ACCELERATED FRAMEWORK FOR SELECTIVE BLURRING WITH OCCLUSION HANDLING USING DEPTH MAPS

Thesis

Submitted in partial fulfilment of the requirements for the degree of

MASTER OF TECHNOLOGY (RESEARCH) in

INFORMATION TECHNOLOGY

By

SUBHAYAN MUKHERJEE

(Reg No: 121114IT12F01)

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

OCTOBER 2014

DECLARATION

By M.Tech (Research) Scholar

I, **SUBHAYAN MUKHERJEE** hereby declare that the Research Thesis entitled "**ACCELERATED FRAMEWORK FOR SELECTIVE BLURRING WITH OCCLUSION HANDLING USING DEPTH MAPS**" which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfillment of the requirements for the award of the Degree of **M. Tech (Research)** in **Information Technology** is a *bonafide report of the research work carried out by me*. The material contained in this Research Thesis has not been submitted to any University or Institute for the award of any degree.

Place: NITK, Surathkal
Date: OCTOBER, 2014

Subhayan Mukherjee
(Reg. No.: IT12F01)

CERTIFICATE

This is to *certify* that the *Research Thesis* entitled "**ACCELERATED FRAMEWORK FOR SELECTIVE BLURRING WITH OCCLUSION HANDLING USING DEPTH MAPS**" is a bonafide work carried out by **SUBHAYAN MUKHERJEE,** (REGISTER NO: **121114IT12F01**) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of **M.Tech (Research)** in **Information Technology**.

Prof. G. Ram Mohana Reddy
(Research Guide)

Prof. Ananthanarayana V. S.
(Chairman - DPGC/DRPC)

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this research thesis.

First and foremost, I would like to express my sincere gratitude to my guide *Prof. Ram Mohana Reddy Guddeti*, Department of Information Technology, National Institute of Technology Karnataka, Surathkal. His advice, constant support, encouragement and valuable suggestions help me throughout the course of my thesis work. Without his continuous support and interest, this report would not have been the same as presented here.

I express my deep gratitude to *Prof. Ananthanarayana V. S.*, Head of the Department of Information Technology, National Institute of Technology Karnataka, Surathkal for his constant co-operation, support and for providing necessary facilities throughout the M.Tech (Research) program.

I would like to take this opportunity to express my thanks towards the teaching and non-teaching staff in the Department of Information Technology. I am also grateful to all my classmates for their help, encouragement and suggestions.

I am thankful to my mother who continuously supported and encouraged me in every possible way for the successful completion of this thesis, and to my deceased father, who had always motivated me strongly to pursue a good academic career.

— Subhayan Mukherjee

ABSTRACT

In recent times, 3D videography, and particularly, 3D movies have become very popular. A 3D video is recorded by capturing temporal 3D image sequences of the same scene using two 2D video recording units separated by a distance similar to that between the two human eyes. Later, when both the left and right streams of the 3D video are played simultaneously, we get the impression of depth. However, depth information can be extracted from 2D videos as well using certain visual cues present in the video frames like variation of focus, contrast etc. Moreover, using special depth cameras like Microsoft Kinect, it is possible to record the depth of a scene while shooting the scene.

Depth-of-field blurring is a technique which has been extensively used in photo-shoots and video recordings. The depth of field (DOF) of an image is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in that image. Nowadays, cameras have DOFs that correspond to a single slab that is perpendicular to their optical axis. However, such a configuration poses severe limitations on how we can selectively focus multiple subjects situated at largely differing depths, and on how we can handle similar challenging focusing scenarios which require a flexible DOF.

Here, the primary intention is to propose a framework to extract depth information from 3D images, and subsequently use that to control the DOF in new and powerful ways. Attempts at post-processing 2D images to create special effects by altering the DOF have been made, but in this respect, 3D is a relatively unexplored area. Furthermore, image processing is an inherently computationally intensive task, and more so for 3D, where we have to process a pair of images instead of just one. Hence, the processing needs to be carried out using parallel algorithms to reduce the execution time. Lastly, the effectiveness of depth estimation from 2D images, 3D images and Microsoft Kinect depth camera are also compared qualitatively. It is believed that flexible DOF imaging can open a new creative dimension in videography and lead to new capabilities in medical, forensic and scientific imaging, vision, and graphics.

Keywords : *3D Video Post-Processing, 3D Camcorder, Stereo Disparity, Microsoft Kinect, Depth Camera, Flexible Depth-of-Field, Depth Blurring, Selective Blurring, Occlusion Handling, Parallel Processing, Depth Map*

Contents

List of Figures

List of Tables

Abbreviations

$2D$	Two Dimensional
$3D$	Three Dimensional
$3D-TOF$	Three Dimensional Time of Flight
$3DTV$	Three Dimensional Television
AD	Absolute Difference
AI	Task Tracker
$APARAPI$	A Parallel Application Programming Interface
API	Application Programming Interface
CGI	Computer Generated Imagery
CID	Computed Image Depth
CIE	Commission Internationale de l'Eclairage
$CMOS$	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CRT	Cathode Ray Tube
$CUDA$	Compute Unified Device Architecture
$DIBR$	Depth Image Based Rendering
DOF	Depth of Field
DP	Dynamic Programming
DTW	Dynamic Time Warp
$GPGPU$	General Purpose computing on Graphics Processing Units
GPU	Graphics Processing Units
GUI	Grapical User Interface
JTP	Java Thread Pool
$LIDAR$	Light Detection and Ranging
LCD	Liquid Crystal Display
LRC	Left Right Consistency
MPI	Message Passing Interface
NCC	Normalized Cross Correlation
$OpenCL$	Open Computing Language
$OpenCV$	Open Source Computer Vision
$OpenMP$	Open Multi-Processing
PC	Personal Computer

$POSIX$	Portable Operating System Interface
RGB	Red Green Blue
$RGB-D$	Red Green Blue Depth
SAD	Sum of Absolute Differences
SD	Squared Difference
SfM	Structure from Motion
SLR	Single Lens Reflex
SO	Scanline Optimization
SSD	Sum of Squared Differences
$TrueHD$	True High Definition
WTA	Winner Takes it All

1 INTRODUCTION

It is well known that we move around in our physical environment which is three-dimensional (3D). Humans are able to perceive the spatial relationship between objects just by looking at them because we have 3D perception, also known as depth perception. As we look around, the retina in each eye forms a two-dimensional (2D) image of our surroundings and our brain processes these two images into a 3D visual experience. However, it is important to note that having vision in both eyes (stereoscopic or binocular vision) is not the only way to see in 3D. People who can only see with one eye (monocular vision) can still perceive the world in 3D, and may even be unaware that they are stereo blind. They are simply missing one of the tools to see in 3D, so they rely on the remaining ones. Below are some of the tools humans use for depth perception.

1.1 Human Depth Perception

Human depth perception is described by the following features:

1. Stereoscopic Vision: Two eyes provide slightly separate images; closer objects appear more separated than distant ones.

2. Accommodation: As we focus on a close or distant object, the lenses in our eyes physically change shape, providing a clue as to how far away the object is.

3. Parallax: As our head moves from side to side, closer objects appear to move more than distant ones.

4. Size Familiarity: If we know the approximate size of an object, we can tell approximately, how far away it is based on how big it looks. Similarly, if we know that two objects are a similar size to each other but one appears larger than the other, we will assume the larger object is closer.

5. Aerial Perspective: As light is scattered randomly by air, distant objects appear to have less contrast than nearby objects. Distant objects also appear less color-saturated and have a slight color tinge similar to the background (usually blue).

In order to represent the 3D world on a flat (2D) surface such as a display screen, it is desirable to simulate as many of these perception tools as possible. Although there is

currently, no way to simulate all of them at the same time, video does use a combination. For example, aerial perspective and size familiarity are automatically captured by the video camera. In CGI scenes, aerial perspective must be added so that distant objects appear less clearly, (this is called distance fog). Of course, the addition of stereoscopic images (a separate image for each eye) is a significant improvement, so much so, that most people think of stereoscopic films as being 3D, and all others as being 2D.

1.2 Videos from 2D Camcorders

A 2D video, more commonly referred to as simply a *video*, is a sequence of moving images having a contextual and temporal relationship with each other. A traditional 2D video image has width and height but technically, it has no depth, i.e. everything in the image is presented at the same distance from the viewer. Still, the viewer does perceive the image as 3D by subconsciously using the techniques listed above–much the same as how stereo-blind people perceive the real world.

1.3 Videos from 3D Camcorders

3D video adds stereoscopic vision, meaning that two separate images are shown simultaneously, one to each eye. This presents enormous technical problems, which is why there is still no perfect system almost 100 years since the first 3D movie was made. Common display methods include the following:

1. Anaglyphic processing (red / cyan glasses): The original 3D system, now largely out of favor.

2. Polarized light system (polarized filter glasses): The most common new system for cinemas.

3. Active shutter system (LCD shutter glasses): The most likely standard for the first generation of 3D televisions and other displays.

1.3.1 Anaglyphic Processing

Anaglyph 3D is the name given to the stereoscopic 3D effect achieved by means of encoding each eye's image using filters of different (usually chromatically opposite) colors, typically red and cyan. Anaglyph 3D images contain two differently filtered colored images, one for each eye. When viewed through the color-coded anaglyph glasses, each of the two images reaches one eye, revealing an integrated stereoscopic image. The visual cortex of the brain fuses this into perception of a 3D scene.

1.3.2 Polarized Light System

A polarized 3D system uses polarization glasses to create the illusion of 3D images by restricting the light that reaches each eye, an example of stereoscopy. To present stereoscopic images and films, two images are projected superimposed onto the same screen or display through different polarizing filters. The viewer wears low-cost eyeglasses which contain a pair of different polarizing filters. As each filter passes only that light which is similarly polarized and blocks the light polarized in the opposite direction, each eye sees a different image. This is used to produce a 3D effect by projecting the same scene into both eyes, but depicted from slightly different perspectives. Several people can view the stereoscopic images at the same time.

1.3.3 Active Shutter System

An active shutter 3D system (a.k.a. alternate frame sequencing, alternate image, AI, alternating field, field sequential or eclipse method) is a technique of displaying stereoscopic 3D images. It works by openly presenting the image intended for the left eye while blocking the right eye's view, then presenting the right-eye image while blocking the left eye, and repeating this so rapidly that the interruptions do not interfere with the perceived fusion of the two images into a single 3D image.

An active shutter 3D system generally uses liquid crystal shutter glasses (also called active shutter glasses). Each eye's glass contains a liquid crystal layer which has the property of becoming dark when voltage is applied, being otherwise transparent. The glasses are controlled by a timing signal that allows the glasses to alternately darken over one eye, and then the other, in synchronization with the refresh rate of the screen. The timing synchronization to the video equipment may be achieved via a wired signal, or wirelessly by either an infrared, radio frequency, Bluetooth or optical transmitter.

Active shutter 3D systems are used to present 3D films in some theaters. It can be used to present 3D images on CRT, plasma, LCD and other types of video displays.

1.4 Microsoft Kinect Depth Camera

A depth camera is a device that can capture the depth information of a scene using special built-in depth sensors and infrared light. A very common example is the Microsoft Kinect Depth Camera shown in Fig. 1.1.

Kinect is a actually a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface using ges-

Figure 1.1: Microsoft Kinect Depth Camera.

tures and spoken commands. The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone running proprietary software, which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

Kinect's sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and the Kinect software is capable of automatically calibrating the sensor based on game-play and the player's physical environment, accommodating for the presence of furniture or other obstacles.

1.5 Depth of Field (DOF)

In optics, particularly as it relates to film and photography, depth of field (DOF) is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. Although a lens can precisely focus at only one distance at a time, the decrease in sharpness is gradual on each side of the focused distance, so that within the DOF, the un-sharpness is imperceptible under normal viewing conditions. In some cases, it may be desirable to have the entire image sharp, and a large DOF is appropriate. In other cases, a small DOF may be more effective, emphasizing the subject while de-emphasizing the foreground and background. In cinematography, a large DOF is often called deep focus, and a small DOF is often called shallow focus.

1.5.1 Digital Techniques Affecting DOF

The advent of digital technology in photography has provided additional means of controlling the extent of image sharpness; some methods allow extended DOF that would be impossible with traditional techniques, and some allow the DOF to be determined

after the image is made. Some of them are explained below in brief.

1. Focus Stacking is a digital image processing technique which combines multiple images taken at different focus distances to give a resulting image with a greater depth of field than any of the individual source images. Available programs for multi-shot DOF enhancement include Adobe Photoshop, Syncroscopy AutoMontage, PhotoAcute Studio, Helicon Focus and CombineZ. Getting sufficient depth of field can be particularly challenging in macro photography.

2. Wavefront Coding is a method that convolves rays in such a way that it provides an image where fields are in focus simultaneously with all planes out of focus by a constant amount.

3. Colour Apodisation is a technique combining a modified lens design with image processing to achieve an increased depth of field. The lens is modified such that each colour channel has a different lens aperture. For example the red channel may be f/2.4, green may be f/2.4, whilst the blue channel may be f/5.6. Therefore the blue channel will have a greater depth of field than the other colours. The image processing identifies blurred regions in the red and green channels and in these regions copies the sharper edge data from the blue channel. The result is an image that combines the best features from the different f-numbers.

Another set of algorithms exist which deal with simulating synthetic depth-of-field effects (like shallow focus) in scenes captured using conventional digital cameras. Such algorithms are important for drawing human attention to regions of interest.

1.6 Depth Maps

A depth map is an image or image channel containing information relating to the distance of the surfaces of scene objects from a viewpoint. Depth maps have a number of uses, including the following:

1. Simulating the effect of uniformly dense semi-transparent media within a scene, such as fog, smoke or large volumes of water.

2. Simulating shallow Depth of fields, where some parts of a scene appear to be out of focus. Depth maps can be used to selectively blur an image to varying degrees. A shallow depth of field can be a characteristic of Macro photography and so the technique may form a part of the process of Miniature faking.

3. Z-buffering and z-culling, techniques which can be used to make the rendering of 3D scenes more efficient. They can be used to identify objects hidden from view and which may therefore be ignored for some rendering purposes. This is particularly important in real time applications such as computer games, where a fast succession of completed renders must be available in time to be displayed at a regular and fixed rate.

4. Shadow mapping, which is part of one process used to create shadows cast by illumination in 3D computer graphics. In this use, the depth maps are calculated from the perspective of the lights, not the viewer.

5. To provide the distance information needed to create and generate Autostereograms and in other related applications intended to create the illusion of 3D viewing through stereoscopy.

6. Subsurface scattering, which can be used as part of a process for adding realism by simulating the semi-transparent properties of translucent materials such as human skin.

1.7 Depth Estimation

Depth estimation or extraction refers to the set of techniques and algorithms aiming to obtain a representation of the spatial structure of a scene. In other terms, to obtain a measure of the distance of, ideally, each point of the scene. This can be done in several ways, including the following:

1. Shooting the scene using special Depth Cameras also known as Ranging Cameras or RGB-D Cameras which capture the depth of the scene in addition to the RGB values of each and every pixel.

2. Depth estimation from visual cues present in the captured scene such as variation of contrast, focus etc. with varying distance from the camera.

3. Depth estimation from using stereo vision techniques like comparing the left and right image of a stereo image pair and correlating them (calculation of stereo disparity). This is the area which will be focused on (primarily) in this thesis, which deals with design of a novel algorithm for depth extraction from stereo image pairs, and further, using the extracted depth information to perform (selective) de-focusing of users' non-interest image regions.

1.8 Parallel Framework

Computer algorithms can be classified as serial and parallel. Serial algorithms present a direct and straight-forward way to translate an algorithm into computer code. However, for algorithms with high computational complexity, as their size of input increases, their execution time also increases drastically. In such cases, we need to convert the serial (or sequential) algorithm to its parallel version, by segregating its mutually independent steps. These steps are run separately from each other, in parallel. Then, the results of these steps are combined to produce the final output. If such parallel algorithms are executed on suitable parallel hardware, like multi-core CPUs and GPGPUs, their execution time decreases significantly.

Thus, a parallel algorithm or concurrent algorithm, as opposed to a traditional sequential (or serial) algorithm, is an algorithm which can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result. Some algorithms are easy to divide up into pieces in this way. For example, splitting up the job of checking all of the numbers from one to a hundred thousand to see which are primes could be done by assigning a subset of the numbers to each available processor, and then putting the list of positive results back together. Most of the available algorithms to compute pi (π), on the other hand, cannot be easily split up into parallel portions. They require the results from a preceding step to effectively carry on with the next step. Such problems are called inherently serial problems. Iterative numerical methods, such as Newton's method or the three-body problem, are also algorithms which are inherently serial. Some problems are very difficult to parallelize, although they are recursive. One such example is the depth-first search of graphs. Parallel algorithms are valuable because of substantial improvements in multiprocessing systems and the rise of multi-core processors. The cost or complexity of serial algorithms is estimated in terms of the space (memory) and time (processor cycles) that they take. Parallel algorithms need to optimize one more resource, the communication between different processors. There are two ways parallel processors communicate, shared memory or message passing, and the details are as follows:

1. Shared Memory processing needs additional locking for the data, imposes the overhead of additional processor and bus cycles, and also serializes some portion of the algorithm.

2. Message Passing processing uses channels and message boxes but this communication adds transfer overhead on the bus, additional memory need for queues and message boxes and latency in the messages. Designs of parallel processors use

special buses like crossbar so that the communication overhead will be small but it is the parallel algorithm that decides the volume of the traffic.

Another problem with parallel algorithms is ensuring that they are suitably load balanced. For example, checking all numbers from one to a hundred thousand for primality is easy to split amongst processors; however, some processors will get more work to do than the others, which will sit idle until the loaded processors complete.

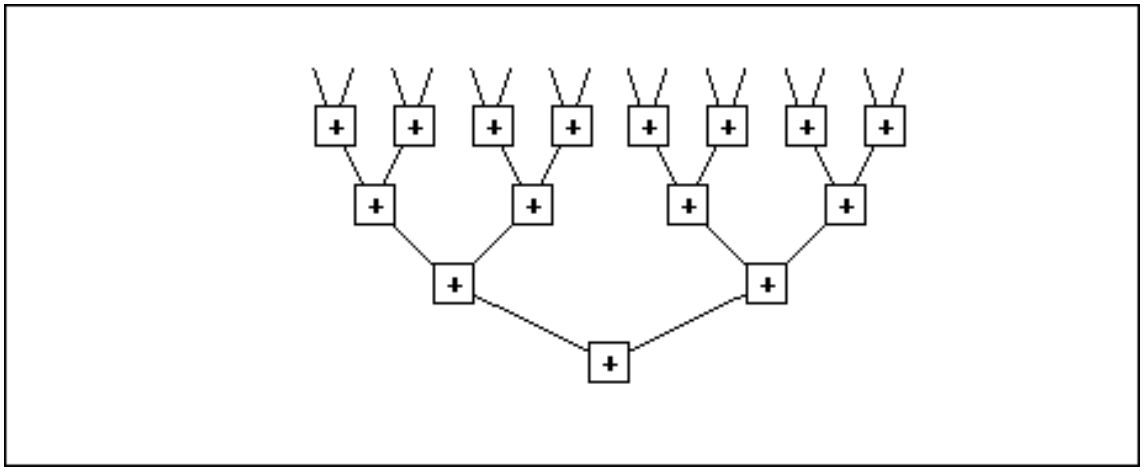


Figure 1.2: Summing of Eight Numbers in Parallel.

Fig. 1.2 shows the classic example of the parallel summation of a list of eight numbers. As is evident from the representation, four processors (or cores) can be allocated, each to sum two numbers in parallel in the first stage, two processors, each to sum two of the previous two sums in parallel in the second stage, and finally, one processor to calculate the final sum. Assuming that each summing operation takes one unit of time, the total time taken in the parallel approach is 3 units, whereas, that in the serial approach is 7. This ratio 7 / 3 is called the speed-up factor and it used to quantitatively describe the increase in speed due to parallelization.

For executing a parallel algorithm, in addition to parallel processing hardware, we need the appropriate software framework as well. The following sections list some existing parallel frameworks with short descriptions of each.

1.8.1 OpenMP

OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming on most processor architectures and operating systems. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior. OpenMP is managed by OpenMP Architecture Review Board (or OpenMP ARB), jointly defined by a group of major computer

hardware and software vendors. OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer. An application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and Message Passing Interface (MPI), or more transparently through the use of OpenMP extensions for non-shared memory systems.

1.8.2 MPI

Message Passing Interface (MPI) is a standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers. The standard defines the syntax and semantics of a core of library routines useful to a wide range of users writing portable message-passing programs in Fortran 77 or the C programming language. There are several well-tested and efficient implementations of MPI, including some that are free or in the public domain. These fostered the development of a parallel software industry, and there encouraged development of portable and scalable large-scale parallel applications.

1.8.3 Pthreads

POSIX Threads, usually referred to as Pthreads, is a POSIX standard for threads. The standard, POSIX.1c, Threads extensions (IEEE Std 1003.1c-1995), defines an API for creating and manipulating threads. Implementations of the API are available on many Unix-like POSIX-conformant operating systems, as well as Windows.

1.8.4 GPGPU

General-purpose computing on graphics processing units (General-purpose graphics processing unit, GPGPU) is the utilization of a graphics processing unit (GPU), which typically handles computation only for computer graphics, to perform computation in applications traditionally handled by the central processing unit (CPU). Additionally, the use of multiple graphics cards in one computer, or large numbers of graphics chips, further parallelizes the already parallel nature of graphics processing. OpenCL is the currently dominant open general-purpose GPU computing language. The dominant proprietary framework is Nvidia's CUDA.

2 LITERATURE SURVEY

A survey of existing literature was carried out to understand the work done by several researchers in the area of depth-based blurring of 3D videos and their parallel implementation using GPGPU. Though attempts at depth blurring of 2D images for drawing human attention to regions of interest have been made, yet it seems like attempts at parallelizing those processes, specially by using GPGPUs are rare. Further, the course of research taken over the last few decades related to depth map extraction methods have been investigated. An attempt has also been made to understand the various methods employed by several researchers to identify and eliminate incorrect depth estimates.

2.1 Related Work

2.1.1 Depth-based Selective Blurring

In [1], authors showed that algorithms for generating synthetic depth-of-field effects have been used to draw human attention to regions of interest [2], to reduce the bit-rate of regions of non-interest [3] and for providing realistic rendering in animated films and computer graphics. It is evident that the focus of this thesis include application of synthetic depth-of-field effects similar to [2].

In [4], the authors tried to overcome the limitations of present day cameras that have DOFs corresponding to a single slab that is perpendicular to the optical axis. In their work, they presented an imaging system that enables one to control the DOF in new and powerful ways. Their approach is to vary the position and/or orientation of the image detector during the integration time of a single photograph. In essence, their work represents the hardware implementation of achieving flexible depth of field effects by using a prototype camera using a micro-actuator to translate the detector along the optical axis during image integration.

In [1], the authors proposed a novel algorithm for enhancing an image or video frame with artificial depth of field effects. The authors claimed that this is similar to the blurring effects produced by real cameras when objects outside the depth of field set on the camera by the photographer are blurred from the sight of the viewer of the photograph. The proposed algorithm is a post-filtering approach which scores over similar earlier methods in that it does not suffer from intensity leakage from sharp boundaries. Intensity leakage occurs in *gather* [2] methods of approximating depth blurring. Gather methods take the local average of pixel values around the desired location, so intensity

from sharp source pixels is spread over surrounding background. The algorithm processes the input image based on the information encoded in the depth map, an in-focus depth (the depth of a chosen object of interest or of the nearest-to-camera object), and an overall blur level, i.e. the desired level of the depth of field effect, and the details are illustrated in the Fig. 2.1.

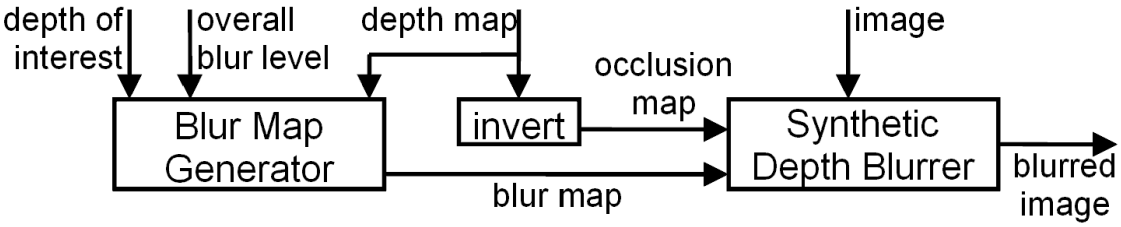


Figure 2.1: Block Diagram of Depth-based Selective Blurring.

The authors computed the blur map from the depth map so that the chosen depth has zero blur level and the other depths have gradually increasing blur away from this depth. However, it must to be borne in mind that a *single* depth of interest will not be of much use to us, as more focus is given in this thesis to the more general case of *multiple* depths. It is similar to [4], where the authors attempted to improve traditional cameras which restrict the photographer to only one range of depths, instead of many.

In [5], authors presented a novel selective blurring algorithm that mimics the optical distance blur effects that occur naturally in cameras and eyes. The proposed algorithm provides a realistic simulation of distance blurring, with the desirable properties of aiming to mimic occlusion effects occurring in natural blurring, and can handle any number of blurring and occlusion levels with the same order of computational complexity.

2.1.2 Depth Estimation and Depth Maps

As stated earlier, the depth of an image can be estimated from various visual cues present in the image itself like variation in focus, contrast etc. at different regions of the image. Depth map of the image can also be directly obtained by using special depth cameras which capture depth of each pixel in addition to their RGB values. Literature survey shows attempts at depth estimation using various such techniques which are discussed in this section.

Depth Estimation From 2D Videos

With the development of 3DTV, the conversion of existing 2D videos to 3D videos has become an important component of 3D content production. In general, the conversion process consists of depth map generation, which estimates the 3D geometry of the

scene, and rendering, which produces output stereo images. Thus, the first key step in 2D to 3D conversion is how to generate a dense depth map from a 2D video, which, technically contains no depth information.

In [6], the authors proposed a novel depth extraction method based on motion and geometric information for 2D to 3D conversion, which consists of two major depth extraction modules, the depth from motion and depth from geometrical perspective. The H.264 motion estimation result is utilized and cooperates with moving object detection to diminish block effect and generates a motion-based depth map. On the other hand, a geometry-based depth map is generated by edge detection and Hough transform. Finally, the motion-based depth map and the geometry-based depth map are integrated into one depth map by a depth fusion algorithm.

The quality of depth maps is crucial for 2D to 3D conversion, but high quality depth map generation methods are usually very time consuming. To overcome this challenge, in [7], the authors proposed an efficient semi-automatic depth map generation scheme based on limited user inputs and depth propagation. First, the original image is oversegmented. Then, the depth values of selected pixels and the approximate locations of T-junctions are specified by user inputs. The final depth map is obtained by depth propagation combining user inputs, color and edge information. The experimental results demonstrated that their scheme was satisfactory in terms of both accuracy and efficiency, and thus can be applied for high quality 2D to 3D video conversion. However, it must be borne in mind that this thesis is not aiming for any user intervention in the depth map extraction process of the proposed system.

In [8], the authors proposed a hybrid algorithm for 2D-to-3D conversion in 3D displays; it is a good way to solve the problem of traditional 2D video contents which need to generate 3D effects in 3D displays. They chose three depth cues for depth estimation: motion information, linear perspective, and texture characteristics. Moreover, they adopted a bilateral filter for depth map smoothing and noise removal. From the experimental results, execution time can be reduced by 25%-35% and the depth perception score is between 75 and 85. Thus, the human eye cannot sense the noticeable differences from the final 3D rendering. Furthermore, it is very suitable to apply their proposed hybrid algorithm to 2D-to-3D conversion in 3D displays.

In [9], the authors presented a novel method for generating the depth map of 2D image sequences from motion information. Different from the typical multi-view stereo method, this approach calculates the motion information directly from two sequential images which are captured by only a single moving camera. Classical constrained optical flow is known to be inaccurate for texture-less regions and complicated areas. The proposed approach uses mean shift segmentation algorithm and optical flow together to

compute the depth map. The motion-based depth map and the segmented map are integrated into one depth map using breadth-first search methods. Authors achieved results with better accuracy compared to the optical flow method.

Depth Estimation From 3D Videos

This thesis mainly focuses on depth estimation from 3D videos, particularly, based on stereo matching between left and right image frame pixels.

Stereo vision based disparity calculation is an important research problem in computer vision with applications such as robotic vision, 3D scene reconstruction, object detection and tracking etc. [10]. Here, the main challenge is to obtain an accurate depth information of a scene by comparing the pixels of left and right images of that scene. It is challenging because individual pixels contain only the colour and spatial information and represent the low level image features [11]. So, to effectively compare a stereo image pair, there is a need to develop a framework for identifying appropriate high level features and thereby comparinging these features across the stereo pair efficiently with reasonable accuracy and speed.

Disparity of a pixel varies inversely as the distance of a point in a scene (that the pixel represents) from the 3D camera. A disparity or depth map contains the mapping of each pixel of an image to its corresponding disparity. A cost function is used to quantify the similarity between pixels of the left and right images of a stereo image pair. Then, pixels disparities are estimated by relative displacement of the corresponding matching counterparts across the image pair.

For finding the matching pixels between the stereo image pair, there as two basic approaches: block-based and region-based; the strengths and weaknesses of these two basic approaches are summarized in Table 2.1 [12]. The key challenge in block-based methods is the determination of an optimal block size, and in region-based methods is the detection of gradually changing disparities. In estimating the stereo disparities using block-based methods, small block sizes produce sharp edges on the depth map but generates errors in the homogeneous regions since only a very limited local information is utilized. On the other hand, large block sizes provide good performance in homogeneous areas but yield very inaccurate disparity measurements along objects edges. This is because, pixels inside the same block may have varying disparities. For region-based methods, apart from the fact that they not only create a limited number of depth levels but also require accurate results of segmentation, which makes them inherently computationally intensive and time-consuming. This provides the motivation to propose a novel hybrid method by combining these block-based and region-based approaches and

thereby overcoming their individual limitations.

Table 2.1: Comparison of Basic Depth Estimation Approaches

	Block-based Method	***Region-based Method***
Approach	Depth estimation based on information contained in pixels and their surroundings.	Depth estimation based on optimal value of cost function for entire image regions of pixels with similar disparities.
Strength	High resolution depth maps.	Sharp edges on depth maps.
Weakness	Need to determine optimal block sizes for different images, or even different regions of same image for creating accurate depth maps.	Unsuitable for finding the disparities that are gradually changing, as all pixels constitute even a large region may share a constant disparity.

Another way to classify stereo depth estimation approaches is sparse vs. dense. In the sparse, feature-based approaches to stereo, only a subset of image pixels (i.e. vertical edge pixels) are matched, with an aim to meet real time processing requirements. However, sparse disparity maps may often contain insufficient data points for supporting object segmentation which is an important prerequisite to subsequent understanding of the scene. On the other hand, dense stereo matching can improve this bottom-up processing chain significantly, but it is computationally complex [13].

In this context, it is also helps to quote a few very important observations about stereo depth estimation approaches (in general) made by several authors, time and again, including the very recent one by Tippetts et al. [14], that the number of pixels that each image contains increases the number of computations required to match it with any number of possible matches, making the correspondence problem a computationally complex one that severely limits the speed at which one can obtain results. Most of the time, accuracy and speed are pitted against each other, making it more difficult to obtain both at the same time. In any given instance of the stereo vision problem, various questions arise; is one of these two attributes more desirable? How can the trade-off between the two be minimized? What options already exist, and how do they compare to each other?

The authors opined that target applications should drive the algorithm designers decision of striking a balance between speed and accuracy. They also observed that authors should not solely rely on advances in hardware technology to bring about radical

increments in processing speeds of stereo vision algorithms. Rather, they must actively engage in improving algorithms to run faster.

When categorizing stereo vision algorithms, one of the most obvious divisions in current literature is that of global versus local methods [14]. Local algorithms are statistical methods usually based on correlation. Global algorithms are based on explicit smoothness assumptions that are solved through various optimization techniques. Computational complexity of majority global algorithms makes them impractical for real-time systems. For global methods, smoothness assumptions are defined through an energy function and the optimization techniques minimize this function. For local methods, the correlation process finds matching pixels in left and right images of a stereo pair by aggregating costs [e.g. sum of absolute differences (SAD), sum of squared differences (SSD), normalized cross correlation (NCC)] within a region or block.

For several years, the most accurate disparity maps were produced by global algorithms [14]. Currently, eight of the ten most accurate stereo vision algorithms ranked by Middlebury evaluation criterion are global energy minimization algorithms. Recently, however, many local algorithms have been developed that are competitive with respect to accuracy. As mentioned, there is always a trade-off between accuracy and speed for stereo vision algorithms and these accurate local algorithms are no exception.

This has provided the motivation to design the proposed local algorithm which can strike a good balance between accuracy of results and speed of computation.

Most existing stereo disparity algorithms use one of the two types of measures of pixel similarity between left and right image: pixel-to-pixel or window-based. Pixel-to-pixel algorithms find the pixel similarity measure solely on individual pixel values, e.g. Absolute Difference (AD) or Squared Difference (SD). On the other hand, window-based methods aggregate the pixel matching cost over a support region around a pixel of interest, e.g. SAD or SSD [13]. A study on comparison of cost functions commonly used in recent methods of stereo disparity determination [15] has come to the conclusion that the performance of a matching cost function depends on the stereo method that uses it. The SAD similarity measure can be computed efficiently by exploiting the fact that neighbouring windows overlap. For neighbouring windows with the same disparity, the overlapping pixels will contain equal absolute difference (AD) values. Therefore, a new SAD can be computed out of an old one by subtracting the values, which are only parts of the old window, and adding the values, that are only parts of the new window [13].

