REAL-TIME ARBITRARY VIEW RENDERING FROM STEREO VIDEO AND TIME-OF-FLIGHT CAMERA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUĞRUL KAĞAN ATEŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2010

Approval of the thesis:

REAL-TIME ARBITRARY VIEW RENDERING FROM STEREO VIDEO AND TIME-OF-FLIGHT CAMERA

submitted by **TUĞRUL KAĞAN ATEŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members

Prof. Dr. Uğur Halıcı _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. A. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Uğur Topay, M.Sc. _____
TÜBİTAK SAGE

Date: **17.12.2010**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Tuğrul Kağan Ateş

Signature :

ABSTRACT

REAL-TIME ARBITRARY VIEW RENDERING FROM STEREO VIDEO AND TIME-OF-FLIGHT CAMERA

Ateş, Tuğrul Kağan

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. A. Aydın Alatan

December 2010, 69 pages

Generating in-between images from multiple views of a scene is a crucial task for both computer vision and computer graphics fields. Photorealistic rendering, 3DTV and robot navigation are some of many applications which benefit from arbitrary view synthesis, if it is achieved in real-time. Most modern commodity computer architectures include programmable processing chips, called Graphics Processing Units (GPU), which are specialized in rendering computer generated images. These devices excel in achieving high computation power by processing arrays of data in parallel, which make them ideal for real-time computer vision applications. This thesis focuses on an arbitrary view rendering algorithm by using two high resolution color cameras along with a single low resolution time-of-flight depth camera and matching the programming paradigms of the GPUs to achieve real-time processing rates. Proposed method is divided into two stages. Depth estimation through fusion of stereo vision and time-of-flight measurements forms the data acquisition stage and second stage is intermediate view rendering from 3D representations of scenes. Ideas presented are examined in a common experimental framework and practical results attained are put forward. Based on the experimental results, it could be concluded that it is possible to realize content production and display stages of a free-viewpoint system in real-time by using only low cost commodity computing devices.

Keywords: free viewpoint television, time-of-flight, graphics processing unit, image based rendering, video plus depth

ÖZ

İKİ GÖRÜNTÜLÜ VİDEO VE UÇUŞ SÜRESİ KAMERASI İLE GERÇEK ZAMANLI GELİŞİGÜZEL GÖRÜ İMGESİ OLUŞTURULMASI

Ateş, Tuğrul Kağan
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Aralık 2010, 69 sayfa

Bir sahnenin, çoklu görüntülerinden arada kalan görüntülerini oluşturmak, hem bilgisayarla görü hem de bilgisayar grafiği alanları için önemli bir iştir. Fotogerçekçi imge oluşturma, 3BTV ve robot yön güdümü, gerçek-zamanlı elde ediliyorsa, gelişigüzel görü imgesi bireşiminden faydalanan birçok uygulamadan bazılarıdır. Tüketicilere yönelik bilgisayar mimarilerinin çoğunluğu Grafik İşleme Birimi (GPU) adında, bilgisayarla oluşturulmuş görüntüler oluşturmak için özelleşmiş, programlanabilir yongalar içermektedir. Bu aygıtların veri dizilimlerini paralel olarak işleyerek yüksek işlem gücü elde etmede üstünlük sağlamaları, onları gerçek-zamanlı bilgisayarla görü uygulamalarında tercih edilir kılar. Bu tezin odağında, iki yüksek çözünürlüklü renk kamerası ile bir düşük çözünürlüklü uçuş-zamanı derinlik kamerası kullanan ve gerçek zamanda işleme hızını elde etmek için GPUların programlama tarzlarına eşlenik olan bir gelişigüzel görü imgesi oluşturma algoritması bulunmaktadır. Önerilen yöntem iki safhaya ayrılmaktadır. İkili görü ve uçuş-zamanı ölçümleri sayesinde derinlik hesaplama ilk safhayı oluşturur ve ikinci safha sahnelerin 3B gösterimlerinden ara görü imgesi oluşturulmasıdır. Sunulan fikirler ortak bir deneysel çerçevede incelenmiş ve elde edilen pratik sonuçlar ortaya konmuştur. Deneysel sonuçlara dayanarak, bir serbest bakış açısı sisteminin içerik üretimi ve görüntüleme aşamalarının düşük maliyetli hesaplama cihazları ile gerçek zamanda gerçekleştiriminin mümkün olduğu çıkarımına varılabilir.

Anahtar Kelimeler: serbest bakış açılı televizyon, uçuş zamanı, grafik işlemci birimi, imge tabanlı çizim, video artı derinlik

To My Beloved Wife

ACKNOWLEDGMENTS

I would like to express my gratitude and deep appreciation to my supervisor Prof. Dr. A. Aydın Alatan for his guidance, positive suggestions and also for the great research environment he had provided.

I would like to also express my thanks for their assistance to Sinan Çanga and Ezgi Can Ozan. Preparation of this thesis became less tedious with their valuable help.

I would like to thank my friends in Video and Audio Processing Group of Space Technologies Research Institute for such a friendly research environment they had provided. I have learned much from their experience and suggestions.

Finally, I would like to thank my wife, Emel, for her never ending love and support. This thesis is dedicated to her.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

3D television (3DTV) is a display system that offers depth cues beyond 2D planar images to enable perception of 3D vision in multimedia. *Free viewpoint television* (FTV) is another system for viewing video with freely adjustable 3D viewpoints to view the content. 3DTV and FTV together define a new inventory of commodity display technologies aiming enriched user experience surpassing those of conventional 2D displays [1]. Research on these devices combines computer vision, computer graphics, multimedia, human-machine interaction, optics and many other fields.

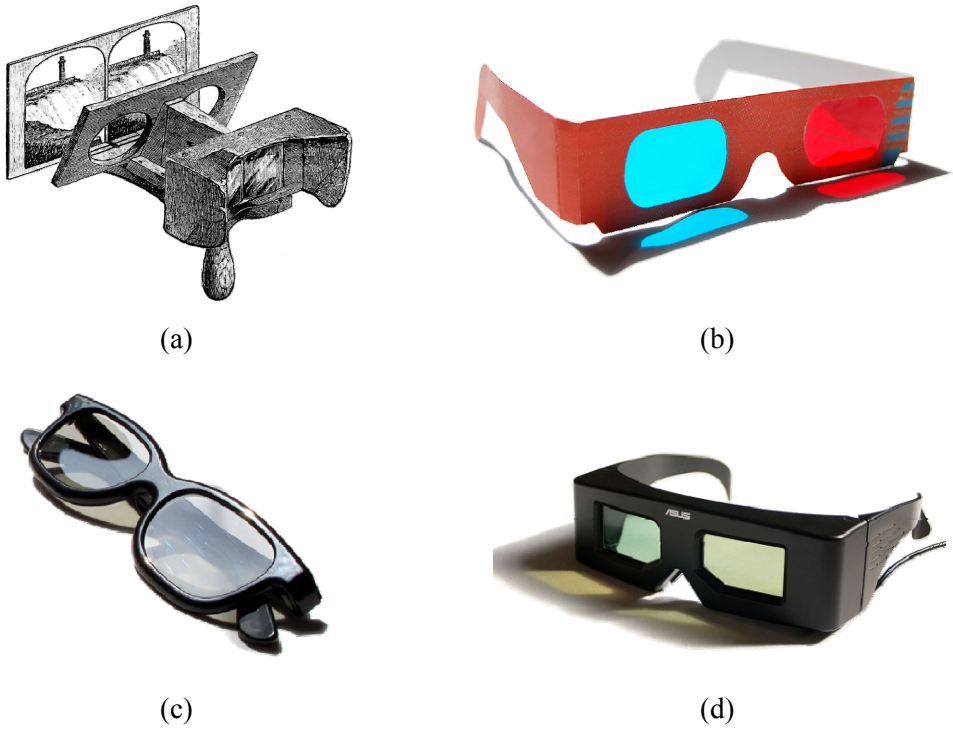
Arbitrary view rendering is generation of images for virtual cameras by using 3D video content and it can be accepted as a common task for certain FTV and 3DTV applications. Design of an arbitrary view rendering algorithm is closely related to content production, transmission and display needs along with corresponding technologies.

