that takes *7 milliseconds* on average on the Middlebury datasets. Q. Yang tested his experiment on a MacBook Air laptop with a 1.8 GHz Intel Core i-7 CPU and 4 GB memory. Another recent method achieves state-of-the-art result on Middlebury stereo datasets that performs stereo matching as a two steps energy-minimization algorithm [35]. The running time of this method is 3 *seconds* only for Tsukuba dataset and 20 *seconds* for Teddy dataset on a computer containing an Intel Core i-5-4300U 1.9-GHz CPU and a 6-GB RAM. Semi-global matching and cost is refined by cross-based aggression has been introduced by J. Zbontar et al. [36]. They also use left-right consistency check to eliminate the errors. The experiment performs on KITTI stereo datasets. At very recent, Fusing Adaptive

Support Weights has been launched by Wenhuan Wu et al. [37]. Local and global support windows are used for each pixel in [37]. Self -guided cost aggression [38] has been determined by deep learning method that depends on two sub-networks. A pyramid stereo matching network [39] also consists of two modules based on pyramid and 3D CNN that have been tested on KITTI 2012 and 2015 datasets. Adaptive Weighted Bilateral Filter [40] is used as main filter at cost aggression step for edge preserve factor.

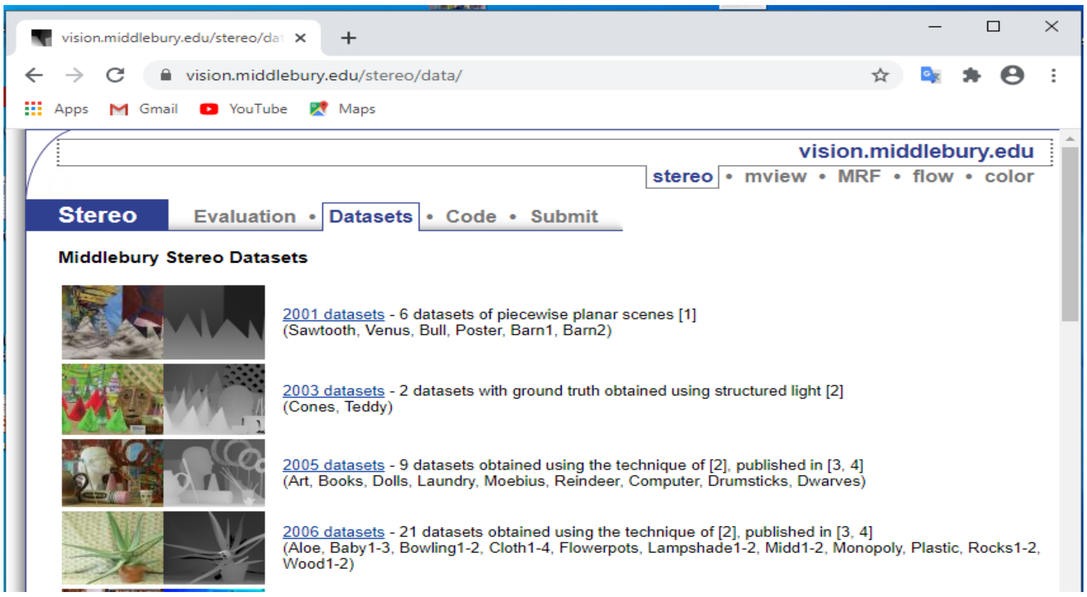
With the above reviews we found that some researchers employed adaptive window-based techniques to calculate the matching costs. But in our proposed methods, we have employed *self-adaptive, self-guided, or two-dimensional computing functions* to calculate the matching costs dynamically. This is one of the distinguishable innovative ideas between our proposed methods and existing state-of-the-art methods. The computational cost calculation formula used in above-mentioned methods is similar to that of our proposed methods, but the search technique is very different. Besides these analysis, the work in [33] requires preprocess and the works in [32], [34] and [36] need post processing steps like refinement, filtering and histogram equalization. Our proposed methods are implemented without preprocessing and post-processing. The experimental dense disparity maps are directly eligible to compare with ground truth dense disparity. So, considering the adaptive similarity, mode of guidance, identical stereo datasets (Middlebury Standard datasets) and hardware platforms, we can consider the articles of [9], [13], [28], [32], [33], [34], [35], [37], [38] and [39] to compare between the state-of-the-art methods and our proposed methods.

1.3 Stereo Matching Standard Datasets

Most of the stereo matching experiments are tested on standard datasets of stereo images. Such types of standard stereo images are Middlebury Standard Stereo Images [12] and [41]. The researchers who wish to work in stereo image processing must have to work with these standard images and should compare their experimental results with respective ground truth image. There are some others online stereo benchmarks, they are:

- Robust Vision Challenge
- KITTI Stereo 2012 evaluation
- KITTI Stereo 2015 evaluation

- ETH3D 2-view stereo benchmark
- Heidelberg HD1K Stereo benchmark


vision.middlebury.edu/stereo/da
vision.middlebury.edu/stereo/data/
Apps
Gmail
YouTube
Maps
vision.middlebury.edu
stereo • mview • MRF • flow • color
Stereo
Evaluation • Datasets • Code • Submit

Middlebury Stereo Datasets

2001 datasets - 6 datasets of piecewise planar scenes [1]
(Sawtooth, Venus, Bull, Poster, Barn1, Barn2)

2003 datasets - 2 datasets with ground truth obtained using structured light [2]
(Cones, Teddy)

2005 datasets - 9 datasets obtained using the technique of [2], published in [3, 4]
(Art, Books, Dolls, Laundry, Moebius, Reindeer, Computer, Drumsticks, Dwarves)

2006 datasets - 21 datasets obtained using the technique of [2], published in [3, 4]
(Aloe, Baby1-3, Bowling1-2, Cloth1-4, Flowerpots, Lampshade1-2, Midd1-2, Monopoly, Plastic, Rocks1-2, Wood1-2)

Fig. 1.3 Middlebury Standard Stereo Images of Different Datasets.

Fig. 1.3 demonstrates the Middlebury Standard Stereo Images only. But Middlebury benchmark is categorized on five types of datasets.

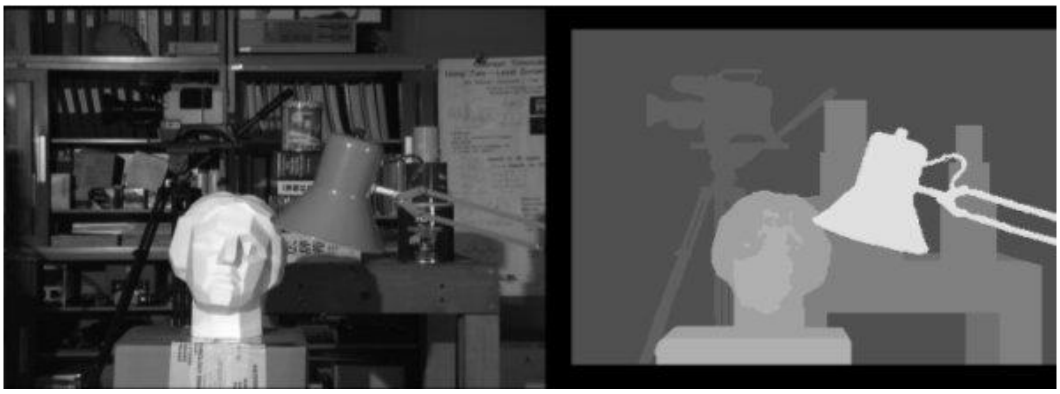
- Stereo
- Mview
- MRF
- Flow
- Color

The experimental results of this research are tested and compared on -

1) Middlebury Standard Stereo Images of different datasets and

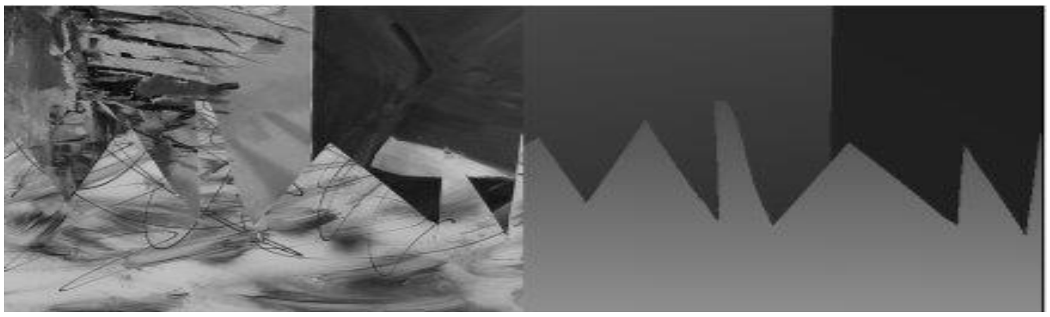
2) Middlebury Optical flow datasets.

Tsukuba, Sawtooth and Venus reference image (Left image) and their ground truth are illustrated below for visual understanding. Middlebury 2001 datasets consists of six datasets (Sawtooth, Venus, Bull, Poster, Barn1, Barn2) of piecewise planar scenes [12]. Middlebury standard 2003 datasets consist of two datasets (Cones, Teddy) with ground truth obtained using structured light. 2006 datasets comprise with 21 datasets (Aloe, Baby1-3, Bowling1-2, Cloth1-4, Flowerpots, Lampshade1-2, Midd1-2, Monopoly, Plastic, Rocks1-2, Wood1-2), is obtained using the technique of high-accuracy stereo depth maps using structured light and published in [41].



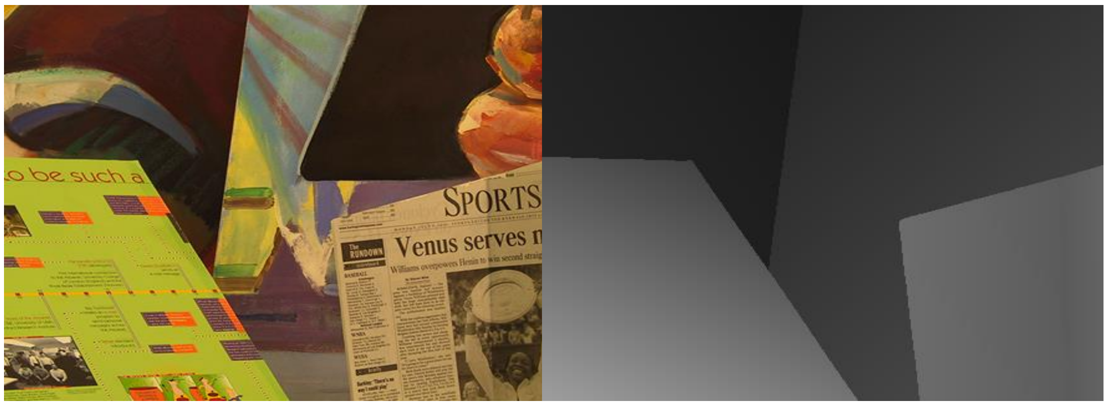
(a) (b)

Fig. 1.4 (a) Left image of Tsukuba stereo pair. (b) Ground truth image.



(a) (b)

Fig. 1.5 (a) Left image of Sawtooth stereo pair. (b) Ground truth image.



(a) (b)

Fig. 1.6 (a) Left image of Venus stereo pair. (b) Ground truth image.

1.4 Inaugurated Motivation

Window-based stereo correspondence estimation technique is widely used due to its efficiency and ease of implementation [9]. In a fixed window-based system a pixel might be compared repeatedly within several windows that lead to increase the computational time, which is relatively high. **Fig. 1.7** shows the redundant comparisons area of two neighbor pixels of (x, y) and (x_1, y_1). The pixels of the overlapping area (redundant area) are compared both for the center pixels (x, y) and (x_1, y_1). So, the computational time is calculated two, three or more times (depends on window size) higher than that of the actual time.

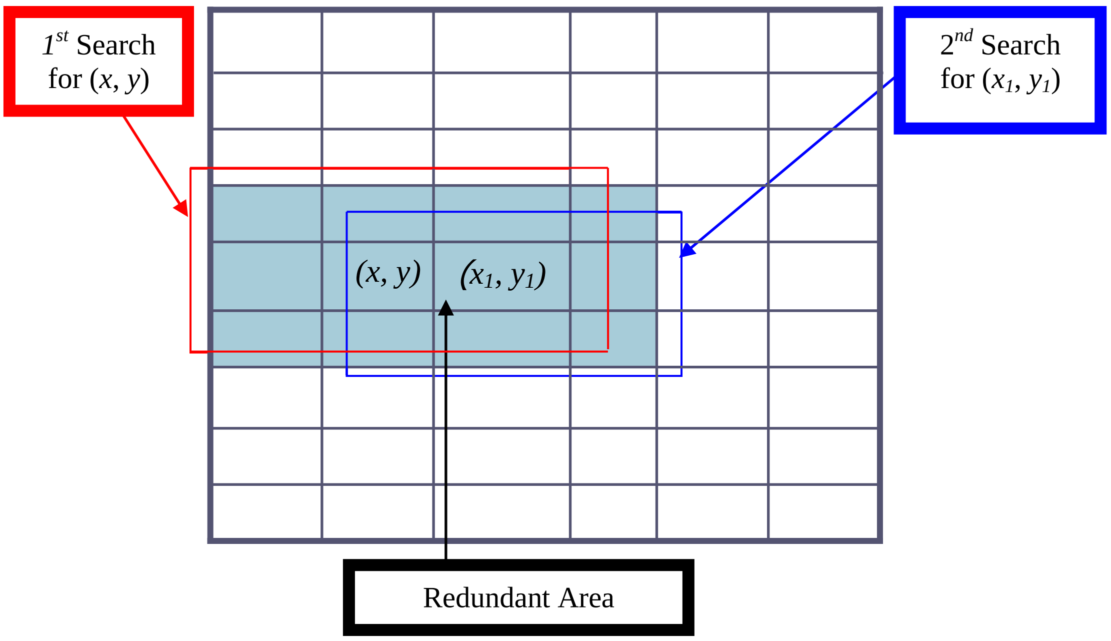


Fig. 1.7 The redundant area of neighbor pixels [(x, y) and (x_1, y_1)].

First search performs on the candidacy nine pixels enclosed by first rectangular large window. Second search occurs on the candidacy nine pixels enclosed by second rectangular large window pixels, delayed by just one pixel along the x axis. So the redundant six pixels of overlapping area centered by the co-ordinate pixels (x, y) and (x_1, y_1) will participate both in first and second window cost calculation processes. These redundant calculations are firstly explored in our research and we resolve this problem by inventing some novels and original stereo matching algorithms which are explained in subsequent chapters.

1.5 Research Objectives

The objectives of this thesis are to develop an optimal algorithm to determine the stereo correspondences from the standard stereo images for real-time application. The goal of the desired algorithm should be fast, robust and accurate. The system should be performed on simple hardware platform without any requirement of parallel processing or Graphics Processing Unit (GPU).

In order to achieve the destination, we developed four algorithms on trial and error basis and invent the optimal one.

The method should employ the standard window cost techniques like SSD, SAD or NCT. But most of the state of the works used SAD, so it should use the same for identical comparisons. The system should be tested on standard benchmark data like Middlebury or KITTI stereo datasets.

The system should work on dense disparity based idea. At first the method calculates the window costs which is presented in this chapter in Section 1.1 with Figure 1.2.

The system should be able to find out the minimum distance between reference pixel and corresponding pixels of minimum window cost. The invention of minimum window cost techniques are proposed in Chapter 2, Chapter 3, Chapter 4 and Chapter 5. The self-guided optimal algorithm for stereo correspondence estimation is presented in Chapter 5.

Different tests should be performed on different standard datasets to evaluate the performance of the system based on computational cost and accuracy. The system should be tested on Middlebury standard stereo datasets of 2001, 2003 and 2006. The system should be also tested on latest Middlebury optical flow datasets.

The system should be performed better than previous related researches with respect to computational time, frame-rate and accuracy. The proposed systems are compared with related researches for same datasets and environment with respect to computational time, frame-rate and accuracy.

1.6 Scope of the Work

Stereo correspondence or disparity has variety of applications, including people tracking, robotics navigation and medical imaging. The proposed system can provide full field of view for 3D measurements. Some of those scope and application areas of this thesis are discussed in this section. It is also used in industrial automation and 3D machine vision to perform task such as 3D object location and identification. The system also can be applied in medical field for monitoring the patients and health care.

The proposed system can be used for developing stereoscopic medical imaging devices to realize the surgical problems accurately. By using the proposed algorithm, stereoscopic medical devices can provide more realistic depth perception to the viewer than conventional imaging technology. The proposed system can permit more accurate understanding and analysis of 3D view and distance of small biological specimens. Therefore, this system can be used to develop the stereo endoscopy and stereo microscopy that can be utilized clinically to improve the surgical accuracy and patient safety.

The proposed SGSC algorithm can be used in the stereo imaging devices of the military and investigative fields. An Unmanned Aerial Vehicle (UAV) is commonly known as a drone. The visional part of UAV is performed by special camera. The stereo vision camera plays a key role in battle field. The proposed system can be used on UAV's camera to measure the distance of enemies' military vehicles and position exactly. Similarly, the system can be used in weather analysis to know the position of cyclone.

1.7 Organization of Doctoral Dissertation

The subsequent Chapters describe disparity estimation techniques employed for this research in details. Chapter 2 introduces a novel method entitled as *Real Time Approximation* (RTA) for stereo correspondence estimation. Some limitations of RTA algorithm encouraged us to invent a new original algorithm called *Two Dimensional Real Time Spiral Search Algorithm* (2DRTSSA) which is described in Chapter 3. The best *Self Adaptive Algorithm* (SAA) is presented in details in Chapter 4. The experimental results and performances of this algorithm have been presented numerically and visually. The original and novel *Self- Guided Stereo Correspondence* (SGSC) Estimation algorithm is the optimal algorithm of state-of-art method and is explained elaborately with experimental results in Chapter 5. The overall conclusion of this research has been drawn in Chapter 6.

1.8 Summary

The content of this Chapter is primarily background information for stereo correspondence estimation. The concepts introduced in Section 1.2, the classifications of stereo correspondence estimation approaches i.e., local techniques and global techniques are introduced here. Their limitations are also mentioned in this section. The subsections 1.2.1, 1.2.2, 1.2.3 and 1.2.4 focus on features and shortcomings on earlier and recent related works. Section 1.3 introduces the image-data of Middlebury Standard Stereo Images of different years. The eagerness of this research is described in Section 1.4. The hierarchical developments of this research are briefly described in Section 1.5.

Chapter 2

Stereo Correspondence Estimation Technique: Real Time Approximation (RTA) Algorithm

2.1 Motivation of Real Time Approximation (RTA) Algorithm

Conventional direct search dominates the intensity levels of each candidate-pixel within the specified search range. In direct search [9], it requires to compute the window costs for all candidate-pixels within the search range -$d_{\max}$ to +$d_{\max}$. This type of algorithm is associated with high computational cost. In direct search, it requires to calculate the all window costs i.e., the SAD values for all candidate-pixels within the search range -$d_{\max}$ to +$d_{\max}$. Some authors used bidirectional search or left-right checking algorithm with a view to discard the non-matching pixels or for smoothing the dense disparity map. This requires the doubling the computational complexity due to the requirement of its reverse matching search [13] and [43]. The disparity estimation algorithm using traditional direct search [9] is mentioned below.

```
For each Pixel ($x$, $y$),
    for $d'$ = $-d_{\max}$ to + $d_{\max}$ do
        Calculate $W_c$ ($x$, $y$, $d'$);
    end
    Find best $W_c$ ($x,y,d$) $\in$ $W_c$($x,y,d'$);
    Disparity of ($x$, $y$) = $d$;
end
```

In order to estimate the disparity with low cost computation, Sum of Square Differences (SSD) or Sum of Absolute Difference (SAD) or some other measure is computed between a window centered in the first image and the same window shifted by in the second image. The shifting mechanism depends on the window/ mask size. The shifting procedure does not require to determine the window cost for all candidate-pixels in right image. This key feature is employed in proposed RTA algorithm to reduce the computational cost. For fast and efficient computation, a rectangular window of fixed size centered at the pixel under consideration is employed.

2.2 Proposed RTA Method

Since the main objective of the proposed method is to reduce the computational cost for making the system useful for real-time applications, the first step of this proposed method is to reduce the size of rectified stereo images. The given left and right images are being reduced in size by nine times using vector quantization method, which ultimately helped to reduce the searching area significantly. After first step, the RTA algorithm produces the quantized shrinked left and right image as shown in **Fig 2.1**.

Secondly, SAD is applied to calculate the window cost for all candidate-pixels in the right image within the reduced search range. According to the proposed RTA method, the experimental disparity image is estimated from shrinked left and right images. But the conventional methods like Fast Area Based [13] stereo matching computed the stereo correspondence directly from left and right image. Various methods can be used for shrinking the left and right images.

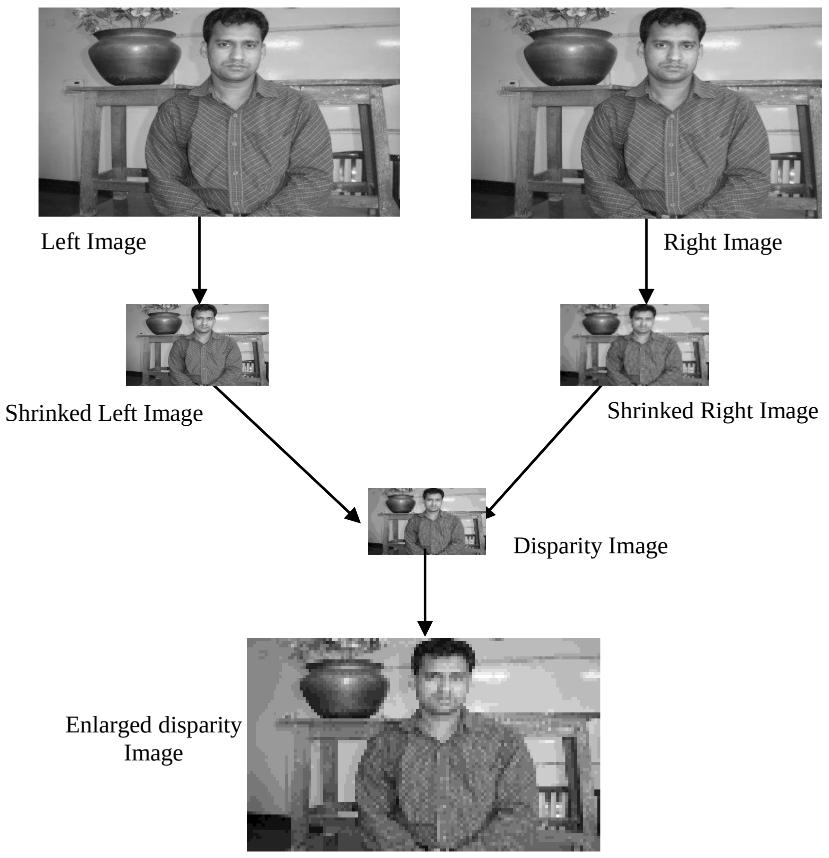


Fig. 2.1 Hierarchical schematic of RTA method.

We employed vector quantization of the window averaging method for image shrinking. Finally, we enlarge the obtained disparity image to its original size for comparing the experimental disparity map with the ground truth image.

2.2.1 Quantization Process for Rectified Stereo Images

Shrinking process can be viewed as under vector quantization. There are many approaches for image shrinking, for instance, to shrink an image by one-half, we delete every other row and column. In this proposed method, images are shrinked by window average method and consideration of a 3 × 3 window pixel as a vector. **Fig. 2.2** illustrates the shrinking process. Suppose the upper image is original left or right image and lower image is shrinking quantized-vector image. According to the proposed RTA algorithm, for first window of original image, nine pixels are converted to the first quantized pixel of quantized image, which is shown in **Fig. 2.2.** The subsequent nine pixels of original image will be the second quantized pixel of quantized shrinked image and so on. So, three times reduction occurs along the *x*-axis and three times reduction occurs along the *y*-axis. This process results the nine times reduced quantized-vector image which is demonstrated at the lower portion of **Fig. 2.2**.

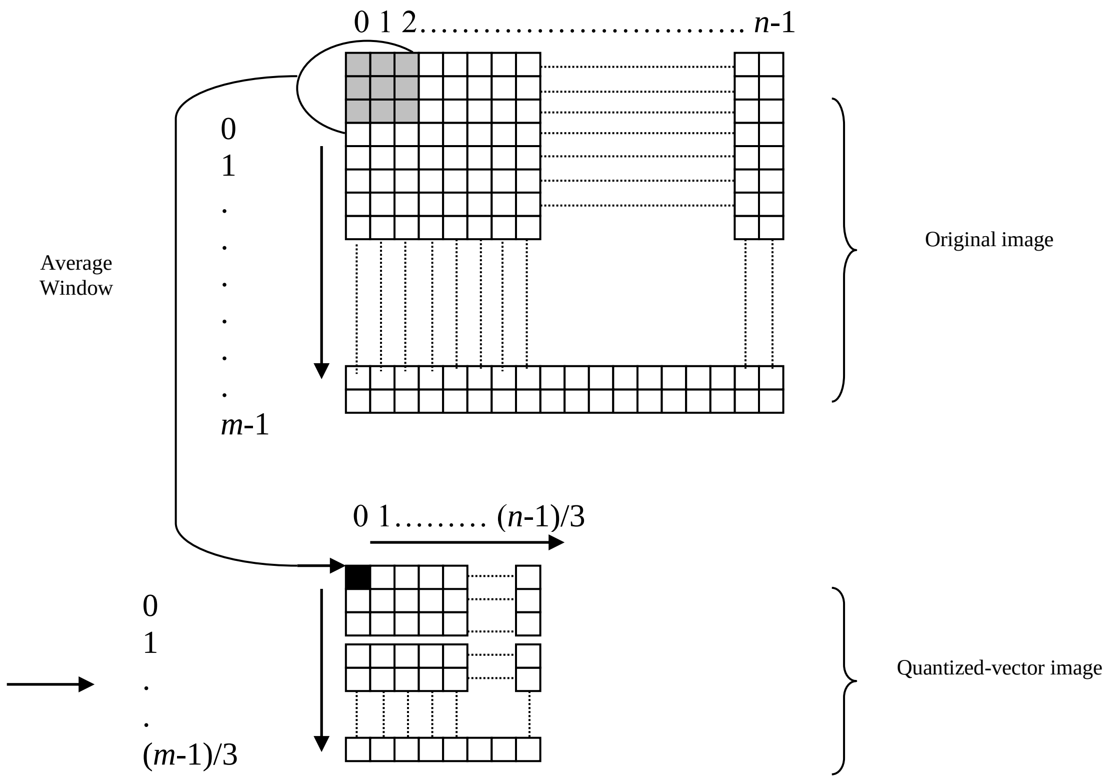
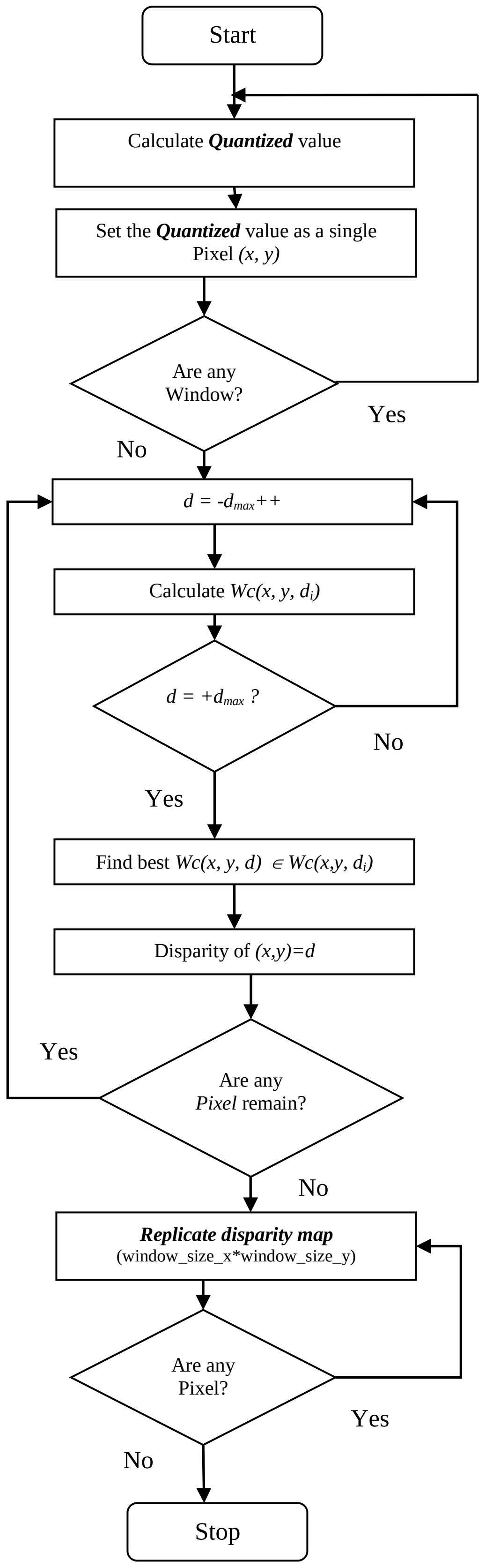


Fig. 2.2 Quantization process of rectified stereo images.

2.2.2 Real Time Approximation Algorithm and Flowchart

The overview of the RTA algorithm employed for this research work is shown in the flowchart in **Fig. 2.3**. The flow chart consists of three stages looping structure. In the first stage, the given left and right images are being compressed nine times by vector quantization of the window averaging method applying a 3×3 window size. Stereo correspondence or disparity estimation is performed at the second stage by matching windows of pixels using Sum of Absolute Differences (SAD) technique. Third stage involves the replication processing in order to retrieve the original size of experimentally estimated dense disparity map. This method has been experimentally evaluated for the computation of stereo under SAD technique. It performs reasonably good and significantly better than the methods based on window-based stereo matching techniques.



Start
Calculate ***Quantized*** value
Set the ***Quantized*** value as a single Pixel *(x, y)*
Are any Window?
Yes
No
$d = -d_{max}++$
Calculate $Wc(x, y, d_i)$
$d = +d_{max}$?
No
Yes
Find best $Wc(x, y, d) \in Wc(x,y, d_i)$
Disparity of $(x,y)=d$
Yes
Are any *Pixel* remain?
No
Replicate disparity map (window_size_x*window_size_y)
Are any Pixel?
Yes
No
Stop

Fig. 2.3 Flow chart of proposed RTA method.

```
Algorithm  RTA  (m,n,temp,temp1,w1,w2,sum1,sum2,sum,d,t0,t1,
v_left,v_right,image_left.pixel,image_right.pixel,image_disp
.pixel,ws1,ws2, k1,k2,kk1,kk2,dmax)

   1. //m,n is the row and column size of an image.
   2. // temp and temp1 are pixels intensity value of left and
   3. // right image.
   4. //w1,w2 is the row and column size of  mask.
   5. //sum1, sum2 and sum are the summation of pixel intensity
   6. //values of left and right images within the mask
   7. // image_left.pixel[1:m][1:n] is the left image pixel
   8. // coordinate that contains m×n number of elements.
   9. //image_right.pixel[1:m][1:n]is the right image pixel
  10. // Coordinate that contains m×n number of elements.
  11. //image_disp.pixel[1:m][1:n]is the disparity image
  12. // that contains m×n disparity values.
  13. // v_left is the pixel intensity value of left image.
  14. // v_right is the pixel intensity value of right image.
  15. //k1,k2 is the number pixels to discard from left and
  16. // right side of image.
  17. //ws1 and ws2 are the local variable within the mask.
  18. //kk1,kk2 is the number pixels to discard from left and
  19. // right side of image within the mask.
  20. //d and dmax is the search range.
  21. // t0 and t1 are time variables.
  22.      for n:=0 to size_y do
  23.       {
  24.      for m:=0 to size_x do
  25.         {
  26.   // Read left and right images and assign the
  27.   //Intensity values in their respective array.
  28.             image_left.pixel[m][n]:= temp;
  29.              image_right.pixel[m][n]:= temp1;
  30.          }
  31.        }
  32.      // Creation of quantized image.
  33.      ws1:= (w1/2), ws2:= (w2/2);
  34.      for n:=kk2 to size_y-kk2 step 3 do
  35.       {
  36.            for m:=kk1 to size_x-kk1 step 3 do
  37.              {
  38.             sum1:= 0; sum2:= 0;// initialization.
  39.             for i:=-1 to ws1 do
  40.                {
  41.                 for j:=-1 to ws2 do
  42.                 {
  43.                 v_left:=image_left.pixel[m+i][n+j];
  44.                 v_right:= image_right.pixel[m+i][n+j];
  45.                 sum1:= sum1+ v_left;
  46.                 sum2:= sum2+ v_right;
  47.                  }
  48.      i:=sum1/(w1*w2); j:= sum2/(w1*w2);
  49. write("quantized values of left image to 1st file",i);
  50. write("quantized values of right image to 2nd file",j);
```

```
        }
   }
// read the quantized images.
   for n:=0 to size_y1 do
    {
   for m:=0 to size_x1 do
      {
   image_left.pixel[m][n]:= temp;// scan the quantized
   //images using different file pointers
        image_right.pixel[m][n]:= temp1;
       }
     }
    t0:= clock();
   // Window cost calculation process using SAD
   for m:=k1 to size_x-k1 do
    {
      for n:=k2 to size_y-k2 do
      {
      for d:= -dmax to +dmax do
       {
         sum:= 0;
         for i:= - ws1 to ws1  do
          {
           for j:= -ws2 to ws2 do
              {
              v_left:= image_left.pixel[m+i][n+j];
              v_right:= image_right.pixel[m+i+d][n+j];
              sum:= sum+abs(v_left - v_right);
               }
            }
           Mtemp[d + dmax].pixel[m][n]:= sum;
      }
image_disp.pixel[m][n]:= minimum(Mtemp,m,n,ws1,&p);
   }
//creating replicated dense disparity map.
   for n:=0 to size_y1-2*k2 do
   {
      for m:=0 to size_x1-2*k1 do
     {
     Write("Values", pixel[m][n], pixel[m][n],pixel[m][n]);
        }
      for m:=0 to 116 do
      {
    Write("Values", pixel[m][n], pixel[m][n], pixel[m][n]);
     }
     for m:=0 to 116 do
     {
    Write("Values", pixel[m][n], pixel[m][n], pixel[m][n]);
  }
}
```

```
t1:= clock();
cpu_speed:= t1 - t0; // time calculation.
write("Total time",cpu_speed);
```

```
Algorithm minimum (temp[2*dmax+1],x,y,a,ws,*p)

// Find the minimum value from temp[0:2*dmax+1] elements
// x, y, i, j, mu, a, min are the integer variables.
//*p is the pointer variable.
j:=1;
  for i:=0 to  2*dmax+1 do
    {
      if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
        j:= i;
     }
     min:= temp[j].pixel[x][y];
 for a:=0 to  2*dmax+1 do
 {
  if(a≠j and temp[a].pixel[x][y] = min)then
  {
     *p+=1;
      break;
     }
   }
 mu:= abs(j-dmax);
 return (mu);
 }
```

2.3 Experimental Settings and Results

The accuracy and frame-rate of this algorithm has been justified over Middlebury standard stereo images of Tsukuba head . Experiments are performed on Intel Core i-3, 2.3 GHz processor PC with 4 GB DDR3 RAM. The algorithm has been implemented by Visual *C*++ programming language with Windows 10 operating system. **Table 2.1** illustrates the summary of comparison between window-based Fast Area Based method [13] and proposed RTA method. The computational time of RTA method is shown in **Table 2.1** using window size of 3 × 3 without any threshold. From this table, it reveals that for the same resolution of image(384 ×288), the proposed RTA method reduced 93.71 % of computational time. The **Fig. 2.4** and **Fig. 2.5** show the Tskuba head of left and right images view. The **Fig. 2.6** and **Fig. 2.7** illustrate the shrinked images of left and right respectively after applying quantization technique. **Fig. 2.8** shows the disparity image that is experimentally estimated from left and right image applying RTA method. The **Fig. 2.9** shows the replicated image of experimental disparity image. Experimental disparity images of **Fig. 2.8** and **Fig. 2.9** are histogram equalized for visualization purpose.

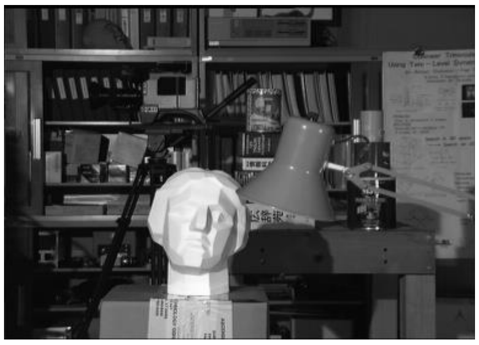


Fig. 2.4 Left image 384 ×288.

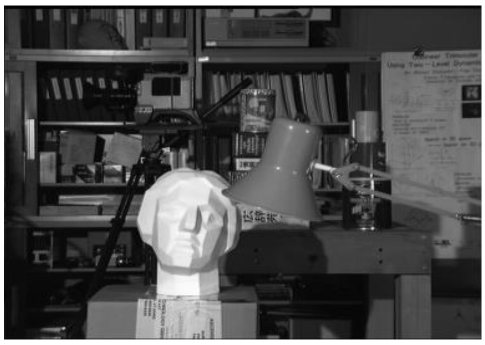


Fig. 2.5 Right image 384 ×288.

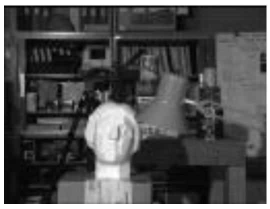


Fig. 2.6 Shrinked left image 128 × 96.

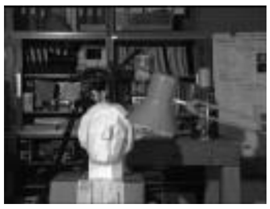


Fig. 2.7 Shrinked Right image 128 × 96.

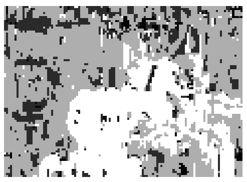


Fig. 2.8 Estimated disparity image 116 × 84.