This advantage of the SAD method, coupled with the fact that it yields better estimates than AD due to its support region, has provided the motivation to use it in the proposed method. The shape of the SAD window is often rectangular. Fewer mismatches will occur with large windows, especially on texture-less regions. However, it

is well known that large windows lead to erroneous disparity estimates at the edges of objects [13]. To overcome these limitations, several cost aggregation strategies based on characteristics of matching windows have been proposed in recent years. These can be classified into cost aggregation based on rectangular windows-based and unconstrained shapes-based strategies; lastly, by assigning different and variable weights to pixels falling in the neighbourhood of two points on the reference and target images for which stereo correspondence is being evaluated. Rectangular window-based methods can be further classified into variable window-size or offset-based, multiple-windows-based, and differential weights of window points-based approaches [16].

The multiple window approach leads to better estimates in non-occluded areas of the input stereo image. However, it is far more expensive compared to single window approach. Further, multiple window approach increases the computation time by more than 50% but with slight improvement in overall accuracy [13]. So, this thesis considers single window approach in proposed depth estimation algorithm.

Moreover, in the proposed approach, SAD is used to determine disparities of only points which lie on segment boundaries, which often coincides with edges of objects, where there is bound to be some amount of variation in intensity. Hence, fixed-size square windows of size 9 x 9 pixels are chosen.

To determine the best matching pixel for a given pixel, the most obvious means involves selecting the point in the other image within a certain disparity range that has the best similarity value, and this has been used in many stereo matching methods. This strategy is called Winner Takes it All (WTA). However, a stereo algorithm based only on WTA does not consider parts of stereo images that are only visible in one of the two images, called occlusions. Moreover, image regions having little or repetitive texture yield similar matching costs. In these areas, WTA is error-prone as there isnt a clear optimum correspondence [13].

It may be noted that the proposed stereo depth *estimation* approach will not focus on addressing occlusion handling as of now. However, the proposed stereo depth estimation *error detection* method will be *generic* in nature, and will try to address *all* types of estimation errors, including those arising out of occlusions.

The smoothness constraint for stereo states that disparity does not change much on object surfaces. This concept is often used to improve disparity estimates. If a correct disparity has been found then it can be used to constrain the range of possible disparities of other points on the same surface [13]. This concept has been incorporated in the fill step of the proposed depth map reconstruction phase and thereby reducing wrong estimates on areas with little or repetitive texture.

In scan-line optimization (SO) schemes, the smoothness constraint for stereo is im-

plemented as a global cost function, which adds penalties to the values in the disparity search interval of each stereo point. Errors can then be suppressed by penalizing large jumps in disparity between scan-line points. The main drawback of SO is its sensitivity to noise [13]. Hence, the SO scheme was not considered.

In dynamic programming (DP) approaches, the problem of finding the correct disparities on a scan line is regarded as a search problem. The matching costs of all points on a scan-line describe the disparity search space. Finding the correct disparities is akin to finding the path in this space which takes the shortest route through the cost values. Special rules for transversing the search space can be added to handle occlusions [13].

An important step in a stereo algorithm is searching for the disparities with the lowest SAD values. For a particular point on the left image line, the possible matches with points on the right image line constitute its search space. Finding the correct match by finding the match with the lowest SAD value is computationally expensive, since the whole disparity interval has to be searched [13]. Hence, some methods like SO and DP try to restrict the search space for faster computation.

The edges of nearby objects do have high disparity jumps between foreground and background. If algorithms such as SO and the DP approaches do not find these jumps, then the wrong disparity is used to define a constraint for a part of the search space. In the disparity images this error is visible in terms of the typical *streaking* error. These misses can be induced if an object is near, or when image noise is present. Algorithms such as SAD used for estimating disparity per point do not suffer from these types of limitations [13].

However, irrespective of the types of algorithms, the scene geometry is known to have a clear influence on the performance of stereo depth estimation. Specially, algorithms like DP which use search space restrictions are more affected [13]. Hence, in the proposed method, there is no imposition of any search space restrictions while determining the disparities of pixels lying on segment boundaries using SAD.

Segmentation has been employed in many recent stereo matching algorithms which considered segments instead of pixels, as the processing elements. But, their segmentation processes mostly dealt with hierarchical approaches and also took into account features of colour, size and shape while processing the input stereo image pairs, viz. object-oriented segmentation methods based on bottom-up region merging [11], colour-based Mean-shift segmentation (refer to Table 2.2) and the use of K-Means clustering to over-segment the reference image [17]. But in all these segmentation-based stereo depth estimation approaches there is no attempt to use, primarily segmentation results of pixel grey level values to identify groups of pixels that could produce reliable disparities when matched. An attempt has been made to explore this possibility in the

proposed work, thus not only making it computationally efficient (compared to time-consuming segmentation methods like Mean-shift etc.), but also simplifying its entire depth estimation process.

However, as only grey level values of image pixels are considered for segmenting the reference image, and no spatial or colour information whatsoever, two steps are included to refine the segment boundaries, using morphological filtering as well as connected components analysis. Since both of these are established techniques for image processing, their efficient implementations are readily available. Hence, the proposed method utilizes the above mentioned techniques.

Morphological filters were used for partitioning a low depth-of-field image into focused object of interest and blurred background [18]; but, the proposed method uses morphological filters for an entirely different task altogether, viz. refining segment boundaries whose disparities are determined later to reconstruct the full disparity map of the scene based on the segment boundaries depth estimates.

Connected components analysis based methods have aided image segmentation for the purpose of text detection [19-21], brain tissue analysis [22, 23] and colour image segmentation using genetic algorithms [24], whereas in the proposed work, connected components analysis is used to refine the boundary map by removing small, isolated connected components (artefacts) which may yield incorrect depth estimates. The proposed approach may seem similar to [22], where the authors first roughly distinguished the region of non-brain tissue through connected component labelling, and then tried to refine those edges using the morphological operations, dilation and erosion. However, a slightly closer look at the proposed method would reveal that connected components analysis is performed after applying morphological filters and thus the ordering of these two operations is entirely reversed.

Several depth estimation techniques [25-28] have been developed in recent years to overcome the limitations of two basic approaches of stereo disparity estimation, viz. block-based and region-based. Their key features are summarized in Table 2.2.

Refinement of Erroneous Depth Estimates

Stereo depth estimation algorithms based on successive refinement of initial disparity estimates have been widely used for multi-view video synthesis [29], DIBR view synthesis [30], 2D-to-3D conversion [31] and refinement of raw depth information obtained from (low-quality) depth sensors [32, 33]. Algorithms like [29] represent an entire class of depth refinement approaches which first classify the pixel-wise initial depth map into two categories, reliable and unreliable, followed by depth refinement

Table 2.2: Comparison of Recent Depth Estimation Methods

Work	***Advantages***	***Limitations***
Choi [25]	Bi-directional consistency check of disparity blocks can improve disparity estimates.	Comparison with similar algorithms using a standard dataset has not been done.
Zhu and Yu [26]	Self-adaptive block matching can increase the efficiency of searching, and reduce errors in matching and block-shape.	Presence of noise in the original left or right images can result in selection of incorrect size for block-matching (8×8 or 16×16).
Wang and Zheng [27]	Cooperative optimization can check or fix disparity errors.	Uses the time-consuming mean-shift algorithm for segmentation.
Lu and Du [28]	Harris corner point extraction and feature-matching yields better corresponding points than traditional methods.	Both left and right images are scanned for corner extraction and matching in the initial steps, making them time-consuming.

for pixels with unreliable depth values. As part of this classification, the authors use the (state-of-the-art) left-right consistency check (LRC) method. In the proposed work, a novel classification technique is developed for such initial disparity estimates, and compare it with LRC. For these (comparison) experiments, a simple, generic, block-based stereo matching method is used to obtain the (said) initial disparity estimates, as this represents a typical approach taken by many depth map refinement based stereo depth estimation approaches, and the details are provided in the next section.

Further, depth map refinement techniques can be classified as pre-filtering and reliability based approaches. The former approach performs pre-processing using adaptive filters, asymmetric Gaussian filters, etc. to smooth the depth map. The latter approach uses reliable warping information from multiple to fill holes in depth maps [30].

Another large class of depth map refinement algorithms are based on two primary objectives: elimination of unnecessary textures and preservation of object boundaries on the depth map [31]. This is carried out in such a manner that transitions of depth values within any object are smoothened and the object boundaries on depth map are aligned to those of input image. To achieve the first objective, traditionally, the *Computed Image Depth* (CID) method was used. This method produces a rough depth map

from local image characteristics (statistics) such as contrast and sharpness. However it was unable to meet the second objective since it ends up with blurring object boundaries, resulting in reduced (human) depth perception. Hence, it was later superseded by bilateral filter based approaches, which keep the said object boundaries intact.

Similar trends can be seen in the case of recent depth refinement algorithms dealing with raw depth maps obtained from (low-quality) sensors like Microsoft Kinect [32]. These methods try filling in the missing (holes) or poor (pseudo-holes) disparity estimates in the initial depth map (hole-filling), as well as correcting the misalignment of object boundaries and edges between input image and the depth map. Authors in [33], considered multi-resolution based approaches with probabilistic Bayesian models using joint guided filtering and belief propagation to de-noise the depth map while filling the regions with missing disparities.

In recent years, entropy has been used to define novel aesthetic measures for 3D stereoscopic imaging [34], particularly to characterize the quality of any given stereo camera setting. But the use of entropy for quantification of reliability, followed by classification of initial disparity estimates as correct or incorrect, is a relatively unexplored area, and hence has been taken up in the proposed work.

2.1.3 Depth Estimation Using Kinect Depth Camera

In [35], authors analyzed Kinect as a 3D measuring device, experimentally investigate depth measurement resolution and error properties and make a quantitative comparison of Kinect accuracy with stereo reconstruction from SLR cameras and a 3D-TOF camera. They proposed Kinect geometrical model and its calibration procedure providing an accurate calibration of Kinect 3D measurement and Kinect cameras. They demonstrated the functionality of Kinect calibration by integrating it into an SfM pipeline where 3D measurements from a moving Kinect are transformed into a common coordinate system by computing relative poses from matches in color camera.

In [36], authors presented an analysis and experimental study of the Kinect's depth sensor with reference to resolution, quantization error and random distribution of depth data. The effects of color and reflectance characteristics of the object are also analyzed. The study examined two versions of Kinect sensors, one dedicated to operate with the Xbox 360 video game console and the more recent Microsoft Kinect for Windows.

While most of the work around the Kinect is on how to take full advantage of its capabilities, not many studies have been carried out on the limitations of this device and how to overcome them by enhancing the precision of its measurements. In [37], the authors reviewed and analyzed current work in this area, and presented and evaluated a temporal de-noising algorithm to reduce the instability of the depth measurements

provided by the Kinect over different distances.

In [38], authors presented a comprehensive review of recent Kinect-based computer vision algorithms and applications classified according to the type of vision problems that can be addressed or enhanced by means of the Kinect sensor including preprocessing, object tracking and recognition, human activity analysis, hand gesture analysis, and indoor 3-D mapping. For method category, they outlined main algorithmic contributions and summarized their pros/cons compared to their RGB counterparts. Finally, they listed challenges and future research trends.

2.1.4 Parallelizing the Depth Map Extraction Process

As described earlier, attempts at depth map extraction from stereo image pairs using GPU acceleration are rare. One such effort was made in [39], where the authors proposed a stereo vision system on GPU based on a hardware-aware algorithm design approach. The system consisted of new algorithms and code optimization techniques emphasizing on keeping the highly data parallel structure in the algorithm design process such that the algorithms can be effectively mapped to massively data parallel platforms. Research has shown that, in the context of parallel extraction of depth maps from stereo image pairs, there exists a trade-off between speed and accuracy. So, in the stated work, authors tried to jointly optimize accuracy speed trade-off by designing high accuracy stereo algorithms that can be effectively mapped to such platform.

On the other hand, several attempts have been made to efficiently extract depth maps from stereo image pairs without using multiple processors cores or GPUs. One such attempt [40] used Dynamic Time Warp (DTW) algorithm to solve the problem of depth map extraction using the dynamic programming approach which is highly efficient in terms of time complexity. The DTW algorithm efficiently solves the problem of stereo matching to match the right and left raster profiles to the accuracy of one pixel distance. To improve the resolution of distance, the authors proposed a method of subpixel disparity estimation, and implemented it to generate a dense disparity map which can resolve one tenth of a pixel distance.

The application of parallel computing architectures to video processing in general have been focused in [41]. Authors demonstrated different optimization strategies in detail using the 3D convolution problem as an example, and further showed, how they affect performance on both many-core CPUs and symmetric multiprocessor CPUs. Applying these strategies to case studies from three video processing domains, they tried to bring out some trends. The highly uniform, abundant parallelism in many video processing kernels means that they are well suited to a simple, massively parallel task-based model such as CUDA. As a result, ten times or greater performances increases

are often seen while running on many-core hardware. Some kernels, however, push the limits of CUDA, because their memory accesses cannot be shaped into regular, vectorizable patterns or because they cannot be efficiently decomposed into small independent tasks. Such kernels may achieve a modest speedup, but they are probably better suited to a more flexible parallel programming model. However, nowadays a wider array of options is available when going for parallelization, such as general purpose computing on multi-core GPUs (GPGPUs).

2.2 Outcome of Literature Survey

Based on the survey of existing literature, the following issues and challenges are identified for further research:

1. Depth data provided by the Kinect sensor presents several noise-related problems like distance-dependent depth maps, spatial noise, and temporal random fluctuations.

2. Most depth-estimation approaches require a trade-off between speed and accuracy, since depth estimation involves a lot of computational complexity.

3. Attempts at parallelizing the depth estimation process from stereo image pairs, especially using a combination of multi-core processor architectures and GPGPUs are rare.

4. Researchers have tried to overcome the *single-DOF* limitation of traditional cameras using hardware implementations that allow multiple (flexible) DOFs [4], but no such effort has been seemingly made using only software for 3D videos.

2.3 Problem Statement

To design and develop a system for selective blurring of 3D video scenes using a novel stereo depth estimation algorithm and further reducing its computational complexity using a CPU-GPU accelerated framework.

2.4 Research Objectives

1. To design a novel accelerated depth map extractor for 3D images to run on CPU and GPU.

2. To develop a system for selective blurring of scenes (frames) from 3D videos to simulate the effect of flexible depth-of-field based on supplied depth maps.

3. To compare three different techniques for depth estimation: firstly, using depth cameras, secondly, using certain visual cues from an image captured using a normal 2D camera, and lastly, from stereo image pairs using a 3D camera using stereo correspondence techniques.

4. To develop a novel method of quantifying the reliability of depth estimates in depth maps.

3 PROPOSED METHODOLOGY

3.1 Depth-based Selective Blurring using both CPU and GPU

3.1.1 Algorithm Design Perspective

It is observed from the literature that, several methods have been proposed to address the stereo matching problem, but the trade-off between speed and accuracy is still an open research area. Hence, throughout algorithm design process, an attempt has been made to give due consideration to both these factors mentioned, viz. accuracy and speed. As an example, an attempt has been made to combine the speed of the sparse, feature-based approaches to stereo with the accuracy of the dense stereo vision methods.

This thesis proposes a novel hybrid method of disparity estimation by segmenting pixels' lightness values by a fast histogram-based K-Means implementation and subsequent refinement of segment boundaries using morphological filters and connected components analysis. We have already provided several motivations behind the design choices in the proposed method in the *Related Work* section.

The core idea behind the proposed method is to use a scalable, block-based approach to estimate the disparities of only pixels lying on the refined segment boundaries, and get reliable disparity estimates in less number of computations. So, the proposed method is scalable to high resolution stereo image pairs. Then, the proposed disparity map reconstruction method is used for estimating the disparities of pixels lying inside segment boundaries. Finally, an application of the proposed algorithm for depth-based selective blurring of stereo images is considered. It uses a Gaussian kernel to blur image regions corresponding to (supplied) depth range(s) of user's non-interest. Thus, proposed methodology for the selective blurring phase is similar to that of [30]; but the proposed method is not at all concerned with the occlusion handling, however the proposed scheme can handle multiple, and discrete depth ranges of users' interest.

Further, the proposed method utilizes two-dimensional intensity based segmentation of the left image whereas Zhen Zhang et al. [12] used a one-dimensional colour-based segmentation process of individual rows of pixels. Similarly, the proposed method uses only L values of pixels, making the process simple and fast; on the other hand the method of [11] is based on object-based segmentation using colour, spatial and shape information, which potentially add to the computational time.

Lastly, the reduction in running time of the proposed algorithm is demonstrated by running some of its mutually independent operations in parallel on the multiple cores

of CPUs and GPUs. Experimentation is also performed by combining the power of the CPU and the GPU to achieve even higher degrees of parallelism.

Following are Key Contributions of the Proposed Work:

- Key Contribution **1**: This is the first work on a hybrid method for stereo depth estimation using disparity information obtained by segmentation of only lightness values of left image pixels, unlike other methods using colour, texture and shape characteristics.
- Key Contribution **2**: This is the first work dealing with morphological filters and connected component analysis to get sparse, but accurate depth estimates for subsequent reconstruction of dense depth maps.
- Key Contribution **3**: This is the first work which proposes an accelerated framework for reducing the running time of a novel algorithm for stereo depth estimation, using Java Thread Pool (CPU) and APARAPI (GPU), and for further combining them, in order to achieve even greater parallelism.

3.1.2 Proposed Stereo Depth Extraction Algorithm

The proposed algorithm obtains the depth map as per the details given in Fig. 3.1.

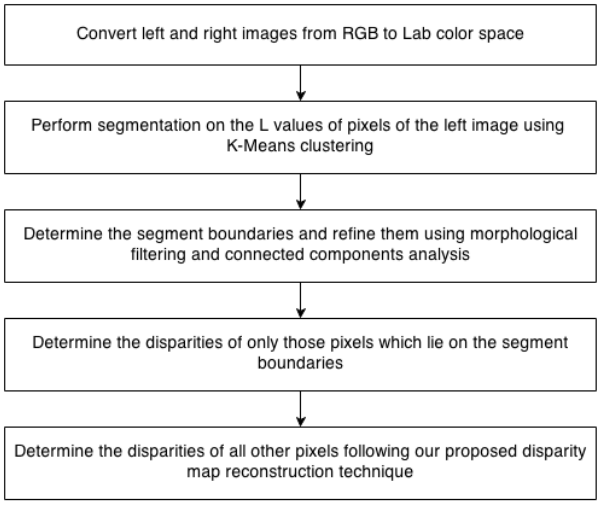


Figure 3.1: Flowchart of Proposed Depth Extraction Algorithm.

Next, the steps of the flowchart of the proposed algorithm are explained:

Colour Space Conversion

Human eyes are more sensitive to changes in brightness than in colour. The L component of the Lab colour space closely matches the human perception of lightness. Hence,

by applying pertinent transformations discussed in [42], left and right images are converted from RGB to Lab colour space and only L values of its pixels are retained.

K-means Clustering

K-means algorithm is based on a basic assumption, that is, a cluster of objects can be modeled by the mean of the cluster members. K-means tries to find the best partitioning of data into k clusters that minimizes the cost function, which is the distance of each point to its closest *centroid*. The cost function is usually the sum of squared error (SSE)

$$SSE = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - c_i\|^2 \tag{3.1}$$

where C_i is the set of members in the i^{th} cluster, and c_i is the centroid of C_i. The K-means algorithm can be described by the following steps:

1. Initialize centers $C = \{c_1, ..., c_k\}$.
2. For each $i \in \{1, ..., k\}$, let μ_i = center of mass of all instances closer to c_i than other centroids.
3. For each $i \in \{1, ..., k\}$, let $c_i = \mu_i$
4. Repeat Steps 2 and 3 until $\hat{\mu}$ converges.

In the computer science community, this iterative local search is also known as *Lloyd's algorithm*. It usually converges after a few iterations. Due to its speed, ease of implementation, and popularity of K-means, many extensions have been proposed. The most famous variation of K-means is *k-median* algorithm, which represents each cluster by the median of its members.

K-means requires three parameters to be specified by the user: Number of clusters k, distance metric, and cluster initialization strategy. These parameters should be ideally specified by expert according to domain knowledge. However, when this knowledge is not available, optimum parameter values should be estimated, or default settings are used instead. Much research have been done on making K-means efficient and effective, by defining some heuristics for each of the above parameters.

- *Number of clusters* k: Many heuristics are available for estimating number of clusters, including gap statistics, Dark Block Extraction (DBE), iK-means, X-means, and PG-means.

- *Distance metric:* Euclidean distance is the default distance metric for K-means clustering and results in spherical clusters in the feature space. For datasets in which Euclidean distance have poor clustering results, statistics community suggests many alternative metrics that find arbitrary cluster shapes.

- *Initialization:* K-means objective function (SSE) is not convex, and hence the optimization procedure can get stuck in local optima. This makes K-means very sensitive to the choice of initial cluster centroids. In original K-means, the initial centroids are chosen randomly. However, various alternatives have been proposed to reduce the sensitivity of K-means to initialization. A simple but expensive solution is to run K-means multiple times with different initial seeds. In this case, the best solution among different clusterings is chosen. Another well-known algorithm for seed selection is K-means++, which selects the centroids through a biased stochastic selection. In K-means++, the first centroid c_1 is chosen at random among data points. For the next $k-1$ centers, probability of each data point x_j for being chosen as c_i (i^{th} center) is $\frac{D^2(x_j,C)}{\sum_{m=1}^{n} D^2(x_m,C)}$, where $D(x,C)$ denotes the distance between a point x and the nearest centroid in C. K-means++ usually converges faster than K-means and its SSE is comparatively smaller.

Regarding computational complexity, finding the optimal solution to the K-means clustering problem for observations in d dimensions is:

- NP-hard in general Euclidean space d even for 2 clusters

- NP-hard for a general number of clusters k even in the plane

- If k and d (the dimension) are fixed, the problem can be exactly solved in time $O(n^{dk+1} \log n)$, where n is the number of entities to be clustered

Thus, a variety of heuristic algorithms such as Lloyd's algorithm are generally used. The running time of Lloyd's algorithm is given as $O(nkdi)$, where n is number of d-dimensional vectors, k the number of clusters and i the number of iterations needed until convergence. On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyd's algorithm is therefore often considered to be of "linear" complexity in practice.

Histogram-based K-Means Clustering for Segmentation

In our proposed histogram-based K-means clustering for image segmentation, L values of left image pixels are segmented using a fast histogram-based implementation of K-

Means clustering algorithm. A histogram of the L values is built and used instead of the actual pixel values for clustering. Thus, the run-time of clustering is significantly reduced, as clustering is performed on a small, fixed number of bins comprising the histogram, because the L values (representing the histogram bins) are bound to fall within the range of [0, 255]. Since there exists a one-to-one correspondence between each pixel and the bin to which it has been mapped, the cluster to which the pixel has been assigned can be easily identified as the cluster to which its bin has been assigned. Further details w.r.t how the proposed histogram-based K-means clustering reduces the time complexity of the image segmentation process can be found in Section 3.2: Sequential Complexity of Depth Extraction Algorithm.

Segment Boundary Detection and Refinement

Segment boundary detection is achieved by comparing the cluster assignment of each pixel with that of its 8-connected pixels (i.e. its Moore neighbourhood). If any of them is found to differ, the pixel is marked as 1 (falling on a segment boundary), else as 0 (not on a segment boundary). So, this step creates the boundary map after segmentation.

But this approach also falsely identifies many pixels as belonging to segment boundaries due to limitations imposed by clustering accuracy, as the image is segmented based on only the pixels' lightness (L) values and not on their colour components (a, b) or spatial locations (x, y) in the image. So, two morphological filters are applied to refine the boundary map by removing such noisy pixels, in the following order.

1. Fill: Fills isolated interior pixels such as the centre pixel in:

   1 1 1
   1 0 1
   1 1 1

2. Remove: Removes interior pixels, i.e., sets a pixel to 0 if all its 4-connected neighbours are 1, thus leaving only the boundary pixels on.

Further, connected components analysis is used to remove small artifacts in the boundary map due to segmentation errors, by ordering connected components present in the boundary map by the number of pixels constituting each connected component to remove the smallest connected components which contribute about 4% of the total number of boundary pixels.

Disparity Measurement of Boundary Pixels

It is assumed that the left and right cameras are calibrated and the left and right images share the same image plane. So, the correspondence of each pixel can only be in the horizontal direction. For e.g., the correspondent point of any pixel on the left image can only appear on the same row of the right image.

SAD (Sum of Absolute Differences) [10] cost function is used to determine only the disparities of boundary pixels, using the L values of the left and right image pixels. It should also be noted here, that by boundary pixels, it is also (implicitly) referring to the pixels of the left image that map to the left and right borders of the disparity map.

Disparity Map Reconstruction from Boundaries

The proposed disparity map reconstruction algorithm scans through each row of the partially computed disparity map and computes the remaining disparities based on disparities that have already been calculated. It operates in two stages:

1. Disparity Propagation (Fill Stage): In the first stage, the disparity map is scanned row-wise, left to rightwhenever consecutive two boundary pixels with equal disparity values are encountered , the intermediate pixels are fill-ed with that disparity value. This reflects the assumption that the pair of points in consideration actually belong to the same object in the original image. So, all their intermediate pixels also belong to that same object, and hence, should have similar disparity values. The process is explained using the pseudo-code given in **Algorithm 1**; disp_map is the matrix containing the disparities of boundary pixels and disparity(cell) refers to the disparity value of a boundary pixel of disp_map.

Algorithm 1 Disparity Propagation along Scan Lines

```
for each row of disp_map, starting from its top do
    for each cell of this row, starting from its left do
        if this cell is a pixel on a segment boundary, then
            if disparity(cell) == disparity(previous boundary pixel), then
                disparity of all intermediate pixels ← disparity(cell)
            end
        end
    end
end
```

2. Estimation from Known Disparities (Peek Stage): In the second stage, for all pixels whose disparities have not yet been determined, their disparities are estimated by peek-ing at disparity values of their two nearest pixels for which the disparities

have been already determined. These two nearest pixels are searched along the same column as the pixel in question and the details are given in **Algorithm 2**.

Algorithm 2 Disparity Estimation from Known Disparities

```
for each row of disp_map, starting from its top do
    for each cell of this row, starting from its left do
        if disparity of this cell has not been already determined, then
            disp_n ← {disparities of two nearest pixels on the same column}
            disp_r ← statistical range of disp_n
            if disp_r > disparity_threshold, then
                disparity of this cell ← smaller of the two values in disp_n
            else
                disparity of this cell ← mean of the two values in disp_n
            end
        end
    end
end
```

3.1.3 Depth-based Selective Blurring

The depth map output by the aforementioned final step is fed to the blur map generator, along with the depth ranges of users interest and an overall blur level. It must be noted here that, the user is solely responsible for selecting the proper depth range(s) which comprise region(s) of his (her) interest in the scene. This is analogous to setting a proper DOF while shooting a scene using a SLR camera. Thus, the blur map (determines which image pixels will be blurred) is fed to a depth-based blurring module. It synthesises a new scene, where all depth ranges of users' non-interest are blurred using Gaussian function $h_g(n_1, n_2)$ and Blurring kernel $h(n_1, n_2)$ as defined by (3.2) and (3.3) respectively, where n_1 and n_2 are dimensions of the blurring kernel, σ^2 is the variance for building the Gaussian kernel and the overall blur level is determined by σ.

$$h_g(n_1, n_2) = e^{-\frac{\left(n_1^2 + n_2^2\right)}{2\sigma^2}} \tag{3.2}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g} \tag{3.3}$$

Fig. 3.2 shows the overall process of this depth-based selective blurring.

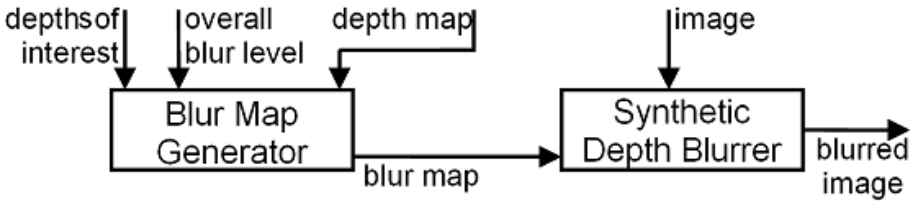


Figure 3.2: Proposed Depth-based Blurring Block Diagram.

3.2 Sequential Complexity of Depth Extraction Algorithm

Here, the step-by-step *time complexity* analysis of the proposed algorithm is presented, and thereby identifying computationally intensive steps and the degrees of interdependency between the operations of each step. Next, the steps eligible for data parallelization are converted to data-parallel workloads in JTP and APARAPI.

The first step involves colour space conversion, in which predefined transformations [42] are applied to the (R, G, B) triplet of each pixel in both the left and right image to extract the L values from them. Thus, the time complexity of this step is O($c_1 N_1$) in which N_1 represents the number of pixels of each image and c_1 the number of operations required to perform each transformation. Further, as transformation applied to each pixel is independent of others, this step lends itself to the most obvious data parallelization.

In the next step, K-Means clustering is performed on the L values obtained in the previous step. The time-complexity derivation for the naive K-Means implementation is first shown. Then, it is shown, how the fast implementation has been able to reduce the complexity. Let t_{dist} be the time to calculate the distance between any two feature vectors on which the K-Means algorithm is run; then, each iteration of K-Means has the complexity O(KN_2t_{dist}), where K denotes the number of cluster centres and N_2 the number of feature vectors (L values, in case of the proposed method). For finding t_{dist}, only one computational step is needed, that of finding the absolute difference between two integers falling within the range [0, 255] (the range of possible values for L). More importantly, since clustering is performed on the bins of the histogram of the left image, and there can be a maximum of 256 bins (representing the range [0, 255]), this drastically reduces the time complexity compared to the naive implementation which would have run each iteration on all N_1 L values (one value for each of the N_1 pixels of the left image).

The next three steps, namely, segment boundary detection and refinement using morphological filters, *fill* and *remove*, are performed in number of comparisons linear in the number of pixels of the left image, i.e., N_1. Hence, they have time complexities of

O(c_2N_1), O(c_3N_1) and O(c_4N_1) respectively with the c_i terms representing the number of operations required for each step. It is also evident, that since the operations for each step are independent for each pixel, they are ideal candidates for data parallelization leading to accelerated processing.

The next step, viz. finding connected components has three basic operations:

1. Search for the next unlabelled pixel, p.
2. Use a flood-fill algorithm to label all the pixels in the connected component containing p.
3. Repeat steps 1 and 2 until all the pixels are labelled.

It is clear that searching for new unlabelled pixels until all image pixels have been labelled requires a number of operations which is linear in the total number of image pixels. Further, a recursive flood-fill algorithm works as outlined below:

1. If the pixel to be labelled is unlabelled, then label it, else stop.
2. Reclusively flood-fill each unlabelled 8-connected pixel of the above pixel.

Since there are *if* clauses, flood-fill only considers unlabelled pixels, which have a static number, say, n. Therefore worst-case complexity becomes O(n). For ordering N_c connected components based on their constituent number of pixels, merge-sort can be used to finish the operation in O(N_c log(N_c)) comparisons.

Next, to find the disparity of each boundary pixel using the SAD cost function for a (constant) window size *w* and a (constant) maximum disparity *d*, O(w^2d) operations are required. Now, for N_b boundary pixels, it becomes O(N_bw^2d).

Lastly, both the *fill* and *peek* stages of the depth map reconstruction phase of the proposed algorithm are primarily based on a (constant number of) computations on the disparities of boundary pixels with time complexities of O(N_b) each, and these can also be converted into data parallel workloads due to its independent nature.

Space complexity of the proposed algorithm can be computed as follows: the pixels' L values and the complete depth map need O(N_1) memory; an auxiliary storage of O(N_b) size is also needed for holding the intermediate results of segment boundary detection and refinement and the disparities of pixels lying on refined segment boundaries. For K-Means clustering, an array of size O(256) is required for storing the count of image pixels mapped to each histogram bin for the present iteration (each histogram bin represents each grey-intensity present in the image).

3.3 Parallel Implementation of Depth Extraction Algorithm

Parallelization of some of the computationally intensive steps of the proposed algorithm are carried out, whose constituent operations are mutually independent. By executing these independent steps on multiple cores of the CPU and GPU, it can reduce the time complexity. Java Thread Pool (JTP) is used for CPU-parallelization and APARAPI (an API which allows suitable data parallel Java code to be executed on GPU via OpenCL), for GPU-parallelization [43]. Experimentation is also done by combining the two methods to achieve even greater performance.

The independent execution units in Java Thread Pool are the *worker threads* and in APARAPI are the *kernels*. Thus, the parts of the developed code which are intended to run in parallel are put in the respective execution units. Moreover, combination of CPU and GPU parallelization leads to sharing the workload between them, so it is required to decide on how many execution units are to be run simultaneously on the CPU and the GPU, so as to achieve optimal balancing of workload between them.

3.4 Parallel Complexity Analysis of Depth Extraction Algorithm

Assuming the availability of p parallel processing units, for four steps of the proposed algorithm, viz. colour space conversion and segment boundary detection and refinement using the morphological filters *fill* and *remove*, each of them can be completed in O($c_i N_1$/p) time. Similarly, both the *fill* and *peek* steps of the depth map reconstruction phase can also be completed in O(N_b/p) time each.

3.5 Proposed Depth Estimation Error Detection Method

3.5.1 Algorithm Design Perspective

Following are key contributions of this thesis w.r.t disparity estimation error detection:

- Key Contribution **1**: This is the first work which proposes a novel entropy-based approach for detection of errors in initial disparity estimates obtained in preliminary stages of dense stereo correspondence algorithms.
- Key Contribution **2**: This thesis is the first to propose a novel confidence measure for initial disparity estimates based on the difference between the entropy of a point on input image and its corresponding point on the depth map.
- Key Contribution **3**: This is the first work which proposes a novel quality metric to quantify error detection capabilities of algorithms on the initial disparity estimates obtained in dense stereo correspondence methods.

3.5.2 Algorithm Design Methodology

Fig. 3.3 shows the flowchart of the overall methodology for the proposed approach w.r.t stereo disparity estimation error detection in initial disparity estimates generated by dense stereo correspondence algorithms. The proposed method uses two entropy filters, one for the left image of stereo image pair, and another for its (initial) depth map, both of which are taken as inputs. The detailed steps are explained subsequently.