Devices and gadgets that offer 3D perception of planar images are invented as early as during 19th century [2]. A *stereoscope* offers depth perception from two close views of a scene via crossing of the eyes as illustrated in Figure 1a. 3D perception devices evolved from stereoscope to parallax stereograms and holograms of 20th century [3]. Contemporary technologies to display 3D moving pictures build upon foundations of these early attempts.

Stereoscopy refers to all 3D perception systems that employ two views of a scene to be perceived with respective eyes. *Anaglyphic stereoscopy* provides two views of the scene content with different colors. Specialized glasses with red-green or red-cyan spectral filters are used to pass each image into a different eye to stimulate binocular depth. *Polarized stereoscopy* solves the color loss problem of anaglyphic methods by filtering through passive polarizing filter glasses. A display system for polarized 3D requires two different types of projectors emitting lights at different orientation of oscillations. *LCD shutter stereoscopy* is a different color preserving and filter lens driven 3D technology that utilizes active viewing gadgets. LCD shutter displays emit two views of a scene with an alternating

sequence, synchronized to electronic shutter glasses which pass each view to its respective eye. Gadgets used for stereoscopic systems are shown in Figure 1.

Autostereoscopy refers to stereoscopic display technologies that do not require special gears for viewing 3D content. *Parallax barrier* devices achieve this effect by pointed lights at two eyes to see different views, if the viewer is positioned in a defined spot in front of the display. *Lenticular lenses* are specialized lenses that magnify different colors for different viewing angles. An array of lenticular lenses is used in an autostereoscopic 3DTV which provides different views of a scene when viewed from different angles.



(a) (b)

(c) (d)

Figure 1 (a) A stereoscope illustration from 1882. Modern variants are (b) anaglyphic, (c) RealD circularly polarized and (d) ASUS LCD shutter glasses (CC, Wikimedia Commons).

Free viewpoint television (FTV) or *freeview television* is any television setup which allows fine control of view angle and position through a remote control. Freeview paradigm adds an enhancement of user experience by adding interactivity and choice over the presentation of video content. FTV can be combined with either traditional 2D displays or 3D technologies discussed here if 3D content is supplied to the device in appropriate formats.

Free viewpoint, inherently, requires rendering of video content for requested arbitrary views. However, arbitrary view rendering can supplement display technologies even

without explicit viewpoint controls. Stereoscopic viewing technologies can benefit from arbitrary view rendering, if head-tracking is employed to estimate the viewing angle and position of the viewer; thus, interactive experience of 3D, which is similar to autostereoscopic displays, is achieved. On the other hand, autostereoscopic devices need to supply many different views of a scene to achieve acceptable results. Current lenticular lens based televisions on the market render up-to 46 views of video content. A feasible solution is dynamic generation of in-between views from multiview content captured with a less number of cameras.

Handhelds device manufacturers made it possible to watch television, movies and other video content in these smaller devices. Sizes of these devices make arbitrary view navigation easier than remote controlled display devices. Viewpoint input through fingers or device orientation [4] to view multimedia content leads to a new human machine interaction paradigm.

Advances in display technologies are coupled with advances in image capturing and generation technologies. 3DTV and FTV are made possible with new data acquisition devices and practices to create the three dimensional content required for novel display systems. Acquisition devices, which are aimed at obtaining 3D representations, are presented in Chapter 3, while Chapter 5 discusses different scene representation alternatives for transmission to 3D display systems. A broad comparison of 3D technologies and conventional broadcast systems is given in Figure 2.

1.1 Scope

Multimedia imaging for 3D video is an active topic where standards for content production, transmission and display are still being stabilized. This thesis work focuses on the scenario for 3D imaging where scenes are captured through a *stereo camera* pair and a *time-of-flight camera*, transmission is applied in compressed *multiview video plus depth* format and arbitrary view rendering is employed for 2D free viewpoint controlled display. Terminology is given in *image based rendering* context and strong emphasis is given upon *graphics processing units*. GPUs and their programming mindset enable high speed realization of the proposed algorithm. A summary of the scope of the thesis and comparison to more generic frameworks are given in Figure 2.

Result of this thesis is a novel framework, which combines several methods in the literature to achieve real-time 3D capture of scenes and their freeview display. Main contribution of

this work is a reinterpretation of these depth estimation and view rendering algorithms to match rendering paradigms of GPUs. Each component in the framework is presented in detail with related alternatives in the literature throughout the thesis.

In this thesis, a depth estimation algorithm through fusion of time-of-flight measurements, stereo matching and bilateral filtering is proposed. Two color images along with their estimated depth maps are used in the rendering algorithm which utilizes pixel-by-pixel post-processing filters in order to create a novel intermediate view. Both depth estimation and view rendering stages are designed to work in real-time with graphical processing units. Solutions to common problems for time-of-flight depth measurement and arbitrary view rendering are formulated taking restrictions of GPUs into account.

An experimental benchmark is also developed as a part of the thesis and results are presented in this work.