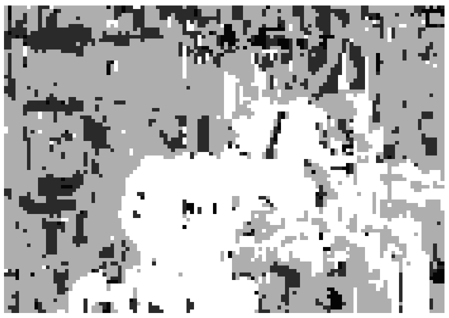


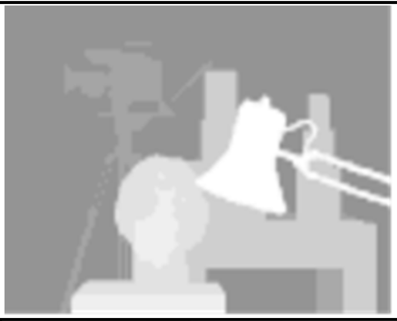
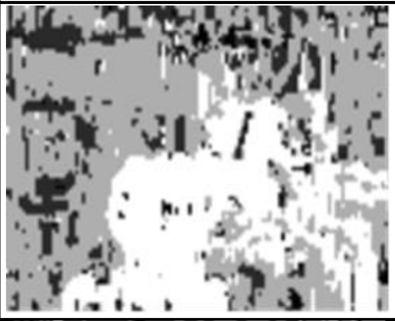
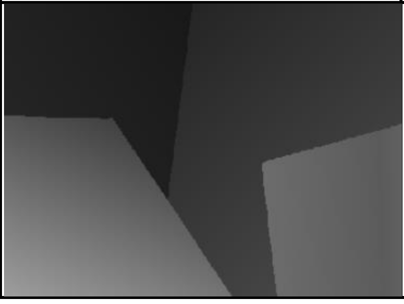
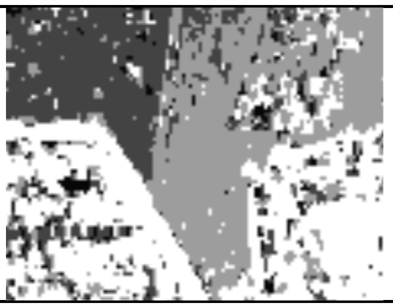
Fig. 2.9 Replicated disparity image 348 × 252.

The size of the left and right image is (width × height) = (384 × 288) pixels, the shrinked image size is (width × height) = (128 × 96) pixels, disparity image size is (width × height) = (116 × 84) pixels and replicated image size is (width × height) = (348 ×252) pixels.

Table 2.1: Computational time reduction (%) compare to window-based method.

Applying Method's Name	**Accuracy (in %)**	**Computational time (in μs)**	**Computational time reduction (in%)**
Fast Area Based Algorithm[13]	86.10	3229	0
RTA method	30.00	203	93.71

Table 2.2: Accuracy of RTA Algorithm

Reference image name	Ground Truth Image	Experimental Dense Disparity Map	Estimated Accuracy
Tsukuba Head			30%
Venus			20%

From the above experimental results it is seen that the computational time of RTA method is sharply reduced to only 203μs compared to 3229 μs. The processing time reduction is due to the following reasons:

1) Image shirking.
2) Reduced window size (3×3 instead of 11×11).

Though the processing time reduction is very high, the accuracy of this method is only 30%. **Table 2.2** demonstrates the estimated accuracy of Middlebury standard stereo datasets of Tsukuba and Venus stereo pair. The experimentally estimated accuracy of Tsukuba head is 30% and accuracy of Venus stereo pair is only 20%. This is happened in shrinking process because nine pixels are quantized at a single pixel.

So, eight (8) pixels might lose some intensity attributes. Beside this some accuracy has been lost during the replication process. This is only shortcoming of proposed RTA algorithm. Since the computational time reduction is very good, the RTA algorithm can be used where a very fast estimation of dense disparity is essential**.**

2.3.1 Experiment on Real Stereo Images by RTA

The performances of RTA algorithm have been further justified on real stereo images acquisitioned by Logitech stereo web camera. This experiment is performed in our software laboratory and images were captured as indoor scenes. The specifications of stereo camera are listed below-

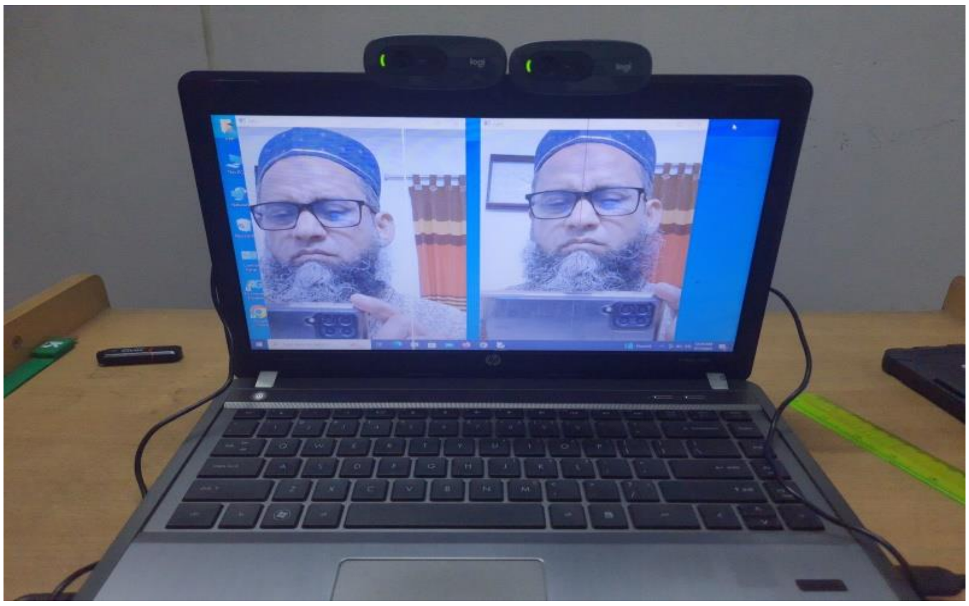
Brand: Logitech C270 Webcam

Country of Origin: Switzerland

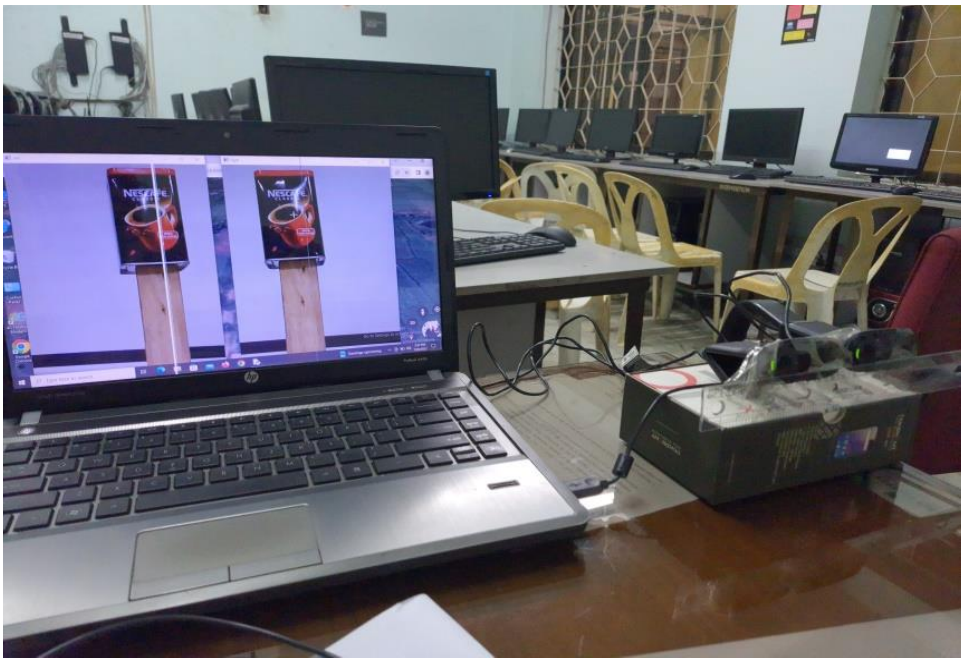
Sensor Resolution (MP) - 3MP, Video Resolution (Pixel) - 1280 ×720, Frame Rate - 30*fps*. Diagonal Field of View 55°.

Stereo Image Capturing Process:

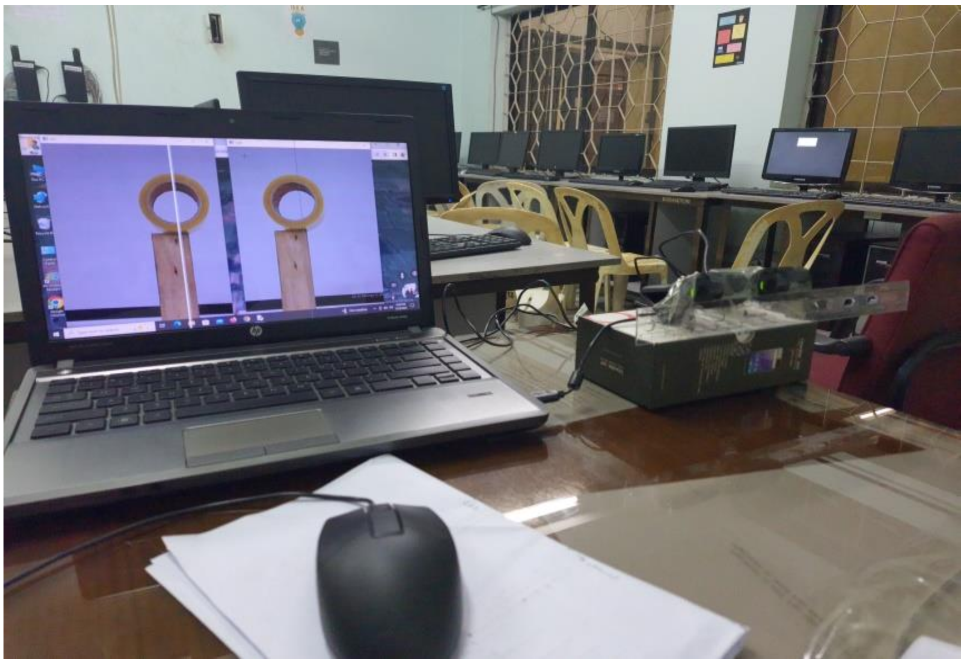
The main objects (Human face, Nescafe coffee stand and Scotch tape stand) of reference images were stood 63.00 cm away from the imaging sensor of the camera. The distance between two cameras was 6.70 cm.



a) Real image acqusition using stereo web camera for dataset-1(Human face).



b) Real image acqusition using stereo web camera for dataset-2(Nescafe coffee stand).



c) Real image acqusition using stereo web camera for dataset-3(Scotch tape stand).

Fig. 2.10 Real image acqusition process using stereo web camera.

Experimental output for Real Stereo Images:

The size of the left and right real-image is (width × height) = (550 × 720) pixels, the shrinked image size is (width × height) = (183× 240) pixels, disparity image size is (width × height) = (171 × 228) pixels and replicated image size is (width × height) = (514 ×684) pixels.

Table 2.3: Visual observation of disparity map of real images generated by RTA algorithm

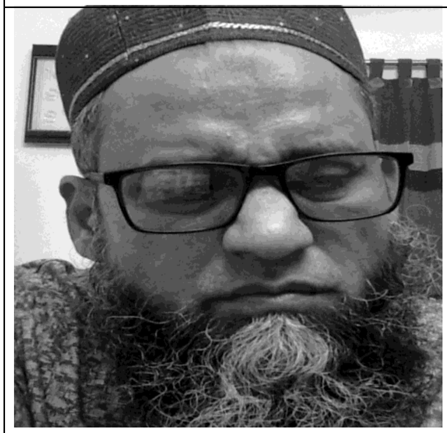
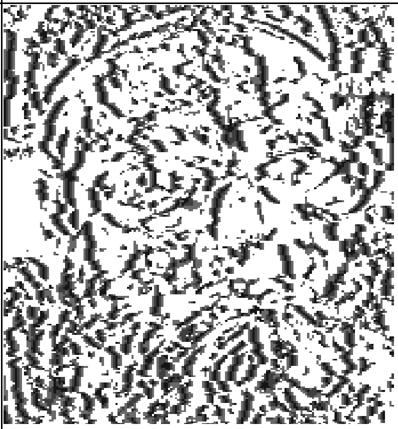
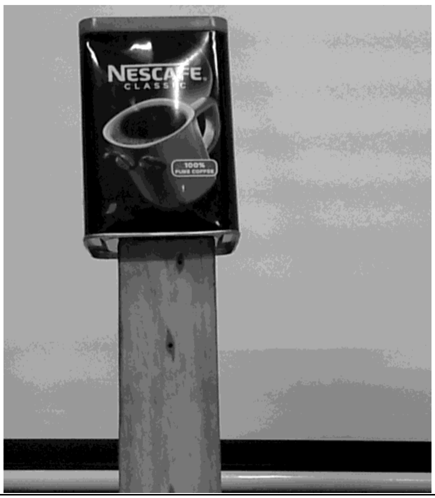
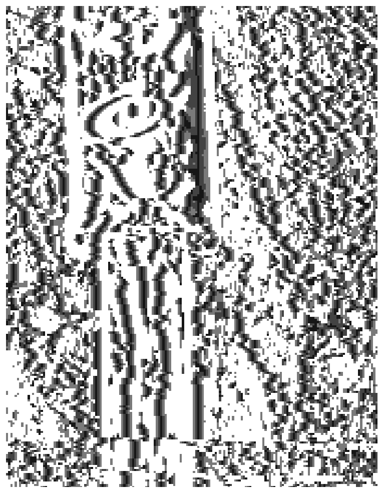
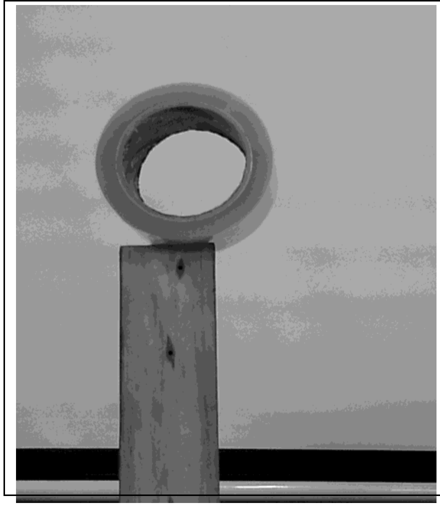
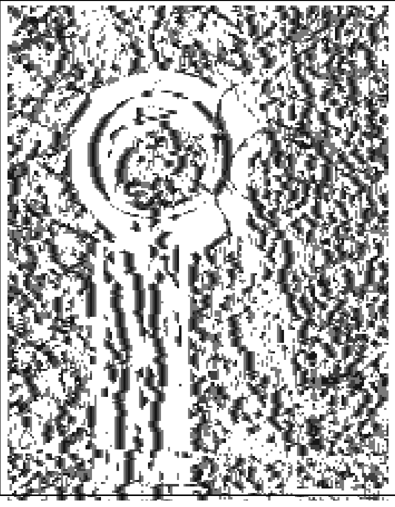
Reference image	Experimental Dense Disparity Map of Real Image	Execution time(μs)
		813
		828
		859

Table 2.3 demonstrates the visual observation of dense disparity maps. The **Table 2.3** illustrates the dense disparity maps of three datasets - Human face, Nescafe coffee stand and Scotch tape stand respectively. The visual qualities of these disparity maps are not good. But the object inside the reference image is visualized and understandable. The disparity maps of output image contain some noise. This is happened due to the following three reasons-

1) We could not provide the equilibrium light condition in our laboratory and hence the mask does not cover enough intensity variation. So the method gives erroneous result due to low signal to noise ratio.
2) Similarly the room temperature was not equilibrium at all the point during the image acquisition process.
3) Moreover, we have tried to our best to calibrate the stereo camera physically. The stereo cameras were manually placed on the same horizontal line, but experimentally, it was not possible. There was a vertical mismatch between two cameras in fractional millimeter (.05 mm approximately) range.

These cause to add a little noise in captured stereo images. Inspite of noise, the objects are demarked, visualized and understandable. So the overall performance of RTA algorithm is good in case of real stereo images.

2.4 Discussion

Experimental results confirm that we can easily reduce the computation time of about 93.71%. Though the accuracy is poor because of quantization error, but the RTA method will be useful for many applications where a very fast estimation of dense disparities is essential.

2.5 Summary

The material of this Chapter establishes on our first original method for disparity estimation. The main objective of this method was to reduce the computational cost and make the system useful for real-time applications. The concepts behind RTA method are described in the Section 2.2 at subsection 2.2.1. The quantization process which was an important technique in RTA algorithm has been described in subsection

2.2.1 of this Chapter. Flow chart and Algorithm are included in subsection 2.2.2 for better understanding. Experimental settings and results are described in Section 2.3. The performances of RTA algorithm have been additionally justified on real images in subsection 3.2.1.

Chapter 3

Stereo Correspondence Estimation Technique: Two-Dimensional Real Time Spiral Search Algorithm-2DRTSSA

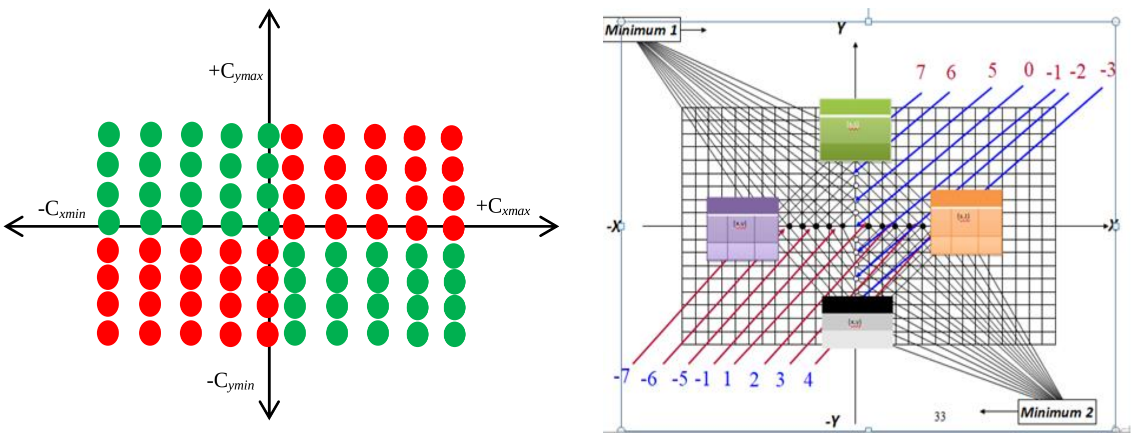
3.1 Motivation of 2DRTSSA Algorithm

The main objective of this research was to invent the optimal algorithm for stereo correspondence estimation, which will be the best trade-off between speed and accuracy in local domain. Although the RTA algorithm of previous chapter had the satisfactory computational speed but its accuracy was poor due to quantization error. Instead of reducing the image size, we were trying to improve the inherent matching accuracy for increasing the quality of RTA algorithm. To achieve this, two-dimensional pixel-wise costs are improved by employing a parallel computing on two axises, which is named as Two-Dimensional Real Time Spiral Search Algorithm - 2DRTSSA.

3.2 Proposed 2DRTSSA

The innovative 2DRTSSA search method can be explained by using co-ordinate geometric concept. The search ranges are outlined in **Fig. 3.1** that shows the search coordinates range ($-C_{xmin}$, $-C_{ymin}$) to ($+C_{xmax}$, $+C_{ymax}$) instead of using $-d_{max}$ to $+d_{max}$ of one dimensional existing system. Accordingly, the proposed method is applied to compute the window costs in two dimensionally. In first phase, first search is done concurrently in the *1st* and *3rd* quadrants (red pixel) of right image as indicated in **Fig. 3.1(a)**. In second phase, second search is performed in the second and fourth quadrants (green pixel) of right image. In both cases the searching commences from the starting point (say $-C_{xmin}$, 0) to the ending point ($+C_{xmax}$, 0) as shown in **Fig. 3.1(b).** Every iterative program sequence tends to reach at the origin point. Each reference pixel of reference image (left image) is hunted in the two-axial coordinate-points according to the above stated procedure. According to the proposed 2DRTSSA method, each pixel of reference image is firstly compared with negative *x*-direction of right image as well as positive *y*-direction of right image. Suppose in two cases, two distinct disparities are determined as d_1 and d_2 from their respective minimum window costs. Secondly, the same pixel is compared with positive *x*-direction of right image as well as negative *y*-direction of right image. Let another two distinct disparities are determined as d_3 and d_4 from their respective minimum costs. All the experimentally estimated disparities $\{d_1, d_2, \ldots d_{+Cxmax}\}$ are passed to the minimum cost function of array d_i. Finally, disparity d is estimated from the set of elements $W_c(x, y, d_i)$, *i.e.*, $W_c(x, y, d) \in W_c(x, y, d_i)$. Therefore, the stereo correspondence or disparity of a reference pixel of left image is $P(x, y) = d$.

The process is then repeated for the successive pixels of reference image along the 2D scan lines from left to right of the whole image. With the above mentioned strategy, the proposed method avoids the repetition of redundant comparisons and false matching, hence increases the frame-rate and accuracy.



(a) (b)

Fig. 3.1 Illustration of 2DRTSSA search method with co-ordinate prefecture.

3.2.1 Algorithm and Flowchart of 2DRTSSA

```
Algorithm 2DRTSSA(m, n, temp, cid,cmin,temp1, w1, w2,
sad1,sad2,v_left,v_right,image_left.pixel,image_right.pixel,
image_disp.pixel,ws1,ws2,k1,k2,abs1,abs2,dmax,t0,t1)

//m,n is the row and column size of an image.
// temp and temp1 are pixels intensity value of left and
// right image.
//cid,cmin is the integer type variables indicate spiral
// distance.
//w1,w2 is the row and column size of the mask.
//sad1 and sad2 is the summation window costs
//values of two axes within the mask.
// image_left.pixel[1:m][1:n] is the left image pixel
// coordinate that contains m×n number of elements.
//image_right.pixel[1:m][1:n]is the right image pixel
// coordinate that contains m×n number of elements.
//image_disp.pixel[1:m][1:n]is the disparity image
// that contains m×n disparity values.
// v_left is the pixel intensity value of left image.
// v_right is the pixel intensity value of right image.
//k1 ,k2 are the number pixels to discard from left and
// right side of image.
//abs1 ,abs2 difference value of left and right pixel.
//ws1 and ws2 are the local variable within the mask.
//dmax is the search range.
// dis counter variable.
// t0, t1 is the variable for time calculation.
      for n:=0 to size_y do
       {
```

```
    for m:=0 to size_x do
     {
     image_left.pixel[m][n]:= temp; // Read left image
     image_right.pixel[m][n]:= temp1;//Read right image
     }
    }
t0:= clock();
 // 2D Window cost calculation process.
   ws1 := (w1/2); ws2 := (w2/2); p:= 0;
   for m:= k1 to size_x-k1 do
   {
     for n:= k2 to size_y-k2 do
       {
     dis:= 0;
        for cid:= -cmin to  cmin do
          {
           sad1:= 0;
           sad2:= 0;
           for i:=- ws1 to ws1 do
             {
                   for j:= -ws2 to ws2 do
                {
   if(((m+i+cid*2)>= 0) and ((m+i+cid*2)< size_x))then
    {
     v_left:= image_left.pixel[m+i][n+j];
     v_right:= image_right.pixel[m+i+cid*2][n+j];
     v_left1:= v_left;
     v_right1:= image_right.pixel[m+i][n+j+cid*(-2)+1];
               abs1:= v_left - v_right;
               abs2:= v_left1 - v_right1;
                  if(abs1 < 0) then abs1:= -1*abs1;
                   if(abs2 < 0) then abs2:= -1*abs2;
                         sad1:= sad1 + abs1;
                         sad2:= sad2 + abs2;
           }
        }
     }
           // Select the minimum window cost.
           if(sad1 <= sad2)then
              Mtemp[dis++].pixel[m][n]:= sad1;
           else
              Mtemp[dis++].pixel[m][n]:= sad2;
        }
  // Find the best window cost using minimum function.
image_disp.pixel[m][n]:= 2*minimum(Mtemp,m,n,ws1,&p);
     }
```

```
72.      }
73.      t1:= clock();
74.      cpu_speed:= t1 - t0; // time calculation.
75.      write("Total time",cpu_speed);
76.    // Creating the dense disparity map.
77.      for n:= k1 to size_y-k1 do
78.        {
79.        for m:= k2 to size_x-k2 do
80.         {
81.      write("dense disparity
   image",image_disp.pixel[m][n]);
82.     }
83.   }
```

```
Algorithm minimum (temp[2*dmax+1],x,y,ws,*p)

1. // Find the minimum value from temp[0:2*dmax+1]elements
2. // x, y, i, j, mu, a, min are the integer variables.
3. // p is the pointer variable.
4.  { j:=1;
5.   for i:=0 to 2*cmin+1 do
6.     {
7.       if(temp[i].pixel[x][y] < temp[j].pixel[x][y]) then
8.         j:= i;
9.      }
10.      min:= temp[j].pixel[x][y];
11.   for a:=0 to  2*cmin+1 do
12.   {
13.    if(a≠j and temp[a].pixel[x][y] = min)then
14.     {
15.       *p+=1;
16.        break;
17.       }
18.     }
19.   abs11:= j-cmin;
20.   mu:= (abs11<0)?(-1*abs11):abs11;
21.   return (mu);
22.    }
23.   }
```

The key idea of this algorithm is that, the search is divided into two axial regions which are well defined in step 48. One cost aggression is estimated along the *x*-axis on photometric point $((x + C_{id} * 2), y)$ while the other cost aggression is estimated along the *y*-axis on photometric point $(x, (y + C_{id} * (-2) + 1))$.

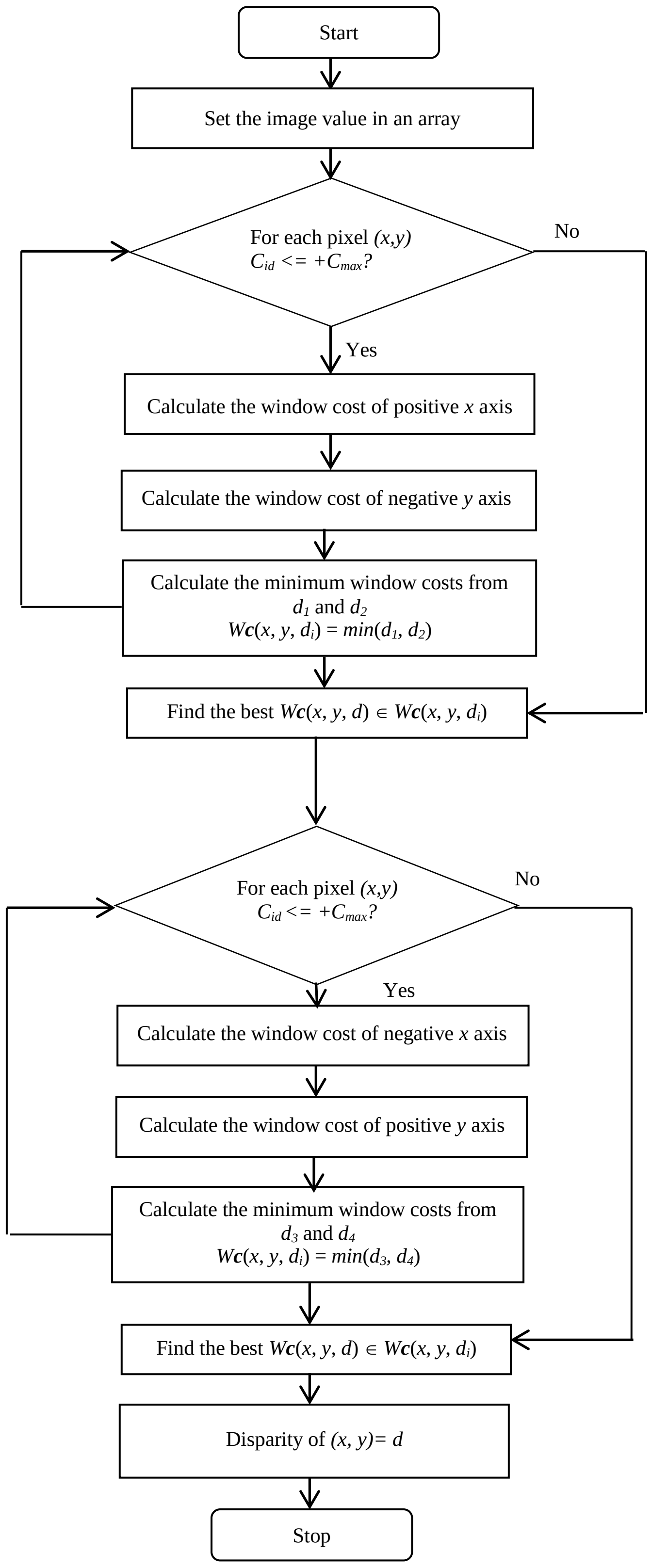

Start
Set the image value in an array
For each pixel (x,y) $C_{id} <= +C_{max}$?
No
Yes
Calculate the window cost of positive x axis
Calculate the window cost of negative y axis
Calculate the minimum window costs from d_1 and d_2 $Wc(x, y, d_i) = min(d_1, d_2)$
Find the best $Wc(x, y, d) \in Wc(x, y, d_i)$
For each pixel (x,y) $C_{id} <= +C_{max}$?
No
Yes
Calculate the window cost of negative x axis
Calculate the window cost of positive y axis
Calculate the minimum window costs from d_3 and d_4 $Wc(x, y, d_i) = min(d_3, d_4)$
Find the best $Wc(x, y, d) \in Wc(x, y, d_i)$
Disparity of $(x, y) = d$
Stop

Fig. 3.2 Flow chart of 2DRTSSA algorithm.

Two axises are selected simultaneously by the expression in the third bracket in steps 51 and 53. The proposed method searches a reference pixel on two probable spaces at a time within a finite range C_{id}. On the contrary, the existing state-of-the-art algorithms search a reference pixel only one space at a time. Therefore, this idea makes the proposed method faster than existing methods.

The concept of 2DRTSSA is illustrated in flowchart of **Fig. 3.2** for better understanding. The flow chart consists of two looping structures. In the first loop, it computes the window costs of positive x and negative y axis simultaneously. Secondly, the 2DRTSSA method measures the window costs on remaining negative x and positive y axis concurrently. Final disparity is the minimum window cost among the four window costs of x and y axis of a pixel (x, y). The process is then repeated for the successive pixels of reference image along the scan line from left to right of the whole image. This method has been experimentally evaluated for the computation of stereo correspondence under SAD technique.

3.3 Computational Complexity Analysis

The computational complexity of 2DRTSSA algorithm is $O(n \times w/2)$, where n is the total number of candidate-pixels and w is the window size of mask. Two matching costs are estimated two different co-ordinates (x and y) at the same time. The main idea of this method is to reduce the window cost by around 50%. These two simultaneous window cost calculations executed per instruction cycle. That is the main reason of reducing the computational time of this method. The required memory depends only the size of n i.e., it directly proportional to image size. It apparently seems that it requires more memory space for two window costs at a time. But actually, the proposed algorithm compares instantly two window costs, selects the minimum one, and discards the other. The total run time for the Tsukuba head image pair is 4480 μs in 2DRTSSA whereas 5844 μs in 1DRTSSA on the same hardware (Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM).

3.4 Experimental Settings and Results

The experiments were done on Middlebury standard stereo images of Tsukuba head stereo pair. The computational time, frame-rate and accuracy of the proposed algorithm have been compared with the Middlebury standard articles. The experimental dense disparity maps are estimated from left and right images applying

2DRTSSA is shown in **Fig. 3.5** to **Fig. 3.10**. Standard dense disparity of ground truth image is shown in **Fig. 3.11**. Experiments were performed on Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM computer. The algorithm is performed by Visual *C++* programming language with Windows10 operating system. To determine the correspondence of a pixel of reference image, the window costs are estimated for the candidate-pixels of right image within the search range -10 to +10. The experimental results show that, the proposed algorithm is currently a better method among the existing state-of-the-art methods. The top performer algorithms are reported in [32], [33], [34] and [35]. All are ranked by Middlebury benchmark [42]. Consequently, we have proved the claim by comparing the time and frame-rate with the top performer algorithms which demonstrated in **Table 3.1**. The disparity maps of the Middlebury datasets for Tsukuba head are estimated by proposed 2DRTSSA method are illustrated in **Fig. 3.3(b). Table 3.1** shows that the proposed 2DRTSSA algorithm outperforms the present top performer algorithms [32-35]. Moreover, the proposed method is faster than top performer algorithms. The accuracy of the proposed algorithm for Tsukuba head is 93.8% i.e., the bad pixel in percentage with the error threshold 2 is only 6.2%, which is almost the same of the top algorithms. Little variation of accuracy occurs due to orientations of pixel redundancy. The experimental results are analyzed in four phases are stated below-

3.4.1 Observation of 3D Reconstruction and Objects Recognition of Experimental Output

The Tsukuba stereo pair of input images contains different objects at different depth of positions. Background and foreground objects are situated at different depth. Almost overlapping objects are found in background of Tsukuba stereo pair those are occlusions and poor objects. Moreover, these stereo pair also contains some special regions like head of the statue, table lamp and video camera. These types of regions are really difficult to separate from other objects by stereo matching process. The first challenge was to distinguish the different depth by marking the different gray level value of output image. Nearest object is shown by more white color and farthest object is shown by dark grey level value or black. It is worth observing that the 3D structure of output image has been reconstructed clearly in **Fig. 3.3(b)** where the face of the statue, table lamp, video camera as well as interesting objects are recognized easily. Consequently, the camera and its nearest objects such as face (head) of

Tsukuba, table lamp are visualized by all most white color. On the contrary, the camera and its farthest objects such as video camera, book shelf, background wall of Tsukuba stereo pair are reconstructed with almost black color. Object borders are clearly recognized in estimated dense disparity image, i.e., border localization problem of article [13] are solved by the proposed method. The object borders of reference image are shown in **Fig. 3.3(a)**. The output image of **Fig. 3.3(b)** is further fed into the object-detection algorithm and the object borders of output image are identified, which are illustrated in **Fig. 3.3(c).**

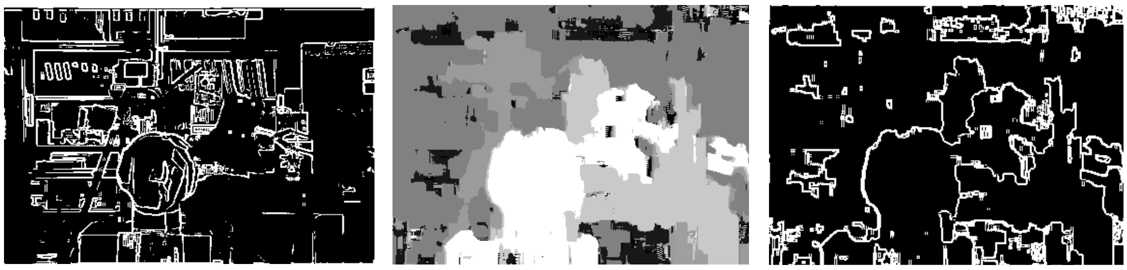

(a) (b) (c)

Fig. 3.3 Localized object borders.

The estimated dense disparity image is compared to ground truth image of Tsukuba head. The experimental results obtained by the 2DRTSSA method on Tsukuba head are very similar to their ground truth image. The estimated dense disparity 3D structure is recovered and its object borders are almost correctly identified which are given in **Fig. 3.3(c)**. The result ensures that the similar depths are found in estimated dense disparity, shown in **Fig. 3.3(b)**.

3.4.2 Computational Cost Calculation and Comparison with state-of-the-art Methods

Disparities of reference image are estimated by Sum of Absolute Difference (SAD) technique using 2DRTSSA search algorithm without any pruning for different window sizes. The disparities are estimated with the search range from -10 to +10. The effects of said search are investigated with respect to computational costs and frame-rate (in *fps*). The computational costs and performances of proposed 2DRTSSA method has been compared with other state-of-the-art methods [32-35]. The 2DRTSSA's experimental results have been compared with the result of methods those are tested on Middlebury standard datasets. The result of ranking in **Table 3.1** indicates that the proposed 2DRTSSA method is ranked *2nd* out of existing five state-of-the-art methods [13] and [32-35]. It shows the second highest frame-rate 223 *fps* and computational time 4480 μs among the top five latest methods with lower configuration of machine. So, it can be claimed that the proposed method is currently the state-of-the-art methods for Tsukuba head image pair with 44.64X, 2.00X, 1.56X, and 669.64X faster than the methods of [32], [33], [34] and [35] respectively. However, the proposed method is 1.38X slower than Fast Area Based method [13].

Table 3.1: Numerical comparison of proposed 2DRTSSA and present state-of-the-art methods.

Method's Name	**Machine Configuration**	**Computational time (in μs)**	**Frame-rate (in *fps*)**	**Accuracy (in %)**	**Rank**
Fast Area Based method[13]	2.3 GHz, Intel Core i-3, RAM: 4GB.	3229	310	86.10	1
2DRTSSA [Proposed]	**2.3 GHz, Intel Core i-3, RAM: 4GB.**	**4480**	**223**	**93.80**	**2**
Tree filtering [34]	1.8 GHz, Intel Core i-7, RAM: 4GB.	7000	143	93.18	3
Edge-aware Geodesic filter[33]	3.0GHz, Intel Core-i-5, Geforce GTX card. RAM: 8GB.	9000	111	93.67	4
DSI & Adaptive Support[32]	2.2 GHz, Core Duo, RAM: NA.	200000	5	90.18	5
Energy Minimization[35]	1.9 GHz, Intel Core i-5.RAM:6GB.	3000000	0.33	92.82	6

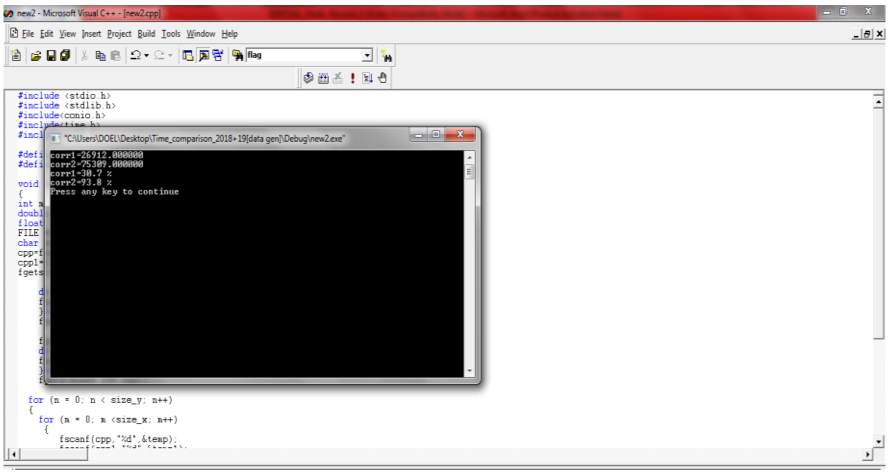
Table 3.2: Computational time reduction (%) compare to window-based existing methods.