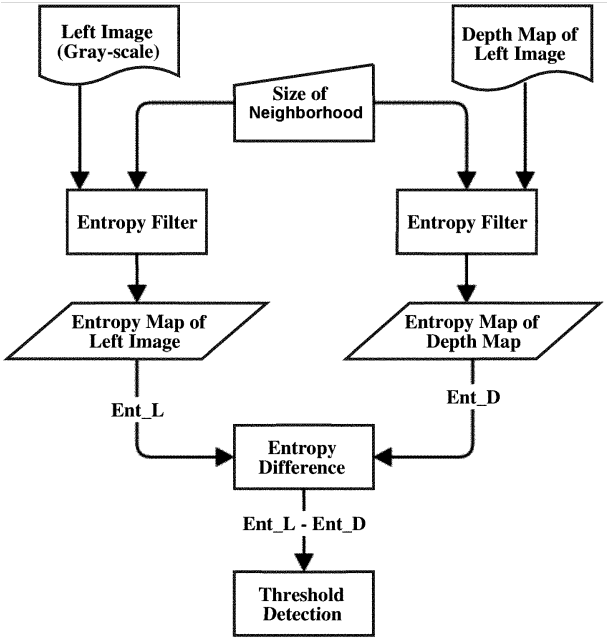


Figure 3.3: Proposed Depth Estimation Error Detection Method.

Inputs

The inputs to the proposed algorithm are the following:

1. Left Image (Gray-scale): The left image of the input stereo pair is converted to gray-scale (if it is not already in gray-scale) before using the proposed algorithm.

2. Depth Map of Left Image: The (initial) depth map output by the dense stereo matching algorithm, corresponding to the left image of the stereo pair is the 2nd input to the proposed algorithm. This (initial) depth map has incorrect disparity estimates for many of the image pixels, and the proposed algorithm will try to identify them.

3. Size of Neighborhood: This is a required parameter for the Entropy Filtering step, where Shannons entropy is calculated (separately) for each pixel of the left

image, and its (initial) depth map, by considering a square area around the pixel. The said pixel always occupies the central position inside this square area; hence, the value supplied for the size of the neighborhood must be *odd*.

Entropy Filter

Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image. The entropy filter computes the local entropy of a gray-scale image. It outputs a new image, in which each output pixel contains the entropy value of the neighborhood around the corresponding pixel in the input image. The size of the said neighborhood is determined by the input parameter, which has been described earlier. For pixels on the borders of the input image, symmetric padding is used, in which all the values of padding pixels are mirror reflections of border pixels of the input image. The entropy is calculated based on the standard Shannons entropy measure, given in (3.4),

$$h = -\sum_{i=1}^{n} p_i. * \log_2 p_i \tag{3.4}$$

where p_i contains the normalized histogram counts of the neighborhood around any given pixel. The number of histogram bins is taken as 256 (gray-scale image).

In the proposed method, the entropy filter is applied to both the (gray-scale) left image, and a (initial) depth map supplied by the stereo correspondence algorithm. Thus, two entropy maps are obtained, one for the left image, denoted by Ent_L, and the other for the (initial) depth map, denoted by Ent_D.

Further, scaling it performed on Ent_L and Ent_D by dividing each with the value of its maximum element and multiplying by 255 (maximum allowed gray-scale intensity value of any pixel), as this procedure of scaling produces better results.

Entropy Assumption

The core assumption underlying the proposed method is that, *entropy of any point on an image will be significantly higher than the entropy of its corresponding point on the depth map of that image*.

The above assumption is based on the observation that, gray-scale intensity can vary significantly (as compared to the depth level) in the neighborhood of any given pixel. This may be due to multiple reasons such as image noise, differences in lighting (like shadows or reflections), differences in surface texture, etc. even when the said neighboring pixels also lie on the same surface and roughly at the same depth level.

Based on this assumption, the proposed method subtracts the value of each pixel of the depth maps entropy map, Ent_D, from the corresponding pixel of the left images entropy map, Ent_L, to generate the entropy difference image, Ent as the output. Some of Ents pixels may have negative values (an absolute difference of entropy is not computed). Thus, high Ent values should correspond to the correct depth estimates.

Threshold Detection

This step deals with deriving a pixel value threshold, Ent_Th, from the entropy difference image, Ent. All pixels in the depth map of the left image corresponding to those in Ent with values lesser than Ent_Th will be classified as incorrect disparity estimates. The detailed steps are as follows:

1. Percentiles Computation: The proposed method first derives all the (integral) percentiles (1 through 100) from the values of Ent, denoted by P_i. One of these P_i will be designated as Ent_Th, as shown below.

2. Tracing Entropy Variations in Ent_D vs. Percentiles: For each percentile P_i, the standard deviation, E_i, of depth map entropys in Ent_D is calculated, such that corresponding pixels in Ent have values less than P_i. Then, a 3rd degree polynomial is fit (using Least Squares) to correlate the percentile value P_i and E_i, by taking them as independent and dependent variables, respectively. In the subsequent steps, the proposed method determine the inflection point of this polynomial (rationale behind all these steps is explained below) to uniquely determine the value of Ent_Th (as a 3rd degree polynomial has exactly one inflection point). It has been observed that, regions of incorrect disparity estimates are often made up of random, uncorrelated disparity values concentrated in small spatial regions. So, entropies of such *unstable* regions of the disparity map are expected to be high. Again, it has been seen that, regions of low entropy difference (Ent) values correspond to the regions of incorrect disparity estimates.

   Thus, combining the above two observations, the motivation behind the proposed method was to examine, how (by gradually taking higher values of Ent or P_i as the threshold, Ent_Th) the entropy of disparity estimates corresponding to Ent values lesser than Ent_Th (or, P_i), varies (as measured by the standard deviation of the entropy values). Further details w.r.t this explanation are provided in the *Results and Discussion* section.

3. Determining the Polynomials Inflection Point: An attempt is made to determine, whether the inflection point of the polynomial (obtained by setting its 2nd deriva-

tive w.r.t its independent variable to zero and solving the resulting equation) falls in between the 20^{th} and the 80^{th} percentile (P_i) value. In that case, the pixel value threshold, Ent_Th is chosen as the value of the inflection point, P_p. However, in case P_p does not lie between P_{20} and P_{80}, P_p (Ent_Th) is taken to be P_{50}, so as to disallow extreme values of the threshold Ent_Th.

4 RESULTS AND DISCUSSION

4.1 Depth-based Selective Blurring

To compare the stereo disparity estimation performance of the proposed approach versus those of some other (existing) algorithms, initially, the proposed algorithm is executed on on three pairs (left and right) of stereo images from the Middlebury data-set, viz. Tsukuba, Sawtooth and Venus and compared the generated depth maps with corresponding ground truth depth maps provided in the data-set. Depth maps obtained by the proposed method are also compared with those output by two established cost functions (SAD and NCC) used in numerous traditional stereo disparity estimation methods, and a recent method [12] which uses the absolute difference (AD) cost function, for those same three stereo image pairs. Error rates of the proposed algorithm are also compared with those of two most competitive disparity estimation techniques, viz. Graph-cut [44] and TV-based [45]. Also, the Middlebury Stereo Evaluation page maintains a table of performance statistics of numerous state-of-the-art disparity estimation methods, submitted by authors, after running their algorithms on four standard stereo pairs supplied as part of the data-set (viz., Tsukuba, Venus, Teddy and Cones) with constant parameters . It is obvious that there is no way to check whether the authors actually ran their algorithms with a constant set of parameter values for all the four image pairs, and algorithms may yield substantially different error rates for different sets of parameter values. Nevertheless, we consider this repository as an additional benchmark for evaluating the performance of the proposed algorithm.

The results of the said comparisons are presented below, along with the outputs of each step of the proposed approach for the Tsukuba image pair (since it presents a typical scene with adequate number of disparity levels in its provided ground truth depth map to mimic real-world depth-based blurring scenarios), to demonstrate how the proposed depth estimation algorithm works.

Characteristics of all Middlebury stereo image pairs used for evaluation of the proposed algorithm and comparison of the output depth maps, are presented in Table 4.1. For experimental purposes, the set of parameter values given in Table 4.2 are used.

Table 4.1: Middlebury Stereo Image Pairs used for Performance Evaluations

Characteristics	***Tsukuba***	***Sawtooth***	***Venus***	***Teddy***	***Cones***
Size (pixels)	384×288	434×380	434×383	450×375	450×375
Disparity Levels	16	20	20	60	60

4.1.1 Outputs of Proposed Method for Tsukuba Image

Colour Space Conversion

Process: Left and right input images are converted from the RGB to the CIE-Lab color space following the pixel value transformations outlined in [42].

Results: Each pixel of both images has an L value for its lightness component, and Fig. 4.1 (left) shows the converted left input image.

Observations: The output image shows the lightness value of each pixel.

Figure 4.1: Segmentation of L Values of Left Image using K-Means.

Segmentation

Process: Converted left input image is fed to the fast implementation of K-Means clustering algorithm.

Results: The result is shown in Fig. 4.1 (right).

Observations: Small variations in L values (noise) have been absorbed into single coherent segments representing image regions of pixels belonging to the same object, with similar values of disparity. However, since the left image is segmented based on just the pixels' L values, ignoring their spatial locations (to expedite the clustering process), it is required to refine the boundaries of objects defined by the clusters above, so that the spatial distribution of the L values are also taken into account.

Segment Boundary Detection and Refinement

Process: The segment boundaries are detected and refined using the proposed approach.

Results: The outputs are shown in Fig. 4.2, with output of segment boundary point detection, followed by that of morphological filtering, and lastly, output of connected components analysis at the bottom.

Observations: It can be inferred from Fig. 4.2 (center), how the redundancies in segment boundary detection have been removed from Fig. 4.2 (left), and from Fig. 4.2 (right), how the small artifacts in Fig. 4.2 (center) have been removed by the connected components analysis technique.

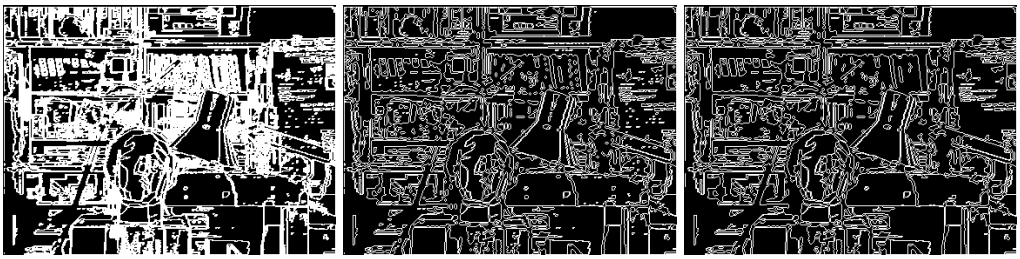


Figure 4.2: Segment Boundary Detection and Refinement.

For the Tsukuba image pair, the segment boundary refinement reduces the number of boundary pixels by nearly 53% for the parameter values of Table 4.2, such that only 19% of the left image pixels are used for disparity calculations. This greatly reduces the number of disparity computations in the next step.

Disparity Measurement of Boundary Pixels

Process: Disparities of the boundary pixels are determined by using the SAD algorithm.

Results: The result in shown in Fig. 4.3 (left).

Observations: Boundaries of objects closer to the camera (like the lamp and the head of the statue) are having a higher intensity (greater value of stereo disparity). This is in direct agreement with the reality that value of disparity of a pixel is inversely proportional to its distance (from the camera lens / human eyes).

Disparity Map Reconstruction from Boundaries

Process: Disparities of the segmented regions in the image are determined from disparities of their boundaries using proposed two-stage disparity map reconstruction method.

Results: The results are shown in Fig. 4.3 (right).

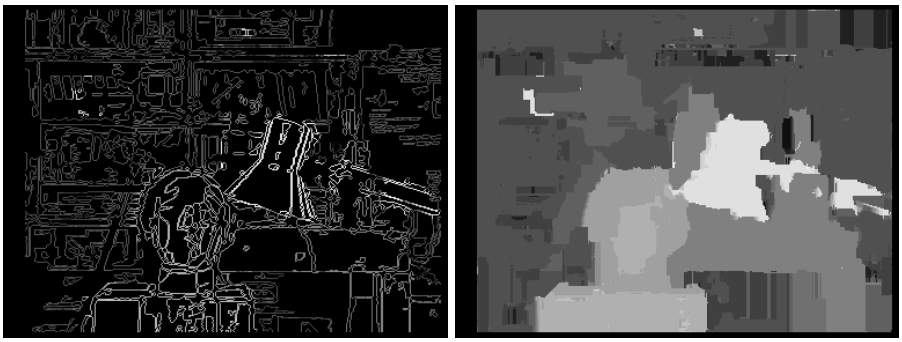


Figure 4.3: Disparity Map Reconstruction from Boundary Disparities.

Observations: The depth map shows the mapping of each image point to a certain level of disparity. Objects which are closer to the camera like the lamp-shade have a greater disparity value resulting in a brighter shade of gray. For objects further away from the camera like the table, the shades of gray get progressively darker, as their disparity values keep on decreasing.

To evaluate the disparity estimation of the proposed approach vis-a-vis those of existing algorithms, initially, three pairs (left and right) of stereo images are chosen from the Middlebury stereo vision data-set, viz. Tsukuba, Sawtooth and Venus. The proposed algorithm is run on those three image pairs. Depth maps obtained by the proposed algorithm are then compared against the corresponding ground truth depth maps provided in the data-set, as well as with the depth maps created by two established methods (SAD and NCC [10]) and a recent work by Zhen Zhang et al. [12] for those same three image pairs. The results of those comparisons are presented below.

4.1.2 Qualitative and Quantitative Comparison with Ground Truth

To evaluate the depth estimates of the proposed method,the approach of computing error statistics with respect to the ground truth image available with the Middlebury dataset is considered, and the quality metric is chosen as *percentage of bad matching pixels* (B) given by (4.1),

$$B = \frac{1}{N} \sum_{(x,y)} \left(\left| d_C\left(x,y\right) - d_T\left(x,y\right) \right| > \delta_d \right) \tag{4.1}$$

where $d_C\left(x,y\right)$ are the computed disparities and $d_T\left(x,y\right)$ are ground truth disparities, and δ_d denotes the disparity error tolerance, which is taken as 1.0 as per published works [46]. It may be further noted that, we are not at all concerned with sub-pixel accuracy in our proposed disparity map generation methodology; hence, the disparity error tolerance can only (meaningfully) have integral values. Thus, a disparity error tolerance

of '1.0' signifies that all "generated" disparities which deviate from the corresponding ground truth disparities by ± 1.0 are considered by our evaluation methodology as "accurate".

Table 4.2 shows the values of the proposed algorithm's parameters which gave best results and hence are considered for performance comparison. From Table 4.2, it is evident that the set of optimal parameter values is almost constant across all three images. It is observed in the experiments that slight variations of the parameter values about the optimal ones effect negligible changes in accuracy of depth estimation. Thus, the proposed method is robust to changes in parameter values, like the one in [27].

Table 4.2: Parameter Values for the Three Image Pairs

Parameter	***Tsukuba***	***Sawtooth***	***Venus***
Number of clusters (K) for K-Means	10	10	10
Block size for cost aggregation (odd)	9×9 pixels	9×9 pixels	9×9 pixels
Disparity threshold for reconstruction	0	1	1

Fig. 4.4 presents a comparison of the depth map produced by the proposed method (left) with ground truth depth map (right) for the Tsukuba image. The black regions in depth maps are those whose disparities are not compared. Further, the percentage of bad matching pixels (including occluded image regions) was found to be 7.8%.

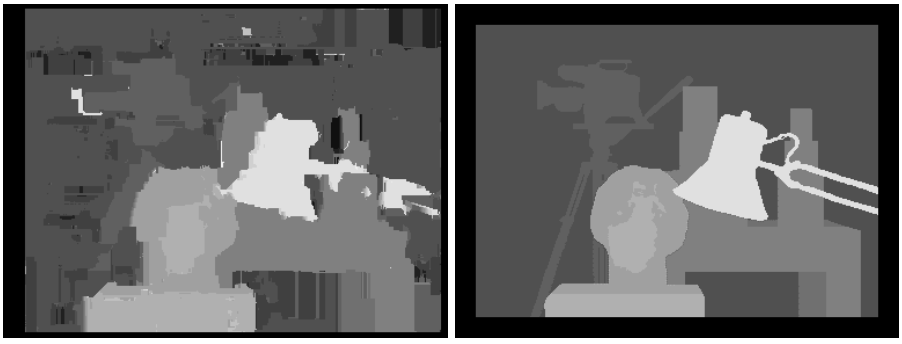


Figure 4.4: Resultant Depth Map Compared to Middlebury's Ground Truth.

It can be seen from the comparison of the depth map output by the proposed method with the ground truth that the proposed algorithm yields fairly accurate depth estimation; its performance is affected mostly in some texture-less and low illumination regions. But as the generated depth maps will be used for the purpose of selective

blurring, inaccurate depth estimates in regions of low texture and illumination will not affect the blurring performance, as human eyes are more sensitive to variations in brightness than colour, and hence perceptual differences resulting from blurring a dark region would be negligible. Likewise, as blurring is performed using a weighted average of neighbouring pixel values, a region of low texture will remain relatively unaffected after a blurring operation, as the variation in pixel values in a texture-less region is negligible. Some incorrect depth estimations also occur in occluded regions which can be reduced using occlusion handling techniques in future. Also, the proposed algorithm does not calculate depth information of pixels lying near the image borders (which will be addressed in future work), but this limitation does not significantly affect the output of depth-based blurring, as the images borders very rarely contain objects or regions of interest to the user.

4.1.3 Qualitative and Quantitative Comparison with State-of-Art

Table 4.3 shows the results of performance comparison of the proposed depth estimation method with two established methods, viz. Sum of absolute differences (SAD) and normalized cross-correlation (NCC) (both of which have been used as cost functions in numerous traditional algorithms for stereo depth estimation), as well as a recent work by Zhen Zhang et al. [12] which considered absolute difference (AD) cost function to predict disparities. Again, the percentage of bad matching pixels is used to quantitatively compare depth maps generated by the proposed method, SAD, NCC and Zhen Zhang et al. [12]. The accuracies of depth maps generated by the proposed algorithm are also separately evaluated by considering the corresponding Middlebury ground truth depth maps, by both including and excluding the occluded regions of the input images, as algorithms like the proposed method and Zhen Zhang et al. [12] often do not deal with occlusion handling explicitly. In this context, it should also be noted that, in the work by Zhen Zhang et al. [12], the authors have not specifically mentioned clearly whether they have evaluated their algorithm only on non-occluded image regions, or on occluded image regions as well.

The results clearly demonstrate that the proposed method outperforms traditional NCC and SAD methods by 33.6%, and even a recent method of Zhen Zhang et al. [12] by 6.1% (including occluded regions of input images).

Further, the percentage of left image pixels used for stereo depth estimation is about 19% for Tsukuba, 18% for Sawtooth and 16% for Venus image pairs, implying that the proposed algorithm is scalable to high resolution stereo images.

Figs. 4.5 and 4.6 show depth maps generated by the proposed method (left) and ground truth depths (right) for the Sawtooth and Venus images. Their quantitative

Table 4.3: Performance Comparison of Proposed Algorithm

	***Tsukuba* %**	***Sawtooth* %**	***Venus* %**
NCC [10]	41.4	9.92	17.4
SAD [10]	36.9	11.9	24.5
Zhen Zhang et al. [12]	13.9	7.22	6.12
Proposed Algorithm (including occluded regions)	7.8	5.26	4.72
Proposed Algorithm (excluding occluded regions)	6.1	3.29	3.78

comparison results are shown above, in Table 4.3. All other depth maps supporting performance comparison data presented in Table 4.3 can be found in [12].

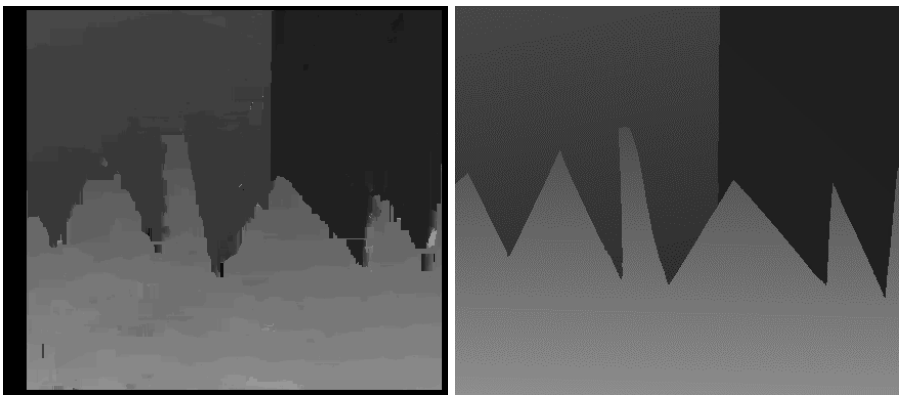


Figure 4.5: Sawtooth Depth Map for Proposed Algorithm (left) and Middlebury's Ground Truth (right).

The proposed method is next compared with two most competitive depth estimation techniques, viz. Graph-cut [44] and TV-based [45], and the results are shown in Table 4.4 and Table 4.5 respectively. The same quality metric, viz., *Percentage of bad matching pixels* is used, (with the value of δ_d denoting the disparity error tolerance, taken as 1.0) to quantitatively compare depth maps generated by the proposed method and the Graph-cut based method [44]. To compare the results of the proposed method with the TV-based method [45], a different quality metric, viz., *Average Absolute Disparity Error* (AADE) is used, similar to that of TV-based method [45].

Analysis of results presented in Table 4.4 and Table 4.5 shows that, in cases where the most competitive recent methods like Graph-cut [44] are able to successfully address the issue of occlusions in the input image (by adopting an expansion-moves approach over swap-moves approach), the proposed method has a definite scope of improvement,

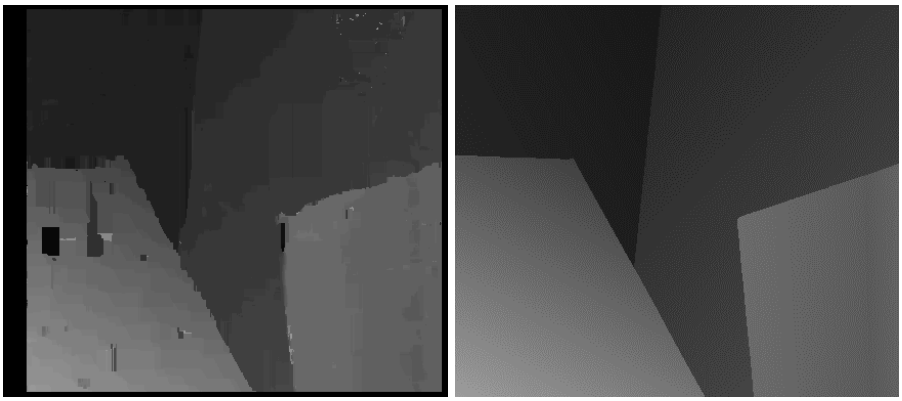


Figure 4.6: Venus Depth Map for Proposed Algorithm (left) and Middlebury's Ground Truth (right).

Table 4.4: Performance Comparison of Proposed Algorithm with Graph-cut Method

Algorithm Execution on Non-Occluded Image Regions	***Tsukuba %***
Proposed Method without Occlusion Handling	6.1
Kolmogorov's Graph-cut Method (using expansion-moves)	1.9
Kolmogorov's Graph-cut Method (using swap-moves)	13.6

Table 4.5: Performance Comparison of Proposed Algorithm with TV-based Method

	Tsukuba $\boldsymbol{B}_{all}$ $\boldsymbol{B}_{untex}$ $\boldsymbol{B}_{disc}$	***Sawtooth*** $\boldsymbol{B}_{all}$ $\boldsymbol{B}_{untex}$ $\boldsymbol{B}_{disc}$	***Venus*** $\boldsymbol{B}_{all}$ $\boldsymbol{B}_{untex}$ $\boldsymbol{B}_{disc}$
Proposed Method	0.56, 0.70, 1.30	0.44, 0.42, 1.58	0.49, 0.59, 1.28
TV-based Method	0.29, 0.24, 0.51	0.23, 0.19, 0.41	0.24, 0.26, 0.39

since occlusions have not been considered as of now. But, using the swap-moves approach, where the Graph-cut based method fails to properly handle occlusions in the input image, the proposed algorithm performs significantly better. Also, it must be noted that handling occlusions in input images entails a higher computational complexity, as evident from the authors run-time calculation [44].

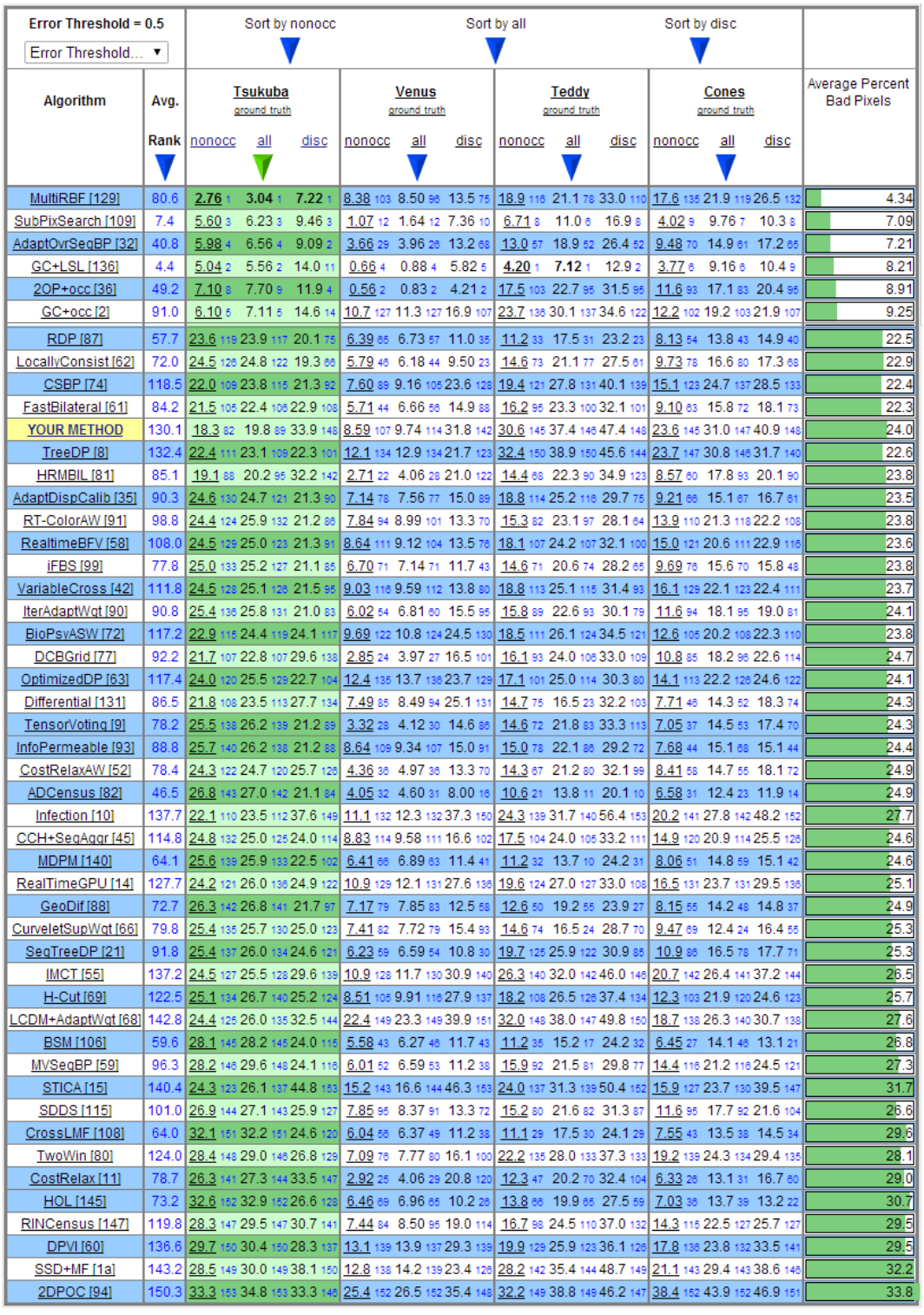
On the other hand, TV-based methods like [45] give better disparity estimates by iteratively refining left-to-right and right-to-left initial disparity maps (obtained using a correlation-based method). The results are shown in Table 4.5 where, using similar evaluation methodology as authors [45], the performance of the proposed algorithm has been quantitatively compared in three different types of areas in the image, classified as un-textured (B_{untex}), discontinuous (B_{disc}) and the entire image (B_{all}), for only non-occluded pixels in all three cases. Understandably, the proposed method yields comparatively greater error rates, as the proposed method does not have any initial disparity

maps to start with, as it builds it from scratch.

For the final set of comparisons w.r.t the proposed algorithms error-rate, once more, the Tsukuba image pair is considered (as it represents a scene which can be a typical candidate for many real-world depth-based blurring scenarios). The Stereo Evaluation table of the Middlebury website is used to compare the proposed algorithms error-rate against those of numerous others, based on their output depth maps submitted to the table by their corresponding authors, and evaluation done by the website itself. A set (fixed) of parameter values is used: Number of K-Means Clusters (K) = 10, Block Size for Cost Aggregation = 9×9 pixels, and Disparity Threshold = 1.0 for all four standard data-set stereo image pairs required for evaluation (Tsukuba, Venus, Teddy and Cones). Then, the generated depth maps are submitted through the Middlebury websites online submission form for evaluation and comparison against their repository of results of several existing state-of-the-art algorithms.

The comparison results are shown in Fig. 4.7, with the proposed method labelled *YOUR METHOD* as per the Stereo Evaluation table of the Middlebury website. The original table has more than 150 entries and cannot fit within the space constraints of this paper, so the table was truncated from the 7^{th} entry onward, till the 5^{th} entry before the one corresponding to the proposed method. The reason for this is, as the table entries are sorted by increasing order of error-rate for the Tsukuba image pair, by truncating the table in the said fashion, one can see some of the best performing methods, followed by few methods which perform slightly better than the proposed method, followed my results of the proposed method, and lastly, those of all the 38 methods which perform worse than the proposed method. The disparity error tolerance, δ_d has been taken to be the most stringent permissible, viz. 0.5, which translates to finding (the complement of) the percentage of *exactly* matching pixel disparities in case methods like the proposed method, which do not compute sub-pixel disparities.