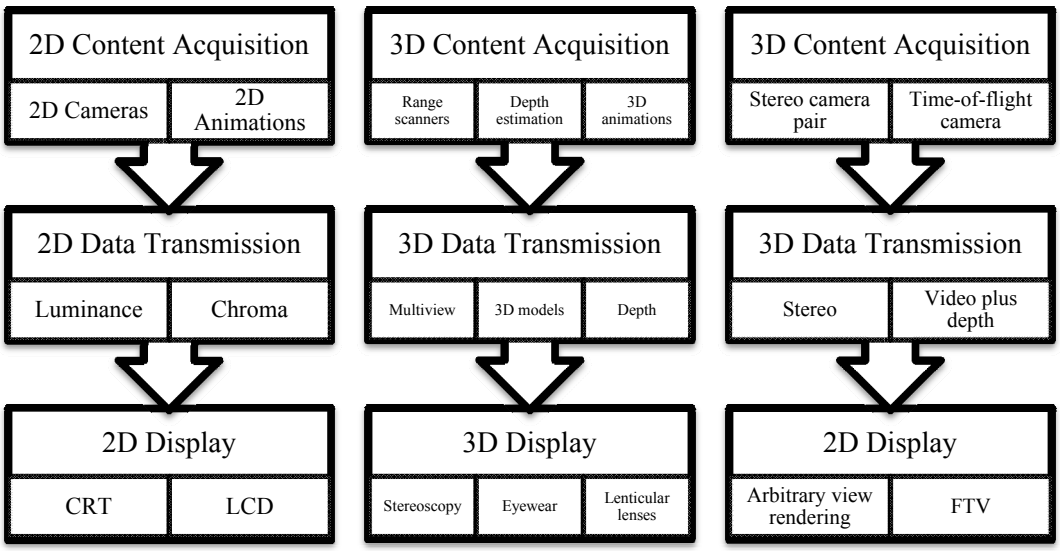


Figure 2 (a) Conventional television broadcast architectures versus (b) 3DTV architectures. (c) Arbitrary view rendering architecture presented in this thesis.

1.2 Related Work

In *computer vision* terms, McMillan and Bishop [5] regard arbitrary view rendering as the estimation of the values of the *plenoptic function* [6] for a certain viewing angle. The plenoptic function maps viewing parameters, namely viewing angle, scene location and

time, to the amount of light intensity that can be seen. Their *plenoptic modeling* approach achieves generation of arbitrary views from projections of interpolation of plenoptic function observations.

Levoy and Hanrahan [7] decomposes source or target views into light fields that produce colors. *Light field rendering* creates intermediate images for aligned views from a database of color observations of image regions grouped by the orientation of the light rays with respect to camera axes.

Yang et al. [8,9] generalizes the *plane sweep* algorithm for stereo disparity estimation to multiview case, provides an extension to estimate scene views and offers novelties for optimal utilization of graphics processing units for solving the freeview rendering problem. Weaknesses of their algorithm at discontinuities are solved by approximative occlusion handling [10].

Full extraction of 3D information from multiview images is offered in works of Yaguchi [11] and Ito [12] where volumetric methods are employed to form model based representations of scenes from correspondences between images, which are required for the camera *calibration problem* presented in Chapter 2. Arbitrary views are generated by rendering obtained models in 3D.

Research in free viewpoint television has led to *multiview video plus depth*, a content format presented by Smolic et al. [13]. Arbitrary view rendering is accomplished by warping color plus depth data from source views to target views.

Jung and Koch [14] offers ray casting on volumetric data to render in between views for full parallax displays of autostereoscopic televisions. Graphics processing units are utilized to accelerate generation of huge amounts of data required for proper operation of these displays.

In *computer graphics* field, view dependent texture mapping, image based modeling and virtual reality are some of many problems which require generation of arbitrary views using several photographs of the scene. These problems are solved under *image based rendering* context.

Szeliski [15] provides an image mosaicing algorithm for registration and resampling of views and Kanade et al. [16] employs multi baseline stereo algorithm for generation of arbitrary depth and color views, both providing discussions on virtual reality applications.

Debevec et al. [17,18] tries to combine traditional model based approaches with image based algorithms for realistic graphics rendering and presents *projective texture mapping* algorithm for view dependent mapping of scene textures. Buehler et al. [19] builds a common framework for view dependent texture mapping algorithms through *unstructured lumigraph rendering* algorithm.

Robotics is a field where generation of depth or color maps for arbitrary views other than mounted camera orientations is utilized. Uyttendaele et al. [20] offers arbitrary view rendering for virtual exploration of real world environments with the help of camera mounted mobile robots. Tanaka et al. [21] applies view rendering to obtain 3D navigation models from images captured with a mobile robot.

Research incorporating computer vision algorithms and time-of-flight cameras are vast. *3D4YOU* is a single and highly related EU funded ongoing project for establishing 3DTV techniques utilizing ToF sensors and high resolution cameras. Scope of this project spans from content acquisition and transmission to 3D displays [22].

1.3 Outline

This thesis is organized into seven chapters, four of which focus on a single important subject of the overall work. These chapters are followed by experimental and conclusive discussions.

Image based rendering, presented in Chapter 2, is a catalogue of algorithms dealing with creating novel views of scenes from their image based representations, such as taken photographs. It is a highly active topic with common grounds in computer vision and computer graphics and is presented in detail for accurate description of rendering methods employed in this thesis. Methods to infer 3D scene information from their 2D projections are given in image based rendering framework. Inverse projection within pinhole camera model, the core problem occurring in arbitrary view rendering is explained and solutions proposed in literature are given in stereo correspondence context. Afterwards, applications that use image based rendering are discussed.

Time-of-flight cameras, presented in Chapter 3, are relatively new, low spatial resolution range sensors capturing planar depth information of scenes at real-time rates. In this thesis, they are used to obtain supplementary depth cues for 3D modeling of scenes, prior to arbitrary view rendering. Working principles of ToF range sensors are explained and their drawbacks are exposed with respective proposed solutions.

Graphics processing units, presented in Chapter 4, are highly parallel microprocessor chips specialized for digital computer graphics rendering. Ever-increasing throughput rates of commodity GPUs and their flexibility for adapting new paradigms via stream programming capabilities make them the ideal choice for 3D content production and display stages presented in this thesis. Design principles of GPUs are explained to achieve understanding of the rendering process and discussions on GPU programming are given.

Central discussion on the proposed method of this thesis is explained in Chapter 5, on foundations of prior chapters. Depth estimation through stereo correspondence is discussed and methods to fuse time-of-flight depth information are presented. After content production via depth estimation, rendering from video plus estimated depth data for 3D display is elaborated.

Chapter 6 provides a presentation of the experimental framework developed as part of this thesis work and puts forward experimental results obtained.

Finally, Chapter 7 begins with a summary of the thesis and interpretations of obtained results are given afterwards. The thesis ends with a discussion on possible future extensions to this work.

CHAPTER 2

IMAGE BASED RENDERING

Realistic rendering is a commonly sought task under computer graphics and computer vision fields when digitized representations of artificial scenes are required. Realistic rendering is achievable with a realistic description of the scene with added details and complex modeling of illumination.

Image based rendering is a category of algorithms that perform modeling and rendering using image descriptions of the scene. These images, commonly in the form of photographs or video streams, provide content for scene description, which becomes complex enough to enable realism yet too compact to be used with little further processing.

This chapter introduces image based rendering, the backbone of the ideas presented in this thesis. The first section elaborates on the topic and gives comparison to other rendering methods. The second section models the mechanics between scenes and their photographic representations and introduces the inverse projection problem whereas the third section extends this discussion to stereo imaging case. Finally, the fourth section discusses applications in which image based rendering becomes useful.

2.1 Rendering and Approaches

In computer graphics, the term *rendering* denotes the process of creating digital images with computers, using the description of a scene, artificial or real-world. Rendering process is traditionally accepted as the final step in any computer graphics application following the production of the scene description.