Computational Time(in μs) of 2DRTSSA method [Proposed]	**Existing state-of-the-art Methods**		**Computational Time Reduction (in %) by 2DRTSSA method compared to the methods of *2nd* column**
	Method's Name	**Computational Time (in μs)**	
4480	Fast Area Based [13]	3229	-38.74
	Tree filtering [34]	7000	36.00
	Edge-aware Geodesic filter[33]	9000	**50.22**

The main achievement of this method is the improvement of accuracy and computational time reduction. The 2DRTSSA achieves 93.80% accuracy. The accuracy is enhanced by 63.80% compared to the RTA method. The recent state-of-art method, Edge-aware Geodesic Filter [33] that takes 9000 μs for execution and our proposed 2DRTSSA requires 4480 μs for the same resolution of Tsukuba head image. Thus the computational time reduction is 50.22% compared to the identical local stereo method [33], which is numerically figured out in **Table 3.2**.

3.4.3 Accuracy of the Proposed 2DRTSSA Method

The accuracy of this algorithm has been justified over standard stereo images of Tsukuba head. **Table 3.3** illustrates the accuracy of the proposed 2DRTSSA method applied on Middlebury standard stereo images of Tsukuba head.


new2 - Microsoft Visual C++ - [new2.cpp]
File Edit View Insert Project Build Tools Window Help
flag
"C:\Users\DOEL\Desktop\Time_comparison_2018+19[data gen]\Debug\new2.exe"

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include<time.h>
#incl
#defi
#defi

void
{
int m
doubl
float
FILE
char
cpp=f
cpp1=
fgets

    d
    f
    }
    f

    f
    d
    f
    }
    f

  for (n = 0; n < size_y; n++)
  {
    for (m = 0; m <size_x; m++)
     {
       fscanf(cpp,"%d",&temp);
```

```
corr1=26912.000000
corr2=75309.000000
corr1=30.7 %
corr2=93.8 %
Press any key to continue
```

Fig. 3.4 Run time snapshot of 2DRTSSA method for accuracy.

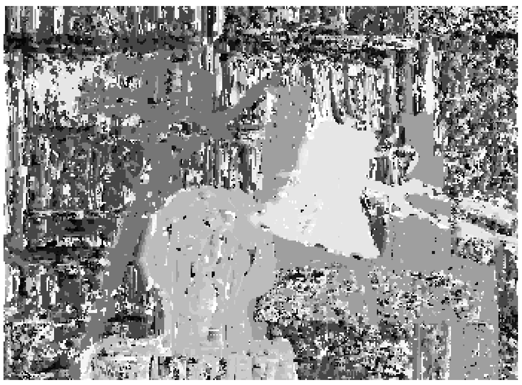


Fig. 3.5 Dense disparity map for window size 3×3.

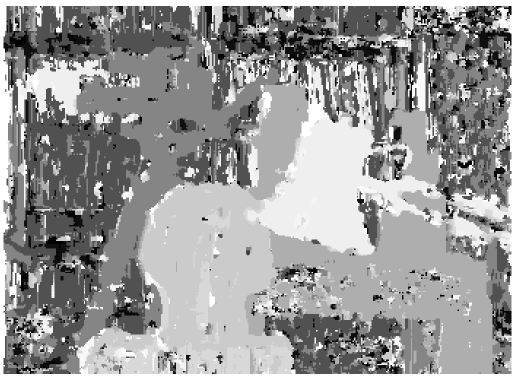


Fig. 3.6 Dense disparity map for window size 5×5.

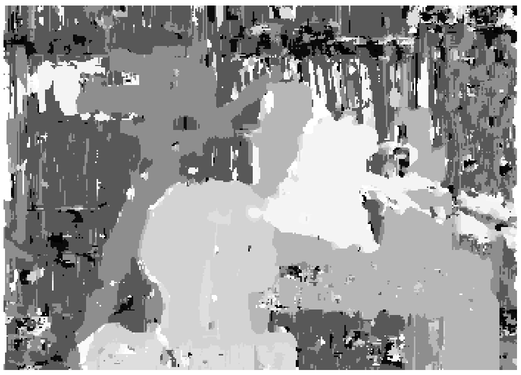


Fig. 3.7 Dense disparity map for window size 7×7.

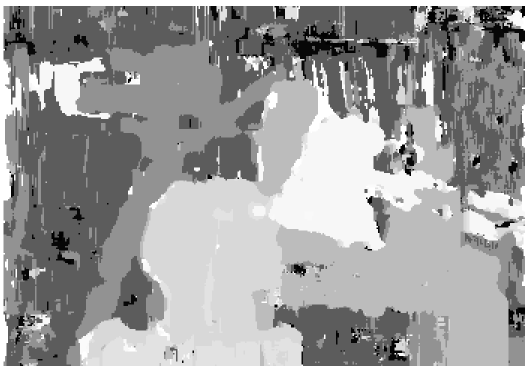


Fig. 3.8 Dense disparity map for window size 9×9.

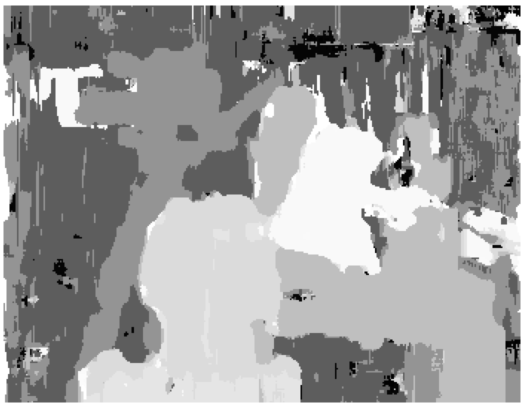


Fig. 3.9 Dense disparity map for window size 11×11.

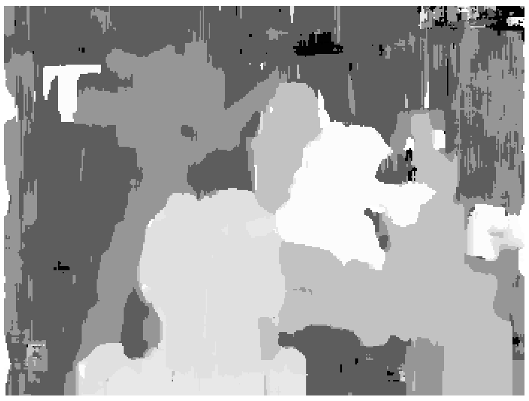


Fig. 3.10 Dense disparity map for window size 15×15.

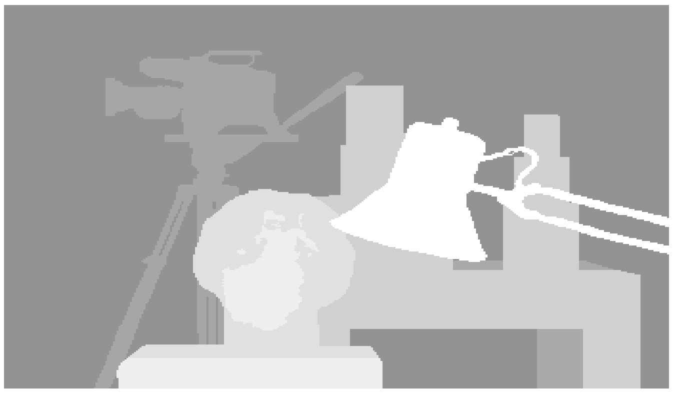


Fig. 3.11 Dense disparity map of ground truth image.

In order to estimate the accuracy of this method we have tested the results using different mask sizes which are mentioned from **Fig. 3.5** to **Fig. 3.10**. From **Table 3.3** the numerical evaluations confirm that the bad pixel is only 6.2%. But using the similar condition the bad pixels in percentage were 6.33%, 7.88%, and 7.18% reported in [33], [34] and [35] respectively for Tsukuba head.

Table 3.3: Accuracy of 2DRTSSA for Tsukuba stereo pair.

Window size (pixel)	**Accuracy (in %)**	**Bad Pixels (in %)**
3×3	73.3	26.7
5×5	79.8	20.2
7×7	92.1	7.9
9×9	93.2	6.8
11×11	**93.8**	**6.2**
15×15	83.6	16.4
17×17	83.3	16.7
19×19	72.9	27.1
21×21	72.4	27.6

We observe from the data of **Table 3.3** the accuracy is gradually increased if widow size increases. But this is valid from window size 3 × 3 to window size 11 × 11 only. The lowest accuracy started from 73.3% for 3 × 3 window size and the highest accuracy occurs at 93.8% for widow size 11 × 11 with bad pixel in percentage only 6.2%. Further increase in widow size (like 15 × 15, 17 × 17 etc.), the accuracy is decreased and it reaches at 72.4% for the widow size 21 × 21. The graphical interpretation is illustrated in **Fig. 3.12** that also prompted the *best operating window* size which is 11 × 11 for best accuracy.

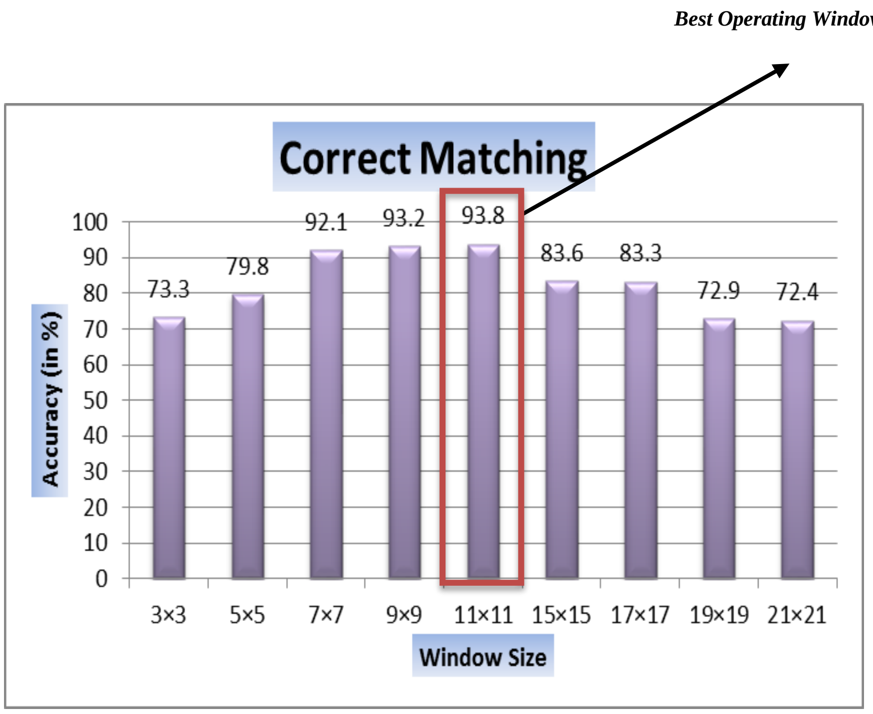


Fig. 3.12 Illustration of correct matching for estimated dense disparity with ground truth image of Tsukuba head.

3.4.4 Experiment on Real Stereo Images by 2DRTSSA

The performances of 2DRTSSA algorithm have been further justified on real stereo images acquisitioned by Logitech stereo web camera. This experiment is performed in our software laboratory and images were captured as indoor scenes. The specifications of stereo camera are listed below-

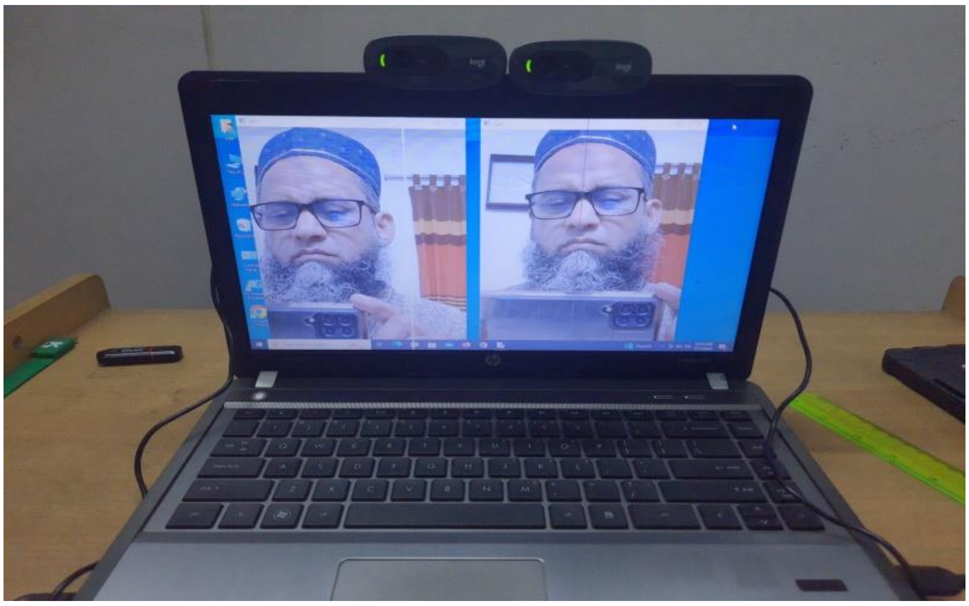
Brand: Logitech C270 Webcam

Country of Origin: Switzerland

Sensor Resolution (MP) - 3MP, Video Resolution (Pixel) - 1280 ×720, Frame Rate - 30*fps*. Diagonal Field of View 55°.

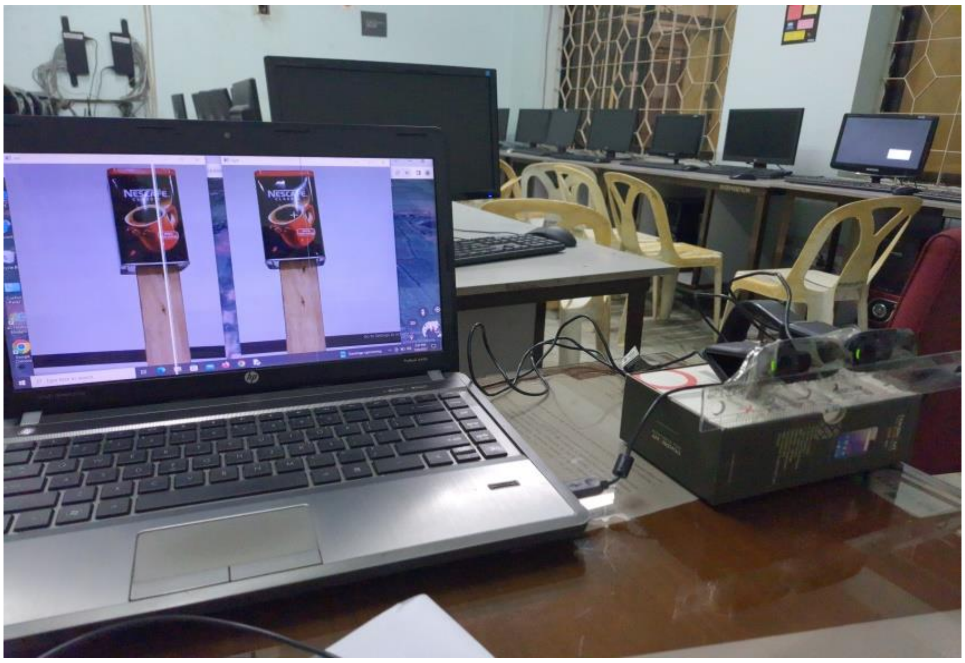
Stereo Image Capturing Process:

The main objects (Human face, Nescafe coffee stand and Scotch tape stand) of reference images were stood 63.00 cm away from the imaging sensor of the camera. The distance between two cameras was 6.70 cm

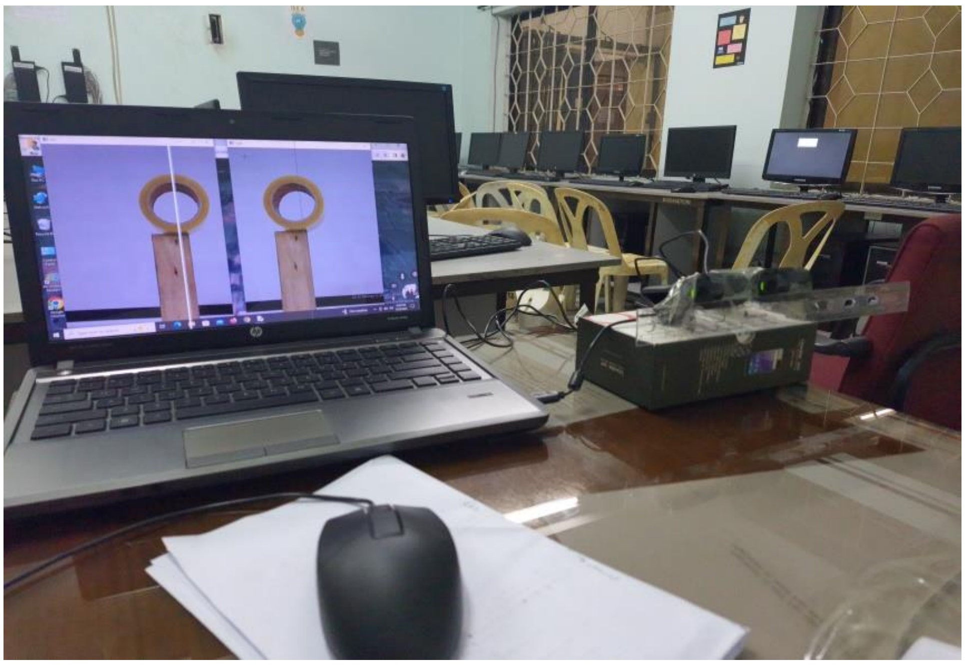


a)Real image acqusition using stereo web camera for dataset-1(Human face).

..



b) Real image acqusition using stereo web camera for dataset-2(Nescafe coffee stand).



c) Real image acqusition using stereo web camera for dataset-3(Scotch tape stand).

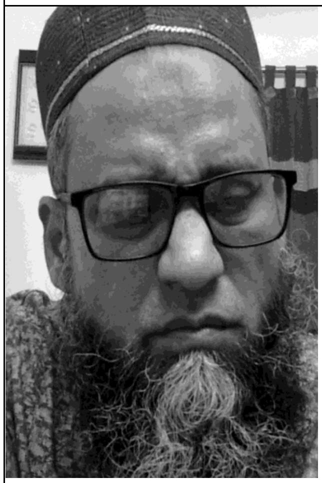
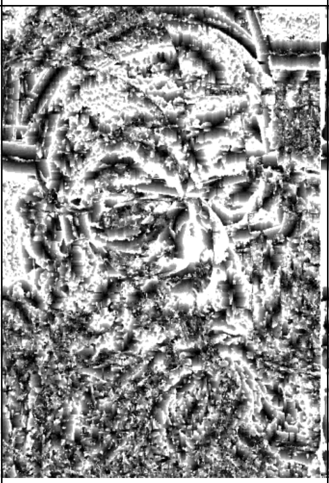
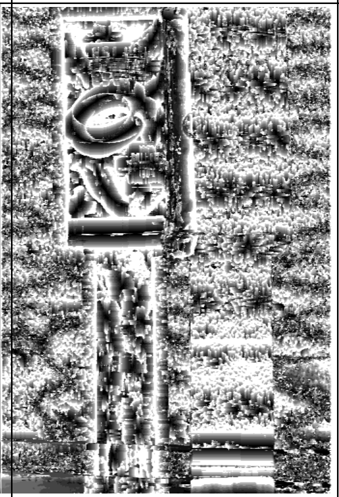
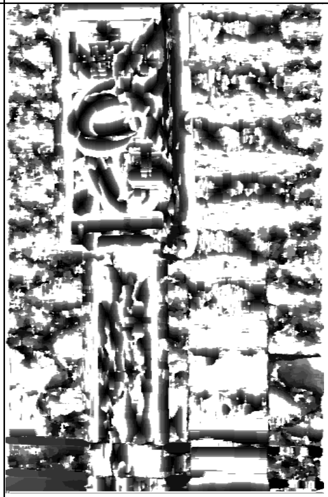
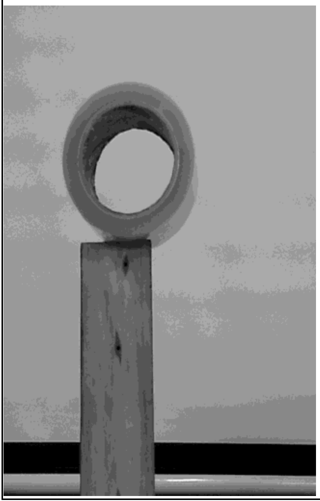
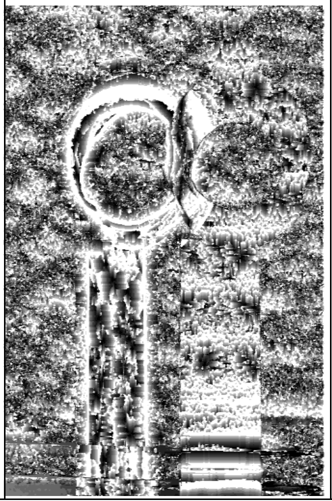
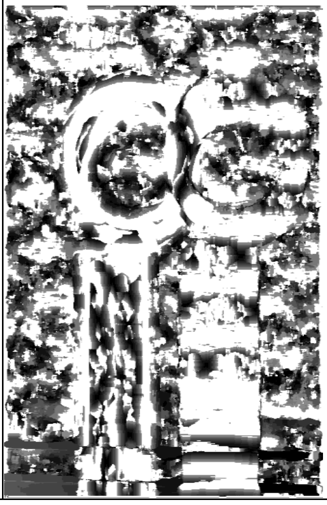
Fig. 3.13 Real image acqusition process using stereo web camera.

Experimental output for Real Stereo Images:

The size of the left and right real-image is (width × height) = (550 × 720) pixels. Disparity image size is (width × height) = (514 × 684) pixels. **Table 3.4** demonstrates the visual observation of dense disparity maps.

The **Table 3.4** illustrates the dense disparity maps of three real datasets- Human face, Nescafe coffee stand and Scotch tape stand respectively. We could not compare the experimental outputs to the ground truth image because it has no ground truth images. In this situation, the disparity maps of output image should be considered and compared visually only. The visual qualities of these disparity maps are not good. This is happened because we could not provide the equilibrium light condition in our laboratory. Similarly the room temperature of laboratory was not equilibrium at all the places during the image acquisition process. Moreover, we have tried to our best to calibrate the stereo camera physically. The stereo cameras were manually placed on the same horizontal line, but experimentally it was not possible. There was vertical difference between two cameras in fractional millimeter (.05 mm approximately) range. These cause to add a little noise in captured stereo images. Inspite of some noise, the objects are demarked and recognized at standard level. The object (Human face, Nescafe coffee stand and Scotch tape stand) inside the reference image is clearly understandable and visualized.

Table 3.4: Visual observation of disparity map of real images generated by 2DRTSSA algorithm

Reference image & Resolution: 550×720	Experimental Dense Disparity Map of Real Image Mask size : 3×3	Experimental Dense Disparity Map of Real Image Mask size : 11×11	Execution time(μs)
			Mask: 3×3 : 1344 Mask: 11×11: 14609
			Mask: 3×3 : 1312 Mask: 11×11: 14984
			Mask: 3×3 : 1531 Mask: 11×11: 15890

The dense disparity map generated by 3×3 mask is more visualized and comprehensive than the disparity map of 11×11 in noisy environment. So the overall performance of 2DRTSSA algorithm is good in case real stereo images.

3.5 Discussion

The main contribution of the proposed method is to increase the performance by reducing the computational cost. Our ultimate aim is to improve the strength of the window-based cost aggression method in order to use in real-time application. The frame-rate of our algorithm is 223 *fps* for input images of Tsukuba head image pair. Hence, it can calculate, process and display output 223 *frames/second* for the case of standard Tsukuba head image pair. The proposed method achieves 93.8% accuracy and enhances 63.80% accuracy compared to RTA method. We have implemented it by 2D parallel costs estimation to reduce the computational costs. Moreover, the 2DRTSSA algorithm does not require any additional hardware like 3D Graphics Processing Unit (GPU). The proposed 2DRTSSA method demonstrates the state-of-the-art results and exceeds most of the existing top methods.

3.6 Summary

A novel method is presented in this Chapter for dense disparity measurement with better improvement of accuracy. This pioneer method leads to compensate the accuracy which was degraded by RTA method of chapter 2. The proposed 2DRTSSA method is described in Section 3.2. According to 2DRTSSA method two simultaneous instruction calculations (in *C++* code) executed in one instruction cycle. Experimental results described in Section 3.4 demonstrate the visual and numerical comparison between the proposed method and top five state-of-the-art methods. The accuracy of 2DRTSSA method is well discussed in subsection 3.4.3. The impact of real stereo images by 2DRTSSA algorithm is reflected in subsection 3.4.4. One of the most appealing aspects of this method is to reduce the computational cost and improvement the accuracy for real-time applications. A significant advantage in terms of implementation is the fact that 2DRTSSA algorithm performed 50.22% of computational time reduction and improved 93.8% accuracy, which means that the algorithm is going to reach at the door of real-time applications.

Chapter 4

Stereo Correspondence Estimation Technique: Self-Adaptive Algorithm (SAA)

4.1 Motivation of SAA Method

We have improved the accuracy of stereo matching by the 2DRTSSA method, which is described in Chapter 3. The accuracy of 2DRTSSA algorithm is raised to 93.8% from 30% and computational time increased to 4480 μs from 203 μs compared to RTA algorithm. In this chapter, we propose a new state-of-the-art method called *Self-Adaptive Algorithm* (SAA) which is far better than 2DRTSSA. The idea behind the SAA method described in the following sections.

4.2 Proposed Self-Adaptive Algorithm

A new stereo imaging search technique has been introduced in this chapter. In traditional window-based searching algorithm, a particular pixel L(x, y) is selected by search method along the corresponding epipolar line in the right image within the search range from $-d_{max}$ *to* $+d_{max}$. Let us assume that the left image is the reference image. So, for most pixels in the left image, there is a corresponding pixel in the right image within a search range from $-d_{max}$ *to* $+d_{max}$.

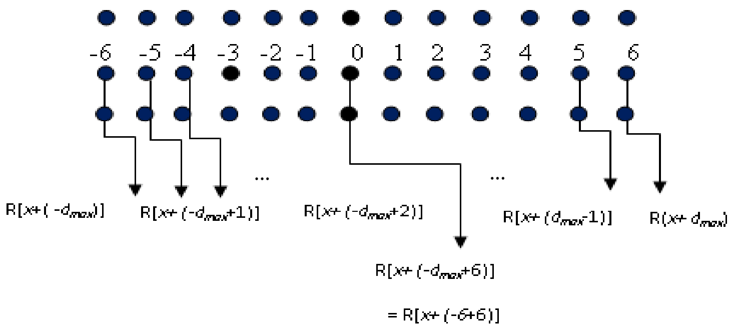


Fig. 4.1 The total search regions of right image for the particular pixel of L(x,y).

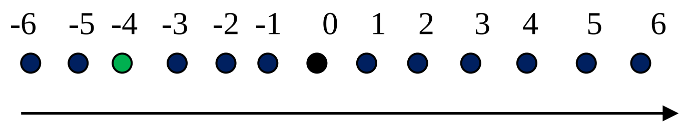


Fig. 4.2 First search interval $\{R(x+(-d_{max})\}$ … $R\{x+(+d_{max})\}$

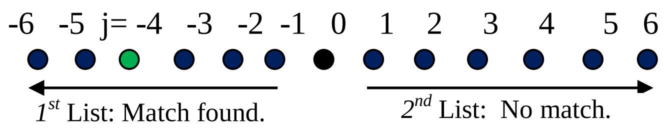


Fig. 4.3 Search range separated by *1st* list and *2nd* list with their candidate-pixels.

First search matching occurs into the *1st* list and the matching pixel indicated by the green color at position $j = -4$.

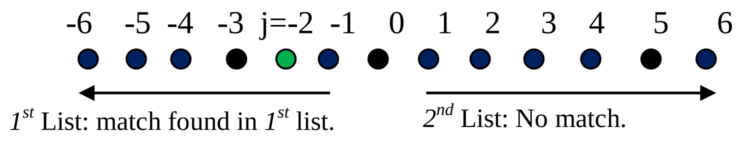

-6 -5 -4 -3 j=-2 -1 0 1 2 3 4 5 6
1st List: match found in 1st list.
2nd List: No match.

Fig. 4.4 Second search occurs in *1st* list too, and the matching pixel is indicated by green color also.

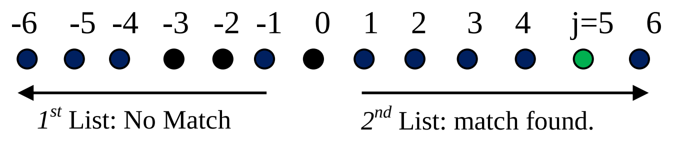


Fig. 4.5 *3rd* search occurs in *2nd* list, match indicated by green color.

Accordingly, the first search of first reference pixel of L(x - d_{max}) is searched to the right image from R[(x +(-d_{max})] , R[x+ (-d_{max}+1)] , R[x+ (-d_{max}+2)] … R[x+ 0)] … R[x+ (d_{max} - 1)] to R[(x + (+d_{max})]. During the matching process the algorithm finds the best candidate-pixel by evaluating its window costs within the interval [R{x + (-d_{max}), y} ... R{x + (+d_{max}), y}]. The method is visually explained in **Fig. 4.1** and **Fig. 4.2** by mortars its coordinate's pixel. Suppose the window cost function $f(wc) = \{wc_1, wc_2, wc_3...wc_n\}$. Let $wc_2 < wc_1$ so the best match occurs for the cost function $f(wc_2)$ and the function associated with the corresponding pixel of right image say R(x_2, y) to indicate that this match from left to right has been established. Assume another pixel R(x_4, y) is associated with the cost function $f(wc_4)$. If $f(wc_4)$ has better score than the previous $f(wc_2)$. *i.e.,* $f(wc_4) < f(wc_2)$, this algorithm will reject the score of wc_2 and will accept wc_4. Therefore, the function $f(wc_4)$ associated with the corresponding pixel of right image say R(x_4, y) indicates the new matching establishment. Thus the coordinate distance from R(x, y) to R(x_4, y) is the final disparity of reference pixel of L(x - d_{max}). The process is then repeated for the successive pixels of reference image along with the scan line from left to right of the whole image.

According to SAA method, if the matching pixel co-ordinate R(x_4, y) of first search is "*j*" (illustrated in **Fig. 4.3**), indicates the x-position of matching point in the *1st* list. In the proposed approach, we use prior knowledge for *2nd* reference pixel matching. In the proposed method we assume that the *2nd* reference pixel obviously resides surrounding to the *jth* position, i.e., within the *1st* list, this is because, most of the cases, the neighbor pixel's photometric properties are approximately same as L(x - d_{max}). So, the searching for *2nd* reference pixel will be within the *1st* list too. This search matching procedure is depicted in **Fig. 4.4**. In case, if the *2nd* reference pixel does not match with the candidate-pixels of *1st* list, then the SAA search sequence goes to new searching zone, i.e., in the *2nd* list with readjusting the search interval. Accordingly, the SAA algorithm will calculate the window costs of *2nd* reference pixel in *2nd* list. This procedure is outlined in **Fig. 4.5**. In this case, the candidates-pixels of *1st* list are not taken into consideration for matching process. Next search for *3rd* reference pixel occurs again in *2nd* list too (which is not shown here). In case of no match in *2nd* list for *3rd* reference pixel, the program search sequence goes to *1st* list by resetting the new starting and ending points of *1st* list. The searching procedure of *3rd* reference pixel will be the same as illustrated in **Fig. 4.4**. Similarly next search for *4th* reference pixel occurs in *1st* list also as per base criteria of the proposed approach. So the matching procedure occurs either in *1st* list or in *2nd* list and so on.

The above stated SAA approach is repeated for the successive pixels of reference image along with the scan line from left to right on the whole image. The algorithm divides automatically the search interval $[R\{x + (-d_{max}), y\} \ldots R\{x + (+d_{max}), y\}]$ into two regions; *1st* list and *2nd* list. The *1st* list ranges from $-d_{max}$ to 0, while *2nd* list from 0 to $+d_{max}$. The capability of proposed SAA algorithm is to adapt itself the search range automatically (either in the *1st* list or *2nd* list). This process reduces the searching cost around 50% in each iteration. The proposed approach relies on j^{th} position of x axis. For a reference pixel (x, y) in the left image, this procedure is repeated for successive pixels in interval $[R\{x + (-d_{max}), y\} \ldots R\{x + (+d_{max}), y\}]$ along with the scan line, and the process is iterated for the whole image. In this paper $(2n + 1) \times (2n + 1)$ mask size is used to estimate the window cost of each pixel; where $n = 1,2,3 \ldots k$. We use $n = 1$, i.e., 3×3 mask is employed in the following figures of this section for making the process easy to understand. However, in the real image the value of n is 5. The right image is scanned by this mask from left to right and from top to bottom during the matching process.

Suppose, $d_{max} = 6$ and $j = 0$, the algorithm resets the starting point at $d_{max1} = -d_{max}$ and ending point $d_{max2} = +d_{max}$. In the beginning, the matching process occurs on the full scan line. Actually, it is treated as *1st* search, where window cost calculations are shown in **Fig. 4.6**. There are thirteen candidates-pixels in right image along the scan line. For each pixel, the window cost is calculated according to the SAD method. So, there are thirteen window costs that have been extracted at *1st* search, although five window costs are shown in **Fig. 4.6** for simplicity. The proposed algorithm arranges the window cost functions in ascending order. Suppose the order for *1st* search is like as $f(wc) = \{wc_2 < wc_1 < wc_3 < ... < wc_{13}\}$. Hence, the proposed method rejects all other window costs except wc_2. Best match occurs for *2nd* window due to its minimum window cost. Thus, the first reference pixel $L(x - d_{max})$ matches with the candidate-pixel (center pixel) of the second window of right image (indicated by green color).

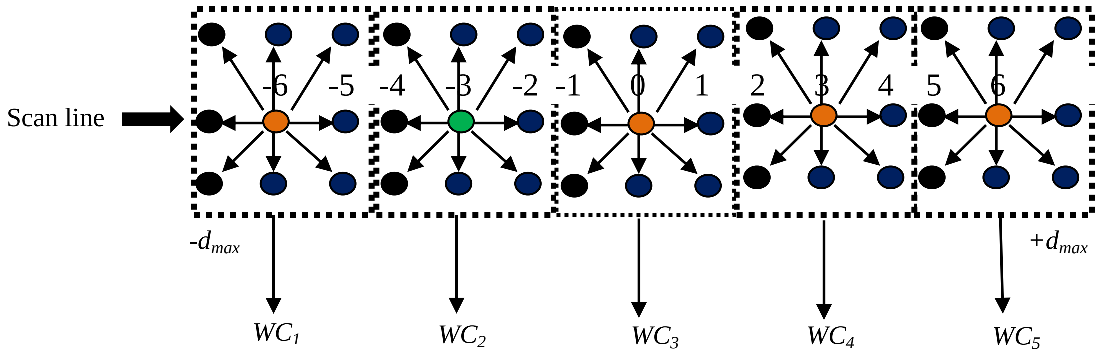


Fig. 4.6 Window cost estimation process over the scan line (*1st* search).

Thus the first matching pixel's position is j = -3 illustrated in **Fig. 4.6**. Since j = -3 (i.e. j < 0) the algorithm resets the search interval $d_{max1} = -d_{max}$ to $d_{max2} = 0$ for *2nd* search. In this region only seven pixels are participants, rest of the candidate-pixels of *2nd* list are ignored for window cost calculations. This idea is illustrated in **Fig. 4.7**.

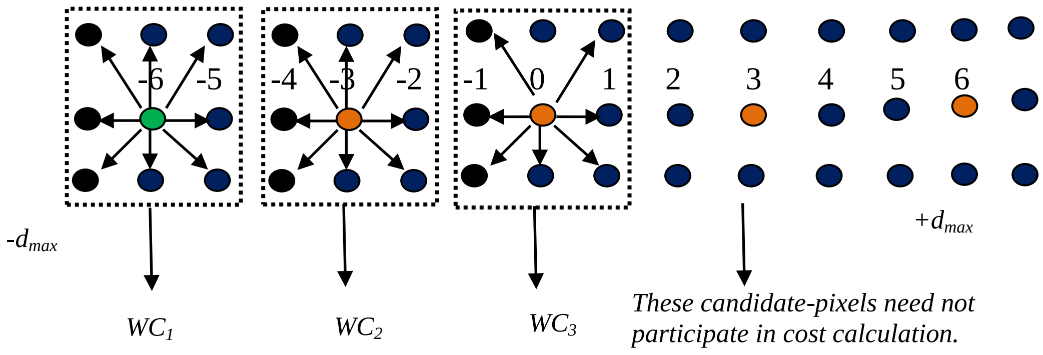

-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
$-d_{max}$
$+d_{max}$
WC_1
WC_2
WC_3
These candidate-pixels need not participate in cost calculation.

Fig. 4.7 Window cost calculation process for *1st* list only (*2nd* Search).

Consequently, there are only seven window costs will be estimated from the *2nd* search, although three window costs are shown in **Fig. 4.7** for simplicity and well understanding. Suppose the order for *2nd* search is like as $f(wc) = \{wc_1 < wc_4 < wc_3 < wc_2 < wc_5 < wc_6 < wc_0\}$. Hence the proposed method rejects all other window costs except wc_1. In this case, the best match occurs for *1st* window cost in *1st* list due to its minimum window cost. Thus, the second reference pixel $L(x - d_{max} + 1)$ matches with the candidate-pixel (center pixel) of the first window of right image (indicated by green color). Since the second reference pixel's $L(x - d_{max}+1)$ matching position is j =

-6 ($j < 0$) shown in **Fig. 4.7**, therefore, the third search (not shown here) will be occurred into *1st* list too. If it falls into a critical situation in the *4th* search, i.e., pixel L(x - d_{max} + 1) does not match with the candidates-pixels of *1st* list, the algorithm readjusts the new searching interval $d_{max1} = 0$ *to* $d_{max2} = +d_{max}$ and the searching sequence goes to *2nd* list. The proposed approach calculates the window costs in *2nd* list using the same procedure applied in the *2nd* search. The cost calculations are outlined in **Fig. 4.8** where the starting and ending points are different from *2nd* and *3rd* search; although three window costs are shown for simplicity. Actually, seven window costs have been estimated in the algorithm.

From the above discussion it is seen that, the proposed SAA method utilizes the prior knowledge of matching-pixel position in the x axis and accordingly it redirects the searching sequence in the *1st* list or in the *2nd* list. As the matching procedure occurs either in *1st* list or in *2nd* list, it reduces around 50**%** of searching costs.

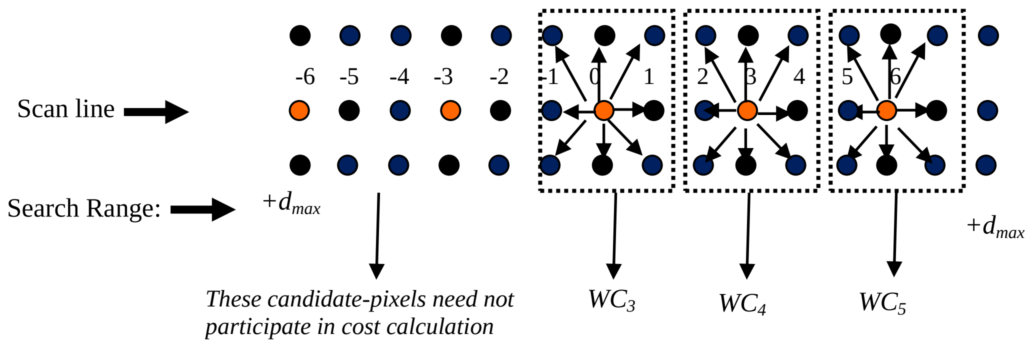


Fig. 4.8 Window cost calculation process for *2nd* list only (*4th* search).

4.2.1 Disparity Estimation Algorithm of SAA Method