From an analysis of the results presented in Fig. 4.7, one can conclude that the proposed method performs better for *nonocc* (non-occluded) and *all* input image regions, but does not produce up-to-the-mark results for the *disc* regions (which denote regions of disparity discontinuities) for all the four (input) test image pairs. Since many *disc* regions coincide with object boundaries, hence improvement of depth estimation performance in these types of image regions can directly translate to an improved overall depth-based blurring performance and so needs to be addressed as part of future work. Moreover, the error-rate for outputs of the proposed method is much higher for the *Teddy* and *Cones* pairs (whose ground truth depth maps have 60 disparity levels each) than the *Tsukuba* and *Venus* pairs (whose respective ground truth depth maps have 16 and 20 disparity levels). Hence, there exists a definite scope of betterment of the pro-



Error Threshold = 0.5 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc			
Algorithm	Avg. Rank	Tsukuba ground truth nonocc	Tsukuba ground truth all	Tsukuba ground truth disc	Venus ground truth nonocc	Venus ground truth all	Venus ground truth disc	Teddy ground truth nonocc	Teddy ground truth all	Teddy ground truth disc	Cones ground truth nonocc	Cones ground truth all	Cones ground truth disc	Average Percent Bad Pixels
MultiRBF [129]	80.6	**2.76** 1	**3.04** 1	**7.22** 1	8.38 103	8.50 96	13.5 75	18.9 116	21.1 78	33.0 110	17.6 135	21.9 119	26.5 132	4.34
SubPixSearch [109]	7.4	5.60 3	6.23 3	9.46 3	1.07 12	1.64 12	7.36 10	6.71 8	11.0 6	16.9 8	4.02 9	9.76 7	10.3 8	7.09
AdaptOvrSegBP [32]	40.8	5.98 4	6.56 4	9.09 2	3.66 29	3.96 26	13.2 66	13.0 57	18.9 62	26.4 62	9.48 70	14.9 61	17.2 65	7.21
GC+LSL [136]	4.4	5.04 2	5.56 2	14.0 11	0.66 4	0.88 4	5.82 5	**4.20** 1	**7.12** 1	12.9 2	3.77 6	9.16 6	10.4 9	8.21
2OP+occ [36]	49.2	7.10 8	7.70 9	11.9 4	0.56 2	0.83 2	4.21 2	17.5 103	22.7 95	31.5 95	11.6 93	17.1 83	20.4 95	8.91
GC+occ [2]	91.0	6.10 5	7.11 5	14.6 14	10.7 127	11.3 127	16.9 107	23.7 136	30.1 137	34.6 122	12.2 102	19.2 103	21.9 107	9.25
RDP [87]	57.7	23.6 119	23.9 117	20.1 75	6.39 85	6.73 57	11.0 35	11.2 33	17.5 31	23.2 23	8.13 54	13.8 43	14.9 40	22.5
LocallyConsist [62]	72.0	24.5 126	24.8 122	19.3 66	5.79 46	6.18 44	9.50 23	14.6 73	21.1 77	27.5 61	9.73 78	16.6 80	17.3 68	22.9
CSBP [74]	118.5	22.0 109	23.8 115	21.3 92	7.60 89	9.16 105	23.6 128	19.4 121	27.8 131	40.1 139	15.1 123	24.7 137	28.5 133	22.4
FastBilateral [61]	84.2	21.5 105	22.4 106	22.9 108	5.71 44	6.66 56	14.9 88	16.2 95	23.3 100	32.1 101	9.10 63	15.8 72	18.1 73	22.3
YOUR METHOD	130.1	18.3 82	19.8 89	33.9 148	8.59 107	9.74 114	31.8 142	30.6 145	37.4 146	47.4 148	23.6 145	31.0 147	40.9 148	24.0
TreeDP [8]	132.4	22.4 111	23.1 109	22.3 101	12.1 134	12.9 134	21.7 123	32.4 150	38.9 150	45.6 144	23.7 147	30.8 146	31.7 140	22.6
HRMBIL [81]	85.1	19.1 88	20.2 95	32.2 142	2.71 22	4.06 28	21.0 122	14.4 68	22.3 90	34.9 123	8.57 60	17.8 93	20.1 90	23.8
AdaptDispCalib [35]	90.3	24.6 130	24.7 121	21.3 90	7.14 78	7.56 77	15.0 89	18.8 114	25.2 116	29.7 75	9.21 66	15.1 67	16.7 61	23.5
RT-ColorAW [91]	98.8	24.4 124	25.9 132	21.2 86	7.84 94	8.99 101	13.3 70	15.3 82	23.1 97	28.1 84	13.9 110	21.3 118	22.2 108	23.8
RealtimeBFV [58]	108.0	24.5 129	25.0 123	21.3 91	8.64 111	9.12 104	13.5 76	18.1 107	24.2 107	32.1 100	15.0 121	20.6 111	22.9 116	23.6
iFBS [99]	77.8	25.0 133	25.2 127	21.1 85	6.70 71	7.14 71	11.7 43	14.6 71	20.6 74	28.2 85	9.69 76	15.6 70	15.8 48	23.8
VariableCross [42]	111.8	24.5 128	25.1 126	21.5 95	9.03 116	9.59 112	13.8 80	18.8 113	25.1 115	31.4 93	16.1 129	22.1 123	22.4 111	23.7
IterAdaptWgt [90]	90.8	25.4 136	25.8 131	21.0 83	6.02 54	6.81 60	15.5 95	15.8 89	22.6 93	30.1 79	11.6 94	18.1 95	19.0 81	24.1
BioPsyASW [72]	117.2	22.9 116	24.4 119	24.1 117	9.69 122	10.8 124	24.5 130	18.5 111	26.1 124	34.5 121	12.6 105	20.2 108	22.3 110	23.8
DCBGrid [77]	92.2	21.7 107	22.8 107	29.6 138	2.85 24	3.97 27	16.5 101	16.1 93	24.0 106	33.0 109	10.8 85	18.2 96	22.6 114	24.7
OptimizedDP [63]	117.4	24.0 120	25.5 129	22.7 104	12.4 135	13.7 136	23.7 129	17.1 101	25.0 114	30.3 80	14.1 113	22.2 126	24.6 122	24.1
Differential [131]	86.5	21.8 108	23.5 113	27.7 134	7.49 85	8.49 94	25.1 131	14.7 75	16.5 23	32.2 103	7.71 48	14.3 52	18.3 74	24.3
TensorVoting [9]	78.2	25.5 138	26.2 139	21.2 89	3.32 28	4.12 30	14.6 86	14.6 72	21.8 83	33.3 113	7.05 37	14.5 53	17.4 70	24.3
InfoPermeable [93]	88.8	25.7 140	26.2 138	21.2 88	8.64 109	9.34 107	15.0 91	15.0 78	22.1 86	29.2 72	7.68 44	15.1 68	15.1 44	24.4
CostRelaxAW [52]	78.4	24.3 122	24.7 120	25.7 126	4.36 36	4.97 36	13.3 70	14.3 67	21.2 80	32.1 99	8.41 58	14.7 55	18.1 72	24.9
ADCensus [82]	46.5	26.8 143	27.0 142	21.1 84	4.05 32	4.60 31	8.00 16	10.6 21	13.8 11	20.1 10	6.58 31	12.4 23	11.9 14	24.9
Infection [10]	137.7	22.1 110	23.5 112	37.6 149	11.1 132	12.3 132	37.3 150	24.3 139	31.7 140	56.4 153	20.2 141	27.8 142	48.2 152	27.7
CCH+SegAggr [45]	114.8	24.8 132	25.0 125	24.0 114	8.83 114	9.58 111	16.6 102	17.5 104	24.0 105	33.2 111	14.9 120	20.9 114	25.5 126	24.6
MDPM [140]	64.1	25.6 139	25.9 133	22.5 102	6.41 86	6.89 63	11.4 41	11.2 32	13.7 10	24.2 31	8.06 51	14.8 59	15.1 42	24.6
RealTimeGPU [14]	127.7	24.2 121	26.0 136	24.9 122	10.9 129	12.1 131	27.6 136	19.6 124	27.0 127	33.0 108	16.5 131	23.7 131	29.5 136	25.1
GeoDif [88]	72.7	26.3 142	26.8 141	21.7 97	7.17 79	7.85 83	12.5 58	12.6 60	19.2 55	23.9 27	8.15 55	14.2 48	14.8 37	24.9
CurveletSupWgt [66]	79.8	25.4 135	25.7 130	25.0 123	7.41 82	7.72 79	15.4 93	14.6 74	16.5 24	28.7 70	9.47 69	12.4 24	16.4 55	25.3
SegTreeDP [21]	91.8	25.4 137	26.0 134	24.6 121	6.23 59	6.59 54	10.8 30	19.7 125	25.9 122	30.9 85	10.9 88	16.5 78	17.7 71	25.3
IMCT [55]	137.2	24.5 127	25.5 128	29.6 139	10.9 128	11.7 130	30.9 140	26.3 140	32.0 142	46.0 146	20.7 142	26.4 141	37.2 144	26.5
H-Cut [69]	122.5	25.1 134	26.7 140	25.2 124	8.51 105	9.91 116	27.9 137	18.2 108	26.5 128	37.4 134	12.3 103	21.9 120	24.6 123	25.7
LCDM+AdaptWgt [68]	142.8	24.4 125	26.0 135	32.5 144	22.4 149	23.3 149	39.9 151	32.0 148	38.0 147	49.8 150	18.7 138	26.3 140	30.7 138	27.6
BSM [106]	59.6	28.1 145	28.2 145	24.0 115	5.58 43	6.27 46	11.7 43	11.2 35	15.2 17	24.2 32	6.45 27	14.1 46	13.1 21	26.8
MVSegBP [59]	96.3	28.2 146	29.6 148	24.1 116	6.01 52	6.59 53	11.2 38	15.9 92	21.5 81	29.8 77	14.4 116	21.2 116	24.5 121	27.3
STICA [15]	140.4	24.3 123	26.1 137	44.8 153	15.2 143	16.6 144	46.3 153	24.0 137	31.3 139	50.4 152	15.9 127	23.7 130	39.5 147	31.7
SDDS [115]	101.0	26.9 144	27.1 143	25.9 127	7.85 95	8.37 91	13.3 72	15.2 80	21.6 82	31.3 87	11.6 95	17.7 92	21.6 104	26.6
CrossLMF [108]	64.0	32.1 151	32.2 151	24.6 120	6.04 56	6.37 49	11.2 38	11.1 29	17.5 30	24.1 29	7.55 43	13.5 38	14.5 34	29.6
TwoWin [80]	124.0	28.4 148	29.0 146	26.8 129	7.09 78	7.77 80	16.1 100	22.2 135	28.0 133	37.3 133	19.2 139	24.3 134	29.4 135	28.1
CostRelax [11]	78.7	26.3 141	27.3 144	33.5 147	2.92 25	4.06 29	20.8 120	12.3 47	20.2 70	32.4 104	6.33 26	13.1 31	16.7 60	29.0
HOL [145]	73.2	32.6 152	32.9 152	26.6 128	6.46 69	6.96 65	10.2 26	13.8 86	19.9 65	27.5 59	7.03 38	13.7 39	13.2 22	30.7
RINCensus [147]	119.8	28.3 147	29.5 147	30.7 141	7.44 84	8.50 95	19.0 114	16.7 98	24.5 110	37.0 132	14.3 115	22.5 127	25.7 127	29.5
DPVI [60]	136.6	29.7 150	30.4 150	28.3 137	13.1 139	13.9 137	29.3 139	19.9 129	25.9 123	36.1 126	17.8 136	23.8 132	33.5 141	29.5
SSD+MF [1a]	143.2	28.5 149	30.0 149	38.1 150	12.8 138	14.2 139	23.4 126	28.2 142	35.4 144	48.7 149	21.1 143	29.4 143	38.6 146	32.2
2DPOC [94]	150.3	33.3 153	34.8 153	33.3 146	25.4 152	26.5 152	35.4 148	32.2 149	38.8 149	46.2 147	38.4 152	43.9 152	46.9 151	33.8

Figure 4.7: Truncated Middlebury Stereo Evaluation Table comparing the Proposed Method vs. State-of-the-Art.

posed method for scenarios requiring large number of depth levels.

4.1.4 Comparison of Serial Execution Time with State-of-Art

To compare the running time of the proposed algorithm with those of state-of-the-art stereo disparity estimation algorithms, the approach described in a very recent work by Tippetts et al. [14] is partly adopted. In their work, authors evaluated numerous published stereo vision algorithms, and evaluate their run-time performance. Their evaluation metric is *millions of disparity evaluations per second* (Mde/s), which is calculated from the time to compute the disparity map by an algorithm for one frame, t, is given by (4.2), where W is the width and H, the height of the input image, and D, the number of disparity levels. Since the authors evaluated many of the algorithms on images of different sizes, this resulted in different runtimes. So, values of execution time they report, include the highest performance each algorithm achieved, along with the corresponding image size.

$$Mde/s = \frac{W \times H \times D}{t} \times \frac{1}{1,000,000} \tag{4.2}$$

The authors observed that multiple factors can influence runtime measurements of stereo vision algorithms, such as, the computational power of a CPU on which it is executed, programming language in which it is implemented, skill and effort of programmers in optimizing the implementation, parallelization techniques used, etc. Also, it is often necessary to make such a comparison when deciding which algorithm to implement for a given application. Thus, they include all published runtimes achieved by the stereo vision algorithms they evaluated, as well as all available hardware details, to let the reader make such comparisons.

The authors also discourage the practice of scaling runtime measurements directly by a processor clock speed, as it ignores other factors, such as processor pipelining, cache design, memory subsystem etc. while comparing running times of two or more algorithms implemented and executed on processors with widely varying configurations and clock speeds. Instead, they collected processor bench-mark values for each of the specific CPUs on which their reviewed methods were executed from the PassMark Software CPU benchmark table. Using these values of CPU benchmarks, they calculated a normalizing factor for each of those CPUs, which were later used to scale the run-times of all the depth estimation algorithms running on those CPUs. In their performance comparison tables and graphs, they provided both un-normalized and normalized runtimes of their surveyed methods to aid the process of comparison. However, they have not mentioned the particular formula or method using which the normalizing factor was

calculated. Further, the processor on which the implementation of the proposed stereo depth estimation method was executed, had a different configuration and clock speed compared to the CPUs mentioned in their work. Hence, the un-normalized run-times of the proposed method are reported, along with the CPU configuration specifications used for our experiments and details of the corresponding input stereo pairs, in terms of width, height, and number of disparity levels of the test images, to let the reader make appropriate comparisons.

Table 4.6: Comparison of Proposed Algorithm with State-of-the-Art Methods w.r.t Computation Time

Method	***Run-Time (seconds)***	***Input Stereo Image Pair Width* $\times$ *Height (Disparities)***	***Mde/s***	***Hardware***
Proposed	1.07	450 $\times$ 375 (60)	9.4626	Core i7-2600 3.40 GHz
za [14]	1.6	450 $\times$ 375 (60)	6.3281	P4 3.0 GHz
we [14]	2.57	450 $\times$ 375 (60)	3.9382	Core 2 Duo 2.66 GHz
Proposed	0.84	434 $\times$ 383 (20)	3.9342	Core i7-2600 3.40 GHz
sd [14]	1.7	434 $\times$ 383 (20)	1.9556	N/A
sc [14]	1.79	434 $\times$ 383 (20)	1.8572	N/A
gp [14]	0.46	384 $\times$ 288 (16)	3.8467	Core 2 Duo 2.80 GHz
Proposed	0.57	384 $\times$ 288 (16)	3.1043	Core i7-2600 3.40 GHz
cg [14]	0.81	384 $\times$ 288 (16)	2.1845	P4 2.8 GHz
Proposed	12.985	1920 $\times$ 1080 (60)	9.5815	Core i7-2600 3.40 GHz
Proposed	10.188	1920 $\times$ 1080 (16)	3.2565	Core i7-2600 3.40 GHz

Table 4.6 shows results of comparison of running time of the implementation of the proposed algorithm in MATLAB version 2013b in terms of Mde/s against some of the best performing methods surveyed by authors [14]. In Table 4.6, the results are grouped according to the input stereo image pair. The entries within each such group was sorted on the decreasing order of Mde/s, following a similar approach of Tippetts et al. [14]. The last two entries of Table 4.6 are for a *True-HD* resolution stereo image pair comprising a stereo frame, which were processed using proposed algorithm and the results show its scalability.

While interpreting the computation time performance of the proposed algorithm vis-a-vis those of other state-of-the-art algorithms presented in Table 4.6, one should not forget, that the proposed methods entire implementation is in MATLAB, which is, largely, an interpreted language. Thus, it may often run comparatively slower than the C / C++ / OpenCV implementations of other methods surveyed by the authors [14].

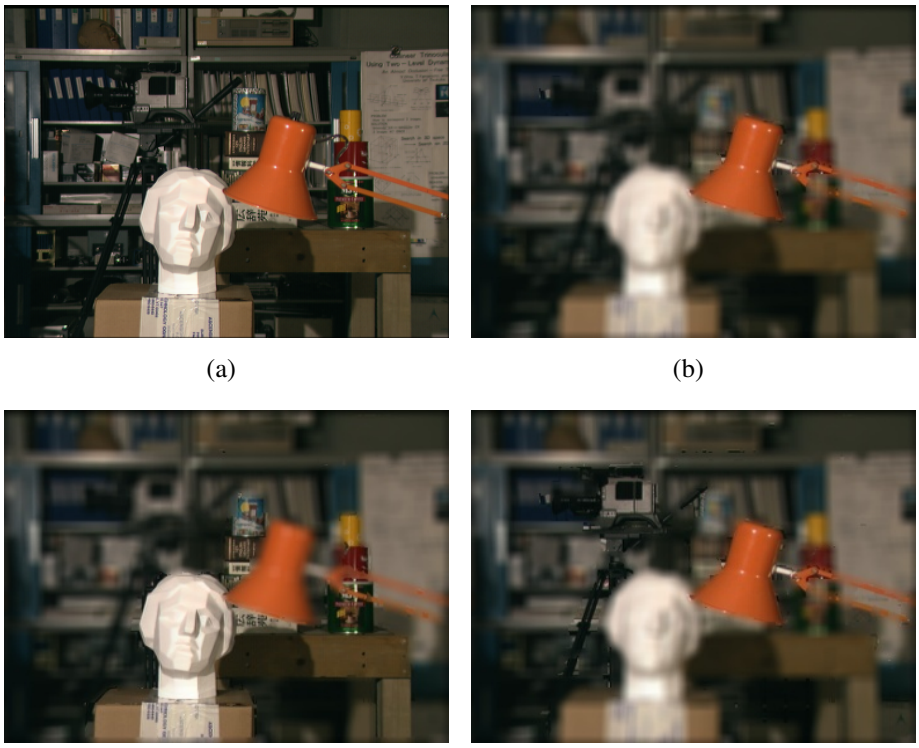
4.1.5 Depth-based Blurring of Tsukuba and Real-World Images

Fig. 4.8 shows the proposed GUI, which lets users visualize the entire range of depth levels present in the depth map output by the proposed method. From this GUI, the user can pick depth range(s) as per his (her) choice of region(s) or object(s) of interest, which he (she) wants to keep in focus. Accordingly, the blur map will be generated.

Figure 4.8: GUI showing Depth Levels in Depth Map output by Proposed Method.

Next, the results of depth-based blurring on the *Tsukuba* image using depth maps output by the proposed algorithm are presented in Fig. 4.9. In the top-right image, only the lamp, and in the bottom-left one, only the statue-head and the table are in focus, as per input depths ranges of interest. Multiple in-focus depth ranges, like the bottom-right image extend the methods like [1]. It is interesting to note that the incorrect depth estimates of the proposed algorithm in the texture-less and low illumination regions do not (noticeably) affect the depth-based blurring.

The results of depth-based blurring on a real-world scene which are captured using a Sony HDR-TD10 3D camcorder and later processed using the proposed method are presented in Fig. 4.10. The input stereo image pair is extracted from the frames of a short 3D (stereo) video clip shot using the said 3D camcorder, and only the left frame is shown in Fig. 4.10. It is interesting to note that even though no stereo rectification algorithm is applied to the extracted stereo image pair prior to its processing with the proposed method, yet the generated output image shows very few inconsistencies.



(a) (b)

(c) (d)

Figure 4.9: Depth-based Blurring on (a) Tsukuba: Original Image, (b & c) Depth-based Blurring with Single Depth Range in Focus, and (d) Depth-based Blurring with Multiple Depth Ranges in Focus.

Figure 4.10: Original Image (left) and Depth-based Blurring with Single Depth Range in Focus (right).

4.2 Results of Parallel Algorithm using JTP and APARAPI

Lastly, the results of parallelization of the computationally intensive, yet parallelizable steps of the proposed depth extraction algorithm using CPU (Java Thread Pool) and GPU (APARAPI) separately, and then by combining them are presented in Table 4.7. Experiments are performed on a PC with an Intel i7-2600 quad-core @ 3.40 GHz CPU with 8 GB RAM and NVIDIA NVS 300 GPU with 2 compute units and 512 MB RAM. In the *combined* implementation, four JTP worker threads and two APARAPI kernels are invoked simultaneously, as it gave best results.

Table 4.7: Results of Parallelization of Proposed Algorithm

No. of Images	***Serial Time (ms)***	***JTP-only Time (ms)***	***APARAPI-only Time (ms)***	***JTP+APRARAPI Time (ms)***
150	73191	32396	17331	13822
200	96361	42391	22982	16595
250	116909	47070	28514	20192

The increase in processing speed obtained by parallelizing serial algorithms and running on CPUs / GPUs is quantitatively described by a metric called speed-up factor as shown in (4.3).

$$Speedup = \frac{ExecutionTime_{Serial}}{ExecutionTime_{Parallel}} \tag{4.3}$$

Speed-ups achieved over serial implementation by parallelizing proposed algorithm and executing serial and parallel versions on CPU and GPU with an input stereo image sequence of 4096 $\times$ 2304 pixels each are given in Table 4.8. Maximum speed-up obtained is 5.8 for a sequence of 250 stereo images, for CPU+GPU (combined) parallelization approach. It is observed that, generally with increase in the number of multi-processing units (cores) and increase in speed of each processing unit, parallel executions become quicker and hence speed-up increases following a sub-linear trend. It is not perfectly linear due to many causes, the most significant of which is a factor called *parallelization overhead*, which deals with coordinating activities of different processing units to collectively achieve a single task, like the distribution of total workload among all parallel processing units, collecting the intermediate processing results from each unit and compiling them to form the output.

From Fig. 4.11, it can be easily inferred that, rate of increase of running time of the proposed algorithm is greatest for the serial implementation, followed successively by the JTP, APARAPI and the JTP+APRARAPI (combined) ones, in decreasing order. This is because GPUs are more suitable for running data-parallel workloads than CPUs,

Table 4.8: Speed-ups Achieved by Parallelization of Proposed Algorithm

No. of Images	***JTP-only***	***APARAPI-only***	***JTP+APRARAPI***
150	2.3	4.2	5.3
200	2.3	4.2	5.8
250	2.5	4.1	5.8

and the combined approach utilizes both for parallelization.

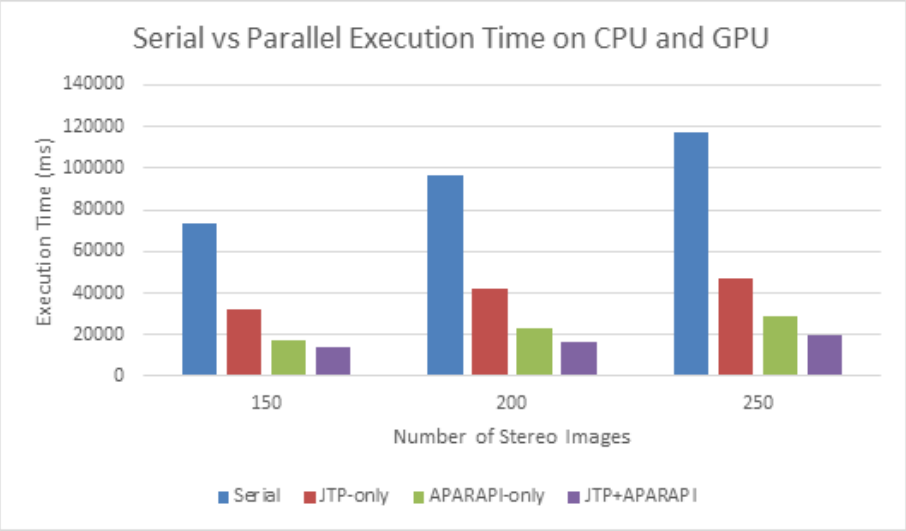


Figure 4.11: Results of Parallelization of Proposed Algorithm.

4.3 Results of Disparity Estimation Error Detection Approach

In this section, both *qualitative* and *quantitative* performance evaluation and comparison of the proposed algorithm with LRC w.r.t both *error detection capability* and *execution time* are presented.

The said *evaluation* is done by considering *different* types of image regions ('all', 'nonocc', and 'disc') of four *rectified* stereo image pairs listed in Table 4.9 and their *ground truth* depth maps from the Middlebury Stereo Vision dataset[1]. Table 4.9 also lists the percentage of total number of left image pixels comprising the 'all' (P_{all}), 'nonocc' (P_{nonocc}), and 'disc' (P_{disc}) regions.

The said *comparison* is performed by *executing* a *popular* technique for detecting estimation errors in initial disparity maps for dense stereo correspondence algorithms, viz. left-right consistency check (LRC), on the *same* four stereo image pairs mentioned

[1] $http://vision.middlebury.edu/stereo/$

above. Then the error detection capability and execution time of LRC and proposed methods were compared.

TABLE 4.9
CHARACTERISTICS OF THE FOUR MIDDLEBURY STEREO IMAGE PAIRS

Stereo Pair	Resolution (pixels)	No. of Disparity Levels	P_{all}	P_{nonocc}	P_{disc}
Tsukuba	384 × 288	16	79.30	77.25	14.28
Venus	434 × 383	20	90.41	88.74	6.34
Teddy	450 × 375	60	97.98	87.50	24.01
Cones	450 × 375	60	96.78	85.29	27.96

4.3.1 Proposed Evaluation Metric

For stereo correspondence algorithms, the *most* widely used evaluation metric to quantify the *error rate* is the percentage of *bad* matching pixels (B) given by (4.4),

$$B = \frac{1}{N} \sum_{(x,y)} \left(\left| d_C\left(x, y\right) - d_T\left(x, y\right) \right| > \delta_d \right) \tag{4.4}$$

where d_C (x, y) and d_T (x, y) are *computed* and *ground truth* disparities, respectively; δ_d denotes disparity error *tolerance*, which is taken as 1.0 as per *published* works [46].

The aforementioned error metric is *more* suitable for performance comparison if and only if *complete* depth maps are obtained as *end*-products of dense stereo correspondence algorithms. However, the proposed method deals with the *intermediate* (preliminary) results, viz. *initial* depth maps generated in the *early* stages of stereo correspondence algorithms. These maps are further *refined* by the algorithm in the later stages. The proposed approach is only trying to *assist* the algorithm by providing a *confidence measure* for the computed set of initial disparities, and a *classification method* for categorizing initial disparities as *good* or *bad* (correct or incorrect).

In order to *quantitatively* evaluate the error detection capability of any algorithm on initial disparity maps using the *traditional* metric given in (4.4), it is required to to compute the percentage of *bad* pixels among the disparity estimates which the algorithm has already classified as *accurate*. Thus, the interpretation of (4.4) reflects the fact that N now represents the total number of pixels (with co-ordinates (x, y)) whose disparity estimates have been *already* classified as *accurate* by the algorithm.

However, such a metric, if used *alone* for performance comparison, would have one *major* drawback: in case, an algorithm is *biased* towards computing a very *stiff* threshold value Ent_Th, it may *always* end up marking a very *few* initial disparities (the most *promising* ones) as *accurate*, ignoring many other (potentially *good*) initial disparity estimates in the process. Moreover, *many* dense stereo correspondence algorithms rely on a *confidence* measure to identify potentially *good* initial disparity estimates, and try

to *reconstruct* the complete depth map from those *reliable* disparities. Now, if such *reliable* disparities are a very *few* in number, it may *adversely* affect the reconstruction process as well as the *final* output, both in terms of *quality* (accuracy) and *efficiency* (computational cost).

Hence, in view of the points discussed above, this thesis proposes a *novel* quality metric, M to quantify and compare the error detection performance of algorithms on initial disparity maps, defined successively through (4.5), (4.6) and (4.7),

$$B_r = \frac{1}{N_r} \sum_{(x_r, y_r)} \left(\left| d_C\left(x_r, y_r\right) - d_T\left(x_r, y_r\right) \right| > \delta_d \right) \tag{4.5}$$

$$P_r = \frac{N_r}{N_a} \tag{4.6}$$

$$M = \frac{P_r}{B_r} \tag{4.7}$$

where N_a represents the total number of pixels of the initial depth map and N_r denotes the total number of pixels (disparity estimates) with co-ordinates (x_r, y_r) that the algorithm has identified as *reliable* or accurate.

Looking at the definition of M in (4.7), it is *intuitive* that *two* ratios are considered in the proposed quality metric: P_r denoting the proportion of total number of initial disparity estimates which the algorithm has marked as *reliable*; B_r denoting the proportion of *truly in*accurate disparity estimates (as compared to the ground truth) among those marked as *accurate* or reliable by the algorithm. The main intention is to propose a quality metric which *favor*s those algorithms that *correctly* classify a *greater* proportion of initial disparity estimates as *accurate*. Thus, it is intuitive to place P_r in the numerator and B_r in the denominator.

4.3.2 Evaluation Methodology

1. Experimental Setup: Both the *proposed* and the left-right consistency check methods (for detecting disparity errors in initial depth maps generated by dense stereo correspondence algorithms) are fully implemented in MATLAB version 2013b. Further, *no* form of task or data parallelism are used *anywhere* in the implementation. All experiments for evaluation and comparison of the two methods are performed on a PC with Intel i7-2600 quad-core, 3.40 GHz CPU, 8 GB RAM running on the Microsoft Windows 7 platform.

2. Generating Initial Disparity Maps: The initial disparity maps for the four rectified stereo image pairs mentioned in Table 4.9 are calculated using a simple

(fixed-sized) block matching algorithm, based on the Sum of Absolute Differences (SAD) cost function and Winner Takes it All (WTA) strategy for determining pixel correspondences. This represents a *typical* method of calculating the initial (dense) disparity estimates for numerous dense stereo correspondence algorithms. Experiments are performed using *moderate* block sizes of 5×5 pixels and 7×7 pixels (separately). Execution time is recorded w.r.t computing *only* the right-to-left depth map in case of proposed method, and *additionally*, the left-to-right depth map in case of the LRC method.

3. Parameter Values: The experiments are performed using *moderate* neighborhood (pixel) sizes of 5×5, 7×7 (separately) for both the proposed and left-right consistency check (LRC) methods. Further, a (standard) *threshold* value of 2.0 is considered for LRC method in all experiments.

4. Evaluation Metrics: In addition to the *proposed* quality metric (M), B_r and P_r, this thesis uses three additional (traditional) metrics, viz. Precision, Recall and Accuracy, as defined in (4.8), (4.9) and (4.10) respectively, to evaluate and compare the proposed method quantitatively with the LRC method.

$$Precision = \frac{tp}{tp + fp} \tag{4.8}$$

$$Recall = \frac{tp}{tp + fn} \tag{4.9}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{4.10}$$

where, 'tp' is the abbreviation for 'true positives', 'tn' for 'true negatives', 'fp' for 'false positives', and 'fn' for 'false negatives'.

In this context, it is meaningful to unambiguously specify the various terms used w.r.t calculation of few traditional metrics which were used (Precision, Recall and Accuracy). Here, *positives* refer to the disparity estimates which are marked as *unreliable* by the proposed method (or the left-right consistency check method), and similarly, all others are referred to as *negatives*, whereas by 'true' this thesis refers to the fact of a positive or negative being classified *correctly* (w.r.t the *ground truth* depth map), and lastly, by 'false', this thesis refers to the case of an *incorrect* classification.

5. Experimentation and Data Collection: Both the *proposed* method and the left-right consistency check method were executed on the *previously* generated initial

disparity maps, and noted the execution time. Further, the proposed quality metric M is calculated, along with as B_r, P_r, Precision, Recall, and Accuracy at the end of each of the executions.

4.3.3 Quantitative Comparison of Outputs

In accordance with the discussed evaluation methodology, the performance data for both proposed method and left-right consistency check method, as obtained from the experiments, are presented in Tables 4.10–4.21, and their execution times are presented in Table 4.22. Tables 4.10–4.15 show the values of all evaluation metrics for experiments where the *size of neighborhood* was taken as 5×5 pixels, and Tables 4.16–4.21 show the results of taking neighborhood size as 7×7 pixels. Further, Tables 4.10, 4.11, 4.16 and 4.17 evaluate the Middlebury stereo pairs for 'all' image regions, while Tables 4.12, 4.13, 4.18 and 4.19 do so for 'nonocc' (non-occluded) ones, and lastly, Tables 4.14, 4.15, 4.20 and 4.21 for 'disc' (depth discontinuities), as per Middlebury's standard nomenclature for (evaluation) dataset image regions[2].

[2] $http://vision.middlebury.edu/stereo/eval/$

TABLE 4.10
PROPOSED METHOD, 'ALL' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	3.87	52.68	13.61	96.13	97.96
Venus	6.43	69.07	10.74	93.57	95.56
Teddy	10.34	48.25	4.67	89.66	95.01
Cones	7.21	49.34	6.84	92.79	96.44

TABLE 4.11
LRC METHOD, 'ALL' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	7.91	83.76	10.59	92.09	93.38
Venus	8.79	84.44	9.60	91.21	92.57
Teddy	15.64	63.56	4.06	84.36	90.06
Cones	15.17	63.99	4.22	84.83	90.29

TABLE 4.12
PROPOSED METHOD, 'NONOCC' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	2.57	53.27	20.76	97.43	98.63
Venus	5.08	69.26	13.62	94.92	96.48
Teddy	7.76	52.32	6.74	92.24	95.94
Cones	4.82	54.28	11.25	95.17	97.38

TABLE 4.13
LRC METHOD, 'NONOCC' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	6.84	84.78	12.40	93.16	94.20
Venus	8.07	85.26	10.56	91.92	93.11
Teddy	14.11	69.64	4.93	85.89	90.17
Cones	13.38	70.72	5.28	86.62	90.53

TABLE 4.14
PROPOSED METHOD, 'DISC' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	5.20	53.70	10.32	94.80	97.21
Venus	18.05	73.00	4.04	81.95	86.82
Teddy	16.18	46.65	2.88	83.81	92.45
Cones	10.61	45.84	4.32	89.39	95.13

TABLE 4.15
LRC METHOD, 'DISC' REGIONS, 5 × 5 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	11.55	82.49	7.14	88.45	90.47
Venus	17.69	76.53	4.32	82.31	86.46
Teddy	24.85	66.52	2.68	75.15	83.47
Cones	18.62	63.67	3.42	81.38	88.14

TABLE 4.16
PROPOSED METHOD, 'ALL' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	4.61	69.46	15.05	95.38	96.79
Venus	2.32	50.58	21.80	97.68	98.83
Teddy	12.41	67.41	5.43	87.59	91.63
Cones	6.04	53.31	8.83	93.96	96.78

TABLE 4.17
LRC METHOD, 'ALL' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	6.95	87.76	12.63	93.05	93.90
Venus	6.10	88.63	14.52	93.90	94.59
Teddy	14.26	65.47	4.59	85.74	90.66
Cones	12.29	67.20	5.46	87.71	91.74

TABLE 4.18
PROPOSED METHOD, 'NONOCC' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	4.86	37.90	7.80	95.14	96.61
Venus	1.20	50.88	42.42	98.80	99.39
Teddy	9.35	72.64	7.77	90.65	93.21
Cones	3.70	58.74	15.89	96.30	97.83

TABLE 4.19
LRC METHOD, 'NONOCC' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	7.09	47.80	6.74	92.91	93.75
Venus	5.30	89.44	16.86	94.69	95.25
Teddy	12.49	71.56	5.73	87.50	91.06
Cones	10.33	74.15	7.18	89.67	92.34

TABLE 4.20
PROPOSED METHOD, 'DISC' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	9.03	64.98	7.19	90.96	94.13
Venus	9.68	46.87	4.84	90.32	95.46
Teddy	23.21	69.37	2.99	76.79	84.00
Cones	12.25	44.66	3.64	87.75	94.53

TABLE 4.21
LRC METHOD, 'DISC' REGIONS, 7 × 7 NEIGHBORHOOD

Stereo Pair	B_r (%)	P_r (%)	M	Precision (%)	Accuracy (%)
Tsukuba	14.05	83.15	5.92	85.95	88.32
Venus	23.82	77.86	3.27	76.18	81.45
Teddy	25.42	68.28	2.69	74.58	82.64
Cones	17.76	65.94	3.71	82.23	88.28

TABLE 4.22
EXECUTION TIMES FOR PROPOSED AND LRC METHOD

Stereo Pair	Proposed (seconds) (5 × 5)	LRC (seconds) (5 × 5)	Proposed (seconds) (7 × 7)	LRC (seconds) (7 × 7)
Tsukuba	2.073	3.584	3.558	6.321
Venus	3.701	6.709	6.445	11.901
Teddy	9.163	17.539	15.987	30.908
Cones	9.137	17.537	15.957	31.008

From experimental results, following *observations* can be made w.r.t performance comparison of proposed method with left-right consistency check (LRC):

1. The error rate, B_r is *significantly* lower in case of the *proposed* method, which clearly indicates its *superiority*.