Digital images created by digital rendering can have a variety of forms. In this work, digital rendering approaches which produce artificially realistic images from 3D scenes are considered.

Scene description consists of the geometry of objects in the scene, their material and texture properties, lighting sources, global shading details and eye geometry of the viewer. Computer representations of these descriptions differ from application to application. Differences in different approaches become eminent when comparing methods for representing objects in the scene. Object surfaces or volumes can be represented with meshes, equations, normal vectors, gradients, color projections and so on. Each approach excels in a different setting of production and rendering task. For some applications, it may be natural to choose one representation over another, because content to be rendered is already available in the corresponding format, e.g. volumetric rendering for computed tomography (CT) or magnetic resonance imaging (MRI) data [23,24]. On the other hand, some applications may utilize certain approaches to suit the needs of rendering stage, such as user interactivity and animation [25]. A general framework for rendering from model based scene descriptions is given in Figure 3.

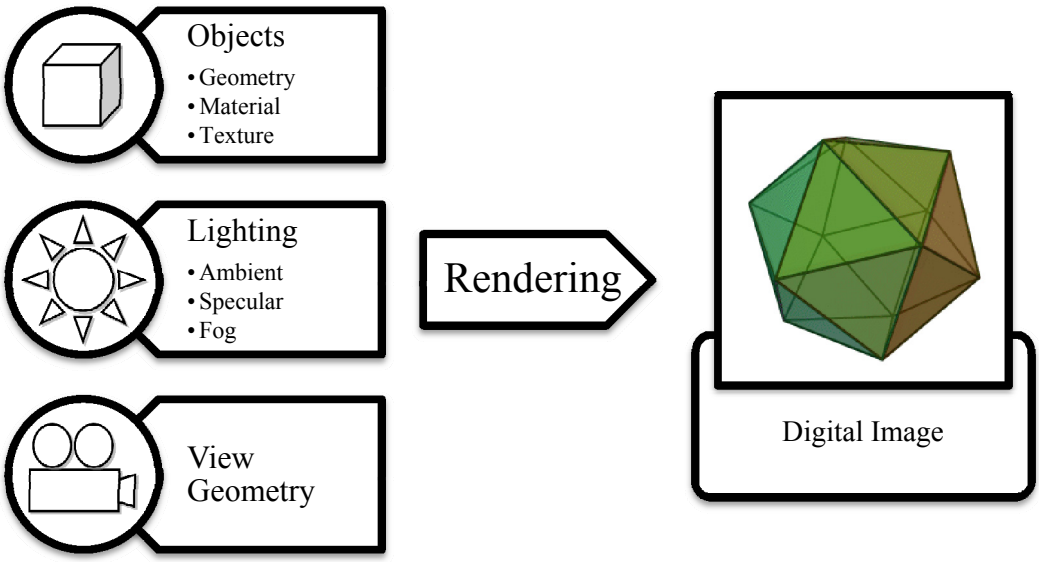


Figure 3 Model based rendering from scene description. (icosahedron image, CC, Wikimedia Commons).

For real-time computer graphics, conventional approach to scene representation has been modeling the surfaces of scene objects with geometric primitives, such as triangles or quadrilaterals [26,27]. Rendering begins with transforming object representations in 3D surface meshes to view coordinates and then to 2D projection coordinates. After objects or portions of objects that fall outside the viewing volume are culled, remaining geometry is

divided into block fragments on which coloring is applied. Finally, these fragments are placed onto the target image. This rendering pipeline is commonly used by most, if not all, of the real-time computer graphics applications [28]. Primary reason for this choice is the availability of mature hardware support in terms of graphical processing chips called *graphics processing units*. These chips are standard components of all commodity computers on the market today. Since utilization of these chips for rendering purposes is one of the primary focuses of this thesis, detailed discussion on the topic is presented in Chapter 4.

By definition, rendering a three dimensional scene into a digital image requires three dimensional geometry representation of the scene. Scene descriptions with 3D models have this information inherently in terms of 3D geometric constructs, such as surface equations and meshes. However, 3D information is also inherent to the photographs or 2D projected representations of scenes and extracting or estimating this information has been one of the main problems of photogrammetry and computer vision fields [5]. Thus, still images can provide the necessary information for describing three dimensional scenes for rendering into digital images. *Image based rendering* (IBR) methods are those that utilize 2D photographic representations of scenes to render them.

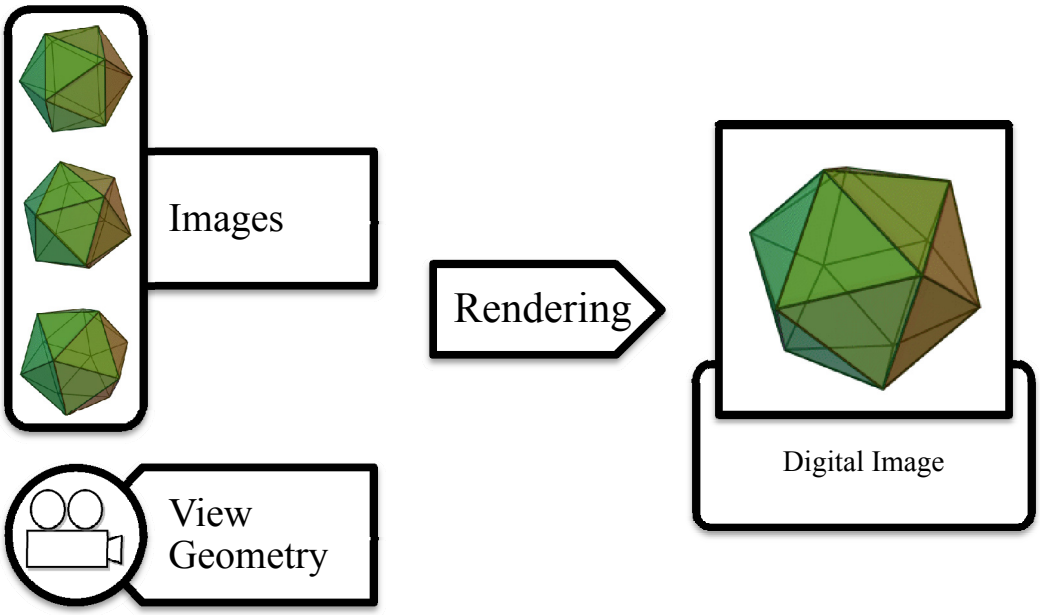


Figure 4 Image based rendering (icoashedron images, CC, Wikimedia Commons).

Image based rendering algorithms perform rendering of digital images from a collection of scene images. Target digital image can be defined as the estimation of the artificial image

to be captured if a camera is positioned at a certain view location and orientation. Thus, IBR provides a solution to arbitrary view generation problem. Figure 4 illustrates image based rendering.

2.2 Pinhole Camera Model and Inverse Projection Problem

One common property of different image based rendering methods is the generation of the final image by reverse mapping of every target pixel to each of the source images supplied to the system. Reverse mapping process is explained by the *pinhole camera model*.