```
Algorithm SAA(m, n, temp, temp1,sum,w1,w2 ,k1,k2,v_left,d,
v_right,image_left.pixel,image_right.pixel,image_disp.pixel
ws1,ws2,dmax,dmax1,dmax2,t0,t1,flag)

    1. //m,n is the row and column size of an image.
    2. // temp and temp1 are pixels intensity value of left and
    3. // right image.
    4. //w1,w2 is row and column size of the mask.
    5. //sum is the summation of window costs.
    6. // v_left is the pixel intensity value of left image.
    7. // v_right is the pixel intensity value of right image.
    8. // image_left.pixel[1:m][1:n] is the left image pixel
    9. // coordinate that contains m×n elements.
   10. //image_right.pixel[1:m][1:n]is the right image pixel
   11. // coordinate that contains m×n elements.
   12. // d is the search range counter variable.
```

```
//k1 ,k2 are the number pixels to discard from left and
// right side of image.
//image_disp.pixel[1:m][1:n]is the disparity image
// that contains m×n disparity values.
// ws1 and ws2 are the local variable within the mask.
//dmax,dmax1 and dmax2 are the search ranges.
// t0, t1 is the variable for time.
//i,j and flag is the integer type counter variables.
     for n:=0 to size_y do
     {
      for m:=0 to size_x do
        {
       image_left.pixel[m][n]:= temp; // Read left image
       image_right.pixel[m][n]:= temp1;//Read right image
        }
      }
  t0:= clock();
   //Region selection.
     ws1 := (w1/2); ws2 := (w2/2);
     for m:= k1 to size_x-k1 do
     {
        for n:= k2 to size_y-k2 do
          {
     if(flag=0) then
       {   dmax1:=-dmax;// initial search region.
           dmax2:= dmax;
        }
     else if (flag<0) then
       {   dmax1:=-dmax;// 1st region.
           dmax2:= 0;
        }
    else
          {dmax1:= 0; // 2nd region.
           dmax2:= dmax;
           }
     for d:= dmax1 to dmax2 do //computing window cost
       {
          sum:= 0;
             for i:=-ws1 to ws1 do
               {
                     for j:= -ws2 to ws2 do
                  {
                 v_left:= image_left.pixel[m+i][n+j];
                v_right:= image_right.pixel[m+i+d][n+j];
                 sum:= sum + abs(v_left - v_right);
             }
          }
        Mtemp[d + dmax].pixel[m][n]:= sum;
  }
        // Select the minimum window cost.
     image_disp.pixel[m][n]:= minimum(Mtemp,m,n,ws1);
        }
     }
     t1:= clock();
     cpu_speed:= t1 - t0; // time calculation.
     write("Total time",cpu_speed);
     // Creating the dense disparity map.
     for n:= k1 to size_y-k1 do
       {
       for m:= k2 to size_x-k2 do
        {
  write("dense disparity image",image_disp.pixel[m][n]);
    }
  }
```

Algorithm minimum (temp[2*dmax+1],x,y,ws)

```
// Find the minimum value from temp[0:2*dmax+1]
// x, y, i, j, mu, a, len are the integer variables.
{ j:=1;
 len:= 2*dmax+1;
 if (flag ≠ 0 ) then
   len:= dmax+1;
    for i:=0 to len do
      {
       if(temp[i].pixel[x][y] < temp[j].pixel[x][y]) then
           j:= i;
       }
    if (j < 10)then // selecting the search region.
       flag:= -1; // for 1st region.
    else
       flag:= 1; // for 2nd region.
// return the best matching co-ordinate distance.
      mu:= abs(j-dmax);
    return (mu);
}
```

The above procedure shows the calculation of window cost for one reference pixel only. The next matching position range is selected by setting the flag pointer either -1 or +1 with the following statements:

```
if(j < 10)
  flag = -1;
else
 flag = 1;
```

Flag value controls the reduced matching range or searching area and thus the computational costs will reduce always.

4.3 Comparison with Existing Matching Algorithms

The proposed matching mechanism is compared with the some of the very popular and established stereo matching methods. They are 1) A Fast Area Based Algorithm 2) Bidirectional Matching or left-right checking 3) Window-Based Fast Algorithm and 4) Hierarchical Disparity estimation. A Fast Area Based approach is kept trace previously matched points[13], while the Bidirectional Matching calculates every possible combination of matches from left to right and right to left. Window-Based Fast Algorithm uses the same idea of first method but it additionally uses the threshold techniques [9]. The hierarchical disparity estimation calculates disparities either for rectified stereo images or uncalibrated pairs of stereo images without known epipolar geometry [28]. This method also uses bidirectional matching to remove the false matches. The coloring area of **Fig. 4.9** defines the probable matching points of right image.

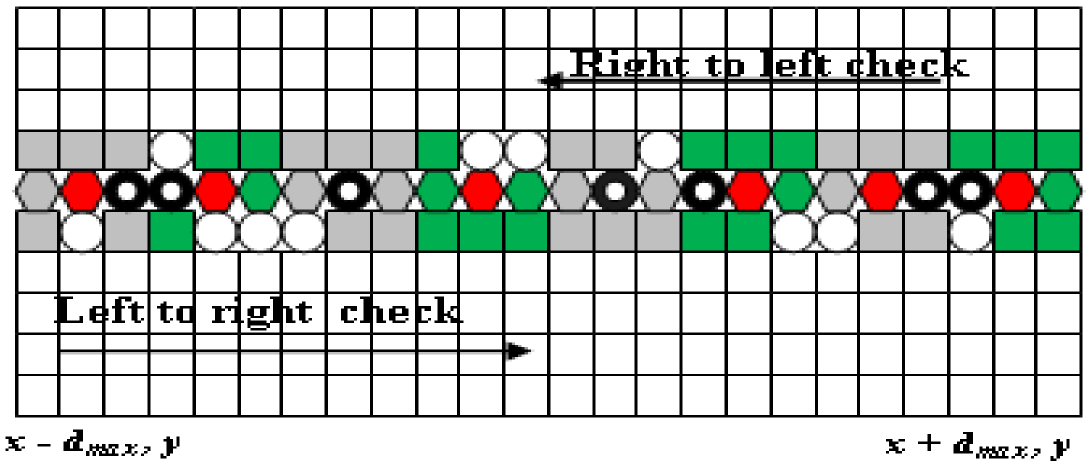


Fig. 4.9 Computational path of bidirectional and unidirectional matching from the computational point of view.

For each point in the left image, bidirectional matching chooses in the direct phase the best score along a middle row in the color area (the matches found when matching left-to-right have been marked with a circle in the **Fig. 4.9**). Then, in the reverse

phase, when matching R(x, y) chooses the best score along the middle row level: a match is accepted only if the match found along this path turns out to be one of those found when matching left-to-right. It is worth noticing that, although during the reverse phase bidirectional matching checks all of the potential matches along the path in middle horizontal row, the allowed ones for R(x, y) turn out to be only those that in the direct phase fall in the middle horizontal row (i.e. the circles lying in the middle horizontal row) [13].
The comparisons of the proposed SAA method and recent related works [29-39] are explained briefly in the last portion of subsection 1.2.4 in Chapter 1.

On the contrary, in our proposed adaptive matching method, the best match occurs by dynamically readjusting the starting and ending points of search region. According to the main concept stated in section 4.2, matching pixel is the nearest neighbor of R(x, y). It is important to mention here that, the *1st* search occurs on all the pixels ranges from R$\{x + (- d_{max}), y\}$ *to* $R\{x + (+d_{max}), y\}$. However, the second, third and consecutive searches are to be adapted according to the best match of the previous search. That is, by the completion of first search the proposed algorithm remembers the position of matching pixel. So, in the second search occurs surrounding to the first matching pixel, as we use the concepts that neighbor pixels have the same photometric properties. Hence depending on the position of matching pixel, the successive search ranges are reselected adaptively. This adaptive matching procedure ultimately is reducing the searching range by half of its original size. This is the main contribution of this research.

The *left-right check* has proven to be particularly effective in detecting and discarding the erroneous matches but it requires two matching phases (direct and reverse). This implies doubling the computational complexity of the matching process. The Fast Area Based Algorithm is based on a matching core that it does not require a reverse matching phase but some details such as for example the lamp’s wire(Tsukuba pair), the lamp’s switch and the two roads that sustain the lamp, have been vanished. Moreover, the disparity map shows the *border-localization* problem, i.e. the objects’ borders are not perfectly localized with respect to their original position [13]. This algorithm requires only direct matching phase. The hierarchical method executed on DirectX 8.1 class 3D hardware (ATI Radeon 9000 Mobility).The disparity map is verified by bidirectional procedure. Window- Based Fast Algorithm uses the different threshold values like 10, 20, 30, 40 and 50. From the experimental result, we found that as soon as the threshold value increased the searching range also increased, causes to high computational costs.

Conversely, the proposed SAA method shows lowest computational cost because, it involves only minimum matching spaces. Suppose there are *n* candidate-pixels appear on the searching range with *n* reference pixels in left image. According to the above analysis, the Fast Area Based Algorithm requires n^2 numbers of matching comparisons and the Bidirectional or Hierarchical search requires $2n^2$ matching comparisons. The proposed SAA method requires only $n+ \{(n/2) \times (n-1)\}$ matching comparisons.

4.4 Optimization of Self- Adaptive Search

The important part of SAA approach is optimization technique. We have already mentioned around 50% window costs are reduced at every reference pixel of left image except first pixel $L(x - d_{max})$. First reference pixel traverses from $R(x - d_{max})$, $R(x - d_{max} + 1)$, $R(x - d_{max} + 2)$ ... $R(x + d_{max})$ along the scan line. The first search occurs from $-d_{max}$ *to* $+d_{max}$ over the scan line. In this case, for first search window costs have been calculated for every candidate-pixel of right image. Suppose the first reference pixel matches with the third pixel($x - d_{max} + 2$) of right image. After tracking the first matching, the proposed method divides the searching space into two regions in right image: i) 1st region and ii) 2nd region. Accordingly, each region contains total d_{max} +1 numbers of candidate-pixels of search range.

Let w is the square mask size and matching range is $-d_{max}$ to $+d_{max}$. So the *first* searching computational cost of SAA method, $C_1= \{(w^2-1) \times (2d_{max} +1)\}$.

Second computational cost,

$$C_2 = [(w^2-1) \times \{(2d_{max}+ 1) - 1\}/ 2 + 1] = (w^2-1) \times (d_{max} + 1).$$

Third, fourth ... n^{th} searching computational costs will be the same i.e., $C_n = (w^2-1) \times (d_{max}+1)$.

Fast Area Based Algorithm [13] requires the computational cost for each reference pixel, $C_{AB}= \{(w^2-1) \times (2d_{max} +1)\}$.

The Hierarchical Disparity [28] or left-right checking algorithm requires the computational cost for each reference pixel, $C_{HD} = 2\times(w^2\text{-}1)\times(2d_{max}+1)$.

For the image size $M \times N$, the total computational costs of Fast Area Based Algorithm [13] is $C_{AB} = (M \times N)\times(w^2\text{-}1)\times(2d_{max}+1)$. Total computational costs of Hierarchical Disparity [28] is $C_{HD} = 2\times[(M \times N)\times(w^2 - 1)\times(2d_{max}+1)]$. The computational costs of proposed SAA method, $C_{SAA} = (w^2\text{-}1)\times\{(2d_{max}+1)+(M\times N\text{-}1)\times(d_{max}+1)\}$.

Suppose there are n reference pixels within the search range $-d_{max}$ *to* $+d_{max}$. If $n = 21$ reference pixels of left image, then the following comparisons are calculated.

- In a stereo matching, searching range varies from *-10 to +10* normally. Let $d_{max} = 10$. The total comparisons for n (here $n = 21$) reference pixels of proposed SAA method = $(2\, d_{max} + 1) + (n\text{-}1)\times(d_{max}+1) = (20 + 1) + (21\text{-}1)\times 11 = 241$.
- Fast Area Based Algorithm [13] requires the total comparisons = $n \times (2\, d_{max} + 1) = 21\times 21 = 441$.
- Hierarchical Disparity method [28] or Bidirectional method requires the total comparisons = $2\times n \times (2d_{max}+1)$. $= 2\times21\times21 = 882$
- Subsequently, comparisons reduction compared to the Fast Area Based algorithm = (220/441) × 100% = 45.35%
- Reduction of comparisons compared to Hierarchical Disparity method or Bidirectional method = (641/882) × 100% = 72.67%

Flag value always controls the matching range by resetting the parameter d_{max1} and d_{max2} that depends upon the previous matching position of x. From d_{max1} to d_{max2} for every window cost is determined by summing up the absolute differences between two pixels by the following pseudo statements-

```
 sum = sum + (abs(v_left - v_right));
```

The values of parameters `v_left and v_right` *of* $(i, j)^{th}$ pixel is set up by -

```
 v_left = image_left.pixel[m+i][n+j];
 v_right= image_right.pixel[m+i+d][n+j];
```

where i and j varies from -5 to +5 for a mask size 11 × 11. The window costs are stored in an array `Mtemp[`$d+d_{max}$`].pixel[`m`][`n`]`.These cost values are gone to minimum cost function as arguments. The minimum function implements the key idea of the proposed method by the assessment of its comparisons among the window costs. The minimum function performs the dual tasks i) it can be able to determine the matching position on ***x*** axis by setting $j = i$ and ii) find the reduced new search range with the following pseudo statement-