2. The proposed quality metric, M is *noticeably* higher for the proposed method, implying its *better* performance.

3. For all stereo image pairs, execution time is *much lower* for proposed method. Not only that, the *difference* between execution times of proposed method and left-right consistency check (LRC) increases *drastically* with increase in the *resolution* and *number of disparity levels* of the stereo image pairs. This shows that proposed method is *much* more *scalable* to *high* resolution stereo pairs with *large* number of disparity levels as compared to LRC method.

4. With a *very* few exceptions, the proportion of initial disparity estimates marked as *reliable*, P_r is *appreciably lower* for the *proposed* method than the LRC method.

5. Both *precision* and *accuracy* are *noticeably higher* for the proposed method as compared to the LRC method, while the 'Recall' value was 100.0 for *all* the experiments, *both* in case of the proposed method and the LRC method.

6. From above *two* points, one can infer, while *all* disparity estimates which were classified as *accurate* by both *proposed* and LRC methods are *actually* so, (as evident from a *perfect* recall score for both of them), the proposed method performs *better* in identifying the *inaccurate* ones (as evident from the *higher precision* and *accuracy* scores for *the proposed* method, when compared to LRC).

7. Looking at performance data presented in Tables 4.10–4.21 and summarizing the above points, it can be concluded that the *proposed* method outperforms the LRC method in *all* cases except *two*: for the 'disc' regions of the 'Venus' pair (in terms of *three* metrics: B_r, M, and Precision) using the size of neighborhood 5×5, and

for the 'disc' regions of the 'Cones' pair (*only* in terms of M) for a neighborhood size of 7 $\times$ 7 pixels. Since image regions denoting depth discontinuities *often* coincide with *occluded* regions, thus it may be concluded that in *very few* instances where both the *proposed* and LRC methods have to identify *incorrect* disparity estimates in *occluded* regions, the LRC method may perform *slightly* better than the proposed method.

4.3.4 Qualitative Comparison of Outputs

Fig. 4.12 shows the left image of Tsukuba stereo pair and Fig. 4.13 shows its ground truth depth map from Middlebury dataset. Fig. 4.14 and Fig. 4.15 show the outputs of left-right consistency check and proposed method respectively using a 5 $\times$ 5 neighborhood, whereas Fig. 4.16 and Fig. 4.17 show outputs (respectively) using a 7 $\times$ 7 neighborhood. The disparity estimates marked as *incorrect,* are shown in *black*. Visual comparison of Fig. 4.14 and Fig. 4.15 show that the *proposed* method detects errors in *texture-less* regions (where *most* dense stereo matching methods typically *fail*, like top-right corner of Tsukuba image) with *much greater* accuracy.

Figure 4.12: Left Image of Tsukuba Stereo Image Pair.

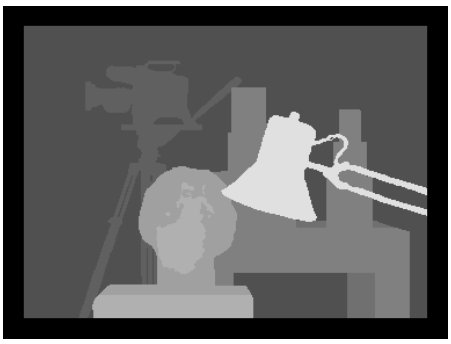


Figure 4.13: Ground Truth Depth Map of Tsukuba.

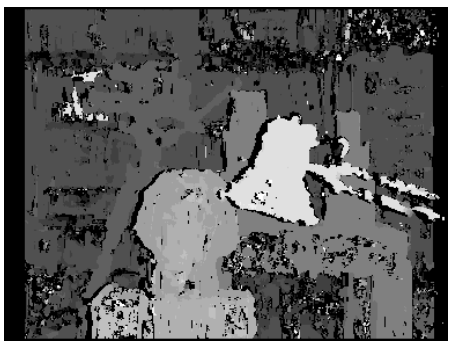


Figure 4.14: Output of LRC Method for Tsukuba (5 $\times$ 5).

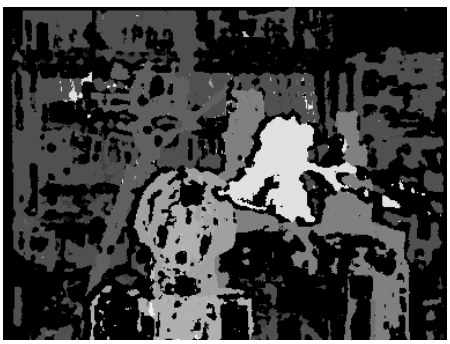


Figure 4.15: Output of the Proposed Method for Tsukuba (5 $\times$ 5).

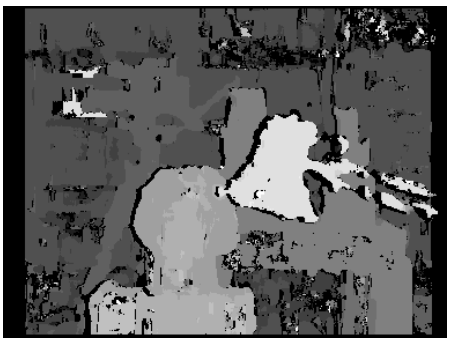


Figure 4.16: Output of LRC Method for Tsukuba (7×7).

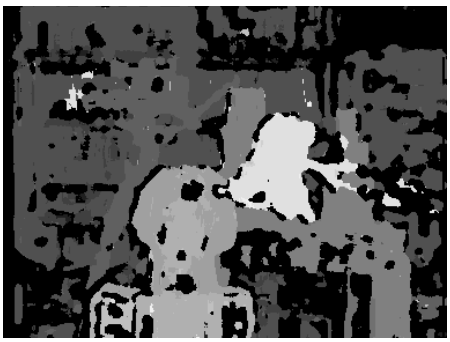


Figure 4.17: Output of the Proposed Method for Tsukuba (7×7).

A *similar* trend is observed for the Venus pair: Fig. 4.18 shows its left image and Fig. 4.19 shows its ground truth depth map, and further, Fig. 4.20 and Fig. 4.21 show the outputs of the left-right consistency check method and proposed method respectively, using a 5×5 neighborhood, whereas Fig. 4.22 and Fig. 4.23 show outputs (respectively) using a 7×7 neighborhood. All the disparity estimates marked as *incorrect* are shown in *black*.

Figure 4.18: Left Image of Venus Stereo Image Pair.

Figure 4.19: Ground Truth Depth Map of Venus.

Figs. 4.24–4.35 show *similar* visual comparisons as shown *above*, for the 'Teddy' and 'Cones' stereo pairs from Middlebury.

4.3.5 Graphical Analysis

Since it is clear from data presented in Tables 4.10–4.21, that the proposed method *outperforms* LRC method, hence, in this subsection, the intention is to investigate, (using graphs) the *extent* by which the proposed method *exceeds* LRC w.r.t both error detection capability and computational speed, for different *types* of image regions ('all', 'nonocc', 'disc'), as well as, using different *size* of neighborhoods (5×5 and 7×7 pixels).

Figure 4.20: Output of LRC Method for Venus (5×5).

Figure 4.21: Output of the Proposed Method for Venus (5×5).

Figure 4.22: Output of LRC Method for Venus (7×7).

Figure 4.23: Output of the Proposed Method for Venus (7×7).

Figure 4.24: Left Image of Teddy Stereo Image Pair.

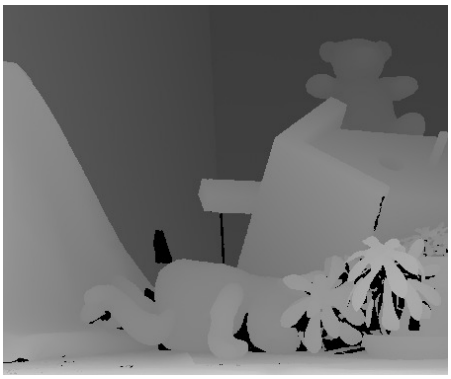


Figure 4.25: Ground Truth Depth Map of Teddy.

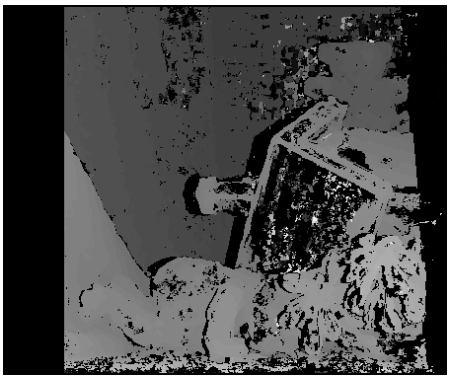


Figure 4.26: Output of LRC Method for Teddy (5 × 5).

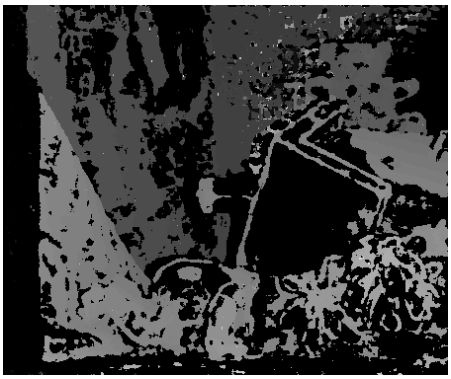


Figure 4.27: Output of the Proposed Method for Teddy (5 × 5).

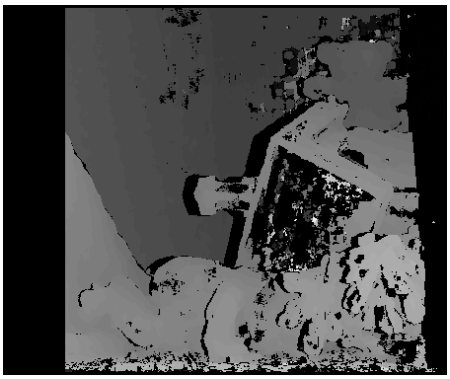


Figure 4.28: Output of LRC Method for Teddy (7 × 7).

Figure 4.29: Output of the Proposed Method for Teddy (7 × 7).

Figure 4.30: Left Image of Cones Stereo Image Pair.

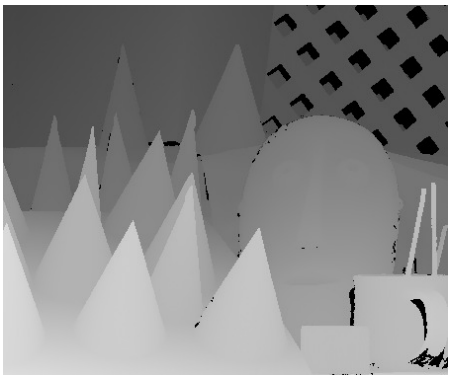


Figure 4.31: Ground Truth Depth Map of Cones.

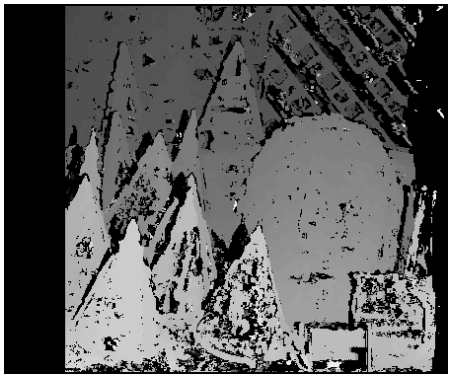


Figure 4.32: Output of LRC Method for Cones (5×5).

Figure 4.33: Output of the Proposed Method for Cones (5×5).

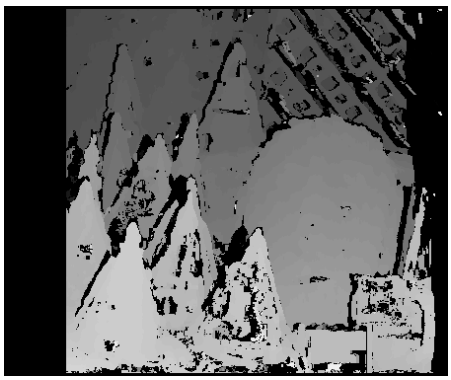


Figure 4.34: Output of LRC Method for Cones (7×7).

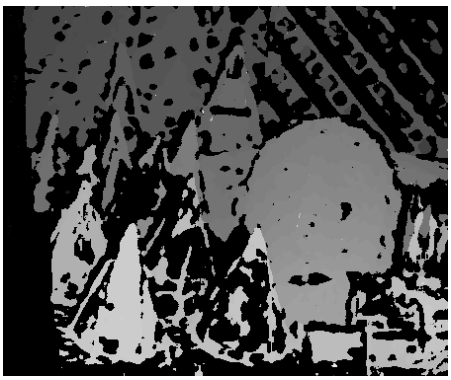


Figure 4.35: Output of the Proposed Method for Cones (7×7).

Thus, the said graphs are plotted based on *difference* in values of *three* metrics, M, Precision (P), and Accuracy (A), between the *proposed* method and the LRC method, for the aforesaid *three* types of image regions and *two* sizes of neighborhood.

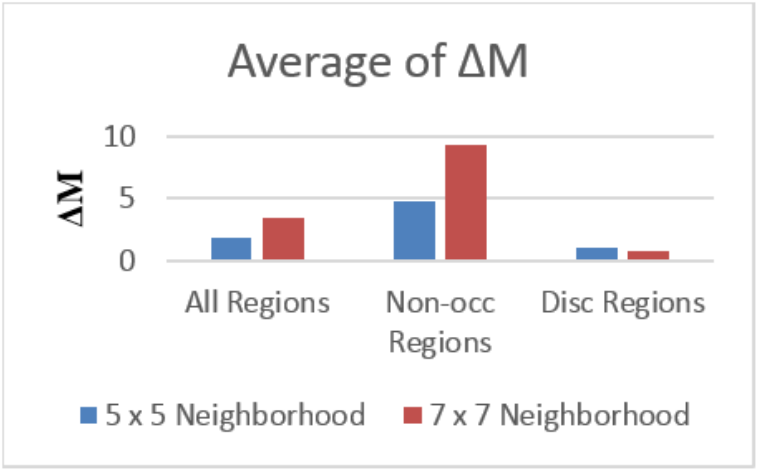


Figure 4.36: Average of ΔM for Proposed Method and LRC.

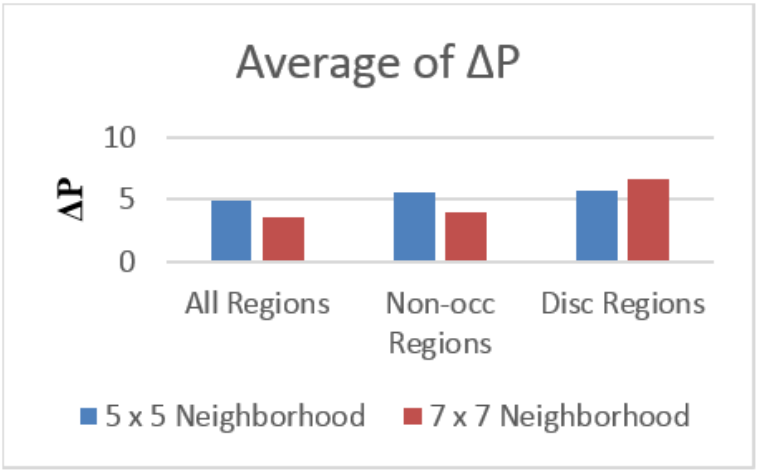


Figure 4.37: Average of ΔP for Proposed Method and LRC.

Figs. 4.36–4.38 clearly show that the *improvement* shown by the *proposed* method over LRC method is of *greater* magnitude for *non-occluded* regions of the test images in terms of ΔM. Similarly, *smaller* improvement is shown by the proposed method for *discontinuous* regions, again in terms of ΔM. Next, another graphical comparison is shown w.r.t *absolute* execution time of the *proposed* and the LRC method, for the abovementioned set of neighborhood sizes and test images. From Fig. 4.39 and Fig. 4.40, it can be clearly seen that the *rate* of *increase* of execution time with *increasing* number of *depth levels* and *image resolution* (Tsukuba–Cones, in an *increasing* order),

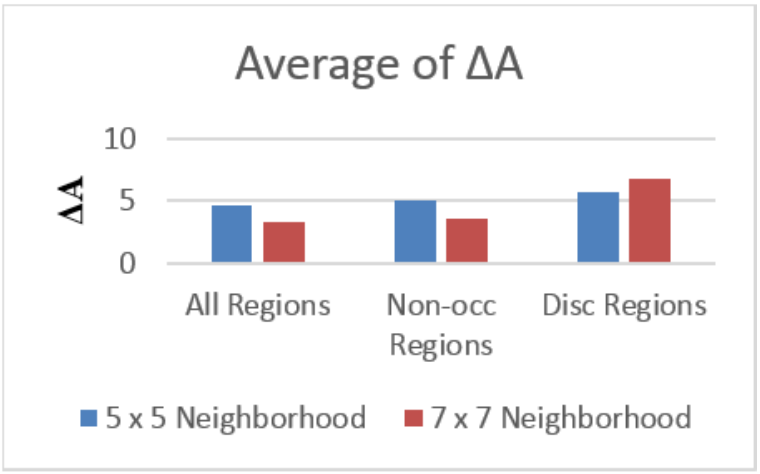


Figure 4.38: Average of ΔA for Proposed Method and LRC.

as well as *increasing* neighborhood size is comparatively *much more* in case of LRC method than the proposed method.

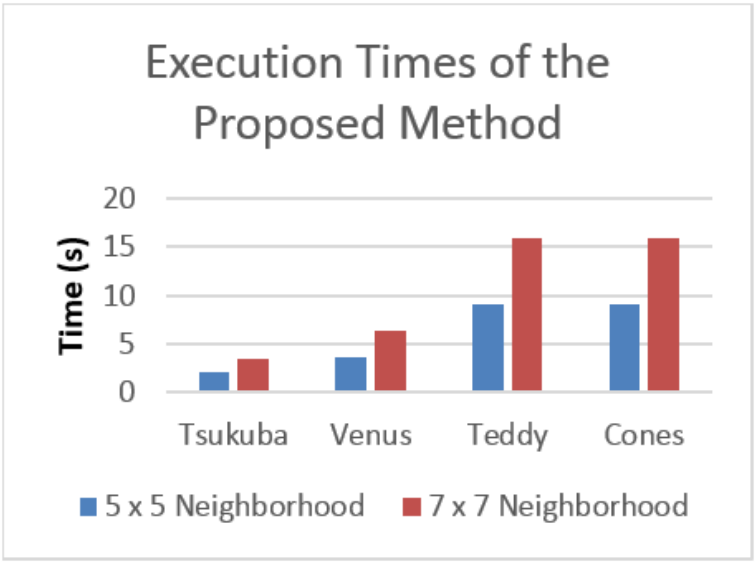


Figure 4.39: Execution Times for Proposed Method.

4.3.6 Outputs of Key Steps of Proposed Method

Lastly, the outputs of certain *important* steps of the proposed method are presented in an attempt to understand *how* they influence the *final* outcome. This will also *possibly* help us to identify performance *bottlenecks*, which may adversely affect the error detection capability of the proposed method.

Figs. 4.41–4.44 show the entropy *difference* images, Ent for all *four* Middlebury stereo image pairs used in the experiments. It may be observed that the *darker* areas

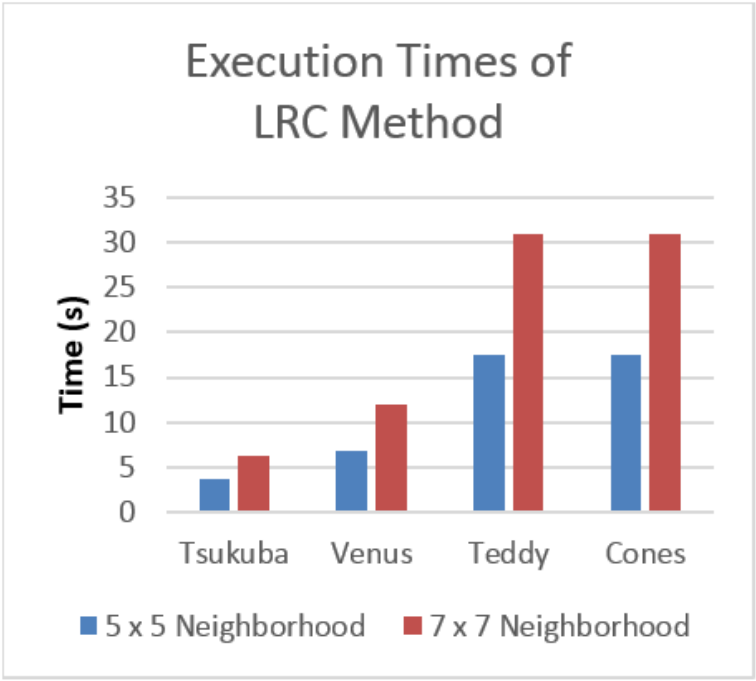


Figure 4.40: Execution Times for LRC Method.

in all entropy difference images correspond to the *black* regions signifying *incorrect* or *unreliable* disparity estimates identified by the proposed method. It may be noted that, a neighborhood size of 5×5 pixels was used for generating all the entropy difference images of Figs. 4.41–4.44.

Figure 4.41: Entropy Difference Image for Tsukuba.

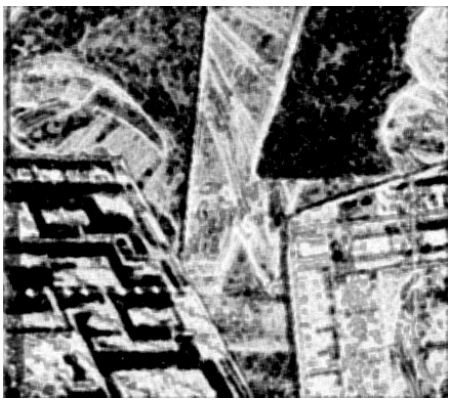


Figure 4.42: Entropy Difference Image for Venus.

Figs. 4.45–4.52 show the 'entropy variations in Ent_D (E) against percentile values' plots (in *red* color) for all four Middlebury stereo image pairs used in the experiments, using *both* 5×5 and 7×7 sizes of neighborhood. The 3^{rd} degree polynomial *approximating* each curve is shown using dotted (*black*) lines.

It is *quite* evident from Figs. 4.45–4.52 that, as the experiments progressively use

Figure 4.43: Entropy Difference Image for Teddy.

Figure 4.44: Entropy Difference Image for Cones.

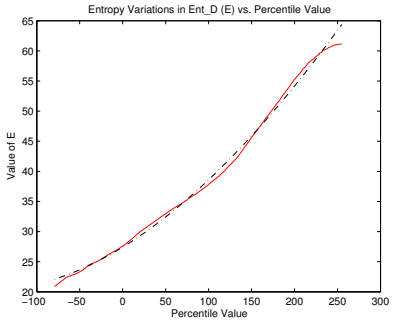


Figure 4.45: Entropy Variations in Ent_D versus Percentiles for Tsukuba (5×5).

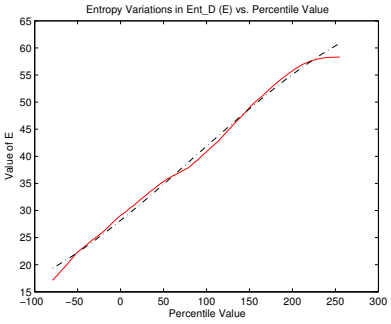


Figure 4.46: Entropy Variations in Ent_D versus Percentiles for Tsukuba (7×7).

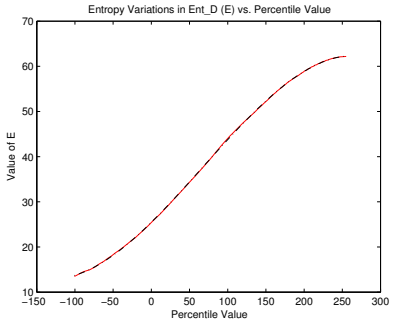


Figure 4.47: Entropy Variations in Ent_D versus Percentiles for Venus (5×5).

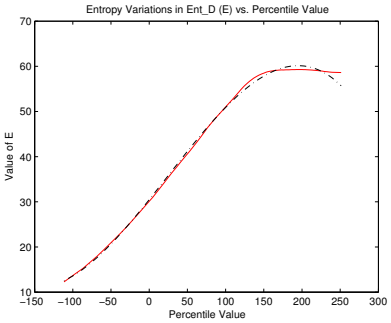


Figure 4.48: Entropy Variations in Ent_D versus Percentiles for Venus (7×7).

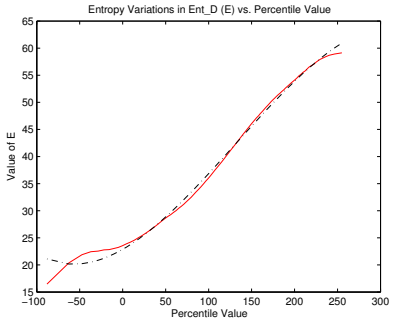


Figure 4.49: Entropy Variations in Ent_D versus Percentiles for Teddy (5×5).

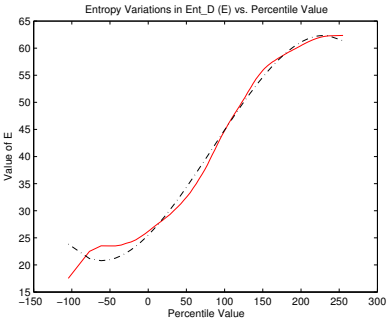


Figure 4.50: Entropy Variations in Ent_D versus Percentiles for Teddy (7×7).

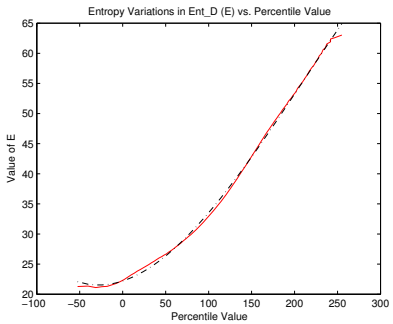


Figure 4.51: Entropy Variations in Ent_D versus Percentiles for Cones (5×5).

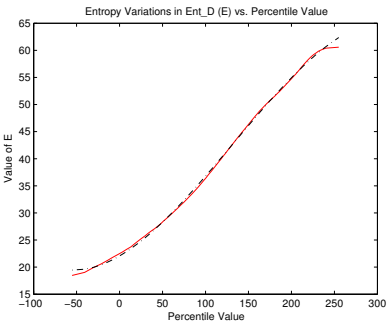


Figure 4.52: Entropy Variations in Ent_D versus Percentiles for Cones (7×7).

higher percentile values (P_i), *initially* the curve rises *steeply* and then, from *some* point onward, there is a *slack* in its rate of ascent. This happens because, at *that* point, *most* of the "unstable" regions of *incorrect* disparity estimates with *high* variations in disparity values (and correspondingly, *entropies* thereof) have *already* been covered; *remaining* areas are relatively "stable", and thus, do not contribute *significantly* in increasing the aforementioned standard deviation (E).

Next, the pixel *threshold* (Ent_Th) values, as determined by the proposed threshold detection technique, are shown, for all results presented in Figs. 4.45–4.52, along with their (nearest) corresponding percentile (P_i) no.'s and values, in Table 4.23.

TABLE 4.23
LOCATION OF INFLECTION POINTS FOR DIFFERENT SIZES OF NEIGHBORHOOD

Stereo Pair	Ent_Th (5 × 5)	Nearest Percentile (No., Val.) (5 × 5)	Ent_Th (7 × 7)	Nearest Percentile (No., Val.) (7 × 7)
Tsukuba	117.65	50, 117.65	86.49	34, 87.08
Venus	67.25	31, 67.10	141.24	50, 141.24
Teddy	130.59	51, 130.90	84.42	31, 84.07
Cones	125.81	50, 125.81	130.92	46, 130.87

From the data presented in Table 4.23, it can be inferred that, *most* inflection points occur *around* the 30^{th} and 50^{th} percentile. Cases where Ent_Th values *exactly* coincide with the 50^{th} percentile (value) are cases where the inflection point determined by proposed method, lay *outside* the percentile range of P_{20}–P_{80}, and hence it was taken as P_{50} by the proposed method.

Figs. 4.53–4.59 show the generated (initial) depth maps for the *Teddy* image (5 × 5 pixels neighborhood is used), showing only *those* pixels that correspond to the *entropy difference* values *greater* than P_i). *Seven* P_i values, viz. the 20^{th}, 30^{th}, 40^{th}, 50^{th}, 60^{th}, 70^{th} and the 80^{th} percentile were considered as *representatives* of the *entire* range of *allowed* percentile values (20^{th}–80^{th} percentile).

Figure 4.53: Teddy Depth Map for 20^{th} Percentile.

Figure 4.54: Teddy Depth Map for 30^{th} Percentile.

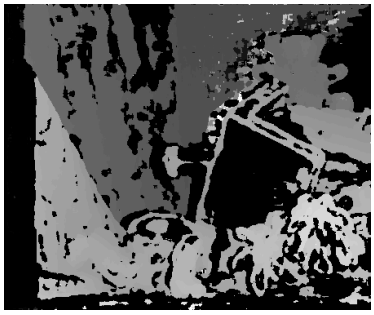


Figure 4.55: Teddy Depth Map for 40th Percentile.

Figure 4.56: Teddy Depth Map for 50th Percentile.

Figure 4.57: Teddy Depth Map for 60th Percentile.

Figure 4.58: Teddy Depth Map for 70th Percentile.

Figure 4.59: Teddy Depth Map for 80th Percentile.

From Figs. 4.53–4.59, it can be easily observed that, with *increasing* values of percentiles (P_i), there is a *steady decline* in number of *erroneous* disparity estimates among depth map pixels. But, along with this, there is *also* a steady decline in the *total* count of disparity estimates (including *possibly accurate* estimates) in the depth maps, as take *higher* percentile values for Ent_Th. Thus, the *primary* aim of the proposed *threshold determination* step (as is evident from *this* discussion) should be to derive the *optimal threshold* value that *retains* as many *reliable* disparities, and *rejects* as many *unreliable* disparities, as possible; the depth map *filtered*, removing *as much* unreliable estimates as possible.

4.4 Comparison of Depth Estimation w.r.t 2D, 3D & Kinect Sensor

For quantitative comparison of depth estimation using 2D and 3D image processing, and Kinect depth camera, we need a device called LIDAR (LIght Detection And Ranging). LIDAR measures distance (or *depth*) by illuminating a target with a laser and analysing the reflected light. Subsequently, the LIDAR builds the complete (*ground truth*) depth map of the scene by analysing the reflected laser beams from each and every point in the scene. However, since the LIDAR device is extremely costly and difficult to procure, this thesis presents a (mostly) qualitative comparison of depth estimation w.r.t 2D, 3D & Kinect. In subsequent *grey-scale* depth maps, lighter shades denote farther distances.

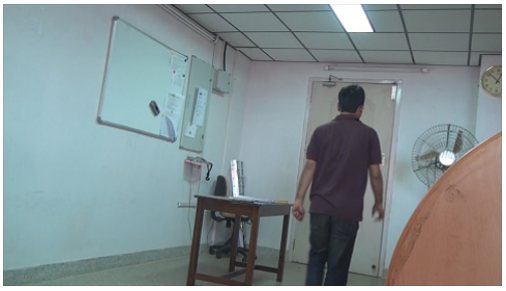


Figure 4.60: Left Image of Stereo Pair captured using 3D Camcorder.

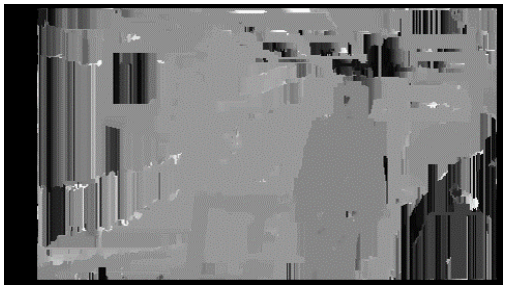


Figure 4.61: Depth Map of Left Image obtained using Proposed Method.

Figs. 4.60, 4.61 and 4.62 show the results of (qualitative) comparison of the output of the proposed stereo depth extraction algorithm with the depth of a similar scene extracted by the Kinect depth camera. As can be clearly seen, Kinect is able to compute depths only for a certain range of distances from the sensor, whereas, the proposed stereo depth extraction algorithm determines the depth for almost the entire scene, except a few rows and columns of pixels near the borders. Concretely, the percentage of *unknown* depth values was 27.5% for the proposed stereo (3D) depth estimation method, while it was 42.7% for Kinect.

Figure 4.62: Kinect Depth Map (left) and Kinect RGB Image (right).

Figure 4.63: Left Image of Stereo Pair.

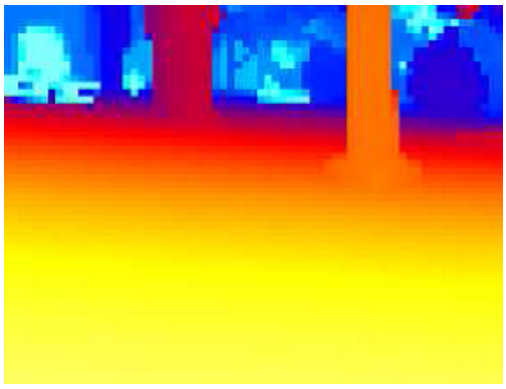


Figure 4.64: Left Image Ground Truth.

Figure 4.65: Proposed 3D Method's output Depth Map.

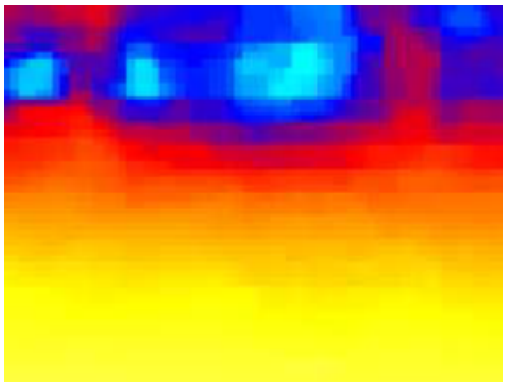


Figure 4.66: Authors' 2D Method's output Depth Map.