Pinhole camera model defines the relation between 3D scene coordinates and their 2D projected correspondences. It is an ideal approximation model where camera aperture is assumed to be dimensionless, distortions caused by the actual physical lenses that perform real world projections are omitted and projection plane is parallel to aperture plane. Projections are usually regarded in front of the camera to work with aligned images as shown in Figure 5. The pinhole camera model is defined by (1) and (2) in a homogeneous right-handed coordinate system. In a homogeneous coordinate system, scalar multiples of a single vector represent the same point and in these equations, scalar equality is denoted with triple bars.

$$p = \boldsymbol{I}\boldsymbol{E}P \tag{1}$$

$$\begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \alpha_x & 0 & -\mu_u & 0 \\ 0 & \alpha_y & -\mu_v & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} & \boldsymbol{R} & & \boldsymbol{T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

In (1) and (2), P denotes the 3D object coordinate of any point on the scene and p denotes the projected point on 2D target image coordinates, u and v. $\boldsymbol{E}$ represents the 4×4 matrix for extrinsic camera transformation which is composed of 3×3 camera orientation matrix $\boldsymbol{R}$ and 3×1 camera translation vector $\boldsymbol{T}$. This matrix transforms scene points to camera coordinates so that the origin is located at the center of the imaginary pinhole where all rays are collected. Coordinate axes match those of principal camera directions in $\boldsymbol{E}$. $\boldsymbol{I}$ represents the 4×4 matrix for intrinsic camera transformation, in which $\boldsymbol{\alpha}$ combines focal length (f) and pixel scale in a single vector and $\boldsymbol{\mu}$ represents the principal point on image coordinates.

This matrix performs *perspective transformation* of 3D coordinates in camera space to 2D projected points on image space.

Any scene point, if its 3D coordinates are known, can be projected to any source or target image with known camera matrices. Thus, pixels can be related in between different images.

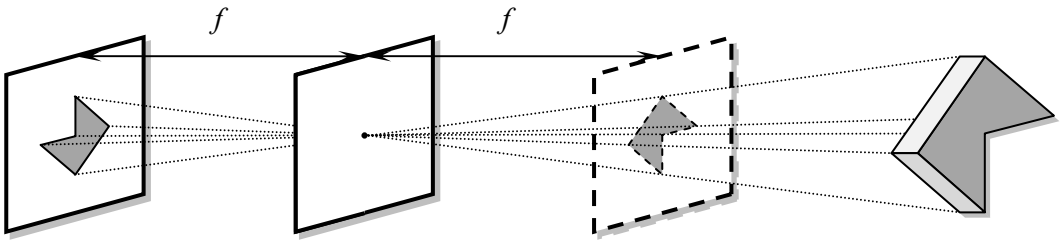


Figure 5 Perspective projection with pinhole camera model. Placing projection plane behind the pinhole results in flipped reflections. Hypothetical projection plane produces non-flipped images of scenes.

Calculating camera models for image sources is not a humane task. Extrinsic matrices cannot be predicted accurately, even if intrinsic camera parameters are provided by the actual hardware, unless actual images taken by the cameras are used. Deduction of camera parameters from image sets is called the *calibration* problem. If some image coordinates from sets of images are accurately known, i.e. some pixels among images can be matched, unknown camera parameters can be estimated with linear regression [29]. Solutions can be refined by iterative approximation or coordinate filtering [30]. Source coordinates can be obtained by the help of a simple pattern with certain easy-to-extract features, e.g. a checkerboard.

Time-of-flight sensors provide low resolution intensity maps, which makes calibration with other cameras possible. However, methods [31,32,33] exist that utilize depth information obtained from these cameras to obtain more accurate calibration results.

Analyzing pinhole camera model helps understand difficulties in most computer vision problems. It is clear from (2) that the reverse transformation requires the explicit knowledge of depth value *z'*. In other words, the 3D image coordinate at which a pixel is to be rendered does not have a unique 3D representation in scene space. Instead, we have an infinite number of candidate coordinates along the line which passes through the pixel on

the projection plane and the origin of the camera space, making *inverse projection* an *ill-posed* problem [34,35,36].

2.3 Stereo Correspondence

Stereo correspondence is a particular and well studied [37,38,39,18,40,41,42,43] related problem, where only corresponding points, rather than their 3D geometry, between different images are needed. If the geometry between two cameras is known beforehand, every coordinate on one image can correspond to any coordinate on the *epipolar line* of the other image. This line is the projection of the solution set of inverse projection problem onto the second image plane as shown in Figure 6.

Block matching is one approach [40,41,42,43] to relate pixels from one image to another in which blocks of pixels under corresponding epipolar lines are compared to each other. Correlation coefficients between raw pixel values inside neighborhood blocks provide a similarity measure to choose correspondences.

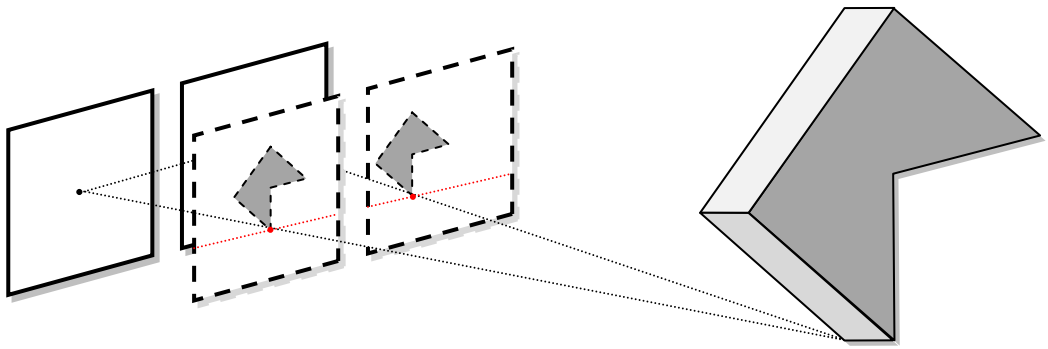


Figure 6 Stereo correspondence for a scene point between a pair of views and respective epipolar lines for each view.

Corresponding epipolar lines for all pixels can be made horizontal by applying *rectification* transformation on images. In this manner, every pixel obtains a horizontal distance to its correspondence, resulting in a *disparity map* which is easier and faster to obtain than a generic correspondence map for some applications [42,44,45,33,46]. Some methods [47] utilize hierarchical partitioning additionally to achieve further speed-ups.

Free viewpoint image generation can be satisfied by finding correspondence pairs for some, if not all, pixels on the target view. These *sparse estimation* methods generally triangulate a mesh between generated pixels and interpolate intermediate pixels by morphing [12].

Figure 7 shows an example to sparse correspondence pair matching. The focus of this thesis remains on *dense estimation* methods [41,44,48], which are more suitable for parallelization with graphics processing units [42,9,8]. Dense methods produce correspondence estimates for all pixels, possibly with a *confidence map* indicating the reliability of each estimate.