```
i) if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
       j = i;
```

and

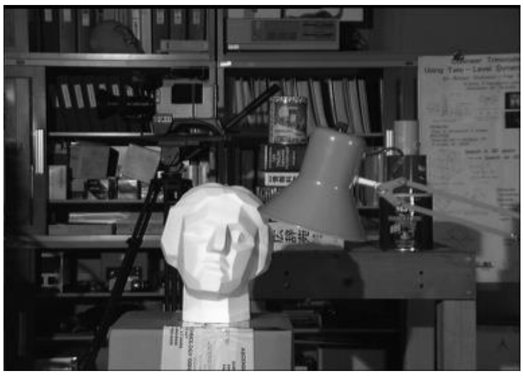
ii) The SAA algorithm selects the upcoming search range by setting the flag value either -1 or +1 with the following statement-

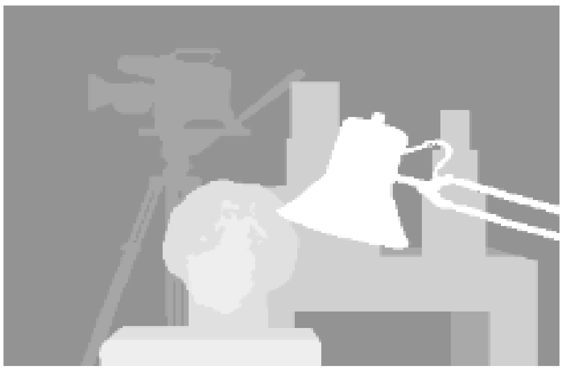
```
if(j < 10)
   flag = -1;
else
   flag = 1;
```

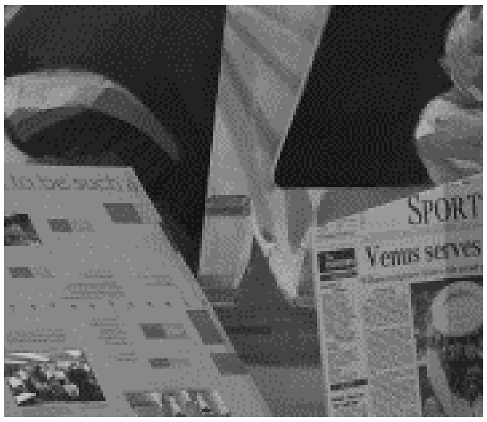
Flag value controls the reduced matching position range and thus the computational costs are low by reducing the number of comparisons.

4.5 Experimental Settings and Results

The experiments have been performed on two Middlebury standard stereo images: i) Tsukuba stereo pair and ii) Venus stereo pair as shown in **Fig. 4.10**. The computational time, frame-rate, accuracy and gain performances of the proposed algorithm have been justified over the standard stereo images. Experiments are performed on Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM. The algorithm has been implemented using Visual C++ programming language with windows 10 operating system. The size of the left and right images of Tsukuba head is (width × height) = (384 × 288) pixels and, the ground truth image size is (width × height) = (348 × 252) pixels. The size of the left and right images of Venus stereo is (width × height) = (434 × 383) pixels and the ground truth image size of Venus is (width × height) = (348 × 252) pixels. Mask size 11 × 11 is used for every operation in this method. The experimental results state that the SAA algorithm is currently the best window- based method among the existing state-of-the-art methods. The top performer existing algorithms are reported in [31], [32], [33], [34] and [35]. All are ranked by Middlebury benchmark [42]. So we have to prove the claim by comparing the time and frame-rate with the top performer algorithms which is compared in **Table 4.1** and **Table 4.3**. The disparity maps of the Middlebury datasets for Tsukuba head and Venus stereo pair are estimated by proposed SAA method are illustrated in **Fig. 4.11** and **Fig. 4.12** respectively.


(a) Left image of Tsukuba Head.


(b) Ground Truth of Tsukuba Head.



(c) Left image of Venus stereo pair.

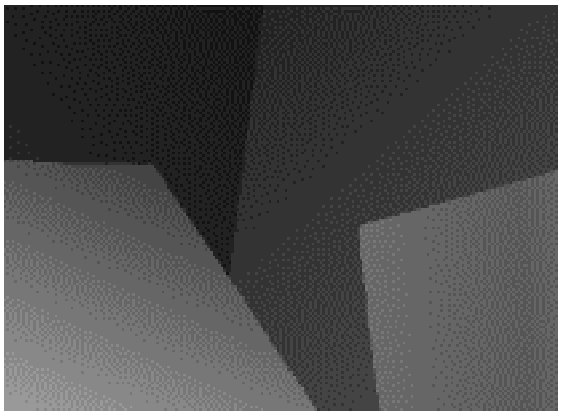
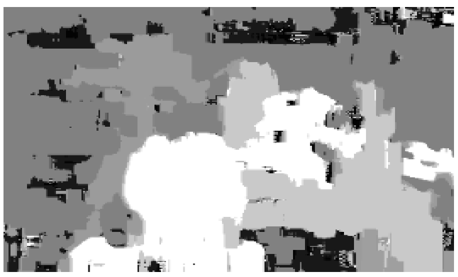

(d) Ground Truth of Venus stereo pair.

Fig. 4.10 Standard Stereo image (Reference image) and their ground truth image.


(a)3D dense disparity map using mask size 11× 11.

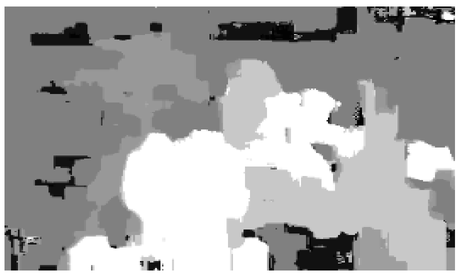
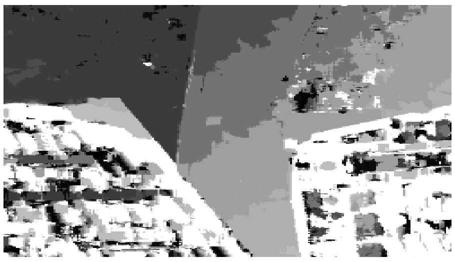

(b) 3D dense disparity map using mask size 15× 15.

Fig. 4.11 Estimated 3D dense disparity map of Tsukuba head using SAA method.


(a) 3D dense disparity map using mask size 11× 11

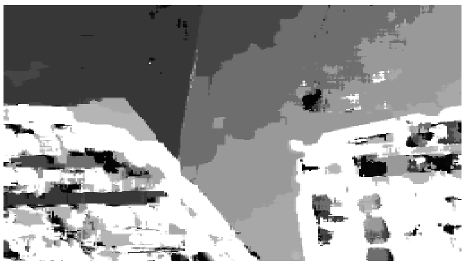
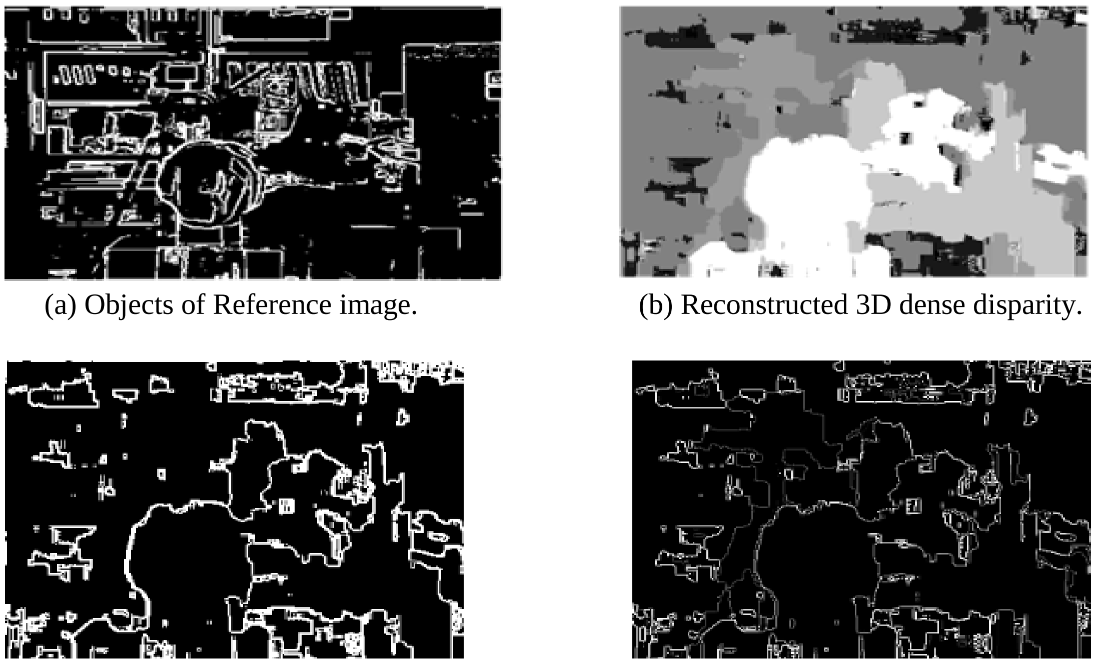

(b) 3D dense disparity map using mask size 15× 15.

Fig. 4.12 Estimated 3D dense disparity map of Venus stereo using SAA method.

The corresponding time reductions of SAA method are compared numerically in **Table 4.2** and **Table 4.4**. From these tables it is evident that in both cases the proposed SAA algorithm outperforms the current and earlier established top performer algorithms. The accuracy of the SAA algorithm for Tsukuba head is 93.8% i.e., the bad pixel in percentage is only 6.2%. The performance enhancement by the SAA method is discussed in section 4.6. **Table 4.5** and **Table 4.6** show the noteworthy gain enhancements of proposed SAA method. The experimental results are analyzed in four phases are explained below.

4.5.1 Observation of 3D Structures of Experimental Output

Both the Tsukuba and Venus stereo pair of input images contain different objects at variable depth of positions. Contextual and forefront objects are positioned at different depth. Four objects are placed at different depth of Venus stereo of input image. Stereo pair also encloses some distinct areas like head of the statue, table lamp, video camera, Venus sport paper, and another paper and background paper wall. Such types of areas are quite challenging to isolate from other objects by stereo matching. So the first work is to differentiate the variable depth of objects by assigning the altered gray level value of output image. Nearby object is presented by deep white color and outermost object is presented by dark or deep black color. It is detecting that the experimental 3D construction of output image is recreated evidently in **Fig. 4.11** and **Fig. 4.12,** where the face of the statue, table lamp, video camera, Venus sport paper, another paper as well as remarkable objects are seen easily. Comparing the output images of **Fig. 4.11(a)** and **Fig. 4.12(a)** with ground truth image of **Fig. 4.10(b)** and **Fig. 4.10(d)** respectively, the camera and its background objects have been recovered almost correctly. The objects depths are noting that nearby objects are realized by additional white color and outermost objects are realized by dark grey level value or black as shown in **Fig. 4.11** and **Fig. 4.12.** Thus, the camera and nearby objects such as face of Tsukuba, table lamp and sports paper, *2nd* paper (to be such: left side) of Venus stereo are visualized by more white color. On the other hand, the camera and outermost objects such as video camera, book shelf, background wall of Tsukuba stereo pair and background paper wall of Venus stereo are recreated with dark grey or black color. Object borders are evidently detecting in computed dense disparity image, i.e. border localization problem of article [13] are resolved by the SAA method. The experimentally estimated images are tested again by the edge detection algorithm and the output object’s borders are acknowledged which are illustrated in **Fig. 4.13.**


(a) Objects of Reference image. (b) Reconstructed 3D dense disparity.

(c) Classified object borders of ground truth. (d) Classified object borders of ground truth.

Fig. 4.13 Localized object borders.

The estimated dense disparity’s 3D structure is recovered and its objects border are correctly identified which are outlined in **Fig. 4.13 (c)** and **(d)**. So, the result ensures that the similar depths are found in estimated dense disparity.

4.5.2 Computational Cost Calculation and Comparison with Existing state-of-the- art Methods for Middlebury Standard Tsukuba Head

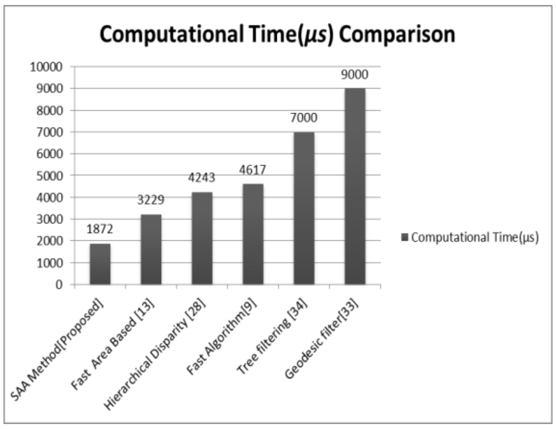
Disparities of reference image are estimated by SAD technique using adaptive search algorithm. The disparities are estimated within the search range from -10 to +10. The effects of self-adaptive search are considered with respect to computational costs, frame-rate (in *fps*), gain and accuracy. The computational costs, frame-rate and gain performance results of adaptive method have been compared with previous fastest literatures [9], [13], [28] and the current state-of-the-art methods [32-35], [37-39]. Fast Area Based algorithm [13] reports 3229μs required by Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM for 384 × 288 image resolution of Tsukuba head. This experiment also results the frame-rate 310 *fps*. Hierarchical Disparity method [28] reports 4243μs required by Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM for 384 × 288 image resolution of Tsukuba head. This experiment also results

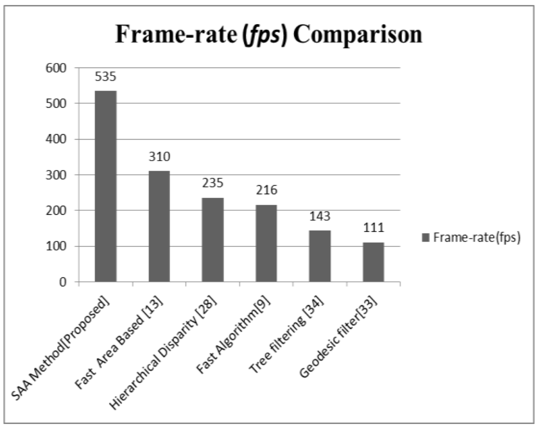
235 *fps* frame-rate. Fast algorithm [9] reports 4617μs required by Intel Core i-3, 2.3 GHz processor with 4 GB DDR3 RAM for 384 × 288 image resolution of Tsukuba head. The frame-rate of this method is 216 *fps*.

Table 4.1: Disparity estimation computational time (in μs) and frame-rate (in *fps*) for the Middlebury standard data of Tsukuba head image using Self-Adaptive Search Method.

Method's Name	**Computational Time(in μs)**	**Frame-rate (in *fps*)**	**Accuracy (in %)**	**Computational Machine**	**Input image & Resolution**	**Rank**
Self-Adaptive Algorithm [Proposed]	1872	535	93.80	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB	Middlebury Standard Tsukuba Head 384×288	1
Fast Area Based [13]	3229	310	86.10	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		2
Hierarchical Disparity [28]	4243	235	92.10	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		3
Fast Algorithm [9]	4617	216	88.23	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		4
Tree filtering [34]	7000	143	93.18	Intel Core i-7 Speed: 1.8 GHz. RAM: 4 GB		5
Edge-aware Geodesic filter[33]	9000	111	93.67	Intel Core i-5 +Geforce GTX card,Speed: 3.0GHz.,RAM: 8GB		6
DSI & Adaptive Support[32]	200000	5	90.18	Core Duo,Speed: 2.2GHz. RAM:NA		7
Pyramid stereo matching [39]	550000	2	97.68	Nvidia GeForce GTX 1080 Ti/PCIe/SSE2	KITTI -2015 Datasets.	8
Deep self-guided[38]	2860000	0.35	91.76	Intel Core i-7 Speed: 3.4GHz. RAM.16GB	Middlebury training datasets and KITTI -2015	9
Energy Minimization[35]	3000000	0.33	92.82	Intel Core i-5 Speed: 1.9 GHz. RAM:6GB	Middlebury Standard datasets	10
Fusing Adaptive Support[37]	40500000	0.025	96.02	Intel Core i-5 Speed: 3.2 GHz. RAM: 8GB	Tsukuba, Venus, Teddy, Cones [Middlebury Benchmark]	11

The proposed SAA algorithm requires only 1872*μs* on the same hardware with 241 comparisons instead of 441 and 882 comparisons as mentioned in section 4.4. The proposed method also performs better frame-rate compared to previous popular methods [9], [13], [28] and recent state-of-the-art methods of [32-35], [37-39]. It shows the highest frame-rate 535 *fps* among the state-of-the art methods. **Table 4.1** illustrates the summary of comparisons among the proposed method and present state-of-the-art methods. The SAA's experimental results have been also compared with the result of methods those are tested on Middlebury standard datasets. The results ranking in **Table 4.1** indicate that the proposed SAA method is ranked *1st* out of existing top state-of-the-art methods of [32-35], [37-39]. It shows the highest frame-rate 535 *fps* and lowest computational time 1872 *microseconds* among the latest methods. The proposed method outperforms all the state-of-the-art methods in frame-rate and computational time on Middlebury standard Tsukuba head image pair.





(a) (b)

Fig. 4.14 Left side graph(a) shows the comparison of computational costs and right side graph(b) shows the comparison of frame-rate (in *fps*) among the proposed and existing state-of-the-art methods for Tsukuba head image.

We claim that the SAA method is currently the state-of-the-art method for Middlebury standard Tsukuba head image pair with 2.4X, 1.7X, 2.2X, 106.8X, 4.8X, 3.7X, 1602.5X, 1527.7X, 293.8X faster than the methods of [9], [13], [28], [32], [33], [34], [35], [38] and [39] respectively.

The SAA method achieves the reduction of 42.02% computational time comparing with the Fast Area Based Stereo Matching Algorithm [13]. The proposed method also performs 59.45% time reduction compare to window-based method by Fast Algorithm [9]. From **Table 4.2,** we observe that 55.88% reduction of computational time is done by the proposed SAA method comparing with the Hierarchical Disparity

Estimation [28]. Finally, the SAA method achieves 99.06%, 79.20%, 73.25%, 99.93% and 99.93% computational time reduction against the recent state-of-the-art methods of [32], [33], [34], [35] and [38] respectively with lower configurations of hardware. The graphical comparison of computational costs and frame-rate (in *fps)* are illustrated in **Fig. 4.14**. Therefore, from the **Table 4.1**, **Table 4.2** and graph of **Fig. 4.14**, the proposed SAA algorithm is the better choice on the basis of computational time and frame-rate.

Table 4.2: Computational time reduction (in %) of proposed method for Tsukuba Head

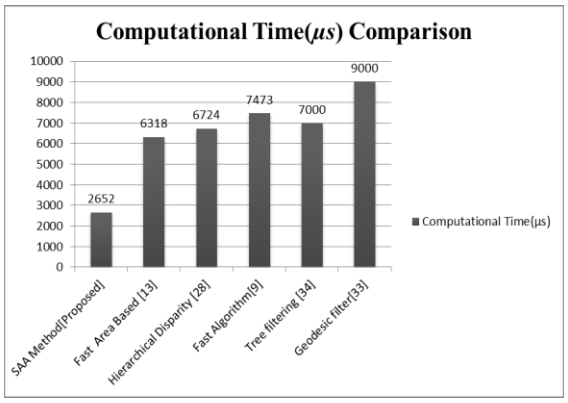
Computational Time(in *µs*) for Self-Adaptive Algorithm [Proposed]	**Existing state-of-the-art Methods**		**Computational Time Reduction (in %) by SAA method compared to the methods of *2nd* column**
	Method's Name	**Computational Time(in *µs*)**	
1872	Fast Area Based [13]	3229	42.02
	Hierarchical Disparity [28]	4243	55.88
	Fast Algorithm[9]	4617	59.45
	Tree filtering [34]	7000	73.25
	Edge-aware Geodesic filter[33]	9000	**79.20**
	Energy Minimization[35]	3000000	99.93
	Pyramid stereo matching[39]	550000	99.65
	Deep self-guided[38]	2860000	99.93

4.5.3 Computational Cost Calculation and Comparison with Existing state-of-the-art Methods for Middlebury Standard Venus Stereo Images

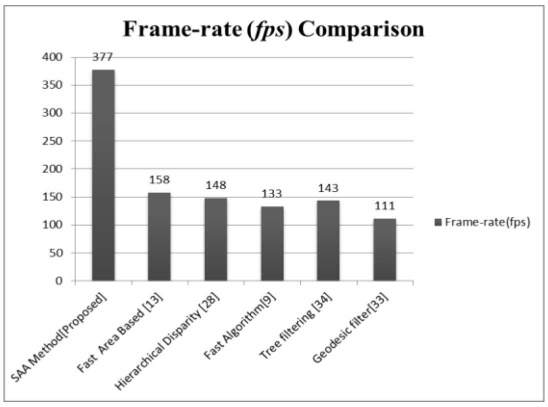
The performances of computational time and frame-rate have been numerically evaluated in **Table 4.3** for Middlebury standard data of Venus stereo pair. The proposed SAA algorithm also outperforms all other algorithms summarized in **Table 4.3**. The comparison of computational costs and frame-rate (in *fps*) of proposed and recent state-of-the-art methods are illustrated in **Table 4.3** with hardware specifications. The SAA algorithm also obtains the better performance in terms of computational time and frame-rate. It requires only 2652 *µs* instead of 6318 *µs*, 6724 *µs*, 7000 *µs*, 7473 *µs* and 9000 *µs* of mentioned methods respectively. It shows the highest frame-rate 377 *fps*. The graphical comparisons of computational time and frame-rate are illustrated in **Fig. 4.15** which depicts that the proposed SAA method runs in lowest time and achieves the highest speed among the current state-of-the-art methods.

Table 4.3: Disparity estimation computational time (in μs) and frame-rate (in *fps*) for the Venus stereo image using Self -Adaptive Search Method.

Method's Name	**Computational Time(in μs)**	**Frame-rate (in *fps*)**	**Computational Machine**	**Input image & Resolution**	**Rank**
Self-Adaptive Algorithm [Proposed]	2652	377	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB	Middlebury Standard Venus Stereo dataset 434×383	1
Fast Area Based [13]	6318	158	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		2
Hierarchical Disparity [28]	6724	148	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		3
Tree filtering [34]	7000	143	Intel Core i-7 Speed: 1.8 GHz. RAM:4GB		4
Fast Algorithm [9]	7473	133	Intel Core i-3 Speed: 2.3 GHz. RAM: 4GB		5
Edge-aware Geodesic filter[33]	9000	111	Intel Core i-5 +Geforce GTX card Speed: 3.0Ghz RAM: 8GB		6



(a)



(b)

Fig. 4.15 Left side graph(a) shows the comparison of computational costs and right-side graph(b) shows the comparison of frame-rate (in *fps*) among proposed and existing state-of-the-art methods for Venus stereo images.

Another numerical evaluation and comparisons are represented in **Table 4.4** in which proposed method performs 64.51% time reduction compared to window-based Fast Algorithm [9] for Venus stereo pair image. The computational time is reduced 60.55% by proposed SAA method compared to the Hierarchical Disparity [28]. The SAA algorithm also reduces 58.02% of computational time comparing with the Fast Area Based method [13]. Finally the SAA method achieves 70.53% and 62.11% computational time reduction over the recent state-of-the-art methods of [33] and [34] respectively using the lower configurations of hardware. So it is evident that the proposed SAA method is currently the state-of-the-art method for Middlebury standard Venus stereo pair with 2.8X, 2.3X, 2.5X,3.3X, 2.6X faster than the top five methods of [9], [13], [28], [33] and [34] respectively.

Table 4.4: Computational time reduction (in %) of proposed method for Venus stereo pair.

Computational Time(in *μs*) for Self-Adaptive Algorithm [Proposed]	**Existing state-of-the-art Methods**		**Computational Time Reduction (in %) by SAA method compared to the methods of *2nd* column**
	Method's Name	**Computational Time(in *μs*)**	
2652	Fast Area Based [13]	6318	58.02
	Hierarchical Disparity [28]	6724	60.55
	Tree filtering [34]	7000	62.11
	Fast Algorithm[9]	7473	64.51
	Edge-aware Geodesic filter[33]	9000	70.53

From the **Table 4.4** and graph of **Fig. 4.15**, the proposed SAA algorithm again proves that it is the best choice on the basis of computational time and frame-rate. In both cases (For Tsukuba and Venus stereo input images) the SAA algorithm performs the lowest computational costs and highest frame-rate.

4.5.4 Accuracy Measurement and Comparisons

The accuracy of this algorithm has been justified over standard stereo images of Tsukuba head. The algorithm is implemented using Visual C++ programming language. To determine the correspondence of a pixel of reference image, the window costs are estimated for the candidate-pixels of right image within the search range -10 to +10 pixels. The proposed SAA method estimates the accuracy in percentage with the error threshold 2. The accuracy of SAA method is 93.8%. The numerical evaluations confirm that the bad pixel in percentage is only 6.2% for our proposed method. But using the same resolution of image, bad pixels in percentage were 6.33%, 7.88%, and 7.18% reported in [33], [34] and [35] respectively for Tsukuba head with the experiments of Middlebury stereo datasets.

4.6 Performance Enhancement Analysis

The performance of the proposed SAA approach has been compared to the state-of-the-art methods. The comparison tools were computational time, frame-rate and gain. Our target was to speed up the computational costs with no degradation of accuracy. Since the accuracies and 3D dense disparity of state-of-the-art algorithms yield very similar to proposed SAA method; confirm the effectiveness of the proposed matching algorithm. **Table 4.5** illustrates the performance enhancement of SAA method compared to the established state-of-the-art methods. The proposed SAA method has done 72% gain enhancement compared to Fast Area Based Method [13]. Enhanced gain is calculated as follows.

Gain Enhancement by proposed SAA method over Fast Area Based Method [13]:

= {(Computational time of Fast Area Based Method ÷ Computational time of SAA Method) × 100 - 100} %

= {(3229/1872) × 100 -100} %

= **72%**

Table 4.5: Quantitative evaluation of performance of proposed SAA method with top five (5) algorithms.

Method's Name	**Computational Time(in *μs*)**	**Frame-rate (in *fps*)**	**Performance Enhanced (in %) by SAA method over the methods of *1st* column**	**Input image & Resolution**
Self-Adaptive Algorithm [Proposed]	1872	535	×	Middlebury Standard Tsukuba Head 384×288
Fast Area Based [13]	3229	310	72	
Hierarchical Disparity [28]	4243	235	126	
Fast Algorithm[9]	4617	216	146	
Tree filtering [34]	7000	143	273	
Edge-aware Geodesic filter[33]	9000	111	380	

Similarly, the experimental gain, enhanced by proposed SAA method over the state-of-the-art methods of [28], [9], [33] and [34] were 126%, 146%, 380% and 273% respectively which is illustrated in **Fig. 4.16.** The performance enhanced graph of proposed SAA method is shown **Fig. 4.16**. Since the speed of proposed method is very high, the estimated gains of this algorithm are automatically high. The numerical measurments of this claim is strongly supported by experimental data of **Table 4.5** in which the SAA algorithm is compared with the top five(5) algorithms. The bar diagrams illustrate the increased gain (in %) of SAA method compared to the cureent methods . The proposed method outpeforms the existing state-of-the-art methods with respect to the performance enhancements too.

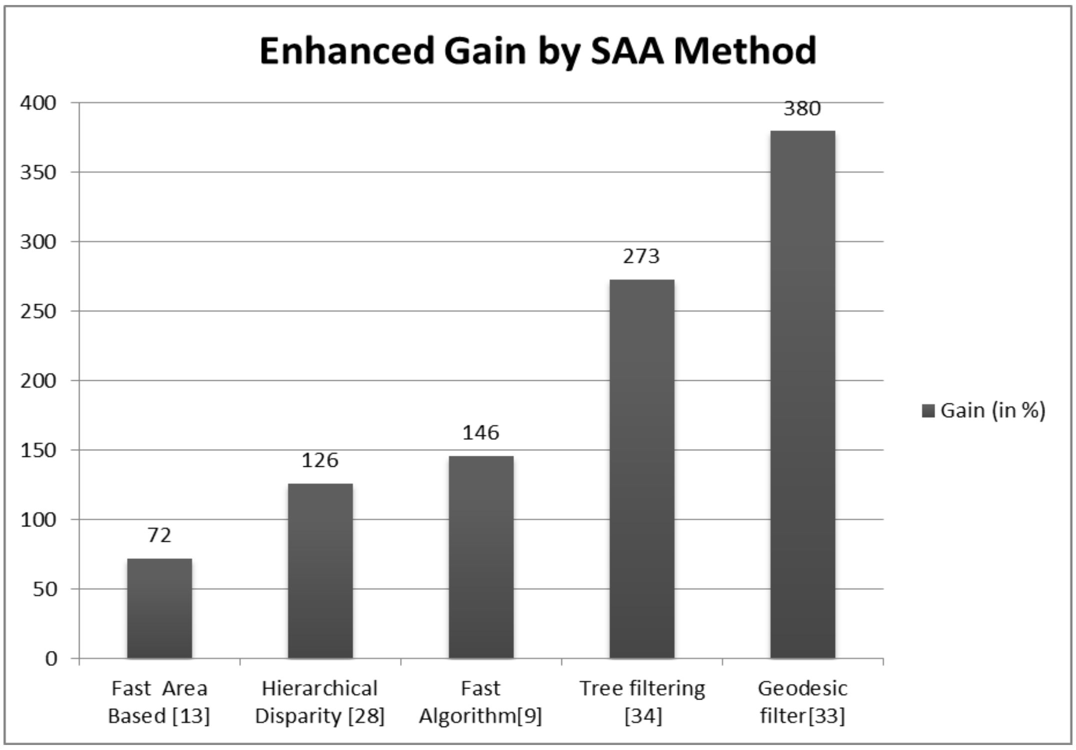


Fig. 4.16 Graph shows the performance enhancement of SAA method comparing to the mentioned state-of-the-art methods.

Moreover, the performance of the SAA method has been tested on Middlebury standard Venus stereo datasets and the estimated experimental data are summarized in **Table 4.6**. From this table, the proposed SAA method also performs better than the existing top five algorithms for Venus stereo datasets. The gains have been estimated using the same procedures as mentioned earlier .The SAA method enhances 138%, 153%, 181%, 239%, 163% gains compared to the established state-of-the-art methods of [13], [28], [9], [33] and [34] respectively. The SAA method enhances 239% gain over Geodesic filter method[33], but it actually increases (239-100)= 139% gain against Geodesic filter method. Similarly , the proposed method enhances 163% gain over the Tree Filtering[34] method and thus it actually increases (163-100)= 63% gain against Tree Filtering method. Therefore, the proposed SAA algorithm outperfoms the existing top five algorithms for Venus stereo datasets. In terms of computational time, frame-rate and gain achievement, the proposed matching approach is faster and better than the existing state-of-the-art methods.

Table 4.6: Quantitative evaluation of performance of proposed SAA method with top five (5) algorithms for Venus stereo pair.

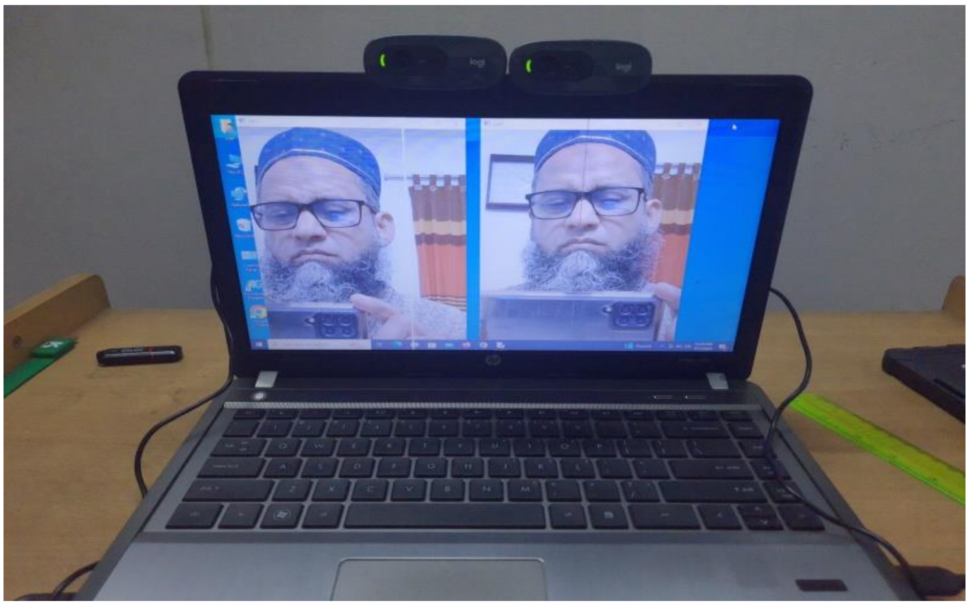
Method's Name	**Computational Time(in *μs*)**	**Frame-rate (in *fps*)**	**Performance enhanced (in %) by SAA method over the methods of *1st* column**	**Input image & Resolution**
Self-Adaptive Algorithm [Proposed]	**2652**	**377**	×	Middlebury Standard Venus Stereo dataset. 434×383
Fast Area Based[13]	6318	158	138	
Hierarchical Disparity [28]	6724	148	153	
Tree filtering [34]	7000	143	163	
Fast Algorithm[9]	7473	133	181	
Edge-aware Geodesic filter[33]	9000	111	239	

4.6.1 Experiment on Real Stereo Images by SAA

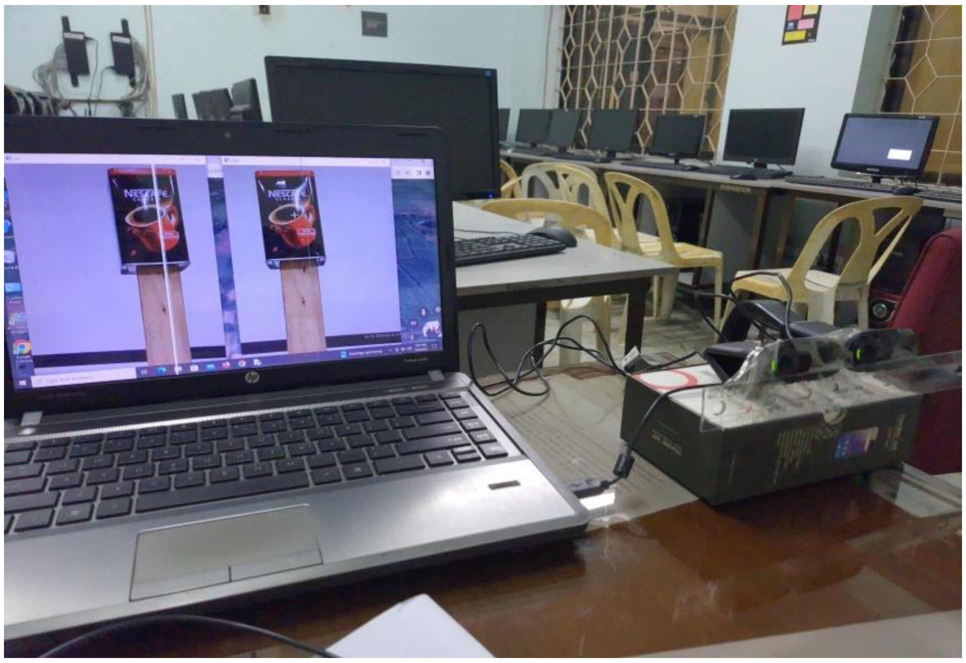
The performances of SAA algorithm have been further tested on real stereo images acquisitioned by Logitech stereo web camera. This experiment is performed in our software laboratory and images were captured as indoor scenes. The specifications of stereo camera are the same as we have mentioned in subsection 2.3.1 and 3.4.4.

Stereo Image Capturing Process:

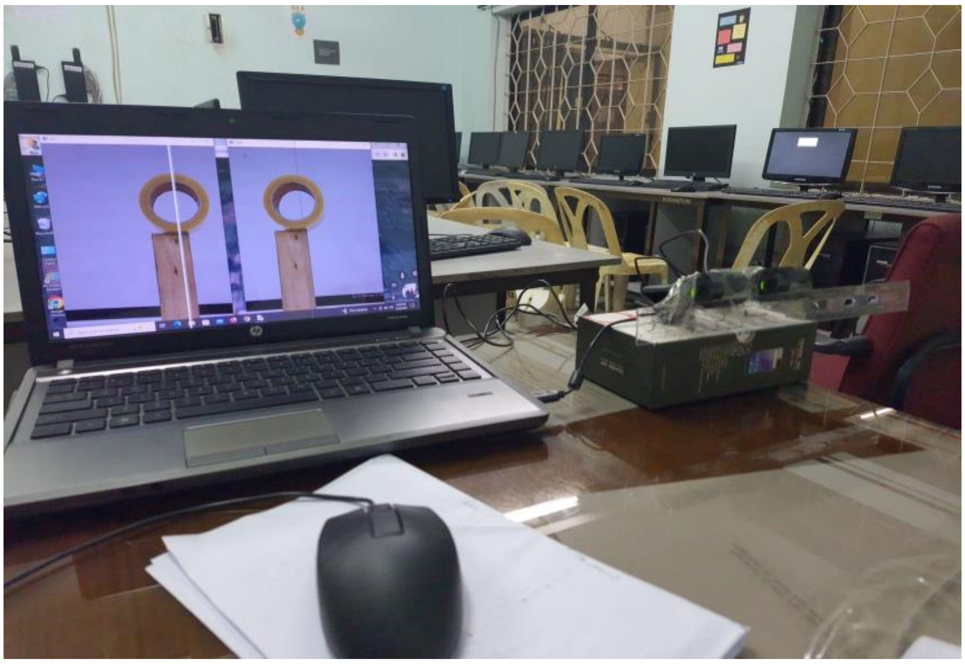
The main objects (Human face, Nescafe coffee stand and Scotch tape stand) of reference images were stood 63.00 cm away from the imaging sensor of the camera. The distance between two cameras was 6.70 cm.



a)Real image acqusition using stereo web camera for dataset-1(Human face).



b) Real image acqusition using stereo web camera for dataset-2(Nescafe coffee stand).



c) Real image acqusition using stereo web camera for dataset-3(Scotch tape stand).

Fig. 4.17 Real image acqusition process using stereo web camera.

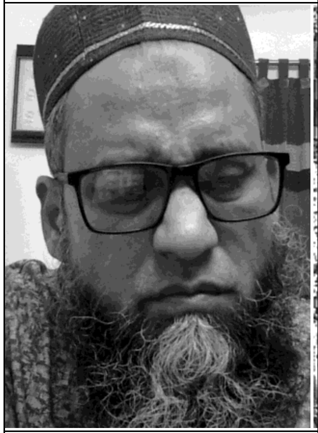
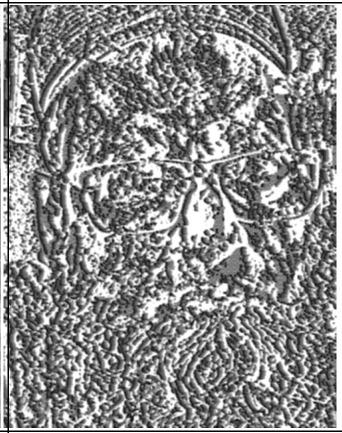
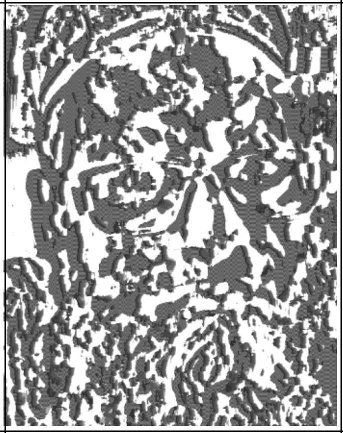
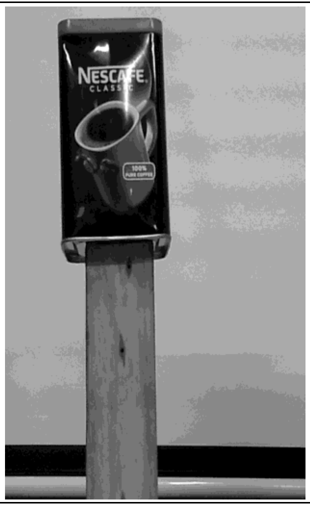
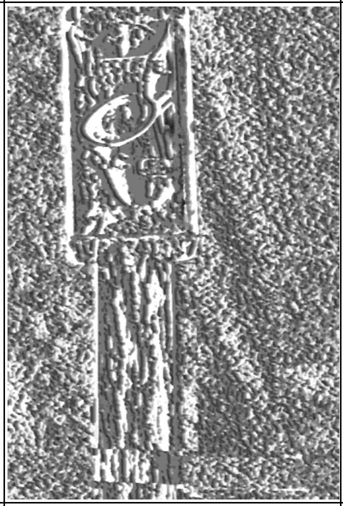
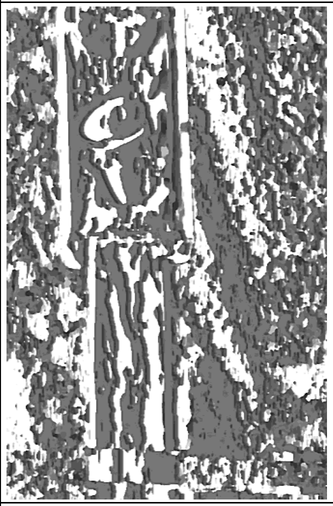
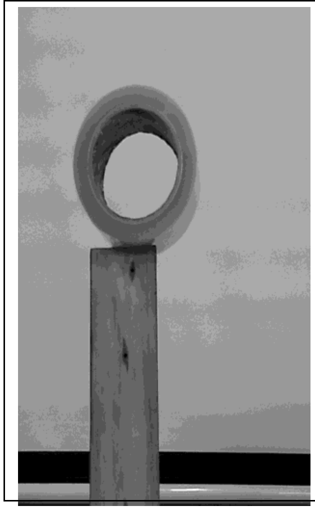
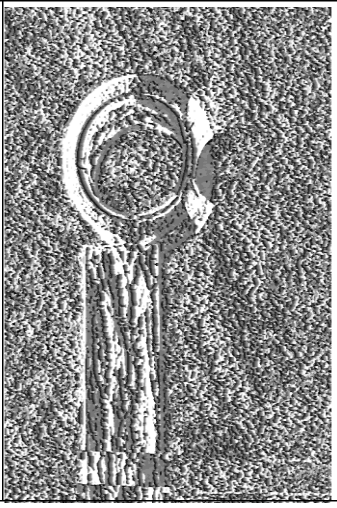
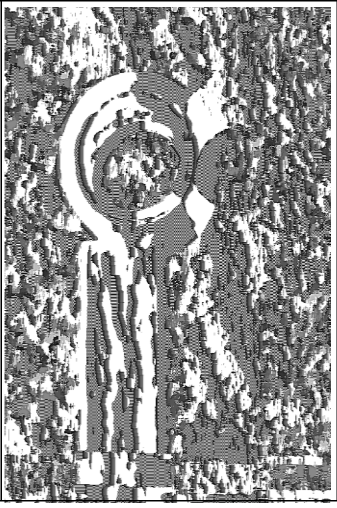
Experimental output for Real Stereo Images:

The size of the left and right real-image is (width × height) = (550 × 720) pixels. Disparity image size is (width × height) = (514 × 684) pixels. **Table 4.7** demonstrates the visual observations and findings of dense disparity map.

Table 4.7: demonstrates the visual observation of dense disparity maps using two different types of mask (3×3 and 11×11). The **Table 4.7** illustrates the dense disparity maps of three real datasets -Human face, Nescafe coffee stand and Scotch tape stand respectively. The 1st column of this table represents the reference image and its size which is captured by our stereo camera. The 2nd and 3rd column represent the estimated dense disparity maps of reference images using 3×3 and 11×11 mask respectively. We could not compare the experimental outputs to the ground truth image because it has no ground truth images. In this situation, the disparity maps of reference image should be considered and compared visually only.

The disparity maps of outputs contain some noise. This is happened because we could not provide the equilibrium light condition in our laboratory.

Table 4.7: Visual observation of disparity map of real images generated by SAA algorithm

Reference image Resolution: 550×720	Experimental Dense Disparity Map of Real Image Mask size : 3×3	Experimental Dense Disparity Map of Real Image Mask size : 11×11	Execution time(μs)
			Mask: 3×3 : 468 Mask: 11×11: 5360
			Mask: 3×3 : 468 Mask: 11×11: 6735
			Mask: 3×3 : 563 Mask: 11×11: 5985

Similarly the room temperature was not equilibrium at all the places during the image acquisition process. Moreover, we have tried to our best to calibrate the stereo camera physically. The stereo cameras were manually placed on the same horizontal line, but experimentally, it was not possible. There was some vertical difference between two cameras in fractional millimeter (.05 mm approximately) range. These cause to add a little noise in captured stereo images. Inspite of noise, the objects are demarked and recognized at standard level. The object (Human face, Nescafe coffee stand and Scotch tape stand) inside the reference image is clearly understandable and visualized.

The dense disparity map generated by 3×3 mask is more visualized and comprehensive than the disparity map of 11×11 in noisy environment. So the overall performance of SAA algorithm is good in case of real stereo images.

4.7 Discussion

The main objective of this method was to speed up the computational time. We have done this by a new technique called Self-Adaptive Algorithm that infers the upcoming matching pixel's position. This algorithm itself readjusts as well as reduces the search range based on remembering the previously matching pixel's position. The frame-rate of our algorithm is 535 *fps* for input images of Tsukuba head pair and 377 *fps* for input images of Venus stereo pair. Thus, it can calculate, process and display output 535 frame/second for the case of standard Tsukuba head image pair and 377 frame/second for the standard Venus stereo pair. The estimated gains of proposed SAA method are 380 and 239 for Tsukuba head and Venus stereo respectively whereas the gain of existing state-of-the-art method is 100. Since the accuracy and 3D dense disparity maps of proposed method are very similar with the existing state-of-the-art algorithm, it confirms the effectiveness of the proposed matching algorithm. Moreover, the SAA algorithm does not require any additional programmable 3D hardware like 3D Graphics Processing Unit (GPU). The proposed SAA method demonstrates the state-of-the-art results and outdoes the present top methods.

4.8 Summary

The compulsion of SAA method is stated in Section 4.1 of this chapter. The proposed Self-Adaptive method is explained both in mathematically and graphically in Section 4.2 for well understanding. The optimization technique of SAA method is described in Section 4.4. The main achievements of this method are discussed in Section 4.5 and its Subsections of Experimental results. Experimental results described in Subsection 4.5.1, 4.5.2 and 4.5.3 demonstrate the visual and numerical comparison between proposed SAA method and the present state-of-the-art methods. The great achievement of this method is Performance enhancement, which is quantified in Section 4.6. The SAA algorithm is tested on real images in subsection 4.6.1. The overall conclusion of this method has been drawn in Section 4.7.

Chapter 5

Stereo Correspondence Estimation Technique: Self-Guided Stereo Correspondence (SGSC) Estimation Algorithm

5.1 Introduction

For binocular machine vision, stereo correspondence is the fundamental issue for viewing the objects exactly. It is well-defined by the parameter d (**Fig. 1.1**) that is calculated by the deviation of x-axis between the reference image and right image, also known as disparity. It is still now open challenge to measure the accurate dense disparity map in 3D environment due to noise, difference in camera parameters, homogenous background of stereo images, variation of light intensities of indoor and outdoor scenes and variable textures of images. Though there are a lot of algorithms to estimate the real-time disparity map, but there is no perfect algorithm till to date to compute the stereo correspondence like human stereo eyes. The said parameter d is expressed by the equation (1.1) in Chapter 1. We can also measure the disparity d by knowing the camera parameters; B, f and Z. Where B is the baseline distance of two horizontally placed cameras, f is the focal length of camera and Z is the depth of information of an object from the camera. The key challenge is to explore the matching position as fast as possible within the whole image or specified searching range. Our proposed SGSC algorithm can recognize the correspondence in right image with very quick response.

The depth of information Z is an important parameter for machine vision, pedestrian navigation, 3D scene tracking and reconstruction. Disparity is experimentally estimated by using Sum of Absolute Differences (SAD) in our proposed SGSC method. Besides this, there is also Sum of Square Differences (SSD) and Normalized Correlation Techniques (NCT) is the alternative techniques to compute the stereo correspondence, which is already defined mathematically in Chapter 1. There are open problems in selection of shape and size of mask or window to scan the searching line. Different complexities are involved for different types of mask or window that is also discussed in Chapter 1.

The current researches of connected problems to the matching costs are stated in [28] and [9]. Bilateral filter and threshold techniques are used to estimate the disparity. Guided image filter is used in the work [29] to estimate the window costs aggression. The authors employ unsupervised and online adaption [45] on KITTI 2012 and KITTI 2015 datasets. Tree filtering method [34] computes the shortage distance between two pixels on designated pixel's tree. This method is tested on Middlebury standard stereo datasets, but it requires more computational costs. The most recent works [35], [37],

[40] and Self-Adaptive Algorithm (SAA) [Chapter 4] worked on Middlebury standard stereo datasets. These methods require high computational costs. Besides these, the above methods except method SAA [Chapter 4] involve post processing step like filtering, refinement or left-right consistency checking those are responsible to increase the computational costs. However, our proposed Self-Guided Stereo Correspondence (SGSC) Estimation method does not require the post processing steps.

5.1.1 Motivation

The software driven parallel model, 2DRTSSA is mentioned in Chapter 3, computes two window costs simultaneously. As we have to find out the best cost among several possible correspondences, we choose the best correspondence carefully and purposefully. Though the performance of 2DRTSSA is satisfactory, i.e., the maximum accuracy was 93.8%, but the computational cost was higher than the RTA method. The SAA method in Chapter 4 reduces the search range around 50%; therefore it takes less time for computation compared to 2DRTSSA. In this method the accuracy is same as 2DRTSSA but the computational cost was lower.

These observations give rise to the idea that the drawbacks of both approaches can be resolved by the newly approached SGSC method. In this method, both the accuracy and performance is enhanced by exactly tracking the matching pixel position in right image using the threshold technique.

5.2 Proposed Self-Guided Search Algorithm

This research work is the enhancement version of Self-Adaptive Algorithm (SAA) of Chapter 4. Some modifications have been imposed on SAA method to achieve the better performance and behaviors of the proposed method. The searching range $-d_{max}$ to $+d_{max}$ is divided into two searching regions; 1) first one is $-d_{max}$ to 0 and 2) second one is 0 to $+d_{max}$. The cost aggression process on the right image is performed either in *1st* region or *2nd* region, reported in SAA method [Chapter 4]. The proposed method follows almost the same procedure of SAA method but it differs mostly that after first search it employs the threshold technique to reduce the search zone as close as needed to create the active zone. Employing the new technique, the modified SGSC algorithm reduces the processing time up to 253 *microseconds* compared to SAA method. The other betterments and performances are presented in Section 5.5. According to the proposed algorithm, the *1st* search of *1st* reference pixel L(x - d_{max}) is searched on R{x + (-d_{max})} , R{x + (-d_{max} + 1)}, R{x + (-d_{max} + 2)} … R(x + *0*) … … R{x + (d_{max} - 1)} to R{(x + (+ d_{max})} in the right image towards the first scanning line (marked by red color pixel) of **Fig. 5.1**. During the first cost aggression process, the algorithm explores the best matching pixel by measuring its lowest value of window cost within the whole coordinate range from R{x + (- d_{max}), y} to R{x + (+d_{max}), y}. Suppose, the *1st* best match is found at P_R (i, y) with intensity I_R in *1st* zone. Actually, this coordinate position is the reference matching position of proposed method. Based on the coordinate position and photometric property of it's, the proposed SGSC algorithm apply threshold technique with a view to more closer the search zone than SAA algorithm. Before starting the *2nd*, *3rd*, *4th* ... *Mth* search the SGSC algorithm apply threshold technique to prevent the false-search area as well as to reduce the search zone for creating needful active zone. Accordingly, until the condition $|P_L(i + 1,y) - P_R\,(i,\ y)| \leq \ \delta_T$ is satisfied, the cost aggression and searching continues at *1st* zone. Where δ_T is the optimal threshold intensity, magnitude of P_R $(i,\ y)$ is I_R and magnitude of P_L $(i\ ,y)$ is I_L.

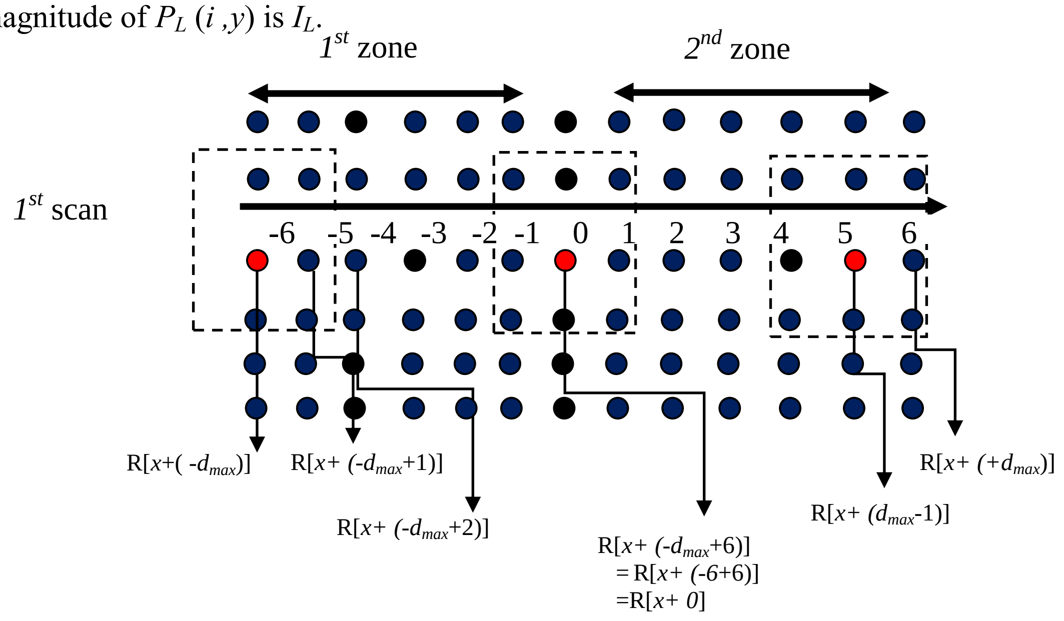


Fig. 5.1 The whole search and sub search zones of right image for the reference pixel of left image.

Thus the overall execution time elapsed for *2nd* zone's candidate-pixels are reduced at every search operation. Actually, this is an extra reduction time for self-decided SGSC algorithm over SAA method. When the above mentioned condition is not satisfied, the search and cost estimation process enter into the *2nd* zone and the procedure will be continued as like as *1st* zone. The mentioned SGSC procedure is reiterated for the consecutive pixels of reference image towards the scanning line from minimum to maximum depth of the whole image.

It is our experimental observation that while scan line crosses over the boundary pixels of a segment then the search toggles the zones. In such case, our algorithm scans both zones to find out the matching correspondence. For better understanding and realization of SGSC, we display a flow chart and algorithm for visual conception at a glance. The flow chart shows only the one scan line pixel of reference image with window cost estimation by self-guided algorithm. The SGSC algorithm always calculates the window costs within only half portion of candidate-pixels in right image with properly handling mismatching zone. The main benefits of this algorithm are that we need not to calculate the costs on half portion of specified search range for every reference pixel of left image. These advantageous procedures are graphically illustrated in **Fig. 5.3**, **Fig. 5.4** and **Fig. 5.5**. The **Fig. 5.3** shows the window cost aggression, calculation and searching cruise of *1st* search where five window costs are considered for simplicity and better realization.

After finding the first stereo correspondence, the SGSC algorithm divides the whole search zone into two sub zone: 1) *1st* zone and 2) *2nd* zone. The territory of *1st* zone starts from R $[(x + (-d_{max})]$ and ends at R$[(x + 0)]$. Similarly, the territory of *2nd* zone begins from R$[x + (0 + 1)]$ and ends at $R[x + (+d_{max})]$. After that the SGSC algorithm assigns the position of first matching correspondence either in *1st* or in *2nd* zone depends on the intensity level of matching pixel I_R. Let it is allocated in *1st* zone.

From the flow chart, all subsequent reference pixels are firstly checked by threshold criteria that prohibit the search sequence to enter into the mismatching zone for false matching. The threshold value δ_T is selected as an optimal value and in our experiment we set the optimal threshold value at 5.

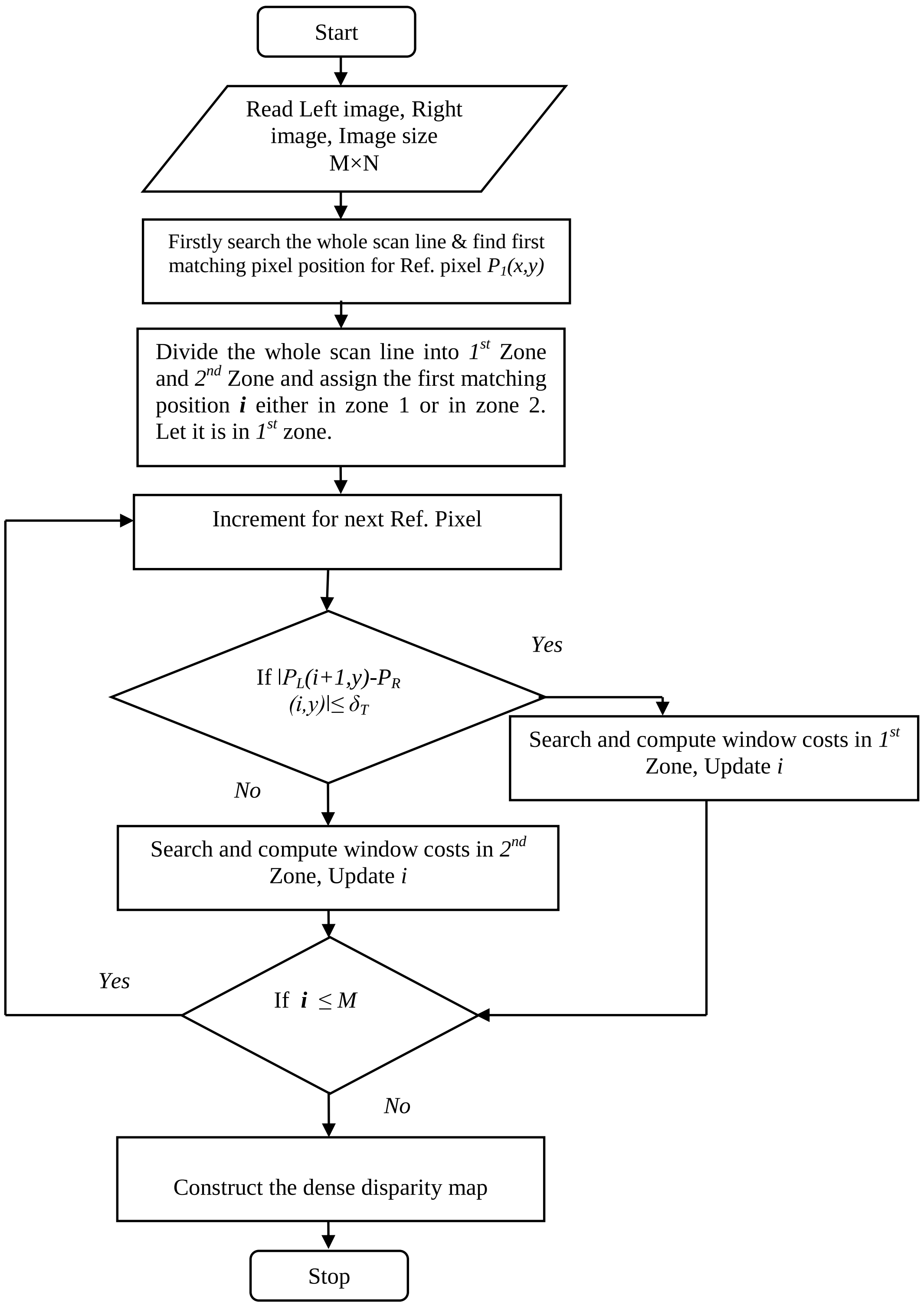

Start
Read Left image, Right image, Image size M×N
Firstly search the whole scan line & find first matching pixel position for Ref. pixel $P_1(x,y)$
Divide the whole scan line into 1st Zone and 2nd Zone and assign the first matching position i either in zone 1 or in zone 2. Let it is in 1st zone.
Increment for next Ref. Pixel
If $|P_L(i+1,y)-P_R(i,y)|\leq \delta_T$
Yes
Search and compute window costs in 1st Zone, Update i
No
Search and compute window costs in 2nd Zone, Update i
Yes
If $i \leq M$
No
Construct the dense disparity map
Stop

Fig. 5.2 Flow chart of the proposed SGSC algorithm.

If the first matching correspondence is found in *1st* zone, then the searching process and window cost calculations are associated with only the *1st* zone's pixels as shown in **Fig. 5.4**. In such case, cost function calculates the window costs for the candidate-pixels of -6,-5,-4,-3,-2,-1,0 only and window costs calculations are discarded for the candidate-pixels of 1,2,3,4,5,6.

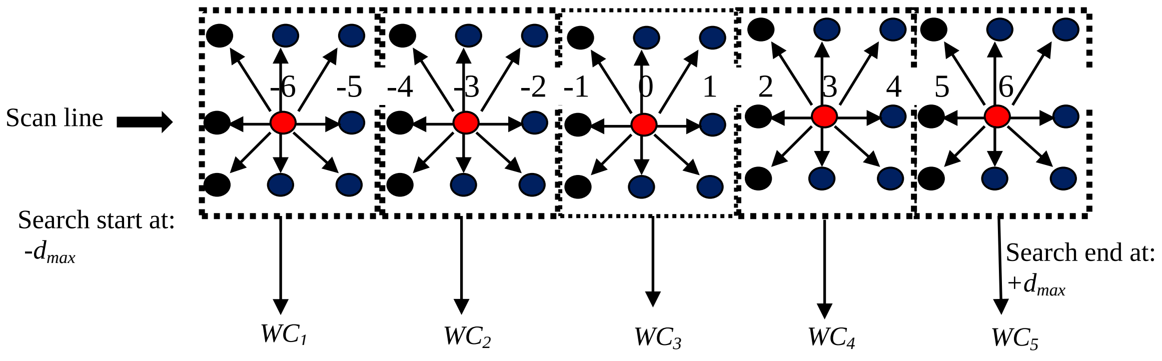


Fig. 5.3 *1st* search for window cost calculation process towards the scanning line.

The window costs calculation continues within the *1st* zone until the condition $|P_L(i+1,y)-P_R(i,y)| \leq \delta_T$ is satisfied. So for every reference pixel the searching, comparing and window cost calculations are always discarded for the half portion of pixels (1, 2, 3, 4, 5, and 6) of *2nd* zone. The search sequence enters into the *2nd* zone while it overpasses the border line of a segment of an input reference image.

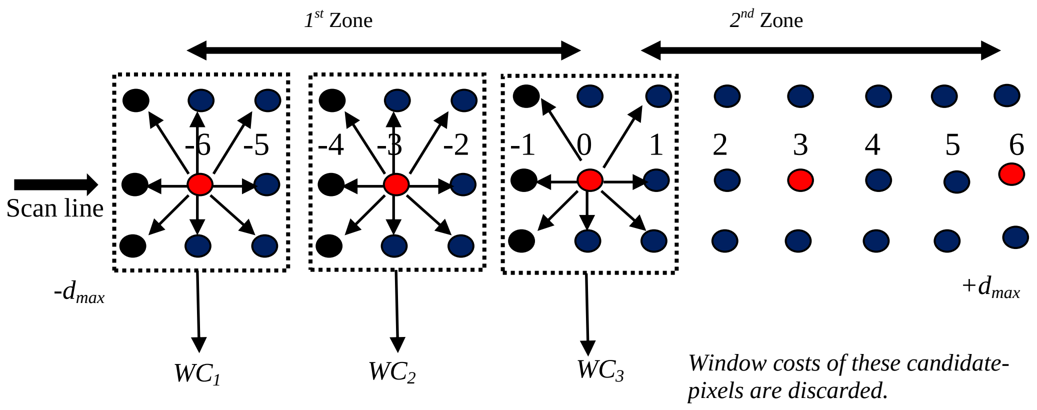


Fig. 5.4 Window cost estimation procedure for *1st* zone (2nd Searching).

If the correspondence is found in *2nd* zone; the searching process and window cost functions are associated with only the *2nd* zone's pixels as shown in **Fig. 5.5**. In such case, cost function calculates the window costs for the candidate-pixels of 0, 1, 2, 3, 4, 5 and 6 only. Window cost calculations are discarded for the candidate-pixels of -6, -5, -4, -3, -2, and -1.

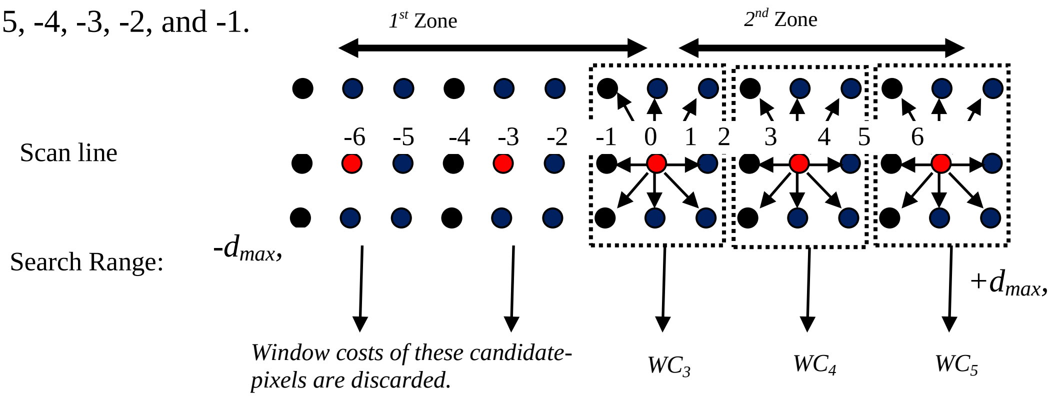


Fig. 5.5 Window cost estimation procedure for 2nd zone (3rd searching).

So, the proposed SGSC method determine the upcoming reference pixel's correspondence on *x* axis either in *1st* zone or in *2nd* zone. Therefore, SGSC method reduces more cost-estimation time than SAA method in local stereo matching domain.

5.2.1 Disparity Estimation Algorithm of SGSC Method