Figure 4.67: Mapping Depth Map Grey Levels & Color Codings to Actual Distances.

Lastly, the qualitative comparison of our proposed method with state-of-the-art work done by the authors in [47] w.r.t depth estimation based on stereo and mono cues, is presented in Figs. 4.63 through 4.67.

5 CONCLUSION AND FUTURE WORK

As part of this thesis, a novel stereo depth estimation method is proposed. The proposed method uses only 18% pixels of either the left or the right image, and outperforms traditional methods like SAD and NCC by up to 33.6% and a recent method developed by Zhen Zhang et al. [12] by up to 6.1%. However, in case of some of the most competitive algorithms like Graph-cut based methods [44], using occlusion handling techniques, better accuracy can be obtained, sometimes at the cost of increased computational complexity. Depth-based Gaussian blurring of image regions is also performed as per depths of users' non-interest.

However, the use of luminance alone in the segmentation step of the proposed method can reduce performance in the presence of shadows. Also, the proposed disparity map reconstruction method may end up altering the shapes of objects, especially in dimly lit regions of the image. However, it is also shown that, despite the proposed algorithm's inaccurate depth estimates in texture-less and low-illumination regions, its blurring performance is not affected. Future work will focus on improving the depth estimates and the depth-map reconstruction technique in texture-less and low-illumination areas.

Moreover, future work can be extended to comparison of the time-complexity of the proposed depth extraction algorithm with the state-of-art algorithms such as those based on Belief Propagation, Graph Cuts and Markov Random Fields.

This thesis also proposes a novel confidence measure for quantifying the reliability of initial disparity estimates produced in preliminary stages of dense stereo correspondence algorithms. A novel classification technique for categorizing the said initial disparity estimates as correct or incorrect is also proposed, based on the values of that confidence measure. Furthermore, a novel evaluation metric is also proposed for evaluation and comparison of the error detection capability of the proposed method with other methods. The performance of the proposed method is compared with left-right consistency check method (which serves a similar purpose as the proposed method). For this comparison, the proposed novel quality metric and traditional metrics such as percentage of bad matching pixels, precision, recall, and accuracy are used. Experiments are performed based on four rectified stereo image pairs from the Middlebury stereo vision dataset. Experimental results show that in majority cases, our proposed method fare significantly better, both in terms of improved error detection capability

and reduced execution time (by a difference of up to 25.6 in terms of quality metrics, and 94.3% in terms of execution time).

One limitation of the proposed work stems from the fact that, in a very few cases, the curve fitting (which uses the Least Squares method and employs a 3rd degree polynomial) fails to properly approximate complex entropy variations in the entropy map of initial disparity map, Ent_D (E) with percentile values of entropy difference image (Ent). In such cases, the proposed method determines a sub-optimal threshold value for classifying the initial disparity estimates as correct or incorrect.

Thus, future work may focus on improving the threshold determination stage of our proposed method and thereby making it more robust w.r.t threshold detection.

Further, the proposed work can be extended to Web-based and mobile Android-based applications, like that of Google Camera.

References

[1] Popkin, T.; Cavallaro, A.; Hands, D., "Efficient depth blurring with occlusion handling," Image Processing (ICIP), 2011 18th IEEE International Conference on , vol., no., pp.2585,2588, 11-14 Sept. 2011, doi: 10.1109/ICIP.2011.6116193

[2] Sungkil, L.; Jounghyun, K. G.; Seungmoon, C., "Real-Time Depth-of-Field Rendering Using Anisotropically Filtered Mipmap Interpolation," Visualization and Computer Graphics, IEEE Transactions on , vol.15, no.3, pp.453,464, May-June 2009, doi: 10.1109/TVCG.2008.106

[3] Popkin, T.; Cavallaro, A.; Hands, D., "Image Coding Using Depth Blurring for Aesthetically Acceptable Distortion," Image Processing, IEEE Transactions on , vol.20, no.11, pp.3039,3050, Nov. 2011, doi: 10.1109/TIP.2011.2145385

[4] Kuthirummal, S.; Nagahara, H.; Changyin, Z.; Nayar, S. K., "Flexible Depth of Field Photography," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.33, no.1, pp.58,71, Jan. 2011, doi: 10.1109/TPAMI.2010.66

[5] Popkin, T.; Cavallaro, A.; Hands, D., "Distance blurring for space-variant image coding," Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on , vol., no., pp.665,668, 19-24 April 2009, doi: 10.1109/ICASSP.2009.4959671

[6] Xiaojun, H.; Lianghao, W.; Junjun, H.; Dongxiao, L.; Ming, Z., "A Depth Extraction Method Based on Motion and Geometry for 2D to 3D Conversion," Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on , vol.3, no., pp.294,298, 21-22 Nov. 2009, doi: 10.1109/IITA.2009.481

[7] Xi, Y.; You, Y.; Guihua, E.; Qionghai, D., "Depth map generation for 2D-to-3D conversion by limited user inputs and depth propagation," 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011 , vol., no., pp.1,4, 16-18 May 2011, doi: 10.1109/3DTV.2011.5877167

[8] Yeong-Kang, L.; Yu-Fan, L.; Ying-Chang, C., "An Effective Hybrid Depth-Generation Algorithm for 2D-to-3D Conversion in 3D Displays," Display Technology, Journal of , vol.9, no.3, pp.154,161, March 2013, doi: 10.1109/JDT.2012.2224637

[9] Chao, L.; Christopher, L., "Depth map estimation from motion for 2D to 3D conversion," Electro/Information Technology (EIT), 2012 IEEE International Conference on , vol., no., pp.1,4, 6-8 May 2012, doi: 10.1109/EIT.2012.6220749

[10] Lazaros, N.; Sirakoulis, G.; Gasteratos, A., "Review of Stereo Vision Algorithms: From Software to Hardware," Optomechatronics, International Journal of, vol.2, no.4, pp.435,462, 2008

[11] Jun, X.; Linyuan, X.; Liqun, L.; Zhentao, Z., "A segment-based stereo matching method with ground control points," Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on , vol.3, no., pp.306,309, 17-18 July 2010, doi: 10.1109/ESIAT.2010.5568363

[12] Zhen, Z.; Yifei, W.; Dahnoun, N., "A novel algorithm for disparity calculation based on stereo vision," Education and Research Conference (EDERC), 2010 4th European, vol., no., pp.180, 184, 1-2 Dec. 2010

[13] Sunyoto, H.; Van der Mark, W.; Gavrila, D. M., "A comparative study of fast dense stereo vision algorithms," Intelligent Vehicles Symposium, 2004 IEEE, vol., no., pp.319, 324, 14-17 June 2004, doi: 10.1109/IVS.2004.1336402

[14] Tippetts, B.; Lee, D.; Lillywhite, K.; Archibald, J., "Review of stereo vision algorithms and their suitability for resource-limited systems," Real-Time Image Processing, Journal of, Springer-Verlag, pp.1,21, 2013

[15] Hirschmuller, H.; Scharstein, D., "Evaluation of Cost Functions for Stereo Matching," Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, vol., no., pp.1, 8, 17-22 June 2007, doi: 10.1109/CVPR.2007.383248

[16] Tombari, F.; Mattoccia, S.; Di Stefano, L.; Addimanda, E., "Classification and evaluation of cost aggregation methods for stereo correspondence," Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, vol., no., pp.1, 8, 23-28 June 2008, doi: 10.1109/CVPR.2008.4587677

[17] Abdollahifard, M.; Faez, K.; Pourfard, M., "Fast stereo matching using two stage color-based segmentation and dynamic programming," Mechatronics and its Applications, 2009. ISMA '09. 6th International Symposium on, vol., no., pp.1, 6, 23-26 March 2009, doi: 10.1109/ISMA.2009.5164848

[18] Changick K., "Segmenting a low-depth-of-field image using morphological filters and region merging," Image Processing, IEEE Transactions on , vol.14, no.10, pp.1503,1511, Oct. 2005, doi: 10.1109/TIP.2005.846030

[19] Xiaobing, W.; Yonghong, S.; Yuanlin, Z., "Natural Scene Text Detection with Multi-channel Connected Component Segmentation," Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, vol., no., pp.1375, 1379, 25-28 Aug. 2013, doi: 10.1109/ICDAR.2013.278

[20] Vishwanath, N.; Somasundaram, S.; Ravi, M. R. R.; Nallaperumal, N. K., "Connected component analysis for Indian license plate infra-red and color image character segmentation," Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on , vol., no., pp.1,4, 18-20 Dec. 2012, doi: 10.1109/ICCIC.2012.6510323

[21] Zirari, F.; Ennaji, A.; Nicolas, S.; Mammass, D., "A Document Image Segmentation System Using Analysis of Connected Components," Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, vol., no., pp.753, 757, 25-28 Aug. 2013, doi: 10.1109/ICDAR.2013.154

[22] Min, L.; Xiaolin, Z.; Xiaoping, W.; Hongyan, L.; Shaoxiang, Z.; Liwen, T., "Segmentation of brain tissue based on connected component labeling and mathematic morphology," Biomedical Engineering and Informatics (BMEI), 2011 4th International Conference on , vol.1, no., pp.482,485, 15-17 Oct. 2011, doi: 10.1109/BMEI.2011.6098294

[23] Moftah, H. M.; ella Hassanien, A.; Shoman, M., "3D brain tumor segmentation scheme using K-mean clustering and connected component labeling algorithms," Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on , vol., no., pp.320,324, Nov. 29 2010-Dec. 1 2010, doi: 10.1109/ISDA.2010.5687244

[24] Bellala, B., Fatima, Z.; Souami, F., "Color image segmentation by a genetic algorithm based clustering and Connected Component Labeling," Microelectronics (ICM), 2012 24th International Conference on, vol., no., pp.1, 4, 16-20 Dec. 2012, doi: 10.1109/ICM.2012.6471432

[25] Kang-Sun C., "Hierarchical block-based disparity estimation," Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on, vol., no., pp.493, 494, 2-5 Oct. 2012, doi: 10.1109/GCCE.2012.6379668

[26] Shiping, Z.; Yang, Y., "Virtual View Rendering Based on Self-adaptive Block Matching Disparity Estimation," Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on , vol., no., pp.947,950, 23-25 Aug. 2012, doi: 10.1109/ICICEE.2012.251

[27] Zeng-Fu, W.; Zhi-Gang, Z., "A region based stereo matching algorithm using cooperative optimization," Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, vol., no., pp.1, 8, 23-28 June 2008, doi: 10.1109/CVPR.2008.4587456

[28] Di, L.; Yu, D., "A two-step stereo correspondence algorithm based on combination of feature-matching and region-matching," Strategic Technology (IFOST), 2013 8th International Forum on , vol.2, no., pp.51,55, June 28 2013-July 1 2013, doi: 10.1109/IFOST.2013.6616858

[29] Shih, H.-C.; Hsiao, H.-F., "A depth refinement algorithm for multiview video synthesis," in Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, March 2010, pp. 742–745

[30] Xu, X.; Po, L.-M.; Cheung, K.-W.; Ng, K.-H.; Wong, K.-M.; Ting, C.-W., "A foreground biased depth map refinement method for dibr view synthesis," in Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, March 2012, pp. 805–808

[31] Chang, Y.-L.; Tsai, Y.-P.; Chang, T.-H.; Chen, Y.-R.; Lei, S., "A depth map refinement algorithm for 2d-to-3d conversion," in Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, March 2012, pp. 1437–1440

[32] Vijayanagar, K.; Loghman, M.; Kim, J., "Refinement of depth maps generated by low-cost depth sensors," in SoC Design Conference (ISOCC), 2012 International, Nov 2012, pp. 355–358

[33] Luo, H.-L.; Shen, C.-T.; Chen, Y.-C.; Wu, R.-H.; Hung, Y.-P., "Automatic multi-resolution joint image smoothing for depth map refinement," in Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on, Nov 2013, pp. 284–287

[34] Tereki, B.; Oittinen, P.; Szirmay-Kalos, L., "Informational aesthetic measure for 3d stereoscopic imaging," in Hungarian Association for Image Processing and Pattern Recognition, 2013. (KEPAF 2013). Proceedings. 9th Conference of, Jan 2013, pp. 57–66. [Online]. Available: http://kepaf2013.mik.uni-pannon.hu/proceedings/ pdfs/F01_05.pdf

[35] Smisek, J.; Jancosek, M.; Pajdla, T., "3D with Kinect," Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on , vol., no., pp.1154,1160, 6-13 Nov. 2011, doi: 10.1109/ICCVW.2011.6130380

[36] Macknojia, R.; Chavez-Aragon, A.; Payeur, P.; Laganiere, R., "Experimental characterization of two generations of Kinect's depth sensors," Robotic and Sensors Environments (ROSE), 2012 IEEE International Symposium on , vol., no., pp.150,155, 16-18 Nov. 2012, doi: 10.1109/ROSE.2012.6402634

[37] Han, J.; Shao, L.; Xu, D.; Shotton, J., "Enhanced Computer Vision with Microsoft Kinect Sensor: A Review," Cybernetics, IEEE Transactions on , vol.PP, no.99, pp.1,1, 0, doi: 10.1109/TCYB.2013.2265378

[38] Essmaeel, K.; Gallo, L.; Damiani, E.; De Pietro, G.; Dipanda, A., "Temporal Denoising of Kinect Depth Data," Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on , vol., no., pp.47,52, 25-29 Nov. 2012, doi: 10.1109/SITIS.2012.18

[39] Wei, Y.; Tsuhan, C.; Franchetti, F.; Hoe, J. C., "High Performance Stereo Vision Designed for Massively Data Parallel Platforms," Circuits and Systems for Video Technology, IEEE Transactions on , vol.20, no.11, pp.1509,1519, Nov. 2010, doi: 10.1109/TCSVT.2010.2077771

[40] Takaya, K., "Dense stereo disparity map for video by sub-pixel dynamic time warp algorithm," Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on , vol., no., pp.1,4, 2-5 May 2010, doi: 10.1109/CCECE.2010.5575257

[41] Lin, D.; Xiaohuang, H.; Quang, N.; Blackburn, J.; Rodrigues, C.; Huang, T.; Do, M. N.; Patel, S. J.; Hwu, W.-M.W., "The parallelization of video processing," Signal Processing Magazine, IEEE , vol.26, no.6, pp.103,112, November 2009, doi: 10.1109/MSP.2009.934116

[42] Tkalcic, M.; Tasic, J. F., "Colour spaces: perceptual, historical and applicational background," EUROCON 2003. Computer as a Tool. The IEEE Region 8, vol.1, no., pp.304, 308 vol.1, 22-24 Sept. 2003, doi: 10.1109/EURCON.2003.1248032

[43] Docampo, J.; Ramos, S.; Taboada, G. L.; Exposito, R. R.; Tourino, J.; Doallo, R., "Evaluation of Java for General Purpose GPU Computing," Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on , vol., no., pp.1398,1404, 25-28 March 2013. DOI= 10.1109/WAINA.2013.234

[44] Kolmogorov, V.; Zabih, R., "Computing visual correspondence with occlusions using graph cuts," Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol.2, no., pp.508, 515 vol.2, 2001, doi: 10.1109/ICCV.2001.937668

[45] Miled, W.; Pesquet, J. C., "Disparity Map Estimation Using A Total Variation Bound," Computer and Robot Vision, 2006. The 3rd Canadian Conference on, vol., no., pp.48, 48, 07-09 June 2006, doi: 10.1109/CRV.2006.28

[46] Scharstein, D.; Szeliski, R.; Zabih, R., "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on, vol., no., pp.131, 140, 2001, doi: 10.1109/SMBV.2001.988771

[47] Ashutosh, S.; Jamie, S.; Andrew, Y. Ng., "Depth estimation using monocular and stereo cues," In Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07), 2007. Rajeev Sangal, Harish Mehta, and R. K. Bagga (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2197-2203

Research Publications (Accepted and Communicated)

Conference Publications

1. Subhayan Mukherjee, Ram Mohana Reddy Guddeti, "A Hybrid Algorithm for Disparity Calculation From Sparse Disparity Estimates Based on Stereo Vision" –IISc Bangalore, India, IEEE; 10th International Conference on Signal Processing and Communications (SPCOM 2014), Jul 22–25, 2014.
2. Subhayan Mukherjee, Ram Mohana Reddy Guddeti, "A Novel Segmentation-Based Algorithm for Stereo Depth Extraction using Sparse Disparity Estimates" –Beijing Jiaotong University, China, IEEE; 12th International Conference on Signal Processing (ICSP 2014), Oct 19–23, 2014.
3. Subhayan Mukherjee, Ram Mohana Reddy Guddeti, "A Novel Entropy-based Approach for Stereo Matching Error Detection" – Indian Institute of Technology Bombay, 21st IEEE National Conference on Communications (NCC 2015), Feb 27–Mar 1, 2015 (**under review**).

Journal Publications

1. Subhayan Mukherjee and Ram Mohana Reddy Guddeti, "Depth-Based Selective Blurring in Stereo Images Using Accelerated Framework", Springer-Verlag Journal "3D Research", Volume 5, Issue 3, Published Online: 25 June 2014, **DOI**: 10.1007/s13319-014-0014-7.
2. Subhayan Mukherjee, Ram Mohana Reddy Guddeti, "A Novel Entropy-based Approach for Detecting Errors in Dense Stereo Disparity Estimation" – IEEE Transactions on Circuits and Systems for Video Technology (**undergoing major revision**).

3D Res (2014) 5:14
DOI 10.1007/s13319-014-0014-7

3DR EXPRESS

Depth-Based Selective Blurring in Stereo Images Using Accelerated Framework

Subhayan Mukherjee · Ram Mohana Reddy Guddeti

Received: 13 March 2014 / Revised: 19 May 2014 / Accepted: 21 May 2014

Abstract We propose a hybrid method for stereo disparity estimation by combining block and region-based stereo matching approaches. It generates dense depth maps from disparity measurements of only 18 % image pixels (left or right). The methodology involves segmenting pixel lightness values using fast K-Means implementation, refining segment boundaries using morphological filtering and connected components analysis; then determining boundaries' disparities using sum of absolute differences (SAD) cost function. Complete disparity maps are reconstructed from boundaries' disparities. We consider an application of our method for depth-based selective blurring of non-interest regions of stereo images, using Gaussian blur to de-focus users' non-interest regions. Experiments on Middlebury dataset demonstrate that our method outperforms traditional disparity estimation approaches using SAD and normalized cross correlation by up to 33.6 % and some recent methods by up to 6.1 %. Further, our method is highly parallelizable using CPU–GPU framework based on Java Thread Pool and APARAPI with speed-up of 5.8 for 250 stereo video frames (4,096 × 2,304).

S. Mukherjee (✉) · R. M. R. Guddeti
National Institute of Technology Karnataka, Surathkal, Mangalore, India
e-mail: subhayan001@gmail.com

R. M. R. Guddeti
e-mail: profgrmreddy@nitk.ac.in

Keywords Stereo depth estimation · Sparse disparity estimates · Java Thread Pool · APARAPI · Morphological filter · Connected components analysis

1 Introduction

Stereo vision based disparity calculation is an important research problem in computer vision with applications such as robotic vision, 3D scene reconstruction, object detection and tracking etc. [1]. Here, the main challenge is to obtain an accurate depth information of a scene by comparing the pixels of left and right images of that scene. It is challenging because individual pixels contain only the colour and spatial information and represent the low level image features [2]. So, to effectively compare a stereo image pair, we need to develop the framework to identify appropriate high level features and thereby comparing these features efficiently with reasonable accuracy and speed.

Disparity of a pixel varies inversely as the distance of a point in a scene (that the pixel represents) from the 3D camera. A disparity or depth map contains the mapping of each pixel of an image to its corresponding disparity. A cost function is used to quantify the similarity between pixels of the left and right images of a stereo image pair. Then, pixels' disparities are estimated by relative displacement of the corresponding matching counterparts across the image pair.

Published online: 25 June 2014

Table 1 Comparison of Basic Depth Estimation Approaches

	Block-based method	Region-based method
Approach	Depth estimation based on information contained in pixels and their surroundings	Depth estimation based on optimal value of cost function for entire image regions of pixels with similar disparities
Strength	High resolution depth maps	Sharp edges on depth maps
Weakness	Need to determine optimal block sizes for different images, or even different regions of same image for creating accurate depth maps	Unsuitable for finding the disparities that are gradually changing, as all pixels constituting even a large region may share a constant disparity

For finding the matching pixels between the stereo image pair, we have two basic approaches: block-based and region-based; the strengths and weaknesses of these two basic approaches are summarized in Table 1 [3]. The key challenge in block-based methods is the determination of an optimal block size, and in region-based methods is the detection of gradually changing disparities. In estimating the stereo disparities using block-based methods, small block sizes produce sharp edges on the depth map but generates errors in the homogeneous regions since only a very limited local information is utilized. On the other hand, large block sizes provide good performance in homogeneous areas but yield very inaccurate disparity measurements along objects' edges. This is because, pixels inside the same block may have varying disparities. For region-based methods, apart from the fact that they not only create a limited number of depth levels but also require accurate results of segmentation, which makes them inherently computationally intensive and time-consuming. This motivates us to propose a novel hybrid method by combining these block-based and region-based approaches and thereby overcoming their individual limitations.

Another way to classify stereo depth estimation approaches is sparse versus dense. In the sparse, feature-based approaches to stereo, only a subset of image pixels (i.e. vertical edge pixels) are matched, with an aim to meet real time processing requirements. However, sparse disparity maps may often contain insufficient data points for supporting object segmentation which is an important prerequisite to subsequent understanding of the scene. On the other hand, dense stereo matching can improve this bottom-up processing chain significantly [4]. But, this dense stereo matching may be computationally complex.

In this context, it is also helps to quote a few very important observations about stereo depth estimation approaches (in general) made by several authors, time and again, including the very recent one by Tippetts et al. [5], that "The number of pixels that each image contains increases the number of computations required to match it with any number of possible matches, making the correspondence problem a computationally complex one that severely limits the speed at which one can obtain results. Most of the time, accuracy and speed are pitted against each other, making it more difficult to obtain both at the same time. In any given instance of the stereo vision problem, various questions arise; is one of these two attributes more desirable? How can the trade-off between the two be minimized? What options already exist, and how do they compare to each other?"

The authors opined that target applications should drive the algorithm designers' decision of striking a balance between speed and accuracy. They also observed that authors should not solely rely on advances in hardware technology to bring about radical increments in processing speeds of stereo vision algorithms. Rather, they must actively engage in improving algorithms to run faster.

Hence, throughout our algorithm design process, we make an attempt to give due consideration to both these factors mentioned, viz. accuracy and speed. As an example, we make an attempt to combine the speed of the sparse, feature-based approaches to stereo with the accuracy of the dense stereo vision methods.

For achieving the above cited goals, we propose a novel hybrid method of disparity estimation by segmenting pixels' lightness values by a fast histogram-based K-Means implementation and subsequent refinement of segment boundaries using morphological filters and connected components analysis. We provide additional motivations behind our design choices in the Sect. 2

The core idea behind our proposed method is to use a scalable, block-based approach to estimate the disparities of only pixels lying on the refined segment boundaries, and get reliable disparity estimates in less

number of computations. So, our method is scalable to high resolution stereo image pairs. Then, the proposed disparity map reconstruction method is used for estimating the disparities of pixels lying inside segment boundaries and the details are given in Sect. 3. Finally, we consider an application of our algorithm for depth-based selective blurring of stereo images, using a Gaussian kernel to blur image regions corresponding to (supplied) depth range(s) of user's non-interest. Thus, proposed methodology for the selective blurring phase is similar to that of [6]; but our method is not at all concerned with the occlusion handling, however the proposed scheme can handle multiple, and discrete depth ranges of users' interest.

Further, our method utilizes two-dimensional intensity based segmentation of the left image whereas Zhang et al. [3] used a one-dimensional colour-based segmentation process of individual rows of pixels. Similarly, our proposed method uses only 'L' values of pixels, making the process simple and fast; on the other hand the method of [2] is based on object-based segmentation using colour, spatial and shape information, which potentially add to the computational time.

Lastly, we demonstrate how the running time of our proposed algorithm can be reduced by running some of its mutually independent operations in parallel on the multiple cores of CPUs and GPUs. We also experiment with combining the power of the CPU and the GPU to achieve even higher degrees of parallelism.

Key contributions of the proposed work are as follows:

- To the best of our knowledge, this is the first work on a hybrid method for stereo depth estimation using disparity information obtained by segmentation of only lightness values of left image pixels, unlike other methods using colour, texture and shape characteristics.
- Further, to the best of our knowledge, this is the first paper dealing with morphological filters and connected component analysis to get sparse, but accurate depth estimates for subsequent reconstruction of dense depth maps.
- Lastly, to the best of our knowledge, this is the first work which proposes an accelerated framework for reducing the running time of a novel algorithm for stereo depth estimation, using Java Thread Pool (CPU) and APARAPI (GPU), and for further combining them, in order to achieve even greater parallelism.

The rest of this paper is organized as follows: Sect. 2 deals with Related Work; Sect. 3 explains our Proposed Algorithm with both Sequential and Parallel Time Complexity Analysis; Sect. 4 gives the Results and Discussion; Finally, Sect. 5 concludes with future directions.

2 Related Work

When categorizing stereo vision algorithms, one of the most obvious divisions in current literature is that of global versus local methods [5]. Local algorithms are statistical methods usually based on correlation. Global algorithms are based on explicit smoothness assumptions that are solved through various optimization techniques. Computational complexity of majority global algorithms makes them impractical for real-time systems. For global methods, smoothness assumptions are defined through an energy function and the optimization techniques minimize this function. For local methods, the correlation process involves finding matching pixels in left and right images of a stereo pair by aggregating costs [e.g. sum of absolute differences (SAD), sum of squared differences (SSD), normalized cross correlation (NCC)] within a region or block.

For several years, the most accurate disparity maps were produced by global algorithms [5]. Currently, eight of the ten most accurate stereo vision algorithms ranked by Middlebury evaluation criterion are global energy minimization algorithms. Recently, however, many local algorithms have been developed that are competitive with respect to accuracy. As mentioned, there is always a trade-off between accuracy and speed for stereo vision algorithms and these accurate local algorithms are no exception.

This has motivated us to design our proposed local algorithm which can strike a good balance between accuracy of results and speed of computation.

Most existing stereo disparity algorithms use one of the two types of measures of pixel similarity between left and right image: pixel-to-pixel or window-based. Pixel-to-pixel algorithms find the pixel similarity measure solely on individual pixel values, e.g. absolute difference (AD) or squared difference (SD). On the other hand, window-based methods aggregate the

pixel matching cost over a support region around a pixel of interest, e.g. sum of absolute differences (SAD) or sum of squared differences (SSD) [4]. A study on comparison of cost functions commonly used in recent methods of stereo disparity determination [7] has come to the conclusion that the performance of a matching cost function depends on the stereo method that uses it. The SAD similarity measure can be computed efficiently by exploiting the fact that neighbouring windows overlap. For neighbouring windows with the same disparity, the overlapping pixels will contain equal AD values. Therefore, a new SAD can be computed out of an old one by subtracting the values, which are only parts of the old window, and adding the values, that are only parts of the new window [4].

This advantage of the SAD method, coupled with the fact that it yields better estimates than AD due to its support region, motivated us to use it in our method.

The shape of the SAD window is often rectangular. Fewer mismatches will occur with large windows, especially on texture-less regions. However, it is well known that large windows lead to erroneous disparity estimates at the edges of objects [4]. To overcome these limitations, several cost aggregation strategies based on characteristics of matching windows have been proposed in recent years. These can be classified into cost aggregation based on rectangular windows-based and unconstrained shapes-based strategies; lastly, by assigning different and variable weights to pixels falling in the neighbourhood of two points on the reference and target images for which stereo correspondence is being evaluated. Rectangular window-based methods can be further classified into variable window-size or offset-based, multiple-windows-based, and differential weights of window points-based approaches [8].

The multiple window approach leads to better estimates in non-occluded areas of the input stereo image. However, it is far more expensive compared to single window approach. Further, multiple window approach increases the computation time by more than 50 % but with slight improvement in overall accuracy [4]. So, we consider single window approach in proposed depth estimation algorithm.

Moreover, in our proposed approach, we use SAD to determine disparities of only points which lie on segment boundaries, which often coincide with edges of objects, where there is bound to be some amount of variation in intensity. Hence, we choose fixed-size square windows of size 9×9 pixels for our method.

To determine the best matching pixel for a given pixel, the most obvious means involves selecting the point in the other image within a certain disparity range that has the best similarity value, and this has been used in many stereo matching methods. This strategy is called 'Winner Takes it All' (WTA). However, a stereo algorithm based only on WTA does not consider parts of stereo images that are only visible in one of the two images, called occlusions. It may be noted that our proposed stereo depth estimation approach will not focus on addressing occlusion handling as of now. Moreover, image regions having little or repetitive texture yield similar matching costs. In these areas, WTA is error-prone as there isn't a clear optimum correspondence [4].

The smoothness constraint for stereo states that disparity does not change much on object surfaces. This concept is often used to improve disparity estimates. If a correct disparity has been found then it can be used to constrain the range of possible disparities of other points on the same surface [4]. We incorporated this concept in the fill step of our depth map reconstruction phase and thereby reducing wrong estimates on areas with little or repetitive texture.

In scan-line optimization (SO) schemes, the smoothness constraint for stereo is implemented as a global cost function, which adds penalties to the values in the disparity search interval of each stereo point. Errors can then be suppressed by penalizing large jumps in disparity between scan-line points. The main drawback of SO is its sensitivity to noise [4]. Hence, we did not consider SO scheme.

In dynamic programming (DP) approaches, the problem of finding the correct disparities on a scan line is regarded as a search problem. The matching costs of all points on a scan-line describe the disparity search space. Finding the correct disparities is akin to finding the path in this space which takes the shortest route through the cost values. Special rules for how to transverse the search space can be added in order to handle occlusions [4].

An important step in a stereo algorithm is searching for the disparities with the lowest SAD values. For a particular point on the left image line, the possible matches with points on the right image line constitute its search space. Finding the correct match by finding the match with the lowest SAD value is computationally expensive, since the whole disparity interval has

Table 2 Comparison of recent depth estimation methods

Work	Advantages	Limitations
Choi [17]	Bi-directional consistency check of disparity blocks can improve disparity estimates	Comparison with similar algorithms using a standard dataset has not been done
Zhu and Yu [18]	Self-adaptive block matching can increase the efficiency of searching, and reduce errors in matching and block-shape	Presence of noise in the original left or right images can result in selection of incorrect size for block-matching (8×8 or 16×16)
Wang and Zheng [19]	Cooperative optimization can check or fix disparity errors	Uses the time-consuming mean-shift algorithm for segmentation
Lu and Du [20]	Harris corner point extraction and feature-matching yields better corresponding points than traditional methods	Both left and right images are scanned for corner extraction and matching in the initial steps, making them time-consuming

to be searched [4]. Hence, some methods like SO and DP try to restrict the search space for faster computation.

The edges of nearby objects do have high disparity jumps between foreground and background. If algorithms such as SO and the DP approaches do not find these jumps, then the wrong disparity is used to define a constraint for a part of the search space. In the disparity images this error is visible in terms of the typical "streaking" error. These misses can be induced if an object is near, or when image noise is present. Algorithms such as SAD which only estimate disparity per point do not have these types of limitations [4].

However, irrespective of the types of algorithms, the scene geometry is known to have a clear influence on the performance of stereo depth estimation. Specially, algorithms like DP which use search space restrictions are more affected [4].

Hence, in our method, we do not impose any search space restrictions while determining the disparities of pixels lying on segment boundaries using SAD.

Segmentation has been employed in many recent stereo matching algorithms which considered segments instead of pixels, as the processing elements. But, their segmentation processes mostly dealt with hierarchical approaches and also took into account features of colour, size and shape while processing the input stereo image pairs. Examples of these are object-oriented segmentation methods based on bottom-up region merging [2], colour-based Mean-shift segmentation (refer to Table 2) and the use of K-Means clustering to over-segment the reference image [9]. But in all these segmentation-based stereo depth estimation approaches there is no attempt to use, primarily segmentation results of pixel grey level values to identify groups of pixels that could produce reliable disparities when matched. We tried to explore this possibility in our proposed work, thus not only making it computationally efficient (compared to time-consuming segmentation methods like Mean-shift etc.), but also simplifying its entire depth estimation process.

However, as we consider only grey level values of image pixels for segmenting the reference image, and no spatial or colour information whatsoever, we include two steps to refine the segment boundaries, using morphological filtering as well as connected components analysis. Since both of these are established techniques for image processing, their efficient implementations are readily available. Hence, our proposed method utilizes the above mentioned techniques.

Morphological filters were used for partitioning a low depth-of-field image into focused object of interest and blurred background [10]; but, our proposed method uses morphological filters for an entirely different task altogether, viz. refining segment boundaries whose disparities are determined later to reconstruct the full disparity map of the scene based on the segment boundaries' depth estimates.