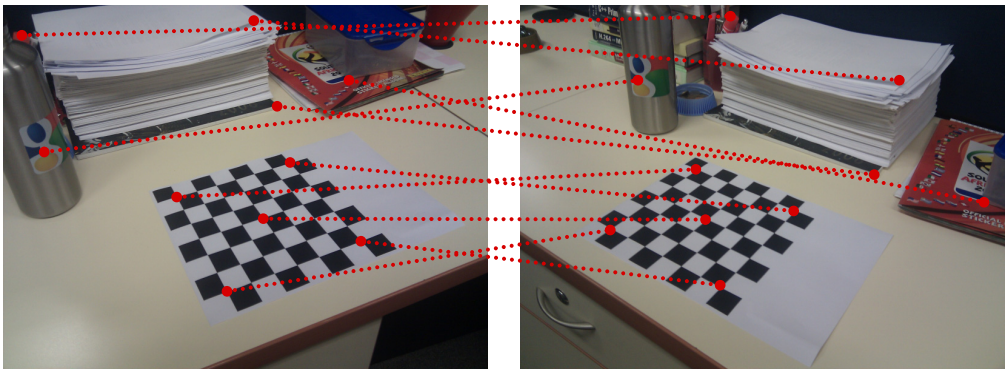


Figure 7 Several stereo correspondence pairs for two views of the same scene.

Disparity and confidence map results can be combined and pruned at a global approximation stage which tries to minimize a cost function over collected data and *smoothness* constraint [49]. Smoothness constraint is trivial to impose upon color and disparity; however, some approaches utilize optical flow as an additional constraint for moving images [45].

The methods utilizing cost minimization to solve stereo correspondence problem differ in the actual minimization algorithm used. Several Bayesian variants [50,36] as well as nonlinear diffusion [51] and graph cut approaches [52,53] exist. Global approximation methods can be backed up with a priori information obtained from time-of-flight cameras [31,54].

Stereo correspondence can also be performed over candidate depth values as shown in Figure 8. This way, correspondence problem can be extended to multiple image case by *plane sweeping* algorithm [9,8,55] where correspondences for all source images are sought at once rather than over several binary pairs. Scharstein and Szeliski [49] provide a detailed taxonomy and survey on dense stereo and correspondence algorithms.

A stereo correspondence result, either a disparity value or 2D coordinate pair on different image planes, originate from a point on 3D scene space and this point is the intersection of

light rays passing through two correspondence points and respective camera origins. Distance of this 3D point to any source or target view is called the *depth* of the point to the image and provides the geometric information needed for image based rendering.

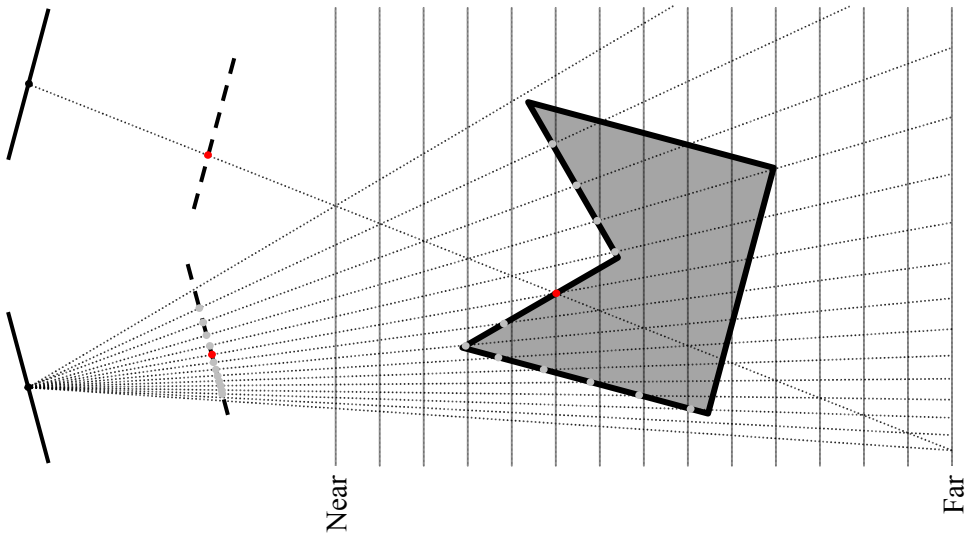


Figure 8 Plane sweeping for stereo depth estimation. A number of depth values are tested by projecting image points on the first view to the second view.

2.4 Applications

Image based rendering methods have higher applicability for rendering problems where photographs of the scene are already available. Intermediate view generation, as discussed in the introduction of this thesis, is one problem where a scene is reconstructed from a set of photographs for an arbitrary viewpoint. Intermediate view generation has an important use in *free viewpoint television*.

Photographs are realistic representations of the scene which they include. Although model based methods are achieving ever-increasing photorealistic results, further realism can be obtained with image base rendering methods in computer graphics [5]. Mixed reality, where artificial objects are augmented into real-world scenes or vice versa, is an application in which an increased realism leads to better user experience. Image based methods to mixed reality provides rendering of virtual objects that look more natural inside real-world photographic scenes [56]. IBR can also be utilized to render real-world scenes captured from multiple cameras onto other real world objects [57]. Photographic nature of both

elements increases the applicability of image based modeling to such virtual reality applications.

Image based rendering methods can also be utilized to render *textures*, a vital supplement for model based graphical applications, to achieve realism. Projective texture mapping [17] generates more realistic view-dependent textures to be used for either image or model based rendering methods. IBR can also be used to build detailed 3D models [58,59] with image registration and mosaicing [15] and to create shadow and lighting effects [60] for further photorealistic rendering.

Since image based rendering depends on creating new images from available ones, new views can also be created by previously generated output images in addition to the initial source images [7].

Image based rendering for arbitrary view generation is a vital component of this thesis work. There are several approaches to arbitrary view generation problem with image based rendering.

Light field rendering [7], provides a solution to arbitrary view rendering problem by enforcing source images taken from cameras aligned on a single focal plane. Patches from the scene projected onto this focal plane are collected in a database along with their projection coordinates and angles. They are referenced when patches with similar angular properties at similar focal plane coordinates are requested. Free viewpoint image generation with light field rendering has limited view ranges due to focal plane alignment constraints.

View dependent texture mapping [18] and *projective texture mapping* [17] can realistically generate an intermediate view by texturing a scene model with image based rendering techniques.

Image based rendering also makes intermediate view generation possible if images are accompanied by *depth maps*, which are single channel images representing the distance of each pixel from the scene to the view. *Video plus depth* [61,62,63,13] is a broadcast standard covering free viewpoint image synthesis. Depth required for each view can be obtained with range sensors, such as *time-of-flight cameras*, or extracted with *stereo correspondence*. Details on video plus depth can be found in Chapter 5 while next chapter introduces time-of-flight cameras.