```
Algorithm SGSC(m, n, temp, temp1,sum,w1,w2 , k1,k2,v_left,d,
v_right,image_left.pixel,image_right.pixel,image_disp.pixel
,ws1,ws2,dmax,dmax1,dmax2,t0,t1,flag)

//m,n is the row and column size of an image.
// temp and temp1 are pixels intensity value of left and
// right image.
//w1,w2 is the row and column size of the mask.
//sum is the summation of window costs.
// v_left is the pixel intensity value of left image.
// v_right is the pixel intensity value of right image.
// image_left.pixel[1:m][1:n] is the left image pixel
// coordinate that contains m×n elements.
//image_right.pixel[1:m][1:n]is the right image pixel
// coordinate that contains m×n elements.
//image_disp.pixel[1:m][1:n]is the disparity image
// that contains m×n disparity values.
// d is the search range counter variable.
//k1 ,k2 are the number pixels to discard from left and
// right side of image.
// ws1 and ws2 are the local variable within the mask.
//dmax,dmax1 and dmax2 are the search ranges.
// t0, t1 is the variable for time.
//i,j and flag is the integer type counter variables.
     for n:=0 to size_y do
     {
      for m:=0 to size_x do
       {
       image_left.pixel[m][n]:= temp; // Read left image
       image_right.pixel[m][n]:= temp1;//Read right image
       }
      }
  t0:= clock();
   // Dividing the searching zone
     ws1 := (w1/2); ws2 := (w2/2);
     for m:= k1 to size_x-k1 do
     {
       for n:= k2 to size_y-k2 do
         {
     if(flag=0) then
      {  dmax1:=-dmax;// initial zone.
          dmax2:= dmax;
       }
     else if (flag<0) then //  1st zone
       {  dmax1:=-dmax;
          dmax2:= 0;
        }
       else           //  2nd  zone
         { dmax1:= 1;
           dmax2:= dmax;
          }
  // window cost calculation
        for d:= dmax1 to dmax2 do
       {
          sum:= 0;
             for i:=-ws1 to ws1 do
             {
                 for j:= -ws2 to ws2 do
                {
                v_left:= image_left.pixel[m+i][n+j];
                v_right:= image_right.pixel[m+i+d][n+j];
                 sum:= sum + abs(v_left - v_right);
                 }
              }
       Mtemp[d + dmax].pixel[m][n]:= sum;

      }
       // Select the minimum window cost.

       image_disp.pixel[m][n]:= minimum(Mtemp,m,n,ws1);
       }
     }
     t1:= clock();
     cpu_speed:= t1 - t0; // time calculation.
     write("Total time",cpu_speed);
     // Creating the dense disparity map.
     for n:= k1 to size_y-k1 do
       {
       for m:= k2 to size_x-k2 do
        {
  write("dense disparity image",image_disp.pixel[m][n]);
    }
  }
```

Algorithm minimum (temp[2*dmax+1],x,y,ws)

```
// Find the minimum value from temp[0:2*dmax+1] elements
// x, y, i, j, mu, a, len , Th are the integer variables.
```

```
{ j:=1,Th:=5;
 len:= 2*dmax+1;
 if (flag ≠ 0 ) then
   len:= dmax+1;
 for i:=0 to len do
   {
     if(temp[i].pixel[x][y] < temp[j].pixel[x][y]) then
          j:= i;
    }
      mu:=abs(j-dmax);
  // Applying Threshold technique
     if (abs(image_disp.pixel[x][y-1]- mu) <= Th)then
        flag:= -1; // Set 1st Zone as active.
     else
       flag:= 1; // Set 2nd Zone as active.
     return (mu);
  }
```

5.3 Computational Complexity Analysis

The complexity of SGSC algorithm is always less than current state-of-the-art methods. The optimization technique is applied in this algorithm in three stages, 1) Total search zone is divided into two zones and window cost calculation occurs always on one zone at a time. Therefore, half portions of candidate-pixels are discarded all times. 2) Threshold technique is employed in second stage to reduce search zone to be closer the effective zone, and 3) The window cost calculation process will not be toggled to opposite zone until or unless $\delta_T > 5$. So, the proposed SGSC algorithm is optimized in three phases that makes the algorithm faster than current state-of-the-art methods. The complexity depends on image size, window size, and search range. Suppose the window size (square size), image size and search range are W, $M \times N$, d respectively. The computational cost of SGSC on half zone for each reference-pixel is $C_{hz} = (W^2 - 1) \times \{(d + 1) / 2\}$. This cost will again be reduced by applying threshold technique and $C_{hz\delta T} = (W^2 - 1) \times \{(d+1) / 2 - \delta_T)\}$. Therefore, the total comparisons for each reference-pixel of SGSC method is $\{(d + 1) / 2 - \delta_T\}$ times. The overall complexity of the proposed stereo matching algorithm

$$C_{SGSC} = [(M \times N) \times (W^2 - 1) \times \{(d + 1) / 2 - \delta_T\}] \tag{5.1}$$

Fast Area-Based [13] Algorithm needs for overall complexity,

$$C_{FAB} = (M \times N) \times (W^2 - 1) \times (d + 1) \tag{5.2}$$

Hierarchical Disparity [28] or left-right checking algorithm needs overall computational cost, $C_{HD} = 2\times(M\times N)\times(W^2-1)\times(d+1)$ (5.3)

The overall complexity of FAS [37] is $C_{FAS} = O(NM^2|D|)$ which is equivalent to our notation $C_{FAS} = \{(M \times N) \times W^2 \times |D|\}$. Where |D| is the total levels of disparity. With the above analysis of complexity, it is clear that the overall complexity of our method is relatively less than the state-of-the-art methods. Moreover, no multiprocessing hardware or accelerated hardware is used in our proposed method.

5.4 Experimental Results

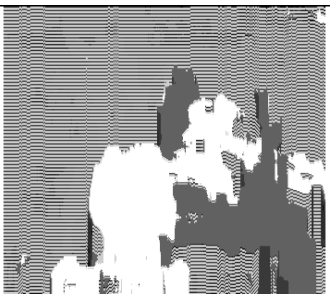
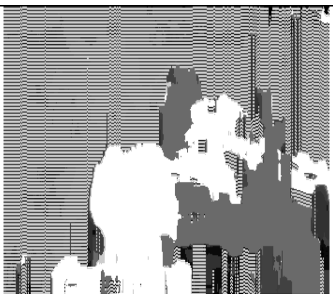
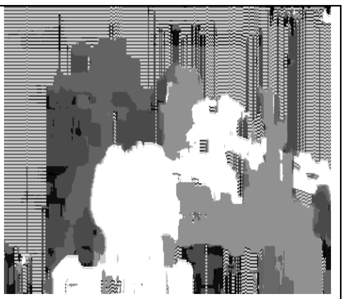
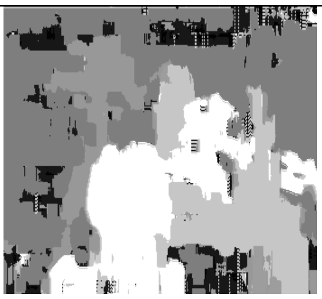
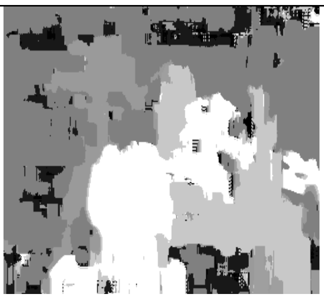
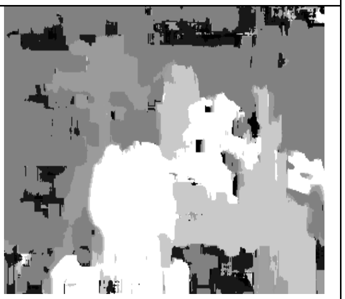
The proposed SGSC algorithm is implemented in Visual *C++* language with Windows 10 operating system. All experimental results are implemented by the processor of Intel Core i-3, 2.3 GHz speed and 4GB RAM. We tested our algorithm on the following datasets and environments.

- Experimental settings and adjustment for threshold value δ_T
- Observation of 3D reconstruction with object borders, size and shape localization.
- Middlebury standard stereo images of Tsukuba and Venus stereo for detailed analysis.
- Middlebury standard stereo datasets 2003 of indoor scenes.
- Middlebury standard stereo datasets 2006 of indoor scenes and
- Middlebury optical flow latest datasets for hidden, synthetic and stereo types of image data.

5.4.1 Experimental Settings and Adjustment for Threshold Value δ_T

The threshold technique is used to make the searching zone as close-fitting as possible. The threshold value (δ_T) is calibrated properly and carefully. **Table 5.1** shows the different dense disparity maps for different values of δ_T with different time reductions. From the experimental data of **Table 5.1** we find that for the case of small values of δ_T (like 1, 2 and 3), the background of the disparity map is condensed and overlapped each other because it abruptly reduced the search region. These maps are shown in first row of **Table 5.1**. The background is expanded with increasing the threshold value (like 4, 5 …10). The large values of threshold increased the search region to make the opportunity for several matchings' and thus expanded both dark and bright values of disparity map. Hence, we set the threshold value at 5 for all subsequent processes.

Table 5.1: Effects of variation of threshold value δ_T.

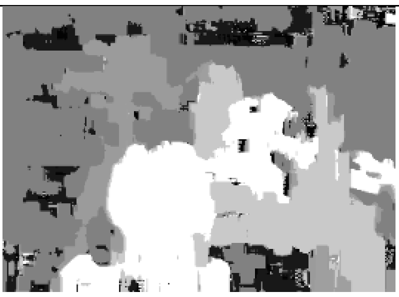
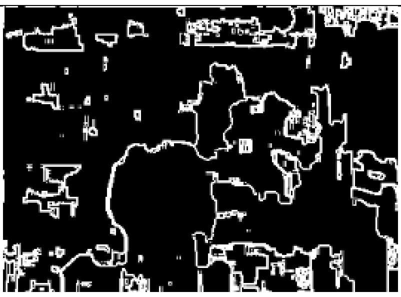
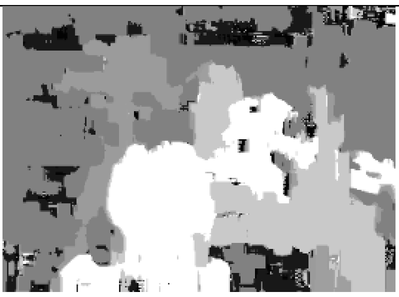
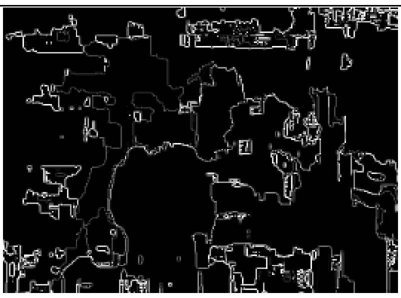
Experimental Output Image			
Threshold Value δ_T	1	2	3
Time(in μs) reduction compared to a SAA [Chap. 4]	86	112	145
Experimental output Image			
Threshold Value δ_T	4	5	10
Time(in μs) reduction compared to SAA [Cha. 4]	167	183	183

We mainly focus on two types of datasets: 1) Middlebury Standard Stereo datasets 2006 and 2003, 2) Middlebury current Optical Flow datasets. Middlebury Standard Stereo datasets 2006 consists of 21 stereo pair and Middlebury Standard Stereo datasets 2003 consist of 6 stereo pair. Optical flow datasets are categorized four types: 1) Hidden Texture, 2) Synthetic, 3) Stereo and 4) High-speed camera (No GT). We evaluated our algorithm on above mentioned datasets both in terms of numerical and visual and compared to the state-of-the-art methods. But detailed comparisons like frame-rate, time and accuracy are presented for Tsukuba and Venus stereo pair in subsection 5.4.3 and 5.4.4 of this section. Moreover, we evaluated our proposed method on Middlebury Optical Flow of datasets for visual comparisons with hidden ground-truth. The window cost calculation mask size of 11 × 11 is used in this proposed experiment.

5.4.2 Observation of 3D Reconstruction with Object Border, Size and Shape Localization

Tsukuba head image encloses with various objects at different depth of locations. Background and foreground objects are positioned at various depths. Some occlusions and bad objects are found in Tsukuba with overlapping condition and it encloses with some special frontal objects; for example, statue's head, lamp of table and camera. The proposed algorithm distinguishes the various depths by setting the dissimilar gray scale values to the output image as shown by 3D reconstruction map in **Table 5.2**. Short-distance object is indicated by white grey level and long-distance object is assigned by black grey level value. Object borders are easily visualized in our estimated border classification map. Border localization problems [13] are resolved by SGSC algorithm properly. The 3D dense disparity map is further checked by the object detection algorithm and the object borders are figured out in *3rd* column of **Table 5.2**. The 3D dense maps of SGSC method are almost similar to their ground truth image. The estimated 3D restructure is convalesced and their object's borders are perfectly recognized. Therefore, the result confirms that the almost same depths are found in experimentally calculated disparity map.

Table 5.2: Visual examination of 3D structures, borders and objects detection on experimental disparity map.

Ref. Image	**3D Reconstruction map.**	**Object borders classification**	**Comments**
Tsukuba Head			Object borders are perfectly localized. (Edge preserved)
Tsukuba Head			Object borders are perfectly localized. (Edge preserved)

5.4.3 Evaluation on Middlebury Standard of Tsukuba Head Image Pair for Detailed Analysis

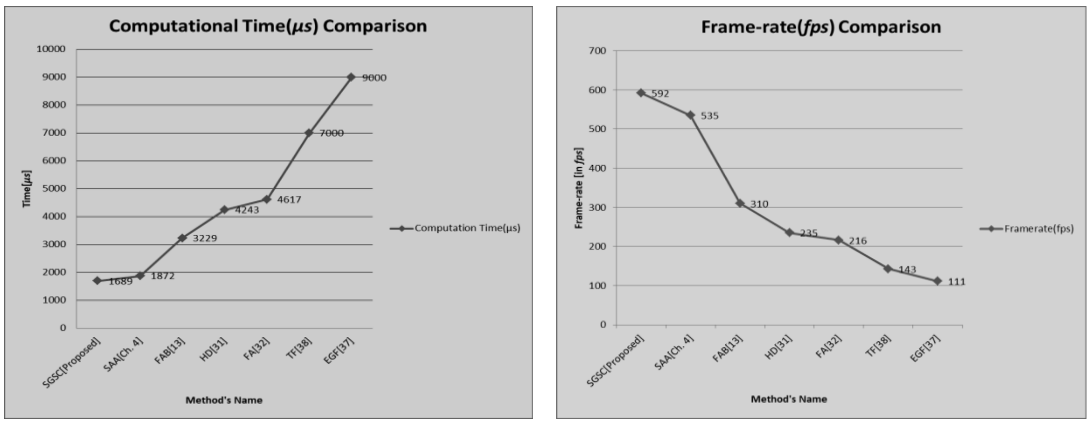
We have evaluated our method on Tsukuba head both on numerically and graphically in this section. Sum of Absolute Differences (SAD) technique is used in our experiment to compare identically with the current state-of-the-art methods. The stereo correspondences are calculated within usual search range (from -10 to +10).

Table 5.3: Numerical comparisons of computing time (in μs) and frame-rate (in *fps*) with top eleven (11) methods for the Middlebury data of Tsukuba head using SGSC algorithm.

Method's Name	**Computing Time (in μs)**	**Frame-rate (in *fps*)**	**Accuracy (in %)**	**Computational Machine**	**Input image**	**Rank**
SGSC [Proposed]	**1689**	**592**	**97.60**	Processor: 2.3 GHz Core i-3, Main-Memory: 4GB.	Middlebury Standard Tsukuba image pair	**1**
SAA [Chapter 4]	1872	535	93.80	Processor: 2.3 GHz Core i-3, Main-Memory: 4GB.		2
FAB[13]	3229	310	86.10	Processor: 2.3 GHz Core i-3, Main-Memory: 4GB.		3
HD[28]	4243	235	92.10	Processor: 2.3 GHz Core i-3, Main-Memory: 4GB.		4
FA[9]	4617	216	88.23	Processor: 2.3 GHz Core i-3, Main-Memory: 4GB.		5
TF[34]	7000	143	93.18	Processor: 1.8 GHz Core i-7, Main Memory: 4GB		6
EGF[33]	9000	111	93.67	Processor: 3.0 GHz Core i-5, Main - Memory: 8GB, GTX card.		7
DSI-AS[32]	200000	5	90.18	Processor: 2.2 GHz Core Duo.		8
PSM[39]	550000	2	97.68	Processor: Nvidia G.F GTX-1080.	KITTI - 2015 Standard	9
DSG[38]	2860000	0.35	91.76	Processor: 3.4 GHz Core i-7, Main – Memory: 16GB	Middlebury and KITTI - 2015 Standard	10
EM[35]	3000000	0.33	92.82	Processor: 1.9 GHz Core i-5, Main - Memory: 6GB	Middlebury datasets	11
FAS[37]	40500000	0.025	96.02	Processor: 3.2 GHz Core i-5, Main - Memory: 8GB	Middlebury & KITTI Standard	12

The experimental outcomes of self-guided search are analyzed with respect to computational time, scanning frame-rate (in *fps*) and its accuracy. The computational time, frame-rate (in *fps*) as well as other achievements of self-guided technique are examined with previous [9], [13], [28] and the present state-of-the-art methods [32], [33, [34], [35] as well as with SAA [Chapter 4]. **Table 5.3** demonstrates the summary of numerical evaluations among the proposed SGSC method and current state-of-the-art methods with their computational machine specifications. From **Table 5.3**, it is seen that proposed SGSC method is ranked *1st* among the non-learning local and learning global state-of-the-art methods. Even our proposed algorithm run with lower configurations of machine and no special hardware or parallel processing technique is used to accelerate the running speed.

This algorithm takes computational time only 1689 *microseconds* with highest frame-rate 592 *fps*. The nearest comparable method is SAA that takes 1872 *microseconds* with frame-rate 535 *fps*. From **Table 5.3** and **Fig. 5.6**, we observe that the SGSC method requires minimum computational time and gives the maximum frame-rate.



(a) (b)

Fig. 5.6 Left-side graph(a) shows the comparison of computational time (in *μs*)and right-side graph(b) shows the comparison of frame-rate (in *fps*) among the proposed SGSC and current state-of-the-art methods for Tsukuba head image.

Therefore, the SGSC method outdoes the present state-of-the-art methods with respect to computational cost and frame-rate on Middlebury Tsukuba head data. So, we can claim that the SGSC method is presently the state-of-the-art method for Tsukuba image with 1.11X, 1.9X, 2.5X, 2.7X, 118.4X, 5.32X, 4.14X, 1776.19X, 1693.3X, 335.6X faster than the methods of SAA [Chapter 4], [13], [28], [9], [32], [33], [34], [35], [38] and [39] respectively.

Table 5.4: Numerical comparisons of time reduction (in %) of SGSC algorithm for Tsukuba stereo image.

Computational Time(in μs) for SGSC[Proposed]	**Existing state-of-the-art methods**		**Computational Time Reduction (in %) by SGSC method compared to the methods of *2nd* column**
	Applied Method	**Computational Time(in μs)**	
1689	SAA[Chapter 4]	1872	09.77
	FAB[13]	3229	47.69
	HD[28]	4243	60.19
	FA[9]	4617	63.41
	TF[34]	7000	75.87
	EGF[33]	**9000**	**81.23**
	EM[35]	3000000	99.94
	PSM[39]	550000	99.69
	DSG[38]	2860000	99.94

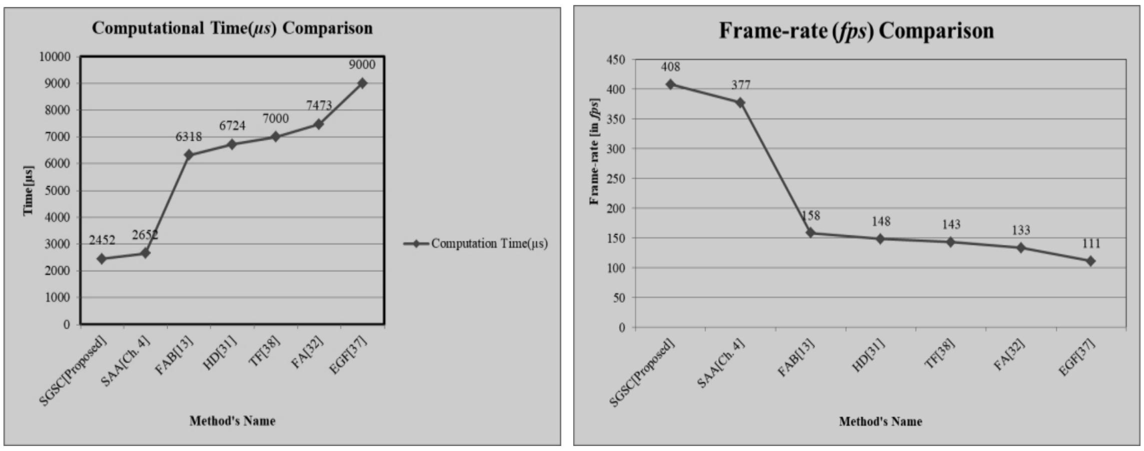
Similarly, computational time reductions of SGSC algorithm are calculated both on local and global method those are demonstrated in **Table 5.4**. First six methods of **Table 5.4** are non-learning local methods and last three methods are learning global methods. Our proposed SGSC method is non-learning and it performs up to 81.23% time reduction in non-learning local method as well as 99.94% time reduction in learning environment.

5.4.4 Evaluation on Middlebury Standard Stereo Images of Venus Stereo Pair for Detailed Analysis

The computational time and frame-rate are assessed by numeric figures in **Table 5.5** for Middlebury Venus stereo datasets. The SGSC method also exceeds to all other non-learning local methods those are tabulated in **Table 5.5**. **Table 5.5** distinguishes the SGSC method and recent state-of-the-art methods with respect to computational time and frame-rate with machine configurations. Our SGSC needs only 2452 μs compared to 2652 μs, 6318 μs, 6724 μs, 7000 μs, 7473 μs and 9000 μs respectively. It runs with highest frame-rate at 408 *fps* among the top six performer algorithms those are represented in **Fig. 5.7**.

Table 5.5: Numerical comparison of time (in *μs*) and frame-rate (in *fps*) for the Venus stereo image using SGSC algorithm.

Applied Method	Computational Time(in *μs*)	Frame-rate (in *fps*)	Computational Machine	Input image	Rank
SGSC [Proposed]	**2452**	**408**	Processor: 2.3 GHz Core i-3, M. Memory: 4GB	Middlebury Venus Stereo dataset	**1**
SAA [Chapter 4]	2652	377	Processor: 2.3 GHz Core i-3, M. Memory: 4GB		2
FAB[13]	6318	158	Processor: 2.3 GHz Core i-3, M. Memory: 4GB		3
HD[28]	6724	148	Processor: 2.3 GHz Core i-3, M. Memory: 4GB		4
TF[34]	7000	143	Processor: 1.8 GHz Core i-7, M. Memory: 4GB		5
FA[9]	7473	133	Processor: 2.3 GHz Core i-3, M. Memory: 4GB		6
EGF[33]	9000	111	Processor: 3.0 GHz Core i-5, M. Memory: 8GB GF, GTX card.		7



(a) (b)

Fig. 5.7 Left-side graph (a) shows the comparison of computational time (in *μs*) and right-side graph (b) shows the comparison of frame-rate (in *fps)* for Venus stereo pair.

Another assessment of the SGSC algorithm is measured numerically based on computational time reduction and their comparisons are represented in **Table 5.6**.

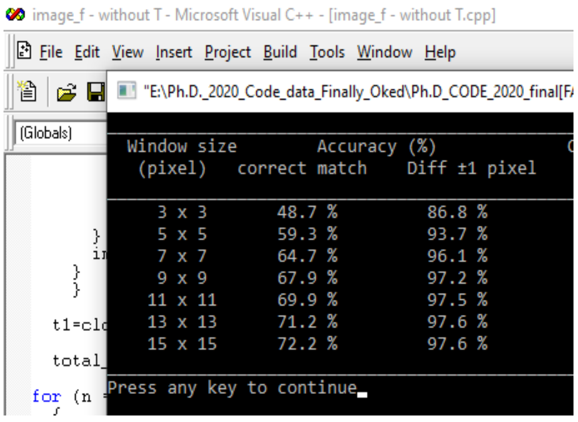
Table 5.6: Numerical comparisons of time reduction (in %) of proposed method for Venus stereo pair.

Computing Time(in μs) for SGSC [Proposed]	**Current state-of-the-art methods**		**Computational Time Reduction (in %) by SGSC method over the methods of 2nd column**
	Method's Name	**Computational Time(in μs)**	
2452	SAA[Chapter 4]	2652	07.54
	FAB [13]	6318	61.19
	HD [28]	6724	63.53
	TF[34]	7000	64.97
	FA[9]	7473	67.18
	EGF[33]	9000	72.75

All methods presented in **Table 5.6** are non-learning local method. Among these methods, the proposed algorithm shows lowest time reduction 7.54% compared with the nearest similar method SAA and highest time reduction 72.75% compared to the recent state-of-the-art method EGF [33]. Therefore, our SGSC algorithm is presently state-of-the-art method for Middlebury Venus stereo datasets with 2.5X, 2.7X, 3.04X, 3.67X, 2.8X, 1.08X faster than the top six methods of [13], [28], [9], [33], [34] and SAA respectively.

5.4.5 Estimation of Accuracy and Comparison with Top State-of-the-Art Methods

The accuracy of proposed SGSC method is estimated on Tsukuba head stereo pair for testing the validity of SGSC algorithm. The SGSC algorithm is executed by Microsoft visual C++ compiler. To calculate the stereo correspondence of a reference image, the cost aggression of window is determined on the candidate-pixel of right image for the usual search range.

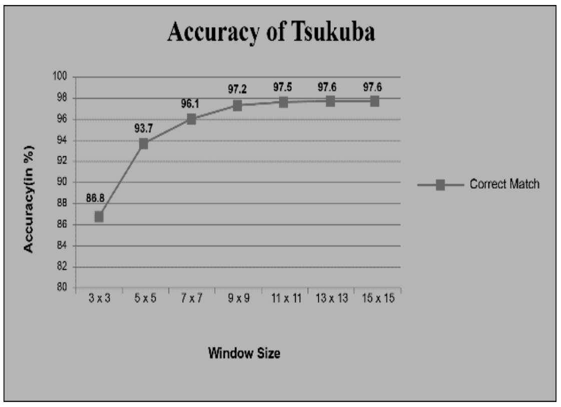

image_f - without T - Microsoft Visual C++ - [image_f - without T.cpp]
File Edit View Insert Project Build Tools Window Help
"E:\Ph.D._2020_Code_data_Finally_Oked\Ph.D_CODE_2020_final[F
(Globals)

```
   }
  i
 }
 }
t1=cl
total_
for (n
```

```
Window size        Accuracy (%)
 (pixel)   correct match    Diff ±1 pixel

  3 x 3        48.7 %         86.8 %
  5 x 5        59.3 %         93.7 %
  7 x 7        64.7 %         96.1 %
  9 x 9        67.9 %         97.2 %
 11 x 11       69.9 %         97.5 %
 13 x 13       71.2 %         97.6 %
 15 x 15       72.2 %         97.6 %