Connected components analysis based methods have aided image segmentation for the purpose of text detection [11–13], brain tissue analysis [14, 15] and colour image segmentation using genetic algorithms [16], whereas in our proposed work, we use connected components analysis to refine the boundary map by removing small, isolated connected components (artefacts) which may yield incorrect depth estimates. Our approach may seem similar to [14], where the authors first roughly distinguished the region of non-brain tissue through connected component labelling, and

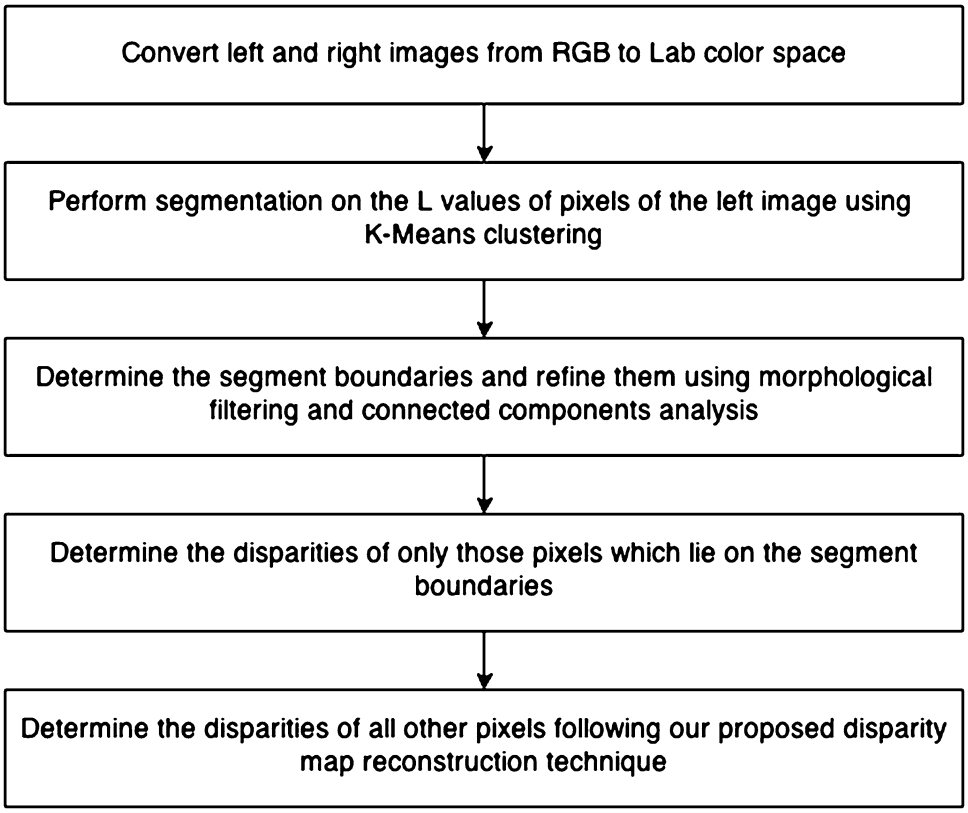


Fig. 1 Flowchart of our proposed depth extraction algorithm

then tried to refine those edges using the morphological operations, dilation and erosion. However, a slightly closer look at our method would reveal that we perform the connected components analysis *after* applying morphological filters and thus the ordering of these two operations is entirely reversed.

Several depth estimation techniques [17–20] have been developed in recent years to overcome the limitations of two basic approaches of stereo disparity estimation, viz. block-based and region-based. Their key features are summarized in Table 2.

3 Proposed Algorithm

Our proposed algorithm obtains the depth map as per its flowchart given in Fig. 1.

We explain the steps of the flowchart of our proposed algorithm, shown in Fig. 1:

3.1 Colour Space Conversion

Human eyes are more sensitive to changes in brightness than in colour. The 'L' component of the Lab colour space closely matches the human perception of lightness. Hence, by applying the pertinent transformations discussed in [21], we convert the left and right images from RGB to the Lab colour space and retain only the 'L' values of its pixels.

3.2 Segmentation

'L' values of the left image pixels are segmented using a fast implementation of the K-Means clustering algorithm. We build a histogram of the 'L' values and use that histogram instead of the actual pixel values for clustering. Thus, the runtime of clustering is significantly reduced, as we perform clustering on a small, fixed number of bins comprising the histogram. Since there exists a one-to-one correspondence between each pixel and the bin to which it has been mapped, we can easily identify the cluster to which the pixel has been assigned as the cluster to which its bin has been assigned.

3.3 Segment Boundary Detection and Refinement

Segment boundary detection is achieved by comparing the cluster assignment of each pixel with that of its 8-connected pixels (i.e. its Moore neighbourhood). If any of them is found to differ, we mark the pixel as '1'

(falling on a segment boundary), else as '0' (not on a segment boundary). So, this step creates the "boundary map" after segmentation.

But this approach also falsely identifies many pixels as belonging to segment boundaries due to limitations imposed by clustering accuracy, as we segment the image based on only the pixels' lightness (L) values and not on their colour components (a, b) or spatial locations (x, y) in the image. So, we apply two morphological filters to refine the boundary map by removing such noisy pixels, in the following order:

Fill: Fills isolated interior pixels such as the centre pixel in:

1 1 1
1 0 1
1 1 1

Remove: Removes interior pixels, i.e., sets a pixel to '0' if all its 4-connected neighbours are '1', thus leaving only the boundary pixels on.

Further, we use connected components analysis and remove small artefacts in the boundary map due to segmentation errors, by ordering connected components present in the boundary map by the number of pixels constituting each connected component to remove the smallest connected components which contribute about 4 % of the total number of boundary pixels.

3.4 Disparity Measurement of Boundary Pixels

We assume that the left and right cameras are calibrated and the left and right images share the same image plane. So, the correspondence of each pixel can only be in the horizontal direction. For e.g., the correspondent point of any pixel on the left image can only appear on the same row of the right image.

We use the SAD [1] cost function to determine only the disparities of boundary pixels, using the 'L' values of the left and right image pixels. It should also be noted here, that by "boundary" pixels, we are also (implicitly) referring to the pixels of the left image that map to the left and right borders of the disparity map.

3.5 Disparity Map Reconstruction from Boundaries

Our disparity map reconstruction algorithm scans through each row of the partially computed disparity map and computes the remaining disparities based on disparities that have already been calculated. It operates in two stages:

3.5.1 Disparity Propagation ('Fill' Stage)

In the first stage, we scan the disparity map row-wise, left to right—whenever we consecutively encounter two boundary pixels with equal disparity values, we 'fill' the intermediate pixels with that disparity value. This reflects our assumption that the pair of points in consideration actually belong to the same object in the original image. So, all their intermediate pixels also belong to that same object, and hence, should have similar disparity values. The process is explained using the pseudo-code below; *disp_map* is the matrix containing the disparities of boundary pixels and disparity(*cell*) refers to the disparity value of a boundary pixel of *disp_map*.

Algorithm 1 Disparity Propagation along Scan Lines

```
for each row of disp_map, starting from its top do
    for each cell of this row, starting from its left do
        if this cell is a pixel on a segment boundary, then
            if disparity(cell) == disparity(previous boundary pixel), then
                disparity of all intermediate pixels ← disparity(cell)
            end
        end
    end
end
```

3.5.2 *Estimation from Known Disparities ('Peek' Stage)*

In the second stage, for all pixels whose disparities have not yet been determined, we estimate their disparities by 'peek'-ing at disparity values of their two nearest pixels for which the disparities have been already determined. We search for these two nearest pixels along the same column as the pixel in question and the details are as follows:

Algorithm 2 Disparity Estimation from Known Disparities

```
for each row of disp_map, starting from its top do
    for each cell of this row, starting from its left do
        if disparity of this cell has not been already determined, then
            disp_n ← {disparities of two nearest pixels on the same column}
            disp_r ← statistical range of disp_n
            if disp_r > disparity_threshold, then
                disparity of this cell ← smaller of the two values in disp_n
            else
                disparity of this cell ← mean of the two values in disp_n
            end
        end
    end
end
```

3.6 Depth-Based Selective Blurring

The depth map output by the aforementioned final step is fed to the blur map generator, along with the depth ranges of users' interest and an overall blur level. It must be noted here that, the user is solely responsible for selecting the proper depth range(s) which comprise region(s) of his (her) interest in the scene. This is analogous to setting a proper DOF while shooting a scene using a SLR camera. Thus, the blur map, determining which pixels in the image will be blurred, is fed to a depth-based blurring module, which synthesises a new scene, where all depth ranges of users' non-interest are blurred using Gaussian function $h_g(n_1, n_2)$ and Blurring kernel $h(n_1, n_2)$ as defined by Eqs. 1 and 2 respectively,

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{2\sigma^2}} \tag{1}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g} \tag{2}$$

where n_1 and n_2 are dimensions of the blurring kernel, σ^2 is the variance for building the Gaussian kernel and the overall blur level is determined by σ.

Figure 2 shows the overall process of this depth-based selective blurring.

3.7 Sequential Complexity Analysis

Here, we present the step-by-step time complexity analysis of our proposed algorithm and thereby identifying computationally intensive steps and the degrees of interdependency between the operations of each step. Next, we convert steps eligible for data-parallelization to data-parallel workloads in JTP and APARAPI.

Our first step involves colour space conversion, in which we apply predefined transformations [21] to the (R, G, B) triplet of each pixel in both the left and right image to extract the 'L' values from them. Thus, the time complexity of this step is $\boldsymbol{O}(\mathbf{c_1}\boldsymbol{N_1})$ in which $\boldsymbol{N_1}$ represents the number of pixels of each image and $\boldsymbol{c_1}$ the number of (constant) operations required to perform

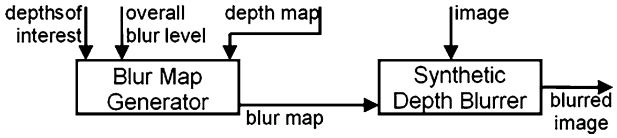


Fig. 2 Depth-based selective blurring

each transformation. Further, as transformation applied to each pixel is independent of others, this step lends itself to the most obvious data-parallelization.

In our next step, we perform K-Means clustering on the 'L' values obtained in the previous step. We first show the time-complexity derivation for the naïve K-Means implementation, and then go on to show how our fast implementation has been able to reduce the complexity. Let 't_{dist}' be the time to calculate the distance between any two feature vectors on which we are running the K-Means algorithm; then, each iteration of K-Means has the complexity $O(\mathbf{K}\mathbf{N_2}\mathbf{t_{dist}})$, where K denotes the number of cluster centres and N_2 the number of feature vectors ('L' values, in our case). For finding 't_{dist}', we need only one computational step, that of finding the AD between two integers falling within the range [**0**, **255**] (the range of possible values for 'L'). More importantly, since we perform clustering on the bins of the histogram of the left image, and there can be a maximum of **256** bins (representing the range [**0**, **255**]), this drastically reduces the time complexity compared to the naïve implementation which would have run each iteration on all N_1 'L' values (one value for each of the N_1 pixels of the left image).

The next three steps, namely, segment boundary detection and refinement using morphological filters 'fill' and 'remove' are performed in number of comparisons linear in the number of pixels of the left image, i.e., N_1. Hence, they have time complexities of $O(c_2N_1)$, $O(c_3N_1)$ and $O(c_4N_1)$ respectively with the constant terms representing the (constant) number of operations required for each step. It is also evident, that since the operations for each step are independent for each pixel, they are ideal candidates for data-parallelization leading to accelerated processing.

The next step, viz. finding connected components has three basic operations:

1. Search for the next unlabelled pixel, p.
2. Use a flood-fill algorithm to label all the pixels in the connected component containing p.
3. Repeat steps **1** and **2** until all the pixels are labelled.

It is clear that searching for new unlabelled pixels until all image pixels have been labelled requires a number of operations which is linear in the total number of image pixels. Further, a recursive flood-fill algorithm works as outlined below:

1. If the pixel we want to label is unlabelled, then label it, else stop.
2. Reclusively flood-fill each unlabelled 8-connected pixel of the above pixel.

Since there are 'if' clauses, flood-fill only considers unlabelled pixels, which have a static number, say, n. Therefore worst-case complexity becomes $O(n)$. For ordering N_c connected components based on their constituent number of pixels, we can use merge-sort to finish the operation in $O(N_c\, log(N_c))$ comparisons.

Next, to find the disparity of each boundary pixel using the SAD cost function for a (constant) window size 'w' and a (constant) maximum disparity 'd', we require $O(w^2d)$ operations. Now, for N_b boundary pixels, it becomes $O(N_bw^2d)$.

Lastly, both the 'fill' and 'peek' stages of the depth map reconstruction phase of our algorithm are primarily based on a (constant number of) computations on the disparities of boundary pixels with time complexities of $O(N_b)$ each, and these can also be converted into data-parallel workloads due to their independent nature.

Space complexity of our proposed algorithm can be computed as follows: the pixels' 'L' values and the complete depth map need $O(N_1)$ memory; we also need an auxiliary storage of $O(N_b)$ size for holding the intermediate results of segment boundary detection and refinement and the disparities of pixels lying on refined segment boundaries. For K-Means clustering, an array of size $O(\mathbf{256})$ is required for storing the count of image pixels mapped to each histogram bin for the present iteration (each histogram bin represents each grey-intensity present in the image).

3.8 Parallel Implementation and Its Time Complexity Analysis

We carry out parallelization of some of the computationally intensive steps of the proposed algorithm whose constituent operations are mutually independent. By executing these independent steps on multiple cores of the CPU and GPU we can reduce the time complexity. We use the Java Thread Pool (JTP) for

Table 3 Characteristics of Middlebury stereo image pairs used for performance evaluations

Characteristics	Tsukuba	Sawtooth	Venus	Teddy	Cones
Size (pixels)	384 × 288	434 × 380	434 × 383	450 × 375	450 × 375
Disparity Levels	16	20	20	60	60

CPU-parallelization and APARAPI [22], an API which allows suitable data parallel Java code to be executed on GPU via OpenCL, for GPU-parallelization. We also experiment with combining the two methods to achieve even greater performance.

The independent execution units in Java Thread Pool are the "worker threads" and in APARAPI are the "kernels". Thus, we put the parts of our code which we want to run in parallel in the respective execution units. Moreover, combination of CPU and GPU parallelization leads to sharing the workload between them, so we have to decide on how many execution units we want to run simultaneously on the CPU and the GPU, so as to achieve optimal balancing of workload between them.

Thus, assuming the availability of 'p' parallel processing units, for four steps of our algorithm, viz. colour space conversion and segment boundary detection and refinement using the morphological filters 'fill' and 'remove', each of them can be completed in $O(c_i N_1/p)$ time. Similarly, both the 'fill' and 'peek' steps of the depth map reconstruction phase can also be completed in $O(N_b/p)$ time each.

4 Results and Discussion

To compare the stereo disparity estimation performance of our approach versus those of some other (existing) algorithms, we initially ran our algorithm on three pairs (left and right) of stereo images from the Middlebury data-set, viz. Tsukuba, Sawtooth and Venus and compared the generated depth maps with corresponding ground truth depth maps provided in the data-set. We also compared our depth maps with those output by two established cost functions (SAD and NCC) used in numerous traditional stereo disparity estimation methods, and a recent method [3] which uses the AD cost function, for those same three stereo image pairs. We further compared our error rates with two most competitive disparity estimation techniques, viz. Graph-cut [23] and TV-based [24]. Also, the Middlebury stereo evaluation page maintains a table of performance statistics of numerous state-of-the-art disparity estimation methods, submitted by authors, after running their algorithms on four standard stereo pairs supplied as part of the data-set (viz., Tsukuba, Venus, Teddy and Cones) with constant parameters.[1] It is obvious that there is no way to check whether the authors actually ran their algorithms with a *constant* set of parameter values for all the four image pairs, and algorithms may yield substantially different error rates for different sets of parameter values. Nevertheless, we consider this repository as an additional benchmark for evaluating the performance of our proposed algorithm.

We present the results of the said comparisons below, along with the outputs of each step of our proposed approach for the Tsukuba image pair (since it presents a typical scene with adequate number of disparity levels in its provided ground truth depth map to mimic real-world depth-based blurring scenarios), to demonstrate how our proposed depth estimation algorithm works.

Characteristics of all Middlebury stereo image pairs used for evaluation of our proposed algorithm and comparison of our output depth maps, are presented in Table 3. We used a set of parameter values and the details are given in Table 4.

4.1 Explanation of Step-wise Outputs of Proposed Method for 'Tsukuba' Image

4.1.1 Colour Space Conversion

Process Left and right input images are converted from the RGB to the CIE-Lab color space following the pixel value transformations outlined in [21].

[1] http://vision.middlebury.edu/stereo/submit/.

Table 4 Parameter values for the three image pairs

Parameter	Tsukuba	Sawtooth	Venus
Number of clusters (K) for K-Means	10	10	10
Block size for cost aggregation (odd)	9 × 9 pixels	9 × 9 pixels	9 × 9 pixels
Disparity threshold for reconstruction	0	1	1

Results Each pixel of both images has an L value for its lightness component. Figure 3 (left) shows the converted left input image.

Observations The output image shows the lightness value of each pixel.

4.1.2 Segmentation

Process Converted left input image image is fed to the fast implementation of K-Means clustering algorithm as discussed earlier.

Results The result is shown in Fig. 3 (right).

Observations Small variations in 'L' values (noise) have been absorbed into single coherent segments representing image regions of pixels belonging to the same object, with similar values of disparity. However, since we have segmented the left image based on just the pixels' 'L' values, ignoring their spatial locations (to expedite the clustering process), we must refine the boundaries of objects defined by the clusters above, so that the spatial distribution of the 'L' values are also taken into account.

4.1.3 Segment Boundary Detection and Refinement

Process We detect and refine the segment boundaries using the approach described in the previous section.

Results The outputs are shown in Fig. 4, with output of segment boundary point detection, followed by that of morphological filtering, and lastly, output of connected components analysis at the bottom.

Observations We can infer from Fig. 4b, how the redundancies in segment boundary detection have been removed from Fig. 4a, and from Fig. 4c, how the small artefacts in Fig. 4b have been removed by the connected components analysis technique.

For the Tsukuba image pair, the segment boundary refinement reduces the number of boundary pixels by nearly 53 % for the parameter values of Table 4, such that only 19 % of the left image pixels are used for disparity calculations. This greatly reduces the number of disparity computations in the next step.

4.1.4 Disparity Measurement of Boundary Pixels

Process Disparities of the boundary pixels are determined by using the SAD algorithm.

Results The result in shown in Fig. 5 (left).

Observations Boundaries of objects closer to the camera (like the lamp and the head of the statue) are having a higher intensity (greater value of stereo disparity). This is in direct agreement with the reality that value of disparity of a pixel is inversely proportional to its distance (from the camera lens/human eyes).

4.1.5 Disparity Map Reconstruction from Boundaries

Process Disparities of the segmented regions in the image are determined from the disparities of their boundaries using the two-stage disparity map reconstruction method discussed earlier.

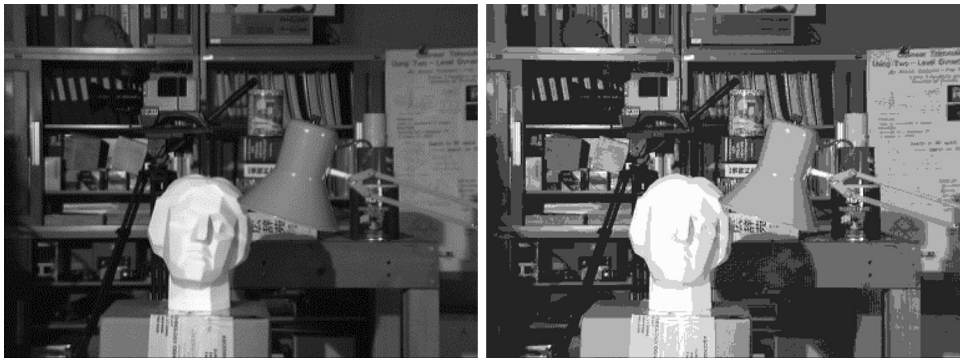
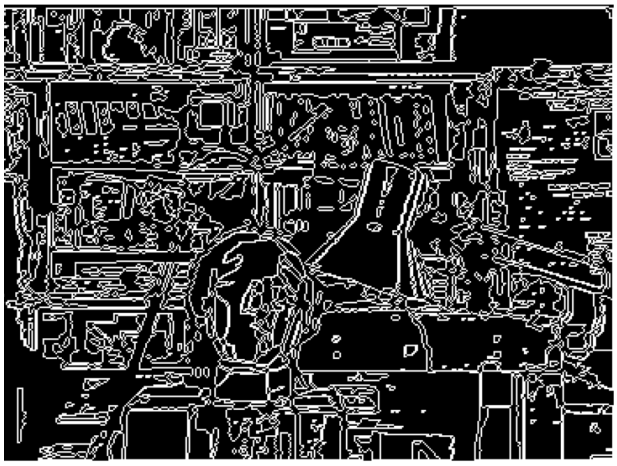
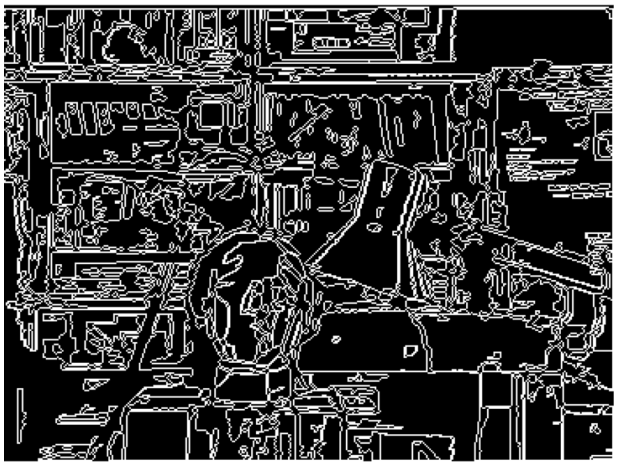


Fig. 3 Segmentation of 'L' values of left image using K-Means

(a)



(b)



(c)

Fig. 4 Segment boundary detection and refinement

Results The results are shown in Fig. 5 (right).

Observations The depth map shows the mapping of each image point to a certain level of disparity. Objects which are closer to the camera like the lamp-shade have a greater disparity value resulting in a brighter shade of gray. For objects further away from the camera like the table, the shades of gray get progressively darker, as their disparity values keep on decreasing.

To evaluate the disparity estimation of our approach vis-a-vis those of existing algorithms, we initially chose three pairs (left and right) of stereo images from the Middlebury stereo vision data-set, viz. Tsukuba, Sawtooth and Venus. We ran our algorithm on those three image pairs. We then compared the depth maps obtained by our algorithm against the corresponding ground truth depth maps provided in the data-set, as well as with the depth maps created by two established methods (SAD and NCC [1]) and a recent work by Zhang et al. [3] for those same three image pairs. We present the results of those comparisons below.

4.2 Qualitative and Quantitative Comparison with Ground Truth Depth Maps

To evaluate the depth estimates of our proposed method, we consider the approach of computing error statistics with respect to the ground truth image available with the Middlebury dataset, and we choose the quality metric as the percentage of bad matching pixels (B) given by Eq. 3,

$$B = \frac{1}{N}\sum_{(x,y)} (|d_C(x,y) - d_T(x,y)| > \delta_d) \tag{3}$$

where $d_C(x, y)$ are the computed disparities and $d_T(x, y)$ are ground truth disparities, and δ_d denotes the disparity error tolerance, which is taken as '1.0' as per published works [25].

Table 4 shows the values of our algorithm's parameters which gave best results are hence, were used for performance comparison. From the table, it's evident that the set of optimal parameter values is almost constant across all three images. We have observed in our experiments that slight variations of the parameter values about the optimal ones effect negligible changes in accuracy of depth estimation. Thus, our method is robust to changes in parameter values, like the one in [19].

Figure 6 presents a comparison of the depth map produced by our method (left) with ground truth depth map (right) for the Tsukuba image. The black regions

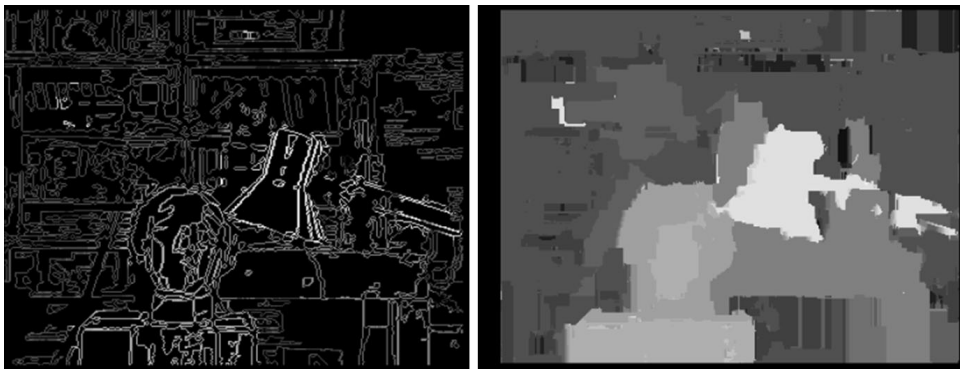


Fig. 5 Disparity map reconstruction from boundary disparities

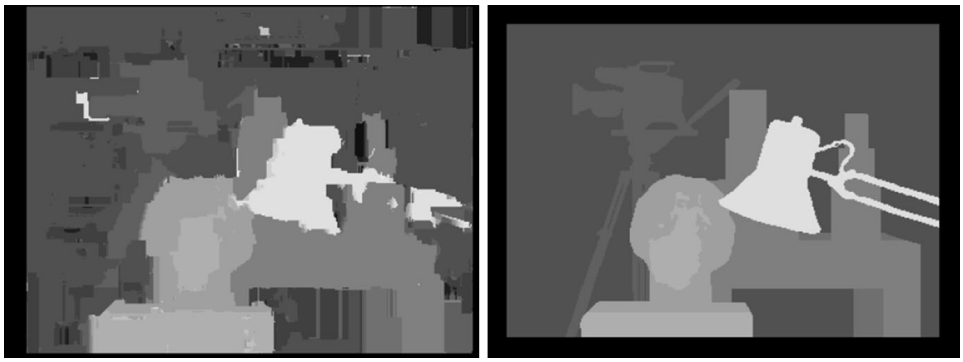


Fig. 6 Resultant depth map compared to Middlebury's ground truth

in depth maps are those whose disparities are not compared. Further, the percentage of bad matching pixels (including occluded image regions) was found to be 7.8 %.

It can be seen from the comparison of our depth map with the ground truth that our algorithm yields fairly accurate depth estimation; its performance is affected mostly in some texture-less and low illumination regions. But as we will be using the generated depth maps for the purpose of selective blurring, inaccurate depth estimates in regions of low texture and illumination will not affect the blurring performance, as human eyes are more sensitive to variations in brightness than colour, and hence perceptual differences resulting from blurring a dark region would be negligible. Likewise, as blurring is performed using a weighted average of neighbouring pixel values, a region of low texture will remain relatively unaffected after a blurring operation, as the variation in pixel values in a texture-less region is negligible. Some incorrect depth estimations also occur in occluded regions which can be reduced using occlusion handling techniques in future. Also, our algorithm does not calculate depth information of pixels lying near the image borders (which we will address in our future work), but this limitation does not significantly affect the output of depth-based blurring, as the images' borders very rarely contain objects or regions of interest to the user.

4.3 Qualitative and Quantitative Comparison with Stereo Matching Methods

Table 5 shows the results of performance comparison of our depth estimation method with two established methods, viz. Sum of absolute differences (SAD) and normalized cross-correlation (NCC) (both of which have been used as cost functions in numerous traditional algorithms for stereo depth estimation), as well as a recent work by Zhang et al. [3] which uses AD cost function to predict disparities. Again, we use the percentage of bad matching pixels to quantitatively compare depth maps generated by our method, SAD, NCC and Zhang et al. [3]. We also separately evaluate the accuracy of depth maps generated by our algorithm by considering the corresponding Middlebury ground truth depth maps, by both including and excluding the occluded regions of the input images, as algorithms like ours and Zhang et al. [3] often do not deal with occlusion handling explicitly. In this context, it should also be

Table 5 Performance comparison of proposed algorithm

	Tsukuba (%)	Sawtooth (%)	Venus (%)
NCC [1]	41.4	9.92	17.4
SAD [1]	36.9	11.9	24.5
Zhang et al. [3]	13.9	7.22	6.12
Proposed algorithm (including occluded regions)	7.8	5.26	4.72
Proposed algorithm (excluding occluded regions)	6.1	3.29	3.78

noted that, in the work by Zhang et al. [3], the authors have not specifically mentioned clearly whether they have evaluated their algorithm only on non-occluded image regions, or on occluded image regions as well.

The results clearly demonstrate that our proposed method outperforms traditional NCC and SAD methods by up to 33.6 %, and even a recent method of Zhang et al. [3] by up to 6.1 % (including occluded regions of input images).

Further, the percentage of left image pixels used for stereo depth estimation was about 19 % for Tsukuba, 18 % for Sawtooth and 16 % for Venus image pairs, implying that our algorithm is scalable to high resolution stereo images.

Figures 7 and 8 show depth maps generated by our method (left) and ground truth depths (right) for the Sawtooth and Venus images. Their quantitative comparison results are shown above, in Table 5. All other depth maps supporting performance comparison data presented in Table 5 can be found in [3].

We next compared our proposed method with two most competitive depth estimation techniques, viz. Graph-cut [23] and TV-based [24], and the results are shown in Table 6 and 7 respectively. We used the same quality metric, viz., "Percentage of bad matching pixels", (with the value of δ_d denoting the disparity error tolerance, taken as '1.0') to quantitatively compare depth maps generated by our method and the Graph-cut based method [23]. To compare our results with the TV-based method [24], we used another quality metric, viz., 'Average Absolute Disparity Error' (AADE) used by the authors of the TV-based method [24].

Analysis of results presented in Table 6 and 7 shows that, in cases where the most competitive recent methods like Graph-cut [23] are able to successfully address the issue of occlusions in the input image (by adopting an expansion-moves approach over swap-moves approach), our method has a definite scope of improvement, since we have not considered occlusions as of now. But, using the swap-moves approach, where the Graph-cut based method failed to properly handle occlusions in the input image, our algorithm performed significantly better. Also, it must be noted that handling occlusions in input images entails a higher computational complexity, as evident from the authors' run-time calculation [23].

On the other hand, TV-based methods like [24] give better disparity estimates by iteratively refining left-to-right and right-to-left initial disparity maps (obtained using a correlation-based method). The results are shown in Table 7 where, using similar evaluation methodology as authors [24], we have quantitatively compared the performance of our algorithm in three different types of areas in the image, classified as un-textured ($\boldsymbol{B_{untex}}$), discontinuous ($\boldsymbol{B_{disc}}$) and the entire image ($\boldsymbol{B_{all}}$), for only non-occluded pixels in all three cases. Understandably, our

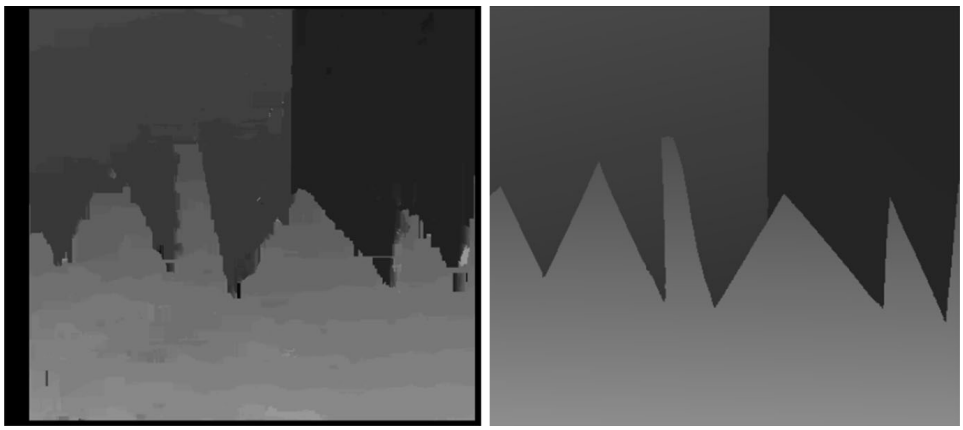


Fig. 7 Sawtooth depth map for proposed algorithm (*left*) and Middlebury's ground truth (*right*)

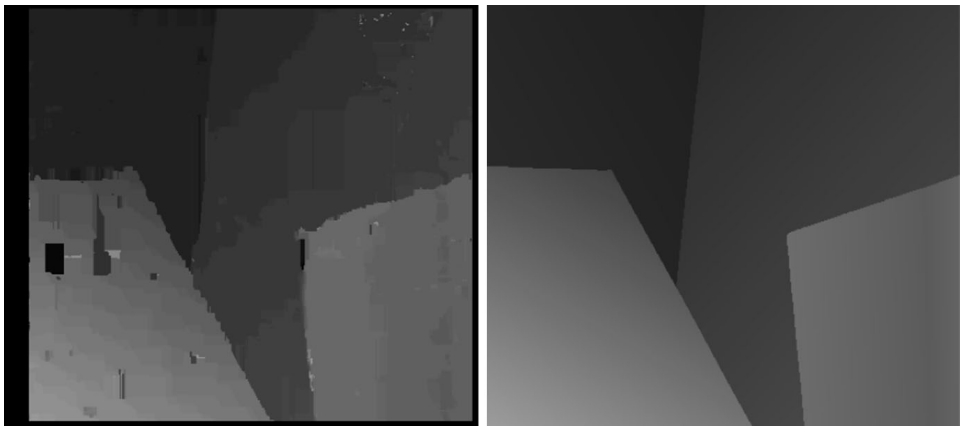


Fig. 8 Venus depth map for proposed algorithm (*left*) and Middlebury's ground truth (*right*)

Table 6 Performance comparison of proposed algorithm with graph-cut method

Algorithm execution on non-occluded image regions	Tsukuba (%)
Proposed method without occlusion handling	6.1
Kolmogorov's graph-cut method (using expansion-moves)	1.9
Kolmogorov's graph-cut method (using swap-moves)	13.6

Table 7 Performance comparison of proposed algorithm with TV-based method

	Tsukuba B_{all} B_{untex} B_{disc}	Sawtooth B_{all} B_{untex} B_{disc}	Venus B_{all} B_{untex} B_{disc}
Proposed method	0.56, 0.70, 1.30	0.44, 0.42, 1.58	0.49, 0.59, 1.28
TV-based method	0.29, 0.24, 0.51	0.23, 0.19, 0.41	0.24, 0.26, 0.39

method yields comparatively greater error rates, as we do not have any initial disparity maps to start with, as we build it from scratch.