CHAPTER 3

TIME-OF-FLIGHT CAMERAS

Depth sensing is a fundamental task in both computer vision [31] and computer graphics [22]. Human machine interaction, virtual reality, robotic navigation, object reconstruction, intermediate view synthesis and many similar applications benefit from existing depth information of a world scene. Depth maps, if not present, can be extracted from inherent 3D information inside accompanying color information.

Given two or more images from separate views of the same scene, *stereo correspondence* can lead to an estimate to the world coordinates of pixel locations on the images. However, state of the art techniques still try to overcome problems occurring at boundaries and textureless regions [31]. Object boundaries present occlusions where correspondence between images may not exist and textureless regions render block matching approaches useless in finding pairs between images.

A hardware solution to the drawbacks of stereo matching algorithms is *time-of-flight sensors,* which are relatively new and superior to previous hardware solutions in many ways [22]. Common examples of time-of-flight solutions to depth sensing on the market is *photonic-mixer-devices* of PMDTechnologies [64] and MESA Imaging SwissRanger cameras [65] as seen in Figure 9.

Time-of-flight sensor used in this thesis work is SwissRanger SR-3000. It provides a depth map of QCIF resolution (176x144), a distance range of 7.5 meters, depth accuracy in the order of a few centimeters at a speed of up to 30 frames per second and by using a default modulation frequency of 20MHz on 850nm infrared light waves [66].

This chapter introduces the time-of-flight cameras which are used in conjunction with color cameras to achieve free viewpoint image synthesis. The first section elaborates on the working principles of ToF sensors; the second section presents the drawbacks of their depth sensing mechanism and their solutions and workarounds; the third, and the last section

provides a discussion on application areas which benefit from real-time planar depth sensing with ToF cameras.

3.1 Working Principle

Time-of-flight is a measure of time for an object, particle or wave to travel a distance through a medium. It can be used to find the duration, the distance or the medium properties of the travel, given other parameters.

Time-of-flight cameras are sensors which measure distance to scene geometry by recording the time light travels from the device and gets reflected back. These are relatively new devices where whole screen depth is captured at once, in comparison to devices in *light detection and ranging* (LIDAR) category which operate with single rays at a time.

(a) (b)

Figure 9 (a) SwissRanger SR-4000 by MESA Imaging and (b) PMDvision CamCube (CC, Wikimedia Commons)

Time-of-flight cameras have two main operations that take stage together. *Illumination* sends light rays with light emitting or laser diodes, preferably at infrared ranges to prevent visible effects. *Sensing* collects emitted rays through a lens and optical filter and the time light rays spent travelling is measured for each pixel on a two dimensional array of light sensing layer based on *charge coupled device* (CCD) principle [67].

Time required for light rays to hit scene objects and return back is proportional to the distance of the objects to the camera. Accurate time measurements are obtained with *continuous wave modulation* method [68,65] where emitted sinusoidal modulated light wave is cross-correlated with the captured wave by demodulation of the incoming signal [65]. Since modulation frequency is known, phase difference leads to time difference.

Operation inside a medium of known refractive index, such as air, enables convert time differences to scene distances.

Emitted modulated signal and the captured signal can be represented by the equations (3) and (4), respectively. Phase offset (φ) is determined by the duration which light travels outside the sensor. Incoming signal has an additive constant component to model background illumination and amplitude component (α) modeling power loss due to light absorption.

$$g(t) = \cos(\omega t) \tag{3}$$

$$s(t) = 1 + \alpha \cos(\omega t - \varphi) \tag{4}$$

Mathematical expression for the cross-correlation between incoming and outgoing signals is defined by (5) [68]. Phase offset between signals can be calculated by sampling at certain phases via (6). The same procedure can be applied with utilizing synchronous sampling and DFT [68].

$$c(\tau) = (s * g)(\tau) = \frac{\alpha}{2}\cos(\varphi + \tau) \tag{5}$$

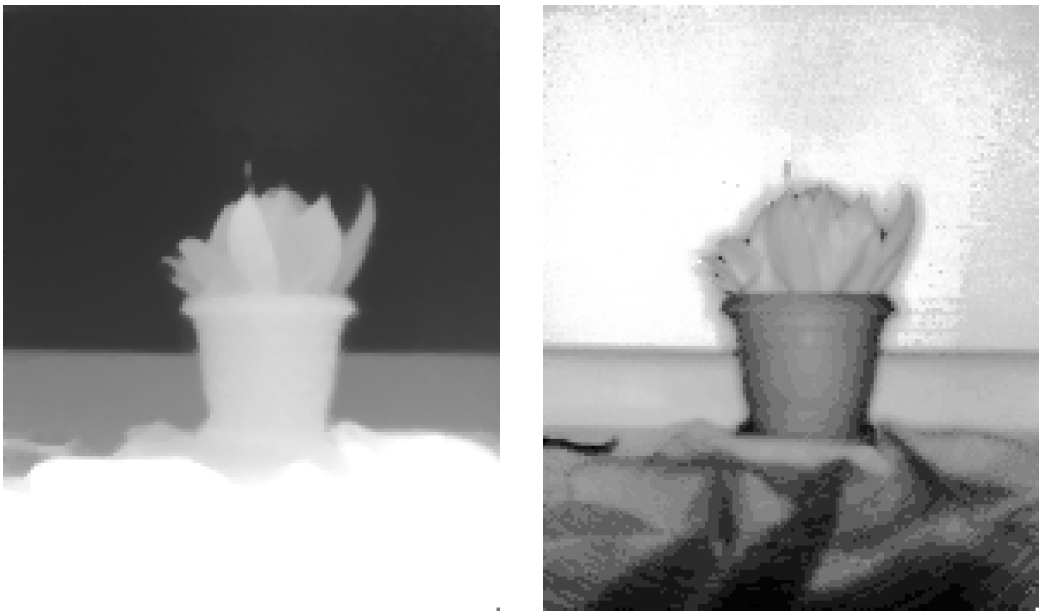
$$\varphi = \tan^{-1}\left(\frac{c(270°) - c(90°)}{c(0°) - c(180°)}\right) \tag{6}$$

Time-of-flight sensors can both record the time traveled in air using phase offset and returning intensity of light rays emitted from the device. These lead to scene depth and intensity correspondingly [68] as shown in Figure 10.

3.2 Challenges

Time-of-flight cameras provide a robust solution to depth sensing problems occurring in many computer graphics and computer vision applications. Although ToF sensors eliminate problems commonly faced in passive extraction methods, new challenges inherent to the nature of the time-of-flight mechanism are unfortunately introduced.

Time-of-flight cameras provide depth images at a rate equal to or greater than real-time speeds. However, their pixel resolution is very low compared to contemporary color cameras. Although highest reported resolution is 484x648 [69], most cameras provide only ten thousands of pixels each frame. One approach to increase resolution of depth images obtained from range sensors is upscaling. Modern graphics processing units provide hardware support for *bilinear filtering* which may be satisfying, but *bilateral filtering* [70] is shown to perform better [71].



(a) (b)

Figure 10 (a) Depth and (b) intensity maps acquired with SwissRanger SR-3000 camera. Brighter values indicate less depth and more intensity.