Press any key to continue
```



(a) (b)

Fig. 5.8 Left side figure(a) shows the run time screen shoot and right side graph(b) demonstrates the accuracy curve of SGSC algorithm.

The proposed SGSC algorithm calculates the accuracy with error threshold 2. From the run time snapshot of **Fig. 5.8(a)**, the highest accuracy of SGSC method is achieved as 97.5% and 97.6% for 11 × 11and 13 × 13 window sizes respectively. The numerical evaluation confirms that the proposed method generates only 2.4% error. The errors are 6.33%, 7.88%, 7.18% and 6.2% reported in [33], [34], [35] and SAA respectively for Middlebury standard stereo data of Tsukuba head using the same resolution of input images. From the graph of **Fig. 5.8(b)**, it is seen that the accuracy curve is almost steady-state after 11 × 11 window size of mask. So, we can conclude that the accuracy goes to saturation level after correct matching 97.6% with 2.4% bad pixel only.

5.5 Performance Analysis with Additional Standard Images

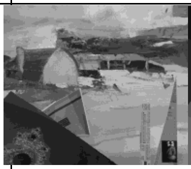
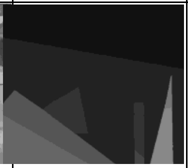
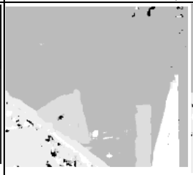
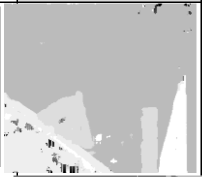
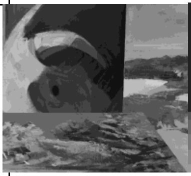
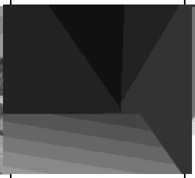
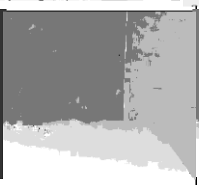
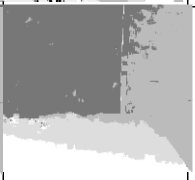
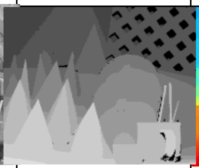
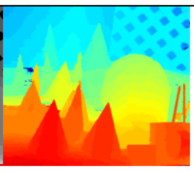
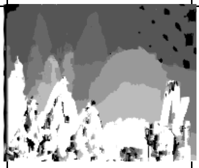
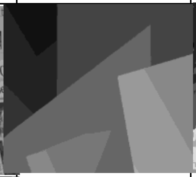
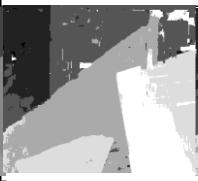
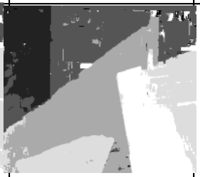
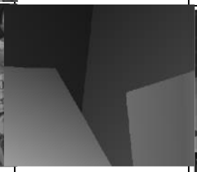
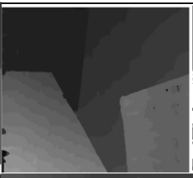
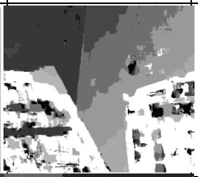
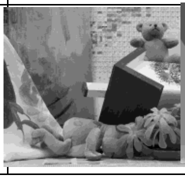
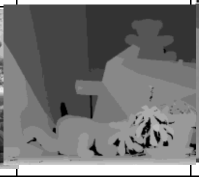
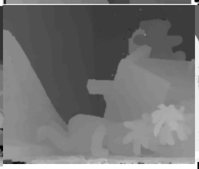
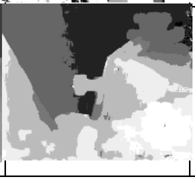
The effectiveness is carried out on Middlebury Standard Tsukuba head and Venus stereo data in Section 5.4 with numerical and graphical comparisons. To justify the adaptability and efficiency of proposed SGSC method we tested our algorithm on complex backgrounds for different types of Middlebury Standard images of different resolutions. Such types of positive results reconfirm again the effectiveness of SGSC method. Versatility testing is made on the following Middlebury Standard Stereo and Optical Flow datasets.

5.5.1 Evaluation on Middlebury Standard Stereo Datasets 2003 and 2001 of Indoor Scenes

The proposed SGSC method is compared with current state-of-the-art methods on Middlebury Standard stereo datasets 2003 and 2001 including four standard stereo pairs of Tsukuba, Venus, Teddy and Cones. The top six current state-of-the-art methods FA[9], FAB[13], TF[34], SAA[Chapter 4], EGF[33] and EM[35] are compared with SGSC method, as they are closely related to our proposed method. The first column of **Table 5.7** represents the stereo image name and their resolution. The stereo images of Cones and Teddy are Middlebury Standard stereo dataset 2003 and their ground truth achieved by structural light.

The stereo images of Burn2, Bull, Poster and Venus are piecewise planar scenes [12] of Middlebury Standard stereo datasets 2001. The *2nd* and *3rd* column's images are directly provided by Middlebury benchmark.

Table 5.7: Numerical and visual comparisons between SGSC method and current state-of-the-art methods on Middlebury standard stereo datasets of 2003 and 2001.

Reference Image Name & Size	Left Image	Ground truth	Output of current state-of-the-art method (Appling last column's method)	**Output of SGSC Method**	Running time for last column's method (in μs)	Running time for SGSC method (in μs)	Compared With
Burn2: 430×381					3109	1390	FA [9]
Bull: 433×381					3145	1391	FAB [13]
Cones: 450×375					7000	1433	TF [34]
Poster: 435×383					1675	1422	SAA [Chap. 4]
Venus: 434×383					9000	2652	EGF [33]
Teddy: 450×375					20000000	1437	EM [35]

The overall matching performance of our proposed SGSC method and its final dense disparity map is evaluated by threshold technique represents at *5th* column marked by shadow color of header in **Table 5.7**. The dense disparity maps of top six current methods are presented at *4th* column. The numerical computational time and comparisons are demonstrated at *6th* and *7th* columns in **Table 5.7**, where whole shadow column (*7th* column) represents the computational cost of the proposed SGSC algorithm. The visual comparisons are placed at *4th* and *5th* column and quantitative comparisons are held on *6th* and *7th* columns of **Table 5.7**. By observing the numerical comparisons carefully, we found that in all cases our proposed method is 2X or more faster than current state-of-the-art methods. The SGSC method takes 253 *microseconds* less than nearest SAA algorithm. From **Table 5.7**, we observe that except SAA method, all top performer algorithms run at double or more times than our proposed method. All 3D reconstructions are properly localized and objects-borders are visualized easily as seen by visual comparisons for different types of image data. Besides these, our algorithm preserves the edges of object correctly. An added feature is found in the proposed SGSC algorithm. By examining the estimated dense disparity map of SGSC method and ground truth image of Poster, Venus and Teddy, it is seen that the SGSC disparity map contains the hidden ground truth, which is not visible in conventional disparity estimation algorithms. Because of this extraordinary feature of SGSC algorithm, it can also detect the variation of optical flow. The detailed discussion is given in subsection 5.5.3 about the optical flow of variation where we discussed with the Middlebury Optical Flow Data.

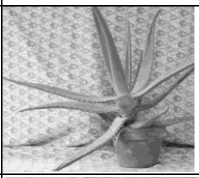
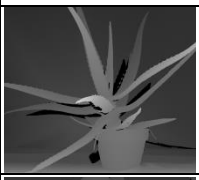
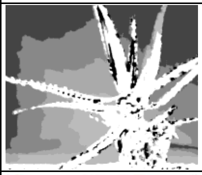
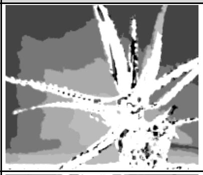
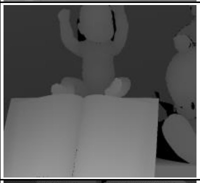
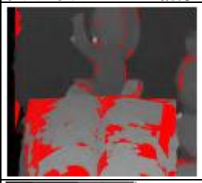
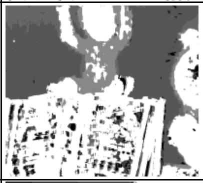
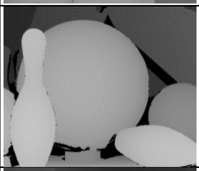
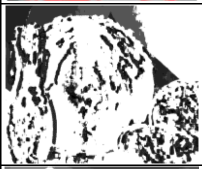
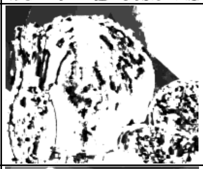
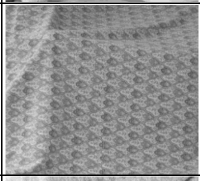
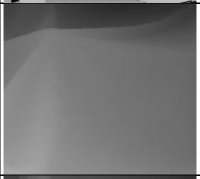
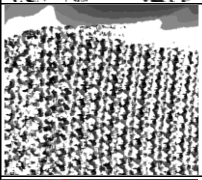
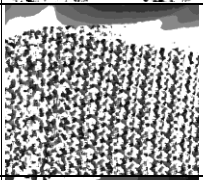
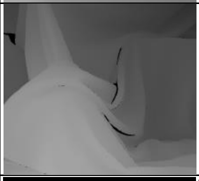
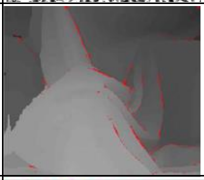
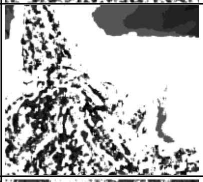
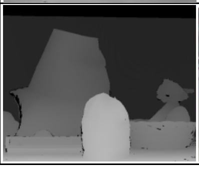
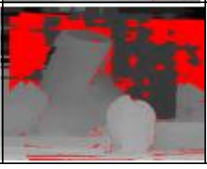
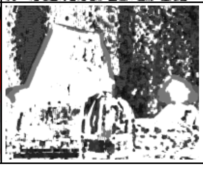
5.5.2 Evaluation on Middlebury Standard Stereo Datasets 2006 of Indoor Scenes

The top six current state-of-the-art methods FA [9], FAB [13], EGF [33], FAS [37], and SAA are compared with SGSC method on Middlebury standard stereo datasets 2006, as they are closely associated to our proposed method. The first column of **Table 5.8** represents the stereo images name and their resolution. The left and ground truth images of *2nd* and *3rd* columns are provided by Middlebury benchmark. The visual comparisons are presented on *4th* and *5th* column and quantitative comparisons are held on *6th* and *7th* columns in **Table 5.8**.

The experimentally evaluated visual and numerical results of our proposed method are represented at column *5th* and *7th* respectively. In SGSC, a photometric threshold technique is embedded on SAA method in order to reduce the search range as close-fitting as needed. Therefore, SGSC algorithm is faster than SAA. The computational time of SAA method for Aloe stereo image is 3405μs whereas the computational time of our SGSC method is 3191μs. So, proposed method takes 214 μs fewer than nearest similar method (i.e., SAA). In the rest cases, proposed SGSC is 2X or more faster than current state-of-the-art methods.

Dual support windows are used in FAS [37] for each candidate-pixel of right image. That is why its computational cost was very high, noted at 40.5 *seconds*.

Table 5.8: Numerical and visual comparisons between SGSC method and current state-of-the-art methods on Middlebury Standard stereo datasets of 2006.

Reference Image Name & Size	Left Image	Ground truth	Output of current state-of-the-art methods (Appling last column's method)	Output of SGSC Method	Running time for last column's method (in μs)	Running time for SGSC method (in μs)	Compared With
ALOE: 641×555					3405	3191	SAA [Chap. 4]
BABY2: 620×555					40500000	3094	FAS [37]
BOWLING2: 665×555					7484	3359	FAB [13]
CLOTH-1: 626×555					7001	3125	FA [9]
CLOTH-3: 626×555					9000	3094	EGF [33]
MIDD-2: 683×555					40500000	3422	FAS [37]

The SGSC method does not require preprocess and post-process like filtering, refinement or left-right checking to discard the outliers from the raw disparity map. The related works of top six methods need post-process techniques in order to make the raw disparity map as close as ground truth image. In our method no post-processing is required, only enhancement technique is used to visualize the raw disparity map. The raw disparity map of our SGSC method is directly comparable with ground truth image. Our proposed method not only takes less time for computing the dense disparity but also more efficient to preserve the object border.

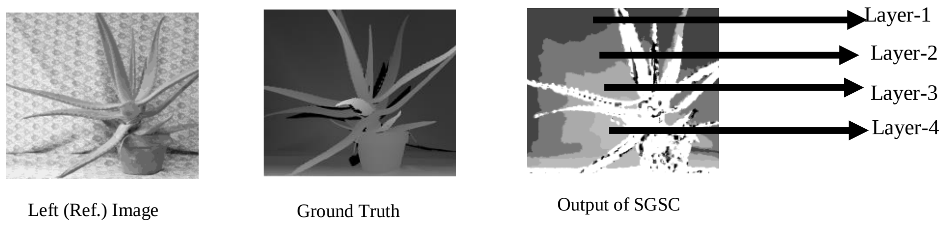


Fig. 5.9 Detection of different layers and hidden structures by SGSC algorithm.

For the case of first image (ALOE) of **Table 5.8**., if we compare the ground truth with the SGSC output like **Fig. 5.9**, it is seen that our proposed method estimates the dense disparity map and detects the hidden layers of background concurrently. The four different layers are indicated by arrow in experimental output image. We observe that background cloth of left image encompassed with four bends shape with different flow of light in the same horizontal line. So, these different flows of light are mapped clearly by our proposed SGSC algorithm. But no hidden structure or layer is found in ground truth image. BABY-2(*2nd* image) contains some noise which is marked by read color in current state-of-the-art method [37]. We utilized threshold technique in our proposed method to reduce the false matching in active zone. Therefore our method contains comparatively less noise.

5.5.3 Evaluation on Middlebury Optical Flow for Hidden, Synthetic and Stereo Datasets

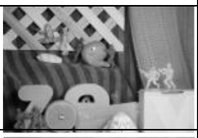
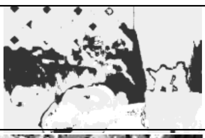
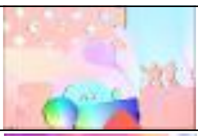
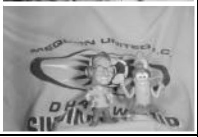
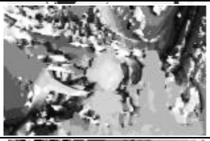
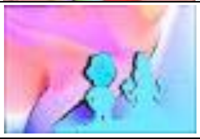
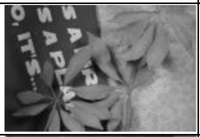
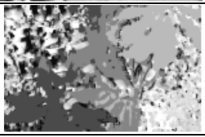
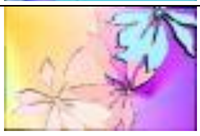
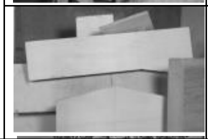
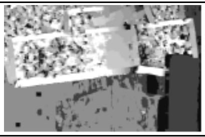
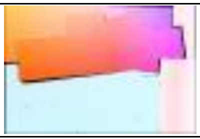
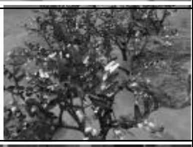
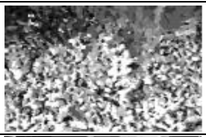
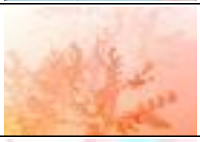
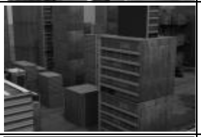
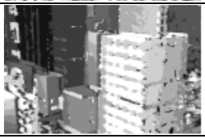
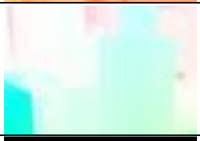
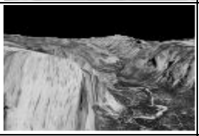
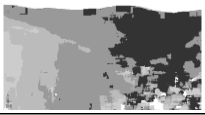
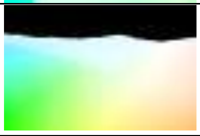
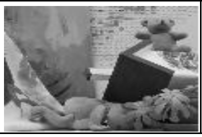
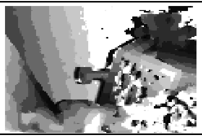
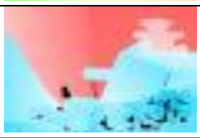
The algorithm has been justified on latest Middlebury Optical Flow datasets. The tested datasets are-

1) Real imagery of nonrigidly moving scenes where dense ground-truth flow is obtained using hidden fluorescent texture painted on the scene [46]. This type of image is called “Hidden Texture” as mentioned in **Table 5.9**.
2) Realistic synthetic imagery texture and
3) Stereo image of rigid scenes modified for optical flow.

The proposed SGSC algorithm shows excellent performance on all types of above mentioned optical flow data. The visual comparisons between SGSC outputs and Middlebury evaluated outputs with hidden ground-truth flow are depicted in **Table**

5.9. The color column (*5th* Column) represents the Middlebury hidden-ground truth flow and *3rd* column represent our experimental hidden ground-truth flow.

Table 5.9: Visual comparisons between SGSC method and current state-of-the-art methods on current Middlebury optical flow datasets.
Source: https://vision.middlebury.edu/flow/data

Image Type	**Reference Image**	**Experimental output of SGSC algorithm**	**Image Name**	**Evaluated Datasets** (With hidden **ground**-truth flow)	**Optical Flow**
Hidden Texture			Army		Yes
			Mequon		Yes
			Schefflera		Yes
			Wooden		Yes
Synthetic Texture			Grove		Yes
			Urban		Yes
			Yosemite		Yes
Stereo			Teddy		Yes

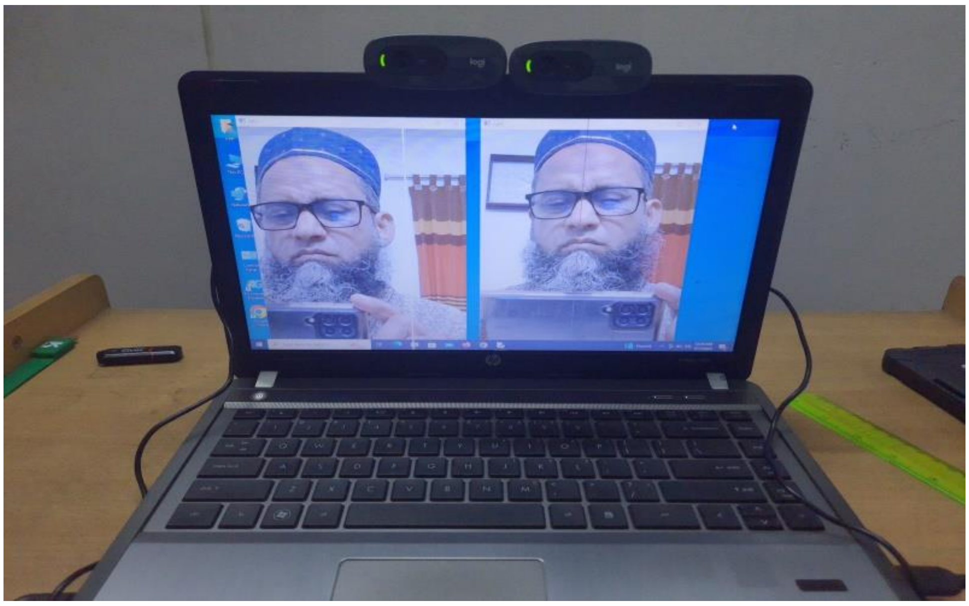
All the color segments, boundaries and objects created by variation of optical flow are detected in our SGSC experimental ground-truth with grey-scale color as like as Middlebury hidden-ground truth. The fluorescent painted textures of two armies are seen at right most bottom of *1st* reference image. These armies are clearly visualized at the same position in our experimental optical flow ground-truth. Whereas, these armies are not clearly visualized in optical flow ground-truth of Middlebury. By observing the *2nd* image (Mequon), it is found that, two heads of Mequon are visualized in both cases. However, in synthetic texture, the urban *1st* building is more clearly detected in our experimental ground-truth than Middlebury. These results confirm us that, the apparent motion of brightness pattern can be sensed by our SGSC algorithm accurately. The unknown difference is found in comparatively more complex Yosemite image data. The difference is that, upper portion of Middlebury ground-truth is black and upper portion of our ground-truth is white. Rests of the flows are captured correctly. Our proposed method not only estimates the stereo ground-truth but also detects the all hidden background textures in modified stereo image of Teddy for optical flow. Therefore, the proposed SGSC algorithm can compute both the stereo and optical flow ground-truth.

5.5.4 Experiment on Real Stereo Images by SGSC

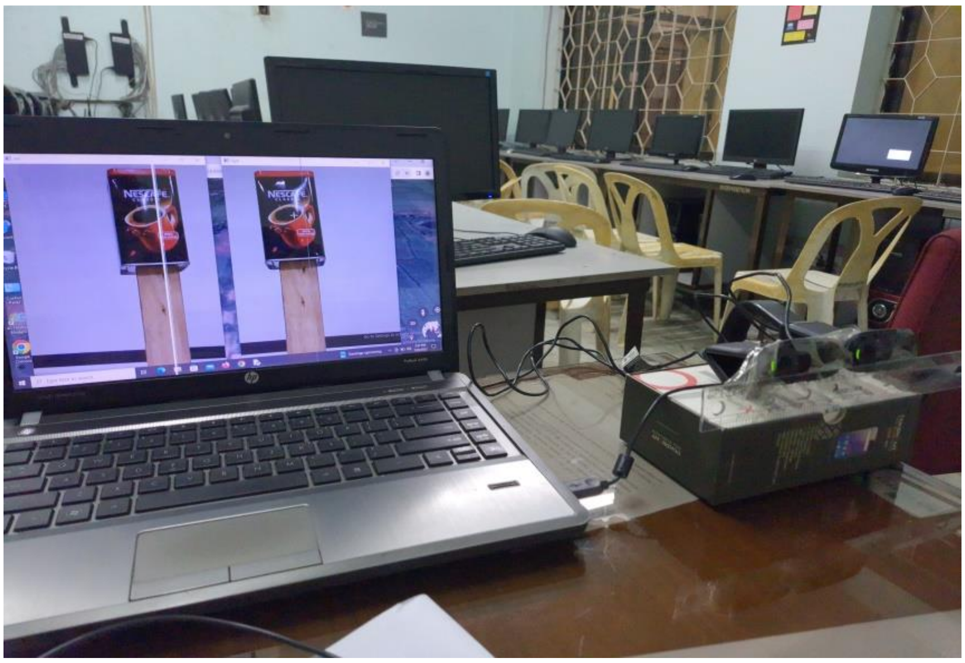
The performances of SGSC algorithm have been further tested on real stereo images acquisitioned by Logitech stereo web camera. This experiment is performed in our software laboratory and images were captured as indoor scenes. The specifications of stereo camera are the same as we mentioned in subsection 2.3.1 and 3.4.4.

Stereo Image Capturing Process:

The main objects (Human face, Nescafe coffee stand and Scotch tape stand) of reference images were stood 63.00 cm away from the imaging sensor of the camera. The distance between two cameras was 6.70 cm. The stereo images are captured by the software at a time.



a) Real image acqusition using stereo web camera for dataset-1(Human face).


b) Real image acqusition using stereo web camera for dataset-2(Nescafe coffee stand).

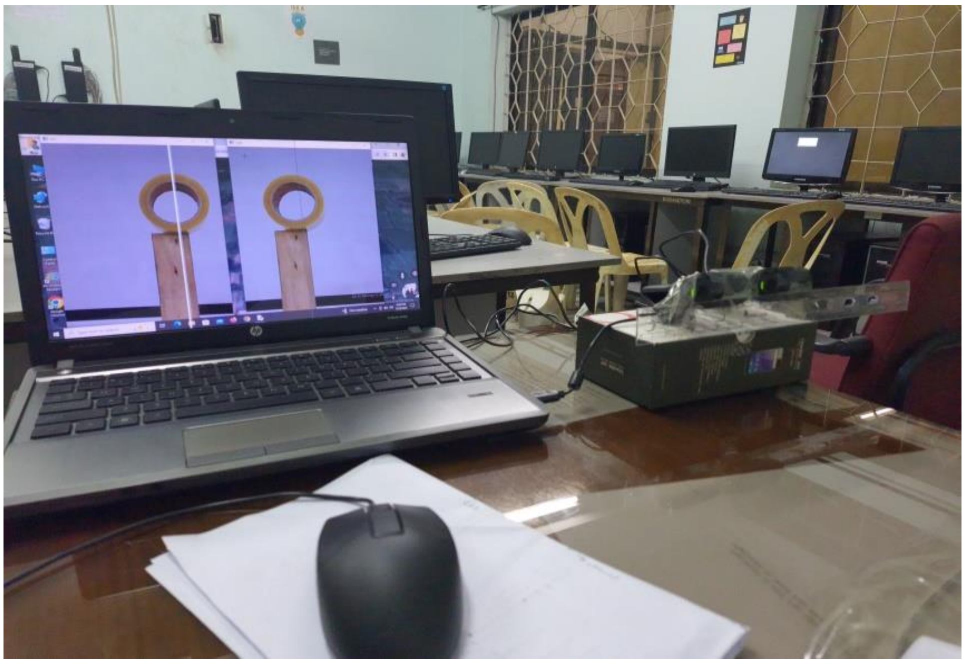
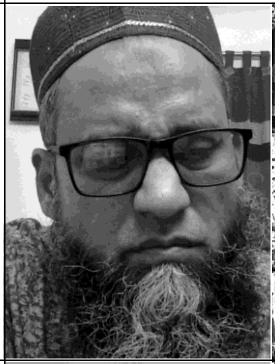
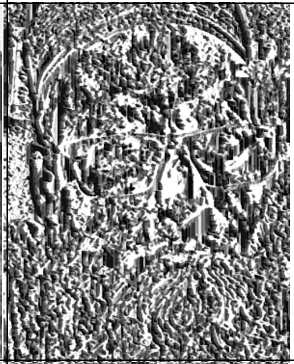
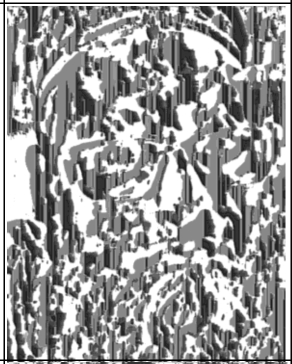
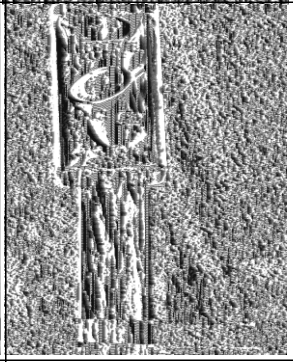
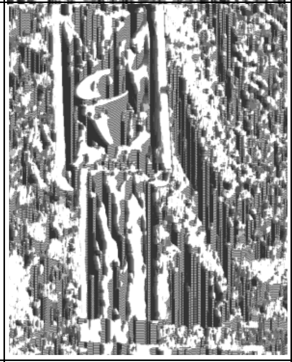
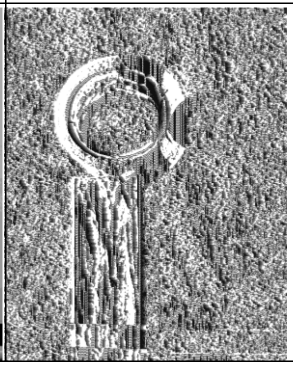
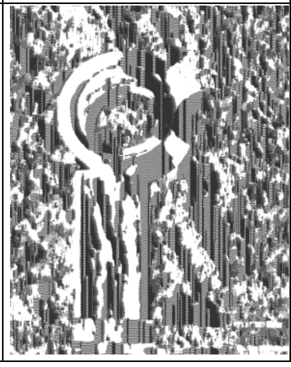

c) Real image acqusition using stereo web camera for dataset-3 (Scotch tape stand).

Fig. 5.10 Real image acqusition process using stereo web camera.

Experimental output for Real Stereo Images:

The size of the left and right real-image is (width × height) = (550 × 720) pixels. Disparity image size is (width × height) = (514 × 684) pixels. **Table 5.10** demonstrates the visual observations and findings of dense disparity map. The **Table 5.10** illustrates the dense disparity maps of three real datasets -Human face, Nescafe coffee stand and Scotch tape stand respectively. Image names and its resolution are mentioned in 1st column. The 2nd column of **Table 5.10** represents the reference image which is captured by our stereo camera. The 3rd and 4th column represent the estimated dense disparity maps of reference images generated by SGSC algorithm using 3×3 and 11×11 mask respectively.

Table 5.10: Visual observation of disparity map of real images generated by SGSC algorithm

Image Name & Resolution	Reference image	Experimental Dense Disparity Map of Real Image Mask size : 3×3	Experimental Dense Disparity Map of Real Image Mask size : 11×11	Execution time(μs)
Dataset-1 (Human face) 550×720				Mask: 3×3 : 469 Mask: 11×11: 5359
Dataset-2 (Nescafe coffee stand) 550×720				Mask: 3×3 : 469 Mask: 11×11: 5374
Dataset-3 (Scotch tape stand) 550×720				Mask: 3×3 : 563 Mask: 11×11: 5985

We could not compare the experimental outputs to the ground truth image because it has no ground truth image. In this situation, the disparity maps of output image should be considered and compared visually only. The disparity maps contain some noise.

This is happened because we could not provide the equilibrium light condition in our laboratory. Similarly the room temperature was not equilibrium at all the places during the image acquisition process. Moreover, we have tried to our best to calibrate the stereo camera physically. The stereo cameras were manually placed on the same horizontal line, but experimentally, it was not possible. There was some vertical mismatch between two cameras in fractional millimeter (.05 mm approximately) range.

These cause to add a little noise in captured stereo images. Inspite of noise, the objects are demarked and recognized at standard level. The object (Human face, Nescafe coffee stand and Scotch tape stand) inside the reference image is clearly understandable and visualized. The dense disparity map generated by 3×3 mask is more visualized and comprehensive than the disparity map of 11×11 in noisy environment. So the overall performance of SGSC algorithm is good in case of real stereo images.

5.6 Discussion

Reducing the computational cost is one of the main aims of this research on stereo correspondence estimation. An inventive method introduced as Self-Guided Stereo Correspondence (SGSC) Estimation Algorithm is implemented here to perform the vision speedily. So, a pioneer core idea of threshold technique is embedded in SAA [Chapter 4] method in order to reduce the search zone as close-fitting as needed. This closefitting impression bans the fake search in its two little territories and thus reduces the computational cost with improvement of 3D structures, object border and accuracies. The frame-rate of our SGSC method is 592 *fps* for Tsukuba image pair and 408 *fps* for Venus stereo images which are the fastest among the current state-of-the-art methods. The outstanding feature of our proposed algorithm is hybrid in nature. It has the adaptability and threshold to process the data both in Middlebury stereo datasets as well as optical flow datasets. This combination makes the SGSC algorithm distinguishable from current state-of-the-art methods.

5.7 Summary

The significant of this Chapter establishes an original optimal method for disparity estimation. The main goal of this method was to reduce the computational cost as well as to increase the accuracy and make the system useful or real-time applications. The concept behind SGSC is described in the Section 5.2. The computational complexity of the algorithm is presented in Section 5.3. The experimental data are tested in different aspects in Section 5.4 and in its Subsections. The performance of SGSC algorithm is tested on additional standard images in Section 5.5 to justify the adaptability of this algorithm. The performance of SGSC algorithm has been further justified on real image datasets in subsection 5.5.4. Experimental results illustrate its efficiencies in **Table 5.7, Table 5.8** and **Table 5.9** both in visually and numerically. An overall achievement has been drawn in Section 5.6 as a discussion.

Chapter 6

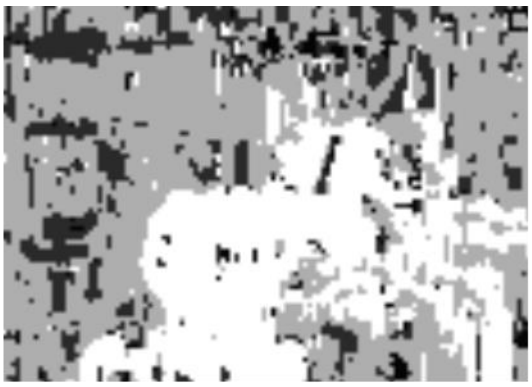
Conclusion and Discussion

6.1 Contributions of the Thesis

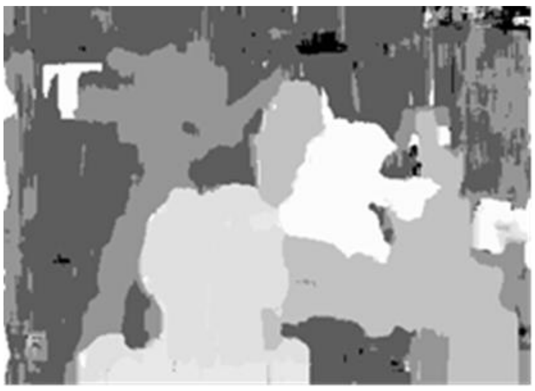
The main objective of this research was to reduce the computational cost, i.e., to improve the performance of stereo matching. The next objective was to improve the accuracy of stereo correspondence estimation in presence of noise.

In this research, we present four contributions on stereo correspondence estimation. We started our first voyage to realize the stereo correspondence using a *Real Time Approximation* (RTA) algorithm. To achieve the first objective, the RTA algorithm has been structured by vector quantization to create the algorithm a real-time one. Then the cost matching scores of the corresponding pixels of the stereo images are approximated to ensure acceptable matching scores. Experimental results prove that this algorithm performs significantly better than the methods employed for window-based existing stereo matching techniques. The computational cost of RTA algorithm is very small compared to the window-based Fast Area Based Algorithm [13]. To justify the effectiveness and validity of this method, we have applied it on the Middlebury standard stereo images. The experimental result of first method is illustrated visually in **Fig. 6.1(a)**. The numerical result is demonstrated in the first row of **Table 6.1** and **Table 6.2**. The computational time reduction is increased by 93.71% and 97.74% compared to reference FAB [13] and EGF [33] methods respectively. Though the computational time reduction of RTA is very good compared to window-based local stereo methods, but its accuracy is only 30%. So, the RTA algorithm can be used where very fast estimation of dense disparity is essential.

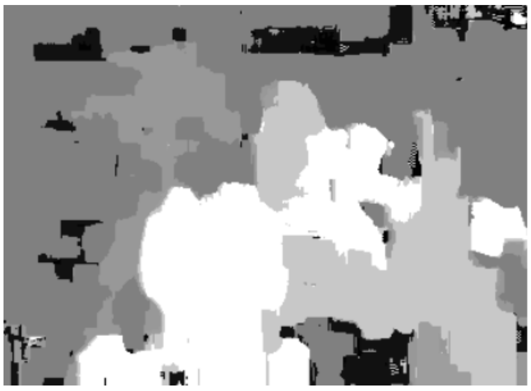
The second achievement of this research is the *2D Real Time Spiral Search Algorithm* (2DRTSSA) for computing the stereo correspondence of the stereo image sequences with a view to implement the first and second objectives at a time. The 2DRTSSA method calculates two window costs; one is along with the $+x$ direction and the other is along with $-y$ direction. The rest two window costs are along with the opposite directions (i.e., $-x$ direction and $+y$ direction) are also calculated using the same procedure. The minimum disparity is estimated from the four window costs. The computational cost of this algorithm is less than the existing state-of-the-art methods [32], [33], [34] and [35]. The visual quality of experimental output is demonstrated in **Fig. 6.1(b),** which shows that the output of 2DRTSSA is better than that of RTA algorithm.



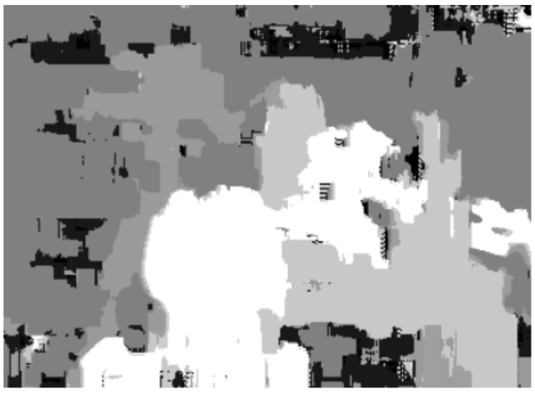
(a) Output image 348 × 252
From RTA Algorithm.



(b) Output image 348 × 252
From 2DRTSSA Algorithm



(c) Output image 348 × 252
From SAA Algorithm.



(d) Output image 348 × 252
From SGSC Algorithm at $\delta_T = 5$

Fig. 6.1 Visual comparison of disparity maps developed by different methods of this research.

The numerical result is demonstrated in the second row of **Table 6.1** and **Table 6.2**. The computational time reduction is decreased by -38.74% and increased by 50.22% compared to reference FAB [13] and EGF [33] methods respectively. The 2DRTSSA method calculates stereo correspondences with good accuracy 93.80%. This method can optimize the speed and accuracy of estimated dense disparity over recent state-of-the-art method.

Table 6.1: Performance comparison of experimental results with Fast Area Based Algorithm, FAB [13] using SAD method. Given that the accuracy of FAB is **86.10%** and computation time is **3229 μs**.

Applied Algorithm	**Computational time (in μs)**	**Time Reduction (in %) compared to FAB**	**Accuracy (in %)**
RTA	203	93.71	30.00
2DRTSAA	4480	-38.74	93.80
SAA	1872	42.02	93.80
SGSC	1689	47.69	97.60

Table 6.2: Performance comparison of experimental results with recent state-of-the-art Edge-aware Geodesic Filter, EGF [33] using SAD method. Given that the accuracy of EGF is **93.67**% and computation time is **9000 μs**.

Applied Algorithm	**Computational time (in μs)**	**Time Reduction (in %) compared to EGF**	**Accuracy (in %)**
RTA	203	97.74	30.00
2DRTSAA	4480	50.22	93.80
SAA	1872	79.20	93.80
SGSC	1689	81.23	97.60

The third and most significant achievement of this research work is the introduction of *Self-Adaptive Algorithm* (SAA) for computing disparity of the stereo images. According to the proposed SAA method, stereo matching search range can be selected dynamically after first matching. That is, by the completion of first search the proposed algorithm remembers the position of matching pixel. So, the second search occurs surrounding to the first matching pixel, as we use the concepts that neighbor pixels have the same photometric properties. Hence depending on the position of matching pixel, the successive search ranges are reselected adaptively. The performance of this algorithm has been tested on Middlebury standard stereo datasets. The performance enhancement of this method was better which is described in details in Section 4.6, Chapter 4. The computational time reduction of SAA method is increased by 42.02% compared to previous fastest literature FAB [13] method and by 79.20% compared to present state-of-the-art, EGF [33] method, with accuracy improvement 7.7% compared to FAB [13] method and 0.13% compared to EGF [33] method.

The fourth and most significant method of this research is *Self-Guided Stereo Correspondence* (SGSC) estimation for disparity of stereo images. Both SGSC and SAA methods are directed by photometric properties of the candidate-pixels. As the photometric properties of reference image pixel and its neighbor's pixel is similar in most cases, so the upcoming corresponding pixel exists in surrounding of the previous matching pixel. Searching performance is greatly improved by utilizing this photometric property of the candidate-pixels. In SGSC the searching performance is further improved by implanting the pioneer threshold technique. We have utilized threshold technique in SAA method; these two techniques jointly reduced the computational costs significantly with further improvement of accuracy over previous methods. The achievements of the proposed SGSC method are testified on Middlebury standard stereo datasets of 2001, 2003, 2006 and Middlebury latest Optical Flow Datasets. Finally, the proposed method is compared with present state-of-the-art methods and SAA. The SGSC experimentally shows that, it outperforms the latest methods in terms of speed, visualization of hidden ground truth, 3D reconstruction and accuracy. The experimental results of SGSC algorithm are figured out in last row of **Table 6.1** and **Table 6.2**, which demonstrates the maximum computational time reduction 81.23% compared to the recent state-of-the-art EGF[33] method and highest accuracy with 97.6%. The accuracy is improved by 11.50% compared to FAB [13] method and 3.93% compared to EGF [33] method. Therefore, SGSC is the optimal algorithm among the algorithms proposed in this research.

6.2 Future Work

This research introduces a novel contribution of four algorithms for stereo correspondence estimation. There are a number of ways to diversify the research in future. Fuzzy logic will be one of the solutions for better understanding the detailed object and to provide more accuracy for visualization purposes. The fuzzy logic may give accurate measurement for the disparity membership function. The reverse-fuzzification technique can be used instead of gray level logic.

Deep learning technique may be employed for better perception of its behaviors hereafter. Besides these, we have a plan to apply our algorithms on color images to observe the capability of our algorithms for color variation. By completion of the said variation, it can be applied to detect the color flow of outdoor scenes of KITTI datasets. We think that variation of both *x* and *y* axis concurrently of SAA or SGSC algorithm will be able to map the hidden ground truth as well as to detect the flow of color variation. In those cases, the modified version of SAA or SGSC may be applied to other benchmarks like KITTI 2015, Robust Vision Challenge and many more with a view to identify the flow of variation and to map the hidden ground truth simultaneously.

6.3 Summary

This chapter describes the main achievements obtained from this research work using four novel algorithms for stereo correspondence. These are: *Real Time Approximation* (RTA), *Two-Dimensional Real Time Spiral Search Algorithm* (2DRTSSA), *Self-Adaptive Algorithm* (SAA) and *Self-Guided Stereo Correspondence* (SGSC) estimation algorithm. The RTA algorithm shows the good performance over existing Fast Area Based Algorithm. The experimental result showed its computational time reduction 93.71%. The second achievement of this research is the *Two-Dimensional Real Time Spiral Search Algorithm* (2DRTSSA) for computing the stereo correspondence of the stereo image sequences with a view to implement the first and second objectives at a time. The performance **Table 6.2** shows that the computational time reduction and accuracy of 2DRTSSA algorithm is 50.22% and 93.80% respectively. The important achievement of this research work is the implementation of *Self-Adaptive Algorithm* (SAA) for computing the stereo correspondence of stereo images with computational time reduction is 79.20% in local stereo matching.

The novel method of this research is the implementation of *Self-Guided Stereo Correspondence* (SGSC) estimation algorithm. The SGSC algorithm achieves 81.23% computational time reduction and 97.60% accuracy. The visual qualities of experimental dense disparity maps outperform the present state-of-the-art methods. Section 6.2 provides some concepts for improving the algorithms for stereo correspondence measurement in future.

References

[1] M. S. Uddin, T. Shioyama, M. H. Chowdhury and A. M. Mondal, “Fast window-based approach for stereo matching,” *Journal of Science, Jahangirnagar University*, vol. 27, pp. 145-154, 2004.

[2] S. T. Barnard and M. A. Fischler, “Stereo vision,” *Encyclopedia of Artificial Intelligence,* (John Wiley, New York, 1987), pp. 1083-1090, 1987.

[3] W. Hoff and N. Ahuja, “Surfaces from stereo: Integrating feature matching, disparity estimation and contour detection,” *IEEE Transections on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 121-136, 1989.

[4] T. Kanade and M. Okutomi, “A stereo matching algorithm with an adaptive window: Theory and experiment,” *IEEE Transections on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, 1994.

[5] O. Veksler, “Stereo matching by compact windows via minimum ratio cycle,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* 2001, pp. 540-547.

[6] S. S. Intille and A. F. Bobick, “Disparity-space images and large occlusion stereo,” in *Proceedings of the European Conference on Computer Vision (ECCV),* 1994, pp. 179-186.

[7] A. Fusiello and V. Roberto, “Efficient stereo with multiple windowing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (ICVPR),* 1997. pp. 858-863.

[8] K. Muhlmann, D. Maier, J. Hesser and R. Manner, “Calculating dense disparity maps from color stereo images, an efficient implementation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (ICVPR),* 2001, pp. 30-36.

[9] M. S. Uddin, “Stereo correspondence estimation using window- based methods by a fast algorithm,” *Journal of Electronics & Computer Science*, vol. 4, pp. 5-11, June 2003.

[10] Md. Abdul Mannan Mondal and Md. Al-Amin Bhuiyan, “Disparity Estimation by a Two-Stage Approximation Real Time Algorithm,” in *Proceedings of the International Management and Technology Conference (IMT)*, 2004, pp. 12-17.

[11] Md. Abdul Mannan Mondal and Md. Haider Ali, “Performance Review of the

Stereo Matching Algorithms," *American Journal of Computer Science and Information Engineering*, vol. 4, no.1, pp- 7-15, 2017.

[12] D.Schartein 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.

[13] Lugi Di Stefano, Massimiliano Marchionni and Stefano Mattoccia, "A fast Area Based Stereo Matching Algorithm," *Image and vision Computing,* vol.22, pp. 983-1005, 2004.

[14] Abijit S. Ogale and Yiannis Aloimonos, "Shape and the Stereo Correspondence Problem," *International Journal of Computer Vision,* vol.65, no. 3, pp.147-167, 2005.

[15] Sukjune Yoon, Sung-Kee Park, Sungehul Kang and Yoon Keun Kwak, "Fast correlation–based stereo matching with the reduction of systematic errors," *Pattern Recognition Letters*, vol. 26, pp. 2221-2231, 2005.

[16] L. Kotoulas, A. Gasteratos, G. Ch. Sirakoulis, C. Georoulas and I. Andreadis, "Enhancement of Fast Acquired Disparity Maps using a 1-D Cellular Automation Filter," in *Proceedings of the fifth IASTED International Conference on Visualization, Imaging and Image Processing*, 2005, pp.7-9.

[17] Lazaros Nalpantidis, Georgios Ch. Sirakoulis and Antonios Gasteratos, "Rieview of stereo matching algorithms for 3D vision," in *Proceedings of the 16th International Symposium on Measurement and Control in Robotics (ISMCR)*, 2007, pp.116-124.

[18] P. H. S. Torra and A. Criminisi, "Dense stereo using pivoted dynamic programming," *Image and Vision Computing,* vol.22, pp. 795-806, 2004.

[19] Salvador Gutierrez and Jose Luis Marroquin, "Robust approach for disparity estimation in stereo vision," *Image and Vision Computing*, vol. 22, pp. 183-195, 2004.

[20] Michael Bleyer and Margrit Gelautz, "A layered stereo matching algorithm using image segmentation and global visibility constraints," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 59, pp.128-150, 2005.

[21] Abijit S. Ogale and Yiannis Aloimonos, "Robust Contrast Invariant Stereo Correspondence," in *Proceedings of the 2005 IEEE International on Robotics and Automation*, 2005, pp. 819-824.

[22] Qingxiong Yang, Liang Wang, Ruigang Yang, Shengnan Wang, Miao Liao and David Nister, " Real-time Global Stereo Matching Using Hierarchical Belief Propagation," in *Proceedings of the British Machine Vision Conference (BMVC),* 2006, pp. 989-998.

[23] Xiaodong Huang and Eric Dubois, "Dense Disparity Estimation based on the continuous wavelet transformation," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE),* 2004, pp. 465-468.

[24] C. Liu, W. Pei, S. Niyokindi, J. C. Song and L. D. Wang, "Micro stereo matching based on wavelet transform and projective invariance," *Measurement Science and Technology*. vol.17, pp. 565 -571, 2006.

[25] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proceeding of the German Conference on Pattern Recognition (GCPR),* September 2014, pp.31-42.

[26] Elisabetta Binaghi, Ignazio Gallo, Giuseppe Marino and Mario Raspanti, "Neural adaptive stereo matching," *Pattern Recognition Letters*, vol.25, pp. 1743-1758, 2004.

[27] Yoon ,Kuk-Jin and In So Kweon, "Adaptive Support –Weight Approach for Correspondence Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650-656, April 2006.

[28] Christopher Zach,Konrad Karner and Horst Bischof, "Hierarchical Disparity Estimation with Programmable 3D Hardware ," *in Proceeding of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2004, pp. 275–282.

[29] D. Min, J. Lu and M. N. Do, "A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy?," *in Proceedings of IEEE International Conference on Computer Vision*, Nov. 2011. pp. 1567-1574.

[30] D.Min, J. Lu and M. N. Do, "Joint histogram-based cost aggregation for stereo matching," *IEEE Transections on Pattern Analysis and Machine Intelligence*, vol.35, no.10, pp. 2539–2545, Oct. 2013.

[31] A. Hosni, C.Rhemann, M. Bleyer, C. Rother and M. Gelautz, "Fast cost volume filtering for visual correspondence and beyond," . *PAMI*, vol.35, pp.504-511, 2013.

[32] Nadia Baha and Slimane Larabi, "Accurate real-time disparity map computation

based on variable support window," *International Journal of Artificial Intelligence & Applications (IJAIA)*, vol.2, no.3, pp. 22-33, July 2011.

[33] Xun Sun , Xing Mei , Shaohui Jiao , Mingcai Zhou , Zhihua Liu and Haitao Wang, "Real-time local stereo via edge-aware disparity propagation," *Pattern Recognition Letters* , vol. 49, pp. 201-206, 2014.

[34] Q. Yang, "Stereo matching using tree filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 834-846, 2015.

[35] Mikhail G. Mozerov and van de Weijer, "Accurate stereo matching by two-step energy minimization," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 1153-1163, March 2015.

[36] Zbontar J. and LeCun Y., " Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition,* October 2015, pp.1592-1599.

[37] Wenhuan Wu,Hong Zhu,Shunyuan Yu and Jing Shi , "Stereo Matching with Fusing Adaptive Support Weights," *IEEE Access*, vol. 7, pp. 61960-61974, May 2019.

[38] Williem and In Kyu Park, "Deep self-guided cost aggregation for stereo matching," *Pattern Recognition Letters*, vol. 112, pp. 168-175, September 2018.

[39] Jia-Ren Chang and Yong-Sheng Chen, "Pyramid Stereo Matching Network," in *Proceeding of the IEEE conference Computer vission and pattern recognition,*June 2018, pp. 5410-5418.

[40] Siti Safwana Abd Razak, Mohd Azlishah Othman and Ahmad Fauzan Kadmin, "The effect of Adaptive Weighted Bilateral Filter on Stereo Matching Algorithm," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 3, pp. 284-287, February 2019.

[41] H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007),*June 2007, pp. 1-8. https://vision.middlebury.edu/stereo/data

[42] D. Scharstein and R. Szeliski, Middlebury Stereo Evaluation – version 2, 2010, https://vision.middlebury.edu/stereo/eval/

[43] Md. Abdul Mannan Mondal, M.S. Thesis, Jahamgirnagar University, Savar, Dhaka, 2007.

[44] Zbontar J. and LeCun Y. ,"Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 2, pp. 1-32, April 2016.

[45] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia and Luigi di Stefano ,"Real-time Self-adaptive deep stereo," in *Proceeding of IEEE conference on Computer vission and pattern recognition(CVPR)*, June 2019, pp. 195-204.

[46] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black and Richard Szeliski ,"A Database and Evaluation Methodology for Optical Flow," *International Journal of ComputerVision*, vol. 92, no. 1, pp.1-31, March 2011.

[47] Computer Vision: Algorithms and Applications, by Richard Szeliski, September 3, 2010, Springer. http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

[48] Mordohai, Philippos and Gerard G. Medioni, "Stereo using monocular cues within the tensor voting framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 28, no. 6, pp. 968-982, 2006.

[49] Veksler,Olga., "Reducing search space for stereo correspondence with graph cuts," *in Proceedings of British Machine Vision Conference*, February 2006, pp.709–718.

[50] Hong, Li and George Chen, "Segment-based stereo matching using graph cuts," *in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2004, pp.74–81.

[51] Kim, Jae Chul, Kyoung Mu Lee, Byoung Tae Choi and Sang Uk Lee, "A dense stereo matching using two-pass dynamic programming with generalized ground control points," *in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2005, pp.1075–1082.

[52] Wang, Liang, Miao Liao, Minglun Gong, Ruigang Yang and David Nister, "High quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proceedings of Third International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006, pp.798–805.

[53] Lei, Cheng, Jason Selzer and Yee-Hong Yang, "Region-tree based stereo using dynamic programming optimization," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2006, pp.2378–2385.

Appendix-A:

Source Code for RTA Algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#define   size_x   384   //original image size
#define   size_y   288

#define   size_x1  128  // 3 times reduced image size
#define   size_y1  96

#define   dmax     10
#define   k1       6
#define   k2       6

#define   kk1      1
#define   kk2      1

#define   w1       3
#define   w2       3

typedef struct{
   int imagesize_x, imagesize_y;
   int **pixel;
}image_t;

image_t allocate_image(const int imagesize_x, const int imagesize_y);
int minimum(const image_t temp[2*dmax+1],int x, int y,int ws,int *p);

void main(){
   image_t image_left, image_right, image_disp;

   int m, n, v_left, v_right, temp, temp1;
   int i, j, ws1, ws2, p, d;
   int sum, sum1, sum2;

   image_t   Mtemp[2*dmax + 1];

   clock_t t0, t1;
   double cpu_speed;
   FILE *cpp, *cpp1, *cpp2, *cpp3, *cpp4, *cpp5, *cpp6;
   char dummy[50] = "";

   cpp  = fopen("f:\\l.pgm", "r+");
   cpp1 = fopen("f:\\r.pgm", "r+");
   cpp2 = fopen("f:\\ls.pgm", "w+");
   cpp3 = fopen("f:\\rs.pgm", "w+");

   fgets(dummy, 50, cpp);
```

```
do{
   fgets(dummy,50,cpp);
}while(dummy[0]=='#');  // All comments lines of .pgm file are
                        //  stored in dummy array.

fgets(dummy,50, cpp);

fgets(dummy, 50, cpp1);
do{
   fgets(dummy,50,cpp1);
}while(dummy[0]=='#');
fgets(dummy,50,cpp1);
fprintf(cpp2,"P2\n%d %d\n255\n", size_x/3,size_y/3);
fprintf(cpp3,"P2\n%d %d\n255\n", size_x/3,size_y/3);
image_left = allocate_image(size_x,size_y);
image_right = allocate_image(size_x,size_y);

t0 = clock ();

for(n = 0; n < size_y; n++){
   for(m = 0; m < size_x; m++){
      fscanf(cpp,"%d", &temp);   // read left input image.
      fscanf(cpp1,"%d", &temp1); // read right input image.
      image_left.pixel[m][n] = temp; //assign the -
                                     //intensity value in array.
      image_right.pixel[m][n] = temp1;
   }
}
// Creation of quantized image.
ws1 = (w1/2); ws2 = (w2/2); p = 0;

for(n = kk2; n < size_y-kk2 ; n += 3){
   for (m = kk1; m < size_x-kk1 ; m += 3){
      sum1 = 0;
      sum2 = 0;
      for(i = -ws1; i <= ws1; i++){
         for(j = -ws2; j <= ws2; j++){
            v_left = image_left.pixel[m+i][n+j];
            v_right = image_right.pixel[m+i][n+j];
            sum1 += v_left;
            sum2 += v_right;
         }
      }
      fprintf(cpp2,"%d ", (int)(sum1/(w1*w2)));
      fprintf(cpp3,"%d ", (int)(sum2/(w1*w2)));
   }
}
fclose(cpp2);
fclose(cpp3);
```

```
cpp4 = fopen("f:\\ls.pgm", "r+");//open the quantized image.
cpp5 = fopen("f:\\rs.pgm", "r+");//open the quantized image.
cpp6 = fopen("f:\\tds.pgm", "w+");// file for dense disparity.
fgets(dummy, 50, cpp4);
do{
   fgets(dummy, 50, cpp4);
}while(dummy[0] == '#');
fgets(dummy, 50, cpp4);

fgets(dummy, 50, cpp5);
do{
   fgets(dummy, 50, cpp5);
}while(dummy[0] == '#');
fgets(dummy, 50, cpp5);

// writing the header and memory allocation for estimated dense
// disparity map.

fprintf(cpp6,"P2\n%d %d\n255\n", size_x1-2*k1, size_y1-2*k2);

image_left = allocate_image(size_x,size_y);
image_right = allocate_image(size_x,size_y);
image_disp = allocate_image(size_x,size_y);

for(d=0; d <= 2*dmax; d++){
   Mtemp[d] = allocate_image(size_x, size_y);
}
// read the quantized images.

for(n = 0; n < size_y1; n++){
   for(m = 0; m <size_x1; m++){
      fscanf(cpp4, "%d", &temp);
      fscanf(cpp5, "%d", &temp1);
      image_left.pixel[m][n] = temp;
      image_right.pixel[m][n] = temp1;
   }
}

t0 = clock();
ws1 = (w1/2);
ws2 = (w2/2); p = 0;
// Window cost calculation process using SAD technique.
for(m = k1; m < size_x-k1 ; m++){
   for(n = k2; n < size_y-k2 ; n++){
      for (d = -dmax; d <= dmax; d++){
         sum = 0;
         for(i =- ws1; i <= ws1; i++){
            for(j = -ws2; j <= ws2; j++){
               v_left = image_left.pixel[m+i][n+j];
               v_right = image_right.pixel[m+i+d][n+j];
               sum += abs(v_left - v_right);
            }
         }
         Mtemp[d + dmax].pixel[m][n] = sum;
      }
      // Assign the best window cost in array using minimum function.
      image_disp.pixel[m][n] = minimum(Mtemp,m,n,ws1,&p);
   }
   for(n = k1; n < size_y1-k1; n++){
      for(m = k2; m < size_x1-k2; m++){
         fprintf(cpp6, "%d ", (image_disp.pixel[m][n]));
      }
   }
   fclose(cpp6);
   int pixel[400][400];
   FILE *cpp7, *cpp8 ;
   cpp7 = fopen("f:\\tds.pgm", "r+");
   // open Shrinked disparity image.
   cpp8 = fopen("f:\\tdz.pgm", "w+"); // Open replicated file.
   fgets(dummy, 50, cpp7);
   do{
      fgets(dummy, 50, cpp7);
   }while(dummy[0] == '#');
   fgets(dummy, 50, cpp7);
   for(n = 0; n < size_y1-2*k2; n++){
      for(m = 0; m <size_x1-2*k1; m++){
         fscanf(cpp7, "%d", &temp);
         // read the Shrinked disparity map.
         pixel[m][n] = temp;
      }
   }
   // write down the header information of replicated image &
   //creating replicated dense disparity map.

   fprintf(cpp8,"P2\n%d %d\n255\n", 348, 252);

   for(n = 0; n < size_y1-2*k2; n++){
      for(m = 0; m <size_x1-2*k1; m++){
         fprintf(cpp8,"%d %d %d ", pixel[m][n], pixel[m][n],pixel[m][n]);
      }
      for (m = 0; m <116; m++){
      fprintf(cpp8,"%d %d %d ", pixel[m][n], pixel[m][n], pixel[m][n]);
   }
   for (m = 0; m <116; m++){
      fprintf(cpp8,"%d %d %d ", pixel[m][n], pixel[m][n], pixel[m][n]);
   }
}
t1 = clock();
```

```
   cpu_speed = ((double) (t1 - t0));
   printf("Total time= %.2f Microseconds\n", cpu_speed);
}


// ------Memory allocation--------.

image_t allocate_image(const int imagesize_x, const int imagesize_y){
   image_t result;
   int x = 0, y = 0;

   result.imagesize_x = imagesize_x;
   result.imagesize_y = imagesize_y;

   result.pixel =(int **) calloc(imagesize_x, sizeof(int*));

   for(x = 0; x < imagesize_x; x++){
      result.pixel[x] =(int*) calloc(imagesize_y, sizeof(int));
      for(y = 0; y < imagesize_y; y++){
         result.pixel[x][y] = 0;
      }
   }
   return result;
}

// -----Best window cost calculation function------.

int minimum(const image_t temp[2*dmax+1],int x, int y, int ws, int *p){
   int i, j = 1, mu, a;
   double min;
   for(i=0; i < 2*dmax+1; i++){
      if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
         j = i;
   }

   min = temp[j].pixel[x][y];
   for(a = 0; a < 2*dmax+1; a++){
      if(a != j && temp[a].pixel[x][y] == min){
         *p+=1;
         break;
      }
   }
   mu = abs(j-dmax);
   return (mu);
}
```