For our final set of comparisons w.r.t our algorithm's error-rate, we focus once more on the Tsukuba image pair (as it represents a scene which can be a typical candidate for many real-world depth-based blurring scenarios). We use the Stereo Evaluation table[2] of the Middlebury website to compare our algorithm's error-rate against those of numerous others, based on their output depth maps submitted to the table by their corresponding authors, and evaluation done by the website itself. We used a set (fixed) of parameter values Number of K-Means' Clusters (K) = 10, Block Size for Cost Aggregation = 9 × 9 pixels, and Disparity Threshold = 1.0 for all four standard data-set stereo image pairs required for evaluation (Tsukuba, Venus, Teddy and Cones). Then, we submitted our generated depth maps through the Middlebury website's online submission form for evaluation and comparison against their repository of results of several existing state-of-the-art algorithms.

The comparison results are shown in Fig. 9, with our method labelled "YOUR METHOD" as per the Stereo Evaluation Table (see footnote 2) of the Middlebury website. The original table has more than 150 entries and cannot fit within the space constraints of this paper, so we chose to truncate the table from the 7th entry onward, till the 5th entry before the one corresponding to our method. The reason for this is, as the table entries are sorted by increasing order of error-rate for the Tsukuba image pair, by truncating the table in the said fashion, one can see some of the best performing methods, followed by few methods which perform slightly better than our method, followed my results of our method, and lastly, those of all the 38 methods which perform worse than ours. The disparity error tolerance, δ_d has been taken to be the most stringent permissible, viz. '0.5', which translates to finding (the complement of) the percentage of *exactly* matching pixel disparities in case methods like ours, which do not compute sub-pixel disparities.

From an analysis of the results presented in Fig. 9, one can conclude that our method performs better for 'nonocc' (non-occluded) and 'all' input image regions, but does not produce up-to-the-mark results

[2] http://vision.middlebury.edu/stereo/eval/.

Error Threshold = 0.5 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc						
Algorithm	Avg. Rank	Tsukuba ground truth nonocc	Tsukuba ground truth all	Tsukuba ground truth disc	Venus ground truth nonocc	Venus ground truth all	Venus ground truth disc	Teddy ground truth nonocc	Teddy ground truth all	Teddy ground truth disc	Cones ground truth nonocc	Cones ground truth all	Cones ground truth disc	Average Percent Bad Pixels
MultiRBF [129]	80.6	2.76 1	3.04 1	7.22 1	8.38 103	8.50 96	13.5 75	18.9 116	21.1 78	33.0 110	17.6 135	21.9 119	26.5 132	4.34
SubPixSearch [109]	7.4	5.60 3	6.23 3	9.46 3	1.07 12	1.64 12	7.36 10	6.71 8	11.0 6	16.9 8	4.02 9	9.76 7	10.3 8	7.09
AdaptOvrSegBP [32]	40.8	5.98 4	6.56 4	9.09 2	3.66 29	3.96 26	13.2 68	13.0 57	18.9 52	26.4 52	9.48 70	14.9 61	17.2 65	7.21
GC+LSL [136]	4.4	5.04 2	5.56 2	14.0 11	0.66 4	0.88 4	5.82 5	4.20 1	7.12 1	12.9 2	3.77 6	9.16 6	10.4 9	8.21
2OP+occ [36]	49.2	7.10 8	7.70 9	11.9 4	0.56 2	0.83 2	4.21 2	17.5 103	22.7 95	31.5 95	11.6 93	17.1 83	20.4 95	8.91
GC+occ [2]	91.0	6.10 5	7.11 5	14.6 14	10.7 127	11.3 127	16.9 107	23.7 136	30.1 137	34.6 122	12.2 102	19.2 103	21.9 107	9.25
RDP [87]	57.7	23.6 119	23.9 117	20.1 75	6.39 65	6.73 57	11.0 35	11.2 33	17.5 31	23.2 23	8.13 54	13.8 43	14.9 40	22.5
LocallyConsist [62]	72.0	24.5 126	24.8 122	19.3 66	5.79 46	6.18 44	9.50 23	14.6 73	21.1 77	27.5 61	9.73 78	16.6 80	17.3 68	22.9
CSBP [74]	118.5	22.0 109	23.8 115	21.3 92	7.60 89	9.16 105	23.6 128	19.4 121	27.8 131	40.1 139	15.1 123	24.7 137	28.5 133	22.4
FastBilateral [61]	84.2	21.5 105	22.4 106	22.9 108	5.71 44	6.66 56	14.9 88	16.2 95	23.3 100	32.1 101	9.10 63	15.8 72	18.1 73	22.3
YOUR METHOD	130.1	18.3 82	19.8 89	33.9 148	8.59 107	9.74 114	31.8 142	30.6 145	37.4 146	47.4 148	23.6 145	31.0 147	40.9 148	24.0
TreeDP [8]	132.4	22.4 111	23.1 109	22.3 101	12.1 134	12.9 134	21.7 123	32.4 150	38.9 150	45.6 144	23.7 147	30.8 146	31.7 140	22.6
HRMBIL [81]	85.1	19.1 88	20.2 95	32.2 142	2.71 22	4.06 28	21.0 122	14.4 68	22.3 90	34.9 123	8.57 60	17.8 93	20.1 90	23.8
AdaptDispCalib [35]	90.3	24.6 130	24.7 121	21.3 90	7.14 78	7.56 77	15.0 89	18.8 114	25.2 116	29.7 75	9.21 66	15.1 67	16.7 61	23.5
RT-ColorAW [91]	98.8	24.4 124	25.9 132	21.2 86	7.84 94	8.99 101	13.3 70	15.3 82	23.1 97	28.1 64	13.9 110	21.3 118	22.2 108	23.8
RealtimeBFV [58]	108.0	24.5 129	25.0 123	21.3 91	8.64 111	9.12 104	13.5 76	18.1 107	24.2 107	32.1 100	15.0 121	20.6 111	22.9 116	23.6
iFBS [99]	77.8	25.0 133	25.2 127	21.1 85	6.70 71	7.14 71	11.7 43	14.6 71	20.6 74	28.2 65	9.69 76	15.6 70	15.8 48	23.8
VariableCross [42]	111.8	24.5 128	25.1 126	21.5 95	9.03 116	9.59 112	13.8 80	18.8 113	25.1 115	31.4 93	16.1 129	22.1 123	22.4 111	23.7
IterAdaptWgt [90]	90.8	25.4 136	25.8 131	21.0 83	6.02 54	6.81 60	15.5 95	15.8 89	22.6 93	30.1 79	11.6 94	18.1 95	19.0 81	24.1
BioPsyASW [72]	117.2	22.9 115	24.4 119	24.1 117	9.69 122	10.8 124	24.5 130	18.5 111	26.1 124	34.5 121	12.6 105	20.2 108	22.3 110	23.8
DCBGrid [77]	92.2	21.7 107	22.8 107	29.6 138	2.85 24	3.97 27	16.5 101	16.1 93	24.0 106	33.0 109	10.8 85	18.2 96	22.6 114	24.7
OptimizedDP [63]	117.4	24.0 120	25.5 129	22.7 104	12.4 135	13.7 136	23.7 129	17.1 101	25.0 114	30.3 80	14.1 113	22.2 126	24.6 122	24.1
Differential [131]	86.5	21.8 108	23.5 113	27.7 134	7.49 85	8.49 94	25.1 131	14.7 75	16.5 23	32.2 103	7.71 46	14.3 52	18.3 74	24.3
TensorVoting [9]	78.2	25.5 138	26.2 139	21.2 89	3.32 28	4.12 30	14.6 86	14.6 72	21.8 83	33.3 113	7.05 37	14.5 53	17.4 70	24.3
InfoPermeable [93]	88.8	25.7 140	26.2 138	21.2 88	8.64 109	9.34 107	15.0 91	15.0 78	22.1 86	29.2 72	7.68 44	15.1 68	15.1 44	24.4
CostRelaxAW [52]	78.4	24.3 122	24.7 120	25.7 126	4.36 36	4.97 36	13.3 70	14.3 67	21.2 80	32.1 99	8.41 58	14.7 55	18.1 72	24.9
ADCensus [82]	46.5	26.8 143	27.0 142	21.1 84	4.05 32	4.60 31	8.00 16	10.6 21	13.8 11	20.1 10	6.58 31	12.4 23	11.9 14	24.9
Infection [10]	137.7	22.1 110	23.5 112	37.6 149	11.1 132	12.3 132	37.3 150	24.3 139	31.7 140	56.4 153	20.2 141	27.8 142	48.2 152	27.7
CCH+SegAggr [45]	114.8	24.8 132	25.0 125	24.0 114	8.83 114	9.58 111	16.6 102	17.5 104	24.0 105	33.2 111	14.9 120	20.9 114	25.5 128	24.6
MDPM [140]	64.1	25.6 139	25.9 133	22.5 102	6.41 66	6.89 63	11.4 41	11.2 32	13.7 10	24.2 31	8.06 51	14.8 59	15.1 42	24.6
RealTimeGPU [14]	127.7	24.2 121	26.0 136	24.9 122	10.9 129	12.1 131	27.6 136	19.6 124	27.0 127	33.0 108	16.5 131	23.7 131	29.5 136	25.1
GeoDif [88]	72.7	26.3 142	26.8 141	21.7 97	7.17 79	7.85 83	12.5 58	12.6 50	19.2 55	23.9 27	8.15 55	14.2 48	14.8 37	24.9
CurveletSupWgt [66]	79.8	25.4 135	25.7 130	25.0 123	7.41 82	7.72 79	15.4 93	14.6 74	16.5 24	28.7 70	9.47 69	12.4 24	16.4 55	25.3
SegTreeDP [21]	91.8	25.4 137	26.0 134	24.6 121	6.23 59	6.59 54	10.8 30	19.7 125	25.9 122	30.9 85	10.9 86	16.5 78	17.7 71	25.3
IMCT [55]	137.2	24.5 127	25.5 128	29.6 139	10.9 128	11.7 130	30.9 140	26.3 140	32.0 142	46.0 146	20.7 142	26.4 141	37.2 144	26.5
H-Cut [69]	122.5	25.1 134	26.7 140	25.2 124	8.51 105	9.91 116	27.9 137	18.2 108	26.5 126	37.4 134	12.3 103	21.9 120	24.6 123	25.7
LCDM+AdaptWgt [68]	142.8	24.4 125	26.0 135	32.5 144	22.4 149	23.3 149	39.9 151	32.0 148	38.0 147	49.8 150	18.7 138	26.3 140	30.7 138	27.6
BSM [106]	59.6	28.1 145	28.2 145	24.0 115	5.58 43	6.27 46	11.7 43	11.2 35	15.2 17	24.2 32	6.45 27	14.1 46	13.1 21	26.8
MVSegBP [59]	96.3	28.2 146	29.6 148	24.1 116	6.01 52	6.59 53	11.2 38	15.9 92	21.5 81	29.8 77	14.4 116	21.2 116	24.5 121	27.3
STICA [15]	140.4	24.3 123	26.1 137	44.8 153	15.2 143	16.6 144	46.3 153	24.0 137	31.3 139	50.4 152	15.9 127	23.7 130	39.5 147	31.7
SDDS [115]	101.0	26.9 144	27.1 143	25.9 127	7.85 95	8.37 91	13.3 72	15.2 80	21.6 82	31.3 87	11.6 95	17.7 92	21.6 104	26.6
CrossLMF [108]	64.0	32.1 151	32.2 151	24.6 120	6.04 56	6.37 49	11.2 38	11.1 29	17.5 30	24.1 29	7.55 43	13.5 38	14.5 34	29.6
TwoWin [80]	124.0	28.4 148	29.0 146	26.8 129	7.09 76	7.77 80	16.1 100	22.2 135	28.0 133	37.3 133	19.2 139	24.3 134	29.4 135	28.1
CostRelax [11]	78.7	26.3 141	27.3 144	33.5 147	2.92 25	4.06 29	20.8 120	12.3 47	20.2 70	32.4 104	6.33 26	13.1 31	16.7 60	29.0
HOL [145]	73.2	32.6 152	32.9 152	26.6 128	6.46 69	6.96 65	10.2 26	13.8 66	19.9 65	27.5 59	7.03 36	13.7 39	13.2 22	30.7
RINCensus [147]	119.8	28.3 147	29.5 147	30.7 141	7.44 84	8.50 95	19.0 114	16.7 98	24.5 110	37.0 132	14.3 115	22.5 127	25.7 127	29.5
DPVI [60]	136.6	29.7 150	30.4 150	28.3 137	13.1 139	13.9 137	29.3 139	19.9 129	25.9 123	36.1 126	17.8 136	23.8 132	33.5 141	29.5
SSD+MF [1a]	143.2	28.5 149	30.0 149	38.1 150	12.8 138	14.2 139	23.4 126	28.2 142	35.4 144	48.7 149	21.1 143	29.4 143	38.6 146	32.2
2DPOC [94]	150.3	33.3 153	34.8 153	33.3 146	25.4 152	26.5 152	35.4 148	32.2 149	38.8 149	46.2 147	38.4 152	43.9 152	46.9 151	33.8

Fig. 9 Truncated Middlebury stereo evaluation table comparing our method versus state-of-the-art

for the 'disc' regions (which denote regions of disparity discontinuities) for all the four (input) test image pairs. Since many 'disc' regions coincide with object boundaries, hence improvement of depth estimation performance in these types of image regions can directly translate to an improved overall depth-based blurring performance and so needs to be addressed as part of future work. Moreover, the error-rate for our outputs is much higher for the 'Teddy' and 'Cones' pairs (whose ground truth depth maps have 60 disparity levels each) than the 'Tsukuba' and 'Venus' pairs (whose respective ground truth depth maps have 16 and 20 disparity levels). Hence, there exists a definite scope of betterment of our method for scenarios requiring large number of depth levels.

4.4 Comparison with State-of-the-Art Algorithms w.r.t Serial Execution Time

To compare the running time of our algorithm with those of state-of-the-art stereo disparity estimation algorithms, we partly adopted the approach described in a very recent work by Tippetts et al. [5]. In their work, authors reviewed numerous published stereo vision algorithms, and evaluated their run-time performance. Their evaluation metric is 'millions of disparity evaluations per second' (Mde/s), which is calculated from the time to compute the disparity map by an algorithm for one frame, t, is given by Eq. 4, where W is the width and H, the height of the input image, and D, the number of disparity levels. Since the authors evaluated many of the reviewed algorithms on images of different sizes, it yielded different runtimes. So, values of execution time they reported, include the highest performance each algorithm achieved, along with the corresponding image size.

$$\text{Mde/s} = \frac{W \times H \times D}{t} \times \frac{1}{1{,}000{,}000} \tag{4}$$

The authors observed that multiple factors can influence runtime measurements of stereo vision algorithms, such as, the computational power of a CPU on which it is executed, programming language in which it is implemented, skill and effort of programmers in optimizing the implementation, parallelization techniques used, etc. Also, it is often necessary to make such a comparison when deciding which algorithm to implement for a given application. Thus, they included all published runtimes achieved by the stereo vision algorithms they reviewed, as well as all available hardware details to let the reader make such comparisons.

The authors also discouraged the practice of scaling runtime measurements directly by a processor clock speed, as it ignores other factors, such as processor pipelining, cache design, memory subsystem etc. while comparing running times of two or more algorithms implemented and executed on processors with widely varying configurations and clock speeds. Instead, they collected processor bench-mark values for each of the specific CPUs on which their reviewed methods were executed from the PassMark Software CPU benchmark table.[3] Using these values of CPU benchmarks, they calculated a normalizing factor for each of those CPUs, which were later used to scale the run-times of all the depth estimation algorithms running on those CPUs. In their performance comparison tables and graphs, they provided both un-normalized and normalized run-times of their surveyed methods to aid the process of comparison. However, they have not mentioned the particular formula or method using which the normalizing factor was calculated. Further, the processor on which we executed the implementation of our proposed stereo depth estimation method is having a different configuration and clock speed compared to the CPUs mentioned in their work. Hence, we report the un-normalized run-times of our method, along with our CPU configuration specifications and details of the corresponding input stereo pairs, in terms of width, height, and number of disparity levels of the test images, to let the reader make appropriate comparisons.

Table 8 shows results of comparison of running time of the implementation of our algorithm in MATLAB version 2013b in terms of Mde/s against some of the best performing methods which the authors surveyed and presented in their work [5]. In Table 8, we have grouped our results according to the input stereo image pair. We have sorted the entries within each such group on the decreasing order of Mde/s, following a similar approach of Tippetts et al. [5]. The last two entries of Table 8 are for a True-HD resolution stereo image pair comprising a stereo frame, which we processed using proposed algorithm and the results show its scalability.

[3] http://www.cpubenchmark.net/cpu_list.php.

Table 8 Comparison of proposed algorithm with state-of-the-art w.r.t computation time

Method	Run-time (seconds)	Input stereo image pair width × height (disparities)	Mde/s	Hardware
Proposed	1.07	450 × 375 (60)	9.4626	Core i7-2600 3.40 GHz
za [5]	1.6	450 × 375 (60)	6.3281	P4 3.0 GHz
we [5]	2.57	450 × 375 (60)	3.9382	Core 2 Duo 2.66 GHz
Proposed	0.84	434 × 383 (20)	3.9342	Core i7-2600 3.40 GHz
sd [5]	1.7	434 × 383 (20)	1.9556	N/A
sc [5]	1.79	434 × 383 (20)	1.8572	N/A
gp [5]	0.46	384 × 288 (16)	3.8467	Core 2 Duo 2.80 GHz
Proposed	0.57	384 × 288 (16)	3.1043	Core i7-2600 3.40 GHz
cg [5]	0.81	384 × 288 (16)	2.1845	P4 2.8 GHz
Proposed	12.985	1920 × 1,080 (60)	9.5815	Core i7-2600 3.40 GHz
Proposed	10.188	1,920 × 1,080 (16)	3.2565	Core i7-2600 3.40 GHz

Fig. 10 GUI showing depth levels present in disparity map generated by proposed method

While interpreting the computation time performance of our algorithm vis-à-vis those of other state-of-the-art algorithms presented in Table 8, we should not forget, that our method's entire implementation is in MATLAB, which is, largely, an interpreted language. Thus, it may often run comparatively slower than the C/C++/OpenCV implementations of other methods surveyed by the authors [5].

4.5 Results of Depth-Based Blurring for 'Tsukuba' and Real-World Stereo Pairs

Figure 10 shows our GUI, which lets users visualize the entire range of depth levels present in the depth map output by our method. From this GUI, the user can pick depth range(s) as per his (her) choice of region(s) or object(s) of interest, which he (she) wants to keep in focus. Accordingly, the blur map will be generated.

Next, we present the results of depth-based blurring on the Tsukuba image using depth maps output by our proposed algorithm, in Fig. 11. In the top-right image, only the lamp, and in the bottom-left one, only the statue-head and the table are in focus, as per input depths ranges of interest. Multiple in-focus depth ranges, like the bottom-right image extend the methods like [6]. It is interesting to note that the incorrect depth estimates of our algorithm in the texture-less and low illumination regions do not (noticeably) affect the depth-based blurring.

We also present the results of depth-based blurring on a real-world scene which we captured using our Sony HDR-TD10[4] 3D camcorder and later processed using our proposed method, in Fig. 12. The input stereo image pair was extracted from the frames of a short 3D (stereo) video clip shot using the said 3D camcorder, and only the left frame is shown in Fig. 12. It is interesting to note that even though no stereo rectification algorithm was applied to the extracted stereo image pair prior to its processing with our method, yet our output shows very few inconsistencies.

4.6 Results of Parallelization of Proposed Algorithm Using JTP and APARAPI

Lastly, we present results of parallelization of the computationally intensive, yet parallelizable steps of our depth extraction algorithm using CPU (Java

[4] http://store.sony.com/gsi/webstore/WFS/SNYNA-SNYUS-Site/en_US/-/USD/ViewProduct-Start?SKU=27-HDRTD10.

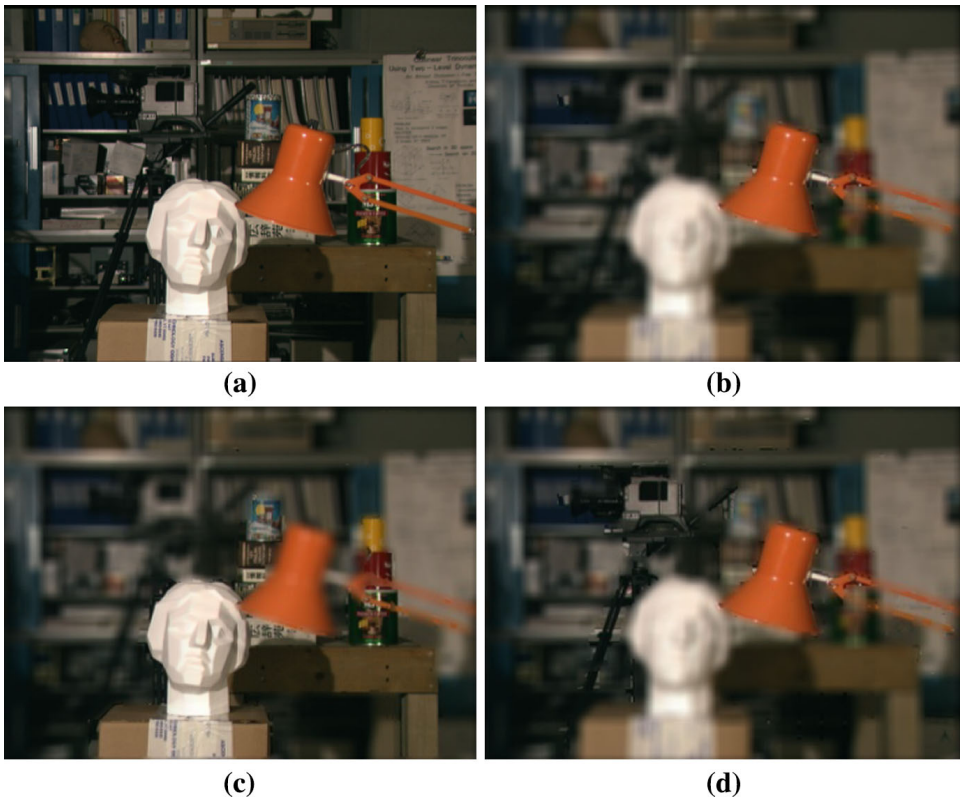


Fig. 11 Original image (**a**), depth-based blurring with single depth range in focus (**b, c**) and depth-based blurring with multiple depth ranges in focus (**d**)

Fig. 12 Original image (*left*) and depth-based blurring with single depth range in focus (*right*)

Table 9 Results of parallelization of proposed algorithm

No. of images	Serial time (ms)	JTP-only time (ms)	APARAPI-only time (ms)	JTP + APRARAPI time (ms)
150	73,191	32,396	17,331	13,822
200	96,361	42,391	22,982	16,595
250	116,909	47,070	28,514	20,192

Thread Pool) and GPU (APARAPI) separately, and then by combining them, in Table 9. The experiments were performed on a PC having an Intel *i*7-2600 quad-core @ 3.40 GHz CPU with 4 GB RAM and NVIDIA NVS 300 GPU having two compute units and 512 MB RAM. In the "combined" implementation, we invoked four JTP worker threads and two APARAPI kernels simultaneously, as it gave best results.

Table 10 Speed-ups Achieved by Parallelization of Proposed algorithm

No. of images	JTP-only	APARAPI-only	JTP + APRARAPI
150	2.3	4.2	5.3
200	2.3	4.2	5.8
250	2.5	4.1	5.8

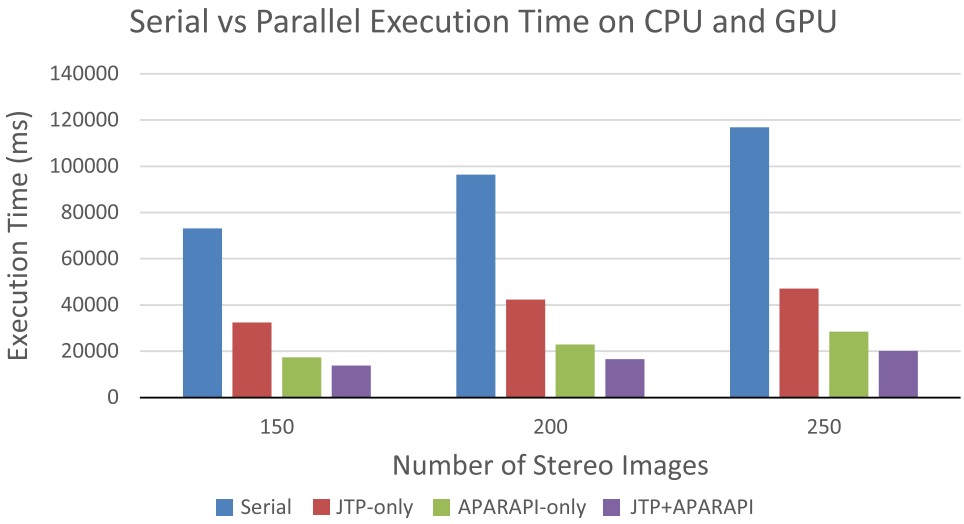


Fig. 13 Results of parallelization of proposed algorithm

The increase in processing speed obtained by parallelizing serial algorithms and running on CPUs/GPUs is quantitatively described by a metric called speed-up factor as shown in Eq. 5:

$$Speedup = \frac{Serial\ Execution\ Time}{Parallel\ Execution\ Time} \quad (5)$$

Speed-ups achieved over serial implementation by parallelizing our proposed algorithm and executing the serial and parallel versions on the CPU and GPU with an input stereo image sequence of 4,096 × 2,304 pixels each are given in Table 10. The maximum speed-up obtained is 5.8 for a sequence of 250 stereo images, for the CPU + GPU (combined) parallelization approach. We observed that, generally with the increase in the number of multi-processing units (cores) and the increase in the speed of each processing unit, the parallel executions become quicker and hence the speed-up increases following a sub-linear trend. It is not perfectly linear due to many causes, the most significant of which is a factor called parallelization overhead, which deals with coordinating activities of different processing units to collectively achieve a single task, like the distribution of total workload among all parallel processing units, collecting the intermediate processing results from each unit and compiling them to form the output.

From the graph of Fig. 13, we can easily infer that, rate of increase of running time of our proposed algorithm is greatest for the serial implementation, followed successively by the JTP, APARAPI and the JTP + APRARAPI (combined) ones, in decreasing order. This is because GPUs are more suitable for running data-parallel workloads than CPUs, and the combined approach utilizes both for parallelization.

5 Conclusion and Future Work

In this paper, we proposed a stereo depth estimation method using only 18 % pixels of either the left or the right image, which outperforms traditional methods like SAD and NCC by up to 33.6 % and a recent method developed by Zhang et al. [3] by up to 6.1 %. We also performed depth-based Gaussian blurring of image regions as per depths of users' non-interest. We showed that despite our algorithm's inaccurate depth estimates in texture-less and low-illumination regions, its blurring performance is not affected. Future work will focus on improving the depth estimates and the depth-map reconstruction technique.

References

1. Lazaros, N., Sirakoulis, G., & Gasteratos, A. (2008). Review of stereo vision algorithms: from software to hardware. *International Journal of Optomechatronics, 2*(4), 435–462.
2. Xiao, J. Xia, L., Lin, L., & Zhang, Z. (2010). A segment-based stereo matching method with ground control points. *International Conference on Environmental Science and Information Application Technology (ESIAT), 2010* (Vol. 3, pp. 306–309). 17–18 July 2010. doi: 10.1109/ESIAT.2010.5568363.
3. Zhang, Z., Wang, Y., & Dahnoun, N. (2010). A novel algorithm for disparity calculation based on stereo vision.

4th European Education and Research Conference (EDERC), 2010 (pp. 180–184). 1–2 Dec 2010.
4. Sunyoto, H., Van der Mark, W., & Gavrila, D. M. (2004) A comparative study of fast dense stereo vision algorithms. *IEEE Intelligent Vehicles Symposium, 2004* (pp. 319–324). 14–17 June 2004. doi: 10.1109/IVS.2004.1336402.
5. Tippetts, B., Lee, D., Lillywhite, K., & Archibald, J. (2013). Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing* 1–21. doi:10.1007/s11554-012-0313-2.
6. Popkin, T., Cavallaro, A. & Hands, D. (2011). Efficient depth blurring with occlusion handling. *18th IEEE International Conference on Image Processing (ICIP), 2011* (pp. 2585–2588). 11–14 Sept 2011. doi: 10.1109/ICIP.2011.6116193.
7. Hirschmuller, H. & Scharstein, D. (2007). Evaluation of cost functions for stereo matching. *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07.* (pp. 1–8) 17–22 June 2007. doi: 10.1109/CVPR.2007.383248.
8. Tombari, F., Mattoccia, S., Di Stefano, L., & Addimanda, E. (2008). Classification and evaluation of cost aggregation methods for stereo correspondence. *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008* (pp. 1–8). 23–28 June 2008. doi: 10.1109/CVPR.2008.4587677.
9. Abdollahifard, M., Faez, K., & Pourfard, M. (2009). Fast stereo matching using two stage color-based segmentation and dynamic programming. *6th International Symposium on Mechatronics and its Applications, 2009. ISMA '09* (pp. 1–6) .23–26 March 2009. doi: 10.1109/ISMA.2009.5164848.
10. Kim, C. (2005). Segmenting a low-depth-of-field image using morphological filters and region merging. *IEEE Transactions on Image Processing, 14*(10), 1503–1511. doi:10.1109/TIP.2005.846030.
11. Wang, X., Song, Y., & Zhang, Y. (2013). Natural Scene Text Detection with Multi-channel Connected Component Segmentation. *12th International Conference on Document Analysis and Recognition (ICDAR), 2013* (pp. 1375–1379). 25–28 Aug 2013. doi: 10.1109/ICDAR.2013.278.
12. Vishwanath, N., Somasundaram, S., Ravi, M. R. R., & Nallaperumal, N. K. (2012). Connected component analysis for Indian license plate infra-red and color image character segmentation. *IEEE International Conference on Computational Intelligence & Computing Research (ICCIC), 2012* (pp. 1–4). 18–20 Dec 2012. doi: 10.1109/ICCIC.2012.6510323.
13. Zirari, F.; Ennaji, A.; Nicolas, S.; Mammass, D. (2013) "A Document Image Segmentation System Using Analysis of Connected Components. *12th International Conference on Document Analysis and Recognition (ICDAR), 2013* (pp. 753–757) 25–28 Aug 2013. doi: 10.1109/ICDAR.2013.154.
14. Li, M., Zheng, X., Wan, X., Luo, H., Zhang, S., & Tan, L. (2011). Segmentation of brain tissue based on connected component labeling and mathematic morphology. *4th International Conference on Biomedical Engineering and Informatics (BMEI), 2011, 1*, 482–485. doi:10.1109/BMEI.2011.6098294.
15. Moftah, H. M., ella Hassanien, A. & Shoman, M. (2010). 3D brain tumor segmentation scheme using K-mean clustering and connected component labeling algorithms. *10th International Conference on Intelligent Systems Design and Applications (ISDA), 2010* (pp. 320–324). Nov 29 2010–Dec 1 2010. doi: 10.1109/ISDA.2010.5687244.
16. Bellala Belahbib, F. Z., & Souami, F. (2012). Color image segmentation by a genetic algorithm based clustering and Connected Component Labeling. *24th International Conference on Microelectronics (ICM), 2012* (pp. 1–4). 16–20 Dec 2012. doi: 10.1109/ICM.2012.6471432.
17. Choi, K. -S. (2012). Hierarchical block-based disparity estimation. *IEEE 1st Global Conference on Consumer Electronics (GCCE), 2012* (pp. 493–494). 2–5 Oct 2012. doi: 10.1109/GCCE.2012.6379668.
18. Zhu, S., & Yu, Y. (2012). Virtual view rendering based on self-adaptive block matching disparity estimation. *International Conference on Industrial Control and Electronics Engineering (ICICEE), 2012* (pp. 947–950). 23–25 Aug 2012. doi: 10.1109/ICICEE.2012.251.
19. Wang, Z. -F., & Zheng, Z. -G. (2008). A region based stereo matching algorithm using cooperative optimization. *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008* (pp. 1–8). 23–28 June 2008. doi: 10.1109/CVPR.2008.4587456.
20. Lu, D., & Du, Y. (2013). A two-step stereo correspondence algorithm based on combination of feature-matching and region-matching. *8th International Forum on Strategic Technology (IFOST), 2013, 2*, 51–55. doi:10.1109/IFOST.2013.6616858.
21. Tkalcic, M., & Tasic, J. F. (2003). Colour spaces: perceptual, historical and applicational background. *EUROCON 2003. Computer as a Tool. The IEEE Region 8 1*, 304–308. 10.1109/EURCON.2003.1248032.
22. Docampo, J., Ramos, S., Taboada, G. L., Exposito, R. R., Tourino, J. & Doallo, R. (2013). Evaluation of Java for general purpose GPU computing. *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2013* (pp. 1398–1404). 25–28 March 2013. doi: 10.1109/WAINA.2013.234.
23. Kolmogorov, V., & Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. *Proceedings of the Eighth IEEE International Conference on Computer Vision ICCV 2001, 2*, 508–515. doi:10.1109/ICCV.2001.937668.
24. Miled, W.; Pesquet, J. C. (2006). Disparity map estimation using a total variation bound. *The 3rd Canadian Conference on Computer and Robot Vision, 2006*. (p 48) 7–9 June 2006. doi: 10.1109/CRV.2006.28.
25. Scharstein, D., Szeliski, R., & Zabih, R. (2001). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision, 2001 (SMBV 2001)* (pp. 131–140). doi: 10.1109/SMBV.2001.988771.

BIODATA

Name:	Subhayan Mukherjee
Address:	122/1/1/3, Monohor Pukur Road, Opp. Elite Nursing Home, P.O: Kalighat, Kolkata - 700026, West Bengal.
E-mail:	subhayan001@gmail.com
Mobile:	9742138607
Qualification:	B.Tech Information Technology (West Bengal University of Technology, Kolkata, West Bengal) M.Tech Information Technology (National Institute of Technology Karnataka, Surathkal)