Appendix-B:

Source Code for 2DTRASSA Algorithm

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
#include<math.h>
#define   size_x  384
#define   size_y  288
#define   cmin     10   // (1/2) Search depth.

#define   k1       18
#define   k2       18
#define   w1       11
#define   w2       11
typedef struct{
   int imagesize_x, imagesize_y;
   int **pixel;
}image_t;

image_t allocate_image(const int imagesize_x, const int imagesize_y);
int minimum(const image_t temp[2*cmin+1],int x, int y,int ws,int *p);
void main(){
   image_t   image_left,image_right,image_disp;
   image_t   Mtemp[2*cmin + 1];
   int m, n, v_left, v_right, temp, temp1, v_right1, v_left1;
   int i, j, cid, ws1, ws2, p;      //cid = Spiral Distance.
   int sad1, sad2, dis=0; //dis = Disparity, sad=Sum of absolute difference.
   clock_t t0,t1;
   double cpu_speed;
   FILE *cpp, *cpp1, *cpp2;
   char dummy[50] = " ";
   cpp = fopen("f:\\l.pgm", "r+");      // open input images.
   cpp1 = fopen("f:\\r.pgm", "r+");
   cpp2 = fopen("f:\\td_2D.pgm", "w"); // open output file
   fgets(dummy,50,cpp);               // comments lines are stored in dummy.
   do{
      fgets(dummy, 50, cpp);
   }while(dummy[0] == '#');
   fgets(dummy, 50, cpp);
   fgets(dummy, 50, cpp1);
   do{
      fgets(dummy, 50, cpp1);
   }while(dummy[0]=='#');
   fgets(dummy, 50, cpp1);
   // writing of output file header information.
   fprintf(cpp2, "P2\n%d %d\n255\n", size_x-2*k1, size_y-2*k2);
   // memory allocation.
   image_left = allocate_image(size_x,size_y);
   image_right = allocate_image(size_x,size_y);
   image_disp = allocate_image(size_x,size_y);
```

```
for(cid = 0; cid <= 2*cmin; cid++){
   Mtemp[cid] = allocate_image(size_x, size_y);
}
// Scanning the left and right images.
for (n = 0; n < size_y; n++){
   for (m = 0; m < size_x; m++){
      fscanf(cpp, "%d", &temp);
      fscanf(cpp1, "%d", &temp1);
      image_left.pixel[m][n] = temp;
      image_right.pixel[m][n] = temp1;
   }
}

t0 = clock();
// 2D Window cost calculation process.
ws1 = (w1/2); ws2 = (w2/2); p = 0;
for (m = k1; m < size_x-k1 ; m++){
   for (n = k2; n < size_y-k2 ; n++){
   dis = 0;
      for (cid = -cmin; cid <= cmin; cid++){
         sad1 = 0;
         sad2 = 0;
         for(i =- ws1; i <= ws1; i++){
            for(j = -ws2; j <= ws2; j++){
               if(((m+i+cid*2) >= 0) && ((m+i+cid*2) < size_x)){
                  v_left = image_left.pixel[m+i][n+j];
                  v_right = image_right.pixel[m+i+cid*2][n+j];
                  v_left1 = v_left;
                  v_right1 = image_right.pixel[m+i][n+j+cid*(-2)+1];
                  int abs1 = v_left - v_right;
                  int abs2 = v_left1 - v_right1;
                  if(abs1 < 0) abs1 = -1*abs1;
                  if(abs2 < 0) abs2 = -1*abs2;
                  sad1 += abs1;
                  sad2 += abs2;
               }
            }
         }
         // Select the minimum window cost.
         if(sad1 <= sad2)
            Mtemp[dis++].pixel[m][n] = sad1;
         else
            Mtemp[dis++].pixel[m][n] = sad2;
      }
      // Find the best window cost using minimum function.
      image_disp.pixel[m][n] = 2 * minimum(Mtemp,m,n,ws1,&p);
   }
}
t1 = clock();
```

```
   cpu_speed = ((double) (t1 - t0)); // time calculation.
   printf("Total time = %lf  microseconds. \n",cpu_speed);

   //  Creating the dense disparity map.
   for (n = k1; n < size_y-k1 ; n++){
     for (m = k2; m < size_x-k2 ; m++){
       fprintf(cpp2,"%d ", (image_disp.pixel[m][n]));
      }
   }
}

//Memory Allocation
image_t allocate_image(const int imagesize_x, const int imagesize_y){
   image_t result;
   int x =  0, y = 0;
   result.imagesize_x = imagesize_x;
   result.imagesize_y = imagesize_y;
   result.pixel = (int **) calloc(imagesize_x, sizeof(int*));
   for (x = 0; x < imagesize_x; x++){
      result.pixel[x] = (int*) calloc(imagesize_y, sizeof(int));
      for(y = 0; y < imagesize_y; y++){
         result.pixel[x][y] = 0;
      }
   }
   return result;
}

// Finding the best window cost. i.e. minimum window cost.
int minimum(const image_t temp[2*cmin+1],int x, int y,int ws,int *p){
   int i,j,mu,a;
   double min;
   j = 1;
   for(i = 0;i < 2*cmin+1; i++ ){
      if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
      j = i;
   }

   min = temp[j].pixel[x][y];
   for(a = 0;a<2*cmin+1; a++){
      if (a != j && temp[a].pixel[x][y] == min){
         *p += 1;
         break;
      }
   }
   int abs11 = j-cmin;
   mu = (abs11<0)?(-1*abs11):abs11;
   return (mu);
}
```

Appendix-C:

Source Code for SAA Algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include<time.h>
#include<math.h>
#define   size_x  384
#define   size_y  288
#define   dmax     10
#define   k1       18
#define   k2       18
#define   w1       11
#define   w2       11

typedef struct{
   int imagesize_x, imagesize_y;
   int **pixel;
}image_t;

image_t allocate_image(const int imagesize_x, const int imagesize_y);
int  minimum(const image_t temp[2*dmax+1],int x, int y,int ws);
int flag = 0;

void main(){
   image_t   image_left,image_right,image_disp;
   image_t   Mtemp[2*dmax + 1];
   int m,n,v_left,v_right,temp,temp1;
   int i,j,d,ws1,ws2;
   int sum;
   clock_t t0,t1;
   FILE *cpp,*cpp1,*cpp2;
   char dummy[50] = " ";
   cpp=fopen("imL.pgm", "r+");// opening left and right images.
   cpp1=fopen("imR.pgm", "r+");
   cpp2=fopen("tdadaptive.pgm", "w");
   fgets(dummy,50,cpp); // all comments lines are stored in dummy.

   do{
      fgets(dummy,50,cpp);
   }while(dummy[0]=='#');
   fgets(dummy,50,cpp);

   fgets(dummy,50,cpp1);

   do{
      fgets(dummy,50,cpp1);
   }while(dummy[0]=='#');
   fgets(dummy,50,cpp1);
   fprintf(cpp2,"P2\n%d %d\n255\n",348,252);// output header.
```

```
image_left = allocate_image(size_x,size_y);// memory allocation.
image_right = allocate_image(size_x,size_y);
image_disp  = allocate_image(size_x,size_y);
for(d = 0;d <= 2*dmax; d++){
   Mtemp[d] = allocate_image(size_x, size_y);
}
// scanning of input images.
for (n = 0; n < size_y; n++){
   for (m = 0; m <size_x; m++){
      fscanf(cpp, "%d", &temp);
      fscanf(cpp1, "%d", &temp1);
      image_left.pixel[m][n] = temp;
      image_right.pixel[m][n] = temp1;
   }
}
t0 = clock();
ws1 = (w1/2); ws2=(w2/2);
for (m = k1; m < size_x-k1; m++){
   for (n = k2; n < size_y-k2; n++){
      int dmax1, dmax2;
      if(flag == 0){ // 1st searching region.
         dmax1 = -dmax;
         dmax2 = dmax;
      }
      else if(flag < 0){ // 1st Region.
         dmax1 = -dmax;
         dmax2 = 0;
      }
      else{              // 2nd Region.
         dmax1 = 0;
         dmax2 = dmax;
      }
   // Window costs are computing either in 1st or in 2nd Region using SAD.
      for(d = dmax1; d <= dmax2; d++){
         sum = 0;
         for(i = -ws1; i <= ws1; i++){
            for(j = -ws2; j <= ws2; j++){
               v_left = image_left.pixel[m+i][n+j];
               v_right = image_right.pixel[m+i+d][n+j];
               sum += abs(v_left - v_right);
            }
         }
         Mtemp[d + dmax].pixel[m][n] = sum;
      }

      // Select the minimum window cost.
      image_disp.pixel[m][n] = minimum(Mtemp, m, n, ws1);
   }
}
```

```
   t1 = clock(); // time calculation.
   printf(" Total time = %d  microseconds\n",(t1 - t0));

   for (n = k1; n < size_y-k1 ; n++){
      for (m = k2; m < size_x-k2 ; m++){
         fprintf(cpp2,"%d ",(image_disp.pixel[m][n]));
      }
   }
}
//Memory Allocation
image_t allocate_image(const int imagesize_x, const int imagesize_y){
   image_t result;
   int x =  0, y = 0;

   result.imagesize_x = imagesize_x;
   result.imagesize_y = imagesize_y;

   result.pixel = (int **) calloc(imagesize_x, sizeof(int*));

   for(x = 0; x < imagesize_x; x++){
      result.pixel[x] = (int*) calloc(imagesize_y, sizeof(int));
      for(y = 0; y < imagesize_y; y++){
         result.pixel[x][y] = 0;
      }
   }
   return result;
}
// Finding the minimum window cost and its co-ordinate distance.

int  minimum(const image_t temp[2*dmax+1],int x, int y,int ws){
   int i, j = 1, mu, a;
   double min;
   int len = 2*dmax+1;
   if(flag != 0)
      len = dmax+1;
   for(i = 0;i < len; i++){
      if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
         j = i;
  }

   if (j < 10) // selecting the search region.
      flag = -1; // for 1st region.
   else
      flag = 1; // for 2nd region.

   mu = abs(j-dmax); // return the best matching co-ordinate distance.
   return (mu);
}
```

Accuracy Measurement Code:

```
//for measuring the accuracy of self-adaptive algorithm
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include<time.h>
#include<math.h>
#define   size_x  384
#define   size_y  288
#define   dmax     20
#define   k1       18
#define   k2       18
#define   k        7
typedef struct{
   int imagesize_x, imagesize_y;
   int **pixel;
}image_t;
image_t allocate_image(const int imagesize_x, const int imagesize_y);
int  minimum(const image_t temp[2*dmax+1],int x, int y,int ws);
int flag = 0;

int main(){
   image_t   image_left,image_right,image_disp,image_tdisp;
   image_t   Mtemp[2*dmax + 1];
   int m, n, v_left, v_right, temp, temp1;
   int i, j, d, ws1, ws2, p, dd;
   int sum, w1, w2;
   double correct1[20] = {0},correct2[20] = {0};
   int total_time,time_dmax;
   double totaldata = (size_x-(2*k1+2*k))*(size_y-2*k2);
   clock_t t0,t1;
   FILE *cpp,*cpp1,*cpp2;
   char dummy[50] = "";
   cpp  = fopen("imL.pgm", "r+");// open input images.
   cpp1 = fopen("imR.pgm", "r+");
   cpp2 = fopen("td.pgm", "r+");// open ground truth image.
   fgets(dummy,50,cpp);
   do{
      fgets(dummy,50,cpp);    // all comments are removed to dummy.
   }while(dummy[0]=='#');
   fgets(dummy,50,cpp);
   fgets(dummy,50,cpp1);
   do{
      fgets(dummy,50,cpp1);
   }while(dummy[0]=='#');
   fgets(dummy,50,cpp1);
   fgets(dummy,50,cpp2);
   do{
      fgets(dummy,50,cpp2);
   }while(dummy[0]=='#');
   fgets(dummy,50,cpp2);
   fprintf(cpp2,"P2\n%d %d\n255\n",348,252);// writing header info.

   // ********* memory allocations************

   image_left  = allocate_image(size_x,size_y);
   image_right = allocate_image(size_x,size_y);
   image_tdisp = allocate_image(size_x,size_y);
   image_disp  = allocate_image(size_x,size_y);

   for(d = 0;d <= 2*dmax; d++){
      Mtemp[d] = allocate_image(size_x, size_y);
   }

   //----------Read input images-----------.
   for (n = 0; n < size_y; n++){
      for (m = 0; m <size_x; m++){
         fscanf(cpp,"%d",&temp);
         fscanf(cpp1,"%d",&temp1);
         image_left.pixel[m][n]  = temp;
         image_right.pixel[m][n] = temp1;
      }
   }
   for (n = k1; n < size_y-k1; n++){
      for (m = k2; m <size_x-k2; m++){
         fscanf(cpp2,"%d",&temp);
         image_tdisp.pixel[m][n] = temp;

      }
   }
   printf("__________________________________________________\n");
   printf("  Window size    Accuracy (%%)  Computational time\n");
   printf("  (pixel)  correct match    Diff %c1 pixel (Microsecond) \n",241);
   printf("________________________________________________\n");
   total_time = 0;

   //-------- Dividing the search area ------------
   for(w1 = 3,w2 = 3;w1 <= 15,w2 <= 15;w1 += 2,w2 += 2){
      t0 = clock();
      ws1 = (w1/2); ws2 = (w2/2); p = 0;
      for (m = k1; m < size_x-k1 ; m++){
         for (n = k2; n < size_y-k2 ; n++){
            int dmax1,dmax2;
            if(flag == 0){
               dmax1 = -dmax;
```

```
               dmax2 = dmax;
            }
            else if(flag < 0){             // 1st Region.
               dmax1 = -dmax;
               dmax2 = 0;
            }
            else{                          // 2nd Region.
               dmax1 = 0;
               dmax2 = dmax;
            }

// **** window cost calculation either in 1st or in 2nd Region***
            for (d = dmax1; d <= dmax2; d++){
               sum = 0;
               for(i =- ws1; i <= ws1; i++){
                  for(j = -ws2; j <= ws2; j++){
                     if(((m+i+d)>=0) && ((m+i+d)<size_x )){
                        v_left =  image_left.pixel[m+i][n+j];
                        v_right= image_right.pixel[m+i+d][n+j];
                        sum += abs(v_left - v_right);
                     }
                  }
               }
               Mtemp[d + dmax].pixel[m][n] = sum;
            }
            image_disp.pixel[m][n] = minimum(Mtemp,m,n,ws1);
         }
      }
      t1 = clock();
      total_time = (t1-t0);

      // ----- Accuracy Measurement steps -------.
      for(n = k1; n < size_y-k1; n++){
         for(m = k2; m <size_x-k2; m++){
            if(abs(image_disp.pixel[m][n]-image_tdisp.pixel[m][n])==0)
               correct1[w1]++;
            if(abs(image_disp.pixel[m][n]-image_tdisp.pixel[m][n])<=2)
               correct2[w1]++;
         }
      }
   }

   // ----- Writing the output data for different Masks-------.
   if(w1 == 3)
      printf("     %d x %d       %.1lf %%         %.1lf %%   %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 5)
      printf("     %d x %d       %.1lf %%         %.1lf %%   %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 7)
      printf("     %d x %d       %.1lf %%          %.1lf %%    %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 9)
      printf("     %d x %d       %.1lf %%          %.1lf %%     %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 11)
      printf("    %d x %d      %.1lf %%          %.1lf %%       %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 13)
      printf("    %d x %d      %.1lf %%          %.1lf %%       %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   else if(w1 == 15)
      printf("    %d x %d      %.1lf %%          %.1lf %%       %d\n", w1, w2,
            (correct1[w1]/totaldata)*100, (correct2[w2]/totaldata)*100,
            total_time);
   printf("_______________________________________________________\n");
   time_dmax = total_time / 3;
}

image_t allocate_image(const int imagesize_x, const int imagesize_y){
   image_t result;
   int x =  0, y = 0;
   result.imagesize_x = imagesize_x;
   result.imagesize_y = imagesize_y;
   result.pixel = (int **) calloc(imagesize_x, sizeof(int*));
   for(x = 0; x < imagesize_x; x++){
      result.pixel[x] = (int*) calloc(imagesize_y, sizeof(int));
      for(y = 0; y < imagesize_y; y++){
         result.pixel[x][y] = 0;
      }
   }
   return result;
}

//**Finding the best window cost and its co-ordinate distance**
int  minimum(const image_t temp[2*dmax+1],int x, int y,int ws){
   int i,j = 1, mu, a;
   double min;
   int len = 2 * dmax + 1;
   if(flag != 0)
      len = dmax + 1;
   for(i = 0;i < len; i++){
      if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
         j = i;
   }
   if(j < 20)
      flag = -1; // set 1st region.
   else
   flag = 1; // set 2nd region.
   mu = abs(j - dmax);
   return (mu);
}
```

Appendix-D:

Source Code for SGSC Algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
#include<time.h>
#include<math.h>

#define   size_x  384
#define   size_y  288
#define   dmax     10
#define   k1       18
#define   k2       18
#define   w1       11
#define   w2       11

typedef struct{
   int imagesize_x, imagesize_y;
   int **pixel;
}image_t;

image_t allocate_image(const int imagesize_x, const int imagesize_y);
int  minimum(const image_t temp[2*dmax+1], int x, int y, int ws);
int flag=0;
image_t image_left,image_right,image_disp;

int main(){
   image_t Mtemp[2*dmax + 1];
   int m, n, v_left, v_right, temp, temp1;
   int i, j, d, ws1, ws2;
   int sum;
   clock_t t0,t1;
   FILE *cpp, *cpp1, *cpp2;
   char dummy[50] = "";
   cpp  = fopen("imL.pgm", "r+");// open input images.
   cpp1 = fopen("imR.pgm", "r+");
   cpp2 = fopen("td_SGSC.pgm", "w");// open dense disparity file.
   fgets(dummy, 50, cpp);

   do{
      fgets(dummy, 50, cpp); // all comments lines are moved to dummy.
   }while(dummy[0]  =='#');
   fgets(dummy, 50, cpp);
   fgets(dummy, 50, cpp1);

   do{
      fgets(dummy, 50, cpp1);
   }while(dummy[0] == '#');
   fgets(dummy, 50, cpp1);
   fprintf(cpp2,"P2\n%d %d\n255\n",348,252);// write file header info.
```

```
// ******** Memory Allocation ************
image_left = allocate_image(size_x,size_y);
image_right = allocate_image(size_x,size_y);
image_disp = allocate_image(size_x,size_y);
for(d = 0;d <= 2*dmax; d++){
   Mtemp[d] = allocate_image(size_x, size_y);
}

// ------ Scanning the input images ----------
for (n = 0; n < size_y; n++){
   for (m = 0; m <size_x; m++){
      fscanf(cpp,"%d", &temp);
      fscanf(cpp1,"%d", &temp1);
      image_left.pixel[m][n] = temp;
      image_right.pixel[m][n] = temp1;
   }
}
t0 = clock();

//--- Dividing the searching zone --------
ws1 = (w1/2); ws2=(w2/2);
for (m = k1; m < size_x-k1; m++){
   for (n = k2; n < size_y-k2; n++){
      int dmax1, dmax2;
      if(flag == 0){
         dmax1 = -dmax;
         dmax2 = dmax;
      }
      else if(flag < 0){    // 1st Zone.
         dmax1 = -dmax;
         dmax2 = 0;
      }
      else{                  // 2nd Zone.
         dmax1 = 1;
         dmax2 = dmax;
      }
      //** Window cost calculation either in 1st or in 2nd Zone using SAD**
      for (d = dmax1; d <= dmax2; d++){
         sum = 0;
         for(i = -ws1; i <= ws1; i++){
            for(j = -ws2; j <= ws2; j++){
               v_left  = image_left.pixel[m+i][n+j];
               v_right = image_right.pixel[m+i+d][n+j];
               sum += abs(v_left - v_right);
            }
         }
         Mtemp[d + dmax].pixel[m][n] = sum;
      }
```

```
         // Save the best window cost using minimum function.
         image_disp.pixel[m][n]=minimum(Mtemp, m, n, ws1);
      }
   }
   t1 = clock();// time calculation.
   printf(" Total time = %d  Microseconds\n",(t1-t0));

   //***** writing the output dense disparity image*******
   for (n = k1; n < size_y-k1 ; n++){
      for (m = k2; m < size_x-k2 ; m++){
         fprintf(cpp2,"%d ", (image_disp.pixel[m][n]));
      }
   }
}
// **** Memory Allocation ***
image_t allocate_image(const int imagesize_x, const int imagesize_y){
   image_t result;
   int x = 0, y = 0;
   result.imagesize_x = imagesize_x;
   result.imagesize_y = imagesize_y;
   result.pixel = (int **) calloc(imagesize_x, sizeof(int*));

   for(x = 0; x < imagesize_x; x++){
      result.pixel[x] = (int*) calloc(imagesize_y, sizeof(int));
      for(y = 0; y < imagesize_y; y++){
         result.pixel[x][y] = 0;
     }
   }
   return result;
}
// ***Finding the best window cost i.e. minimum window cost***
int minimum(const image_t temp[2*dmax+1],int x, int y, int ws){
   int i, j = 1, mu, a, Th=5;
   double min;
   int len = 2*dmax+1;
   if(flag != 0)
     len = dmax+1;
   for(i = 0; i < len; i++){
     if(temp[i].pixel[x][y] < temp[j].pixel[x][y])
       j = i;
   }
   mu = abs(j - dmax);
   // --- Applying Threshold technique in active Zone----
   if(abs(image_disp.pixel[x][y-1]- mu) <= Th)
      flag = -1; // Set 1st Zone as active.
   else
      flag = 1; // Set 2nd Zone as active.
   return (mu);
}
```