Cross-correlation for phase estimation requires sampling of the incoming signal at least four times. This introduces *motion artifacts* where depth values at object boundaries become erroneous for a dynamic scene [72].

Time-of-flight principle introduced previously depends on generation of ideal sinusoidal modulated signals and accurate sampling for cross-correlation. Since obtaining perfect results are not possible for both, acquired depth image has an error component caused by the sensor mechanism itself. Kolb et al. [72] call this component the *systematic error* of the camera and presents a theoretical discussion on a possible solution [22]. A solution can be extending the sinusoidal correlation model to multiple Fourier coefficients but complexity of circuitry for demodulation of higher frequencies and required sampling rate will

increase. Some solutions by means of look-up tables [73] and B-splines [74] are also proposed.

Other challenges related to the mechanics of time-of-flight cameras include intensity related distance error and depth inhomogeneity [22]. Moreover, utilization of multiple ToF cameras may lead to interference for older models [72].

3.3 Applications

Time of flight cameras are utilized mainly for content acquisition for computer vision applications. Accurate disparity estimation is possible with incorporation of time-of-flight cameras with traditional stereo methods [44,31,54]. Generation of layered depth video [75] content is also subject to improvements with initial depth estimates from ToF sensors [76]. ToF sensors can also be utilized in estimation of patchlets, image patches with surface normals [77].

Mixed reality [78,79], user tracking [80] and gesture recognition [81] are some of human machine interface related applications that utilize environmental sensing nature of these depth cameras to provide interactive experience for larger systems.

CHAPTER 4

GRAPHICS PROCESSING UNITS

Graphics processing units (GPU) are specialized microprocessor chips intended to accelerate computer graphics applications. GPUs existing in modern commodity computers provide hardware support for dedicated floating point operations, 3D transform and lighting, primitive rendering and framebuffer manipulation enabling real-time high quality computer graphics.

Hardware support for computer graphics is dated back to the emergence of first graphical displays on computers. Earlier chips that provided acceleration for common computer graphics tasks are ANTIC of Atari and VIC series of Commodore, both being graphical components of 8-bit personal computer architectures [82]. These graphical chips provide page flipping, sprite and mixed text-bitmap support, common to many 2D graphical rendering chips. Successor 2D graphics devices provide *bit block image transfer* (bitblit) [83] allowing combining multiple raster images with binary and arithmetic operators at once.

Although evolution of 2D acceleration chips still continues, these devices are primarily found in low performance handheld devices today. Modern usage of *graphics processing unit* denotes 3D integrated hardware found in personal computers. One of the first devices to support complete support for 3D rendering pipeline is GeForce 256 of NVIDIA [84] with support for transform and lighting, triangle setup with clipping and rendering 10 million polygons per second.

Contemporary 3D acceleration chips have hardware capabilities for image filtering, programmability for certain stages in rendering pipeline and digital video decoding; thus, extending their scope beyond simple computer graphics rendering. *General purpose computing on graphics processing units* (GPGPU) is an emerging paradigm that converts general engineering applications to computer graphics context to achieve high parallel computation power on GPUs. In contrast to other computing alternatives, such as FPGAs or

DSPs, GPUs exist on all commodity personal computers. In this manner, developed software using GPUs for high performance demanding applications are ready to run without accompanying dedicated hardware. Besides, GPUs have mature software and driver support. OpenGL and Direct3D are two major computer graphics software libraries whose design and development is tightly coupled with those of mainstream GPU models.

This chapter provides information on graphics processing units and their programmability detailed to the extent of understanding the ideas of the scope of this thesis. The first section introduces the fixed graphics pipeline existing in modern GPUs. The second section focuses on programming the rendering pipeline to achieve user controlled behavior of these chips. Finally, the third chapter provides a brief discussion on compute unified frameworks which can utilize GPUs with general computing paradigms.

4.1 Graphics Pipeline

Rendering is a broad concept of converting scene description into digital images. Each rendering application is distinguished by the format of scene description and the actual process in which target image is formed. Chapter 2 provides detailed discussion on rendering concepts with a focus on image based rendering.

Modern GPUs provide rendering support around a single data streaming framework called the *3D graphics pipeline* [28]. GPUs act as a *stream processor* which transforms incoming 3D object descriptions into pixel operations on the target image according to preset *rendering state*. Multiple data flow in one direction to another, from 3D representation to pixel values, hence the name pipeline. Figure 11 provides a flowchart of the data flow in graphics pipeline.

Objects on the scene to be rendered are described with surface primitives, such as triangles or quadrilaterals and texture images that define their surface appearance [26,27]. Rendering begins with the supply of surface primitives to the GPU. A primitive is defined by its vertices and its description lies within *vertex units* which encapsulate 3D location, surface normal vectors, texturing and coloring information and material properties.

Transform and lighting is the initial stage of rendering where vertex units are transformed through various coordinate spaces and colored using per vertex properties and global lighting state. Vertex locations are transformed from object coordinates to first scene coordinates and then to view (eye) coordinates consecutively. This transformation is

accomplished with the *modelview matrix*, which is part of the rendering state. Vertex coloring is accomplished with global ambient lighting, directed light sources and environment fog. These concepts are called *vertex operations* and defined inside the *vertex kernel* of the rendering pipeline.

Primitives within the rendering pipeline can be filtered out to eliminate redundant primitives or new primitives can be generated to increase rendering quality. Final primitives are projected to target image coordinates through the *projection matrix* of the rendering state. Projections can either be *orthographic* or *perspective* depending on the application. Orthographic projection preserves Euclidean ratios in scene description and is suitable for *computer aided design* applications and scenes with complex 2D descriptions. Perspective projection emulates the biological eye and other lens driven cameras where distant objects project onto smaller portions of the target image. Enumeration and projection of primitives are called *geometry operations* and defined inside the *geometry kernel* of the pipeline.

Primitive operations are followed by *rasterization* in which 3D geometry is converted to 2D in the form of rendering *fragment units*. Each fragment unit corresponds to a single pixel on the target framebuffer and embodies rendering information including color, depth, texture coordinates and transparency. During rasterization, vertex information is spread by interpolation in between primitive corners. Linear interpolation is a common approach to achieve realism in this stage.

Fragments can be re-colored after texturing. Target pixels on the framebuffer are modified according to fragments and pixel operations defined in the rendering state. These operations can be copying, alpha compositing, selection (min, max) or a combination of these. Conversion of fragments to pixel values is called *fragment operations* and these are defined inside the *fragment kernel* of the pipeline.

Standard pipeline is controlled through graphics APIs, such as OpenGL and Direct3D, which modify the rendering state and supply input to the pipeline. Applicability of the standard rendering pipeline to problems other than graphics rendering is limited due to its fixed nature. However, when combined with programmable parts of the GPU, rendering pipeline provides an efficient framework for most applications. Computer vision applications, in particular, can benefit from the hardwired algorithms inside the GPU for common tasks between computer vision and graphics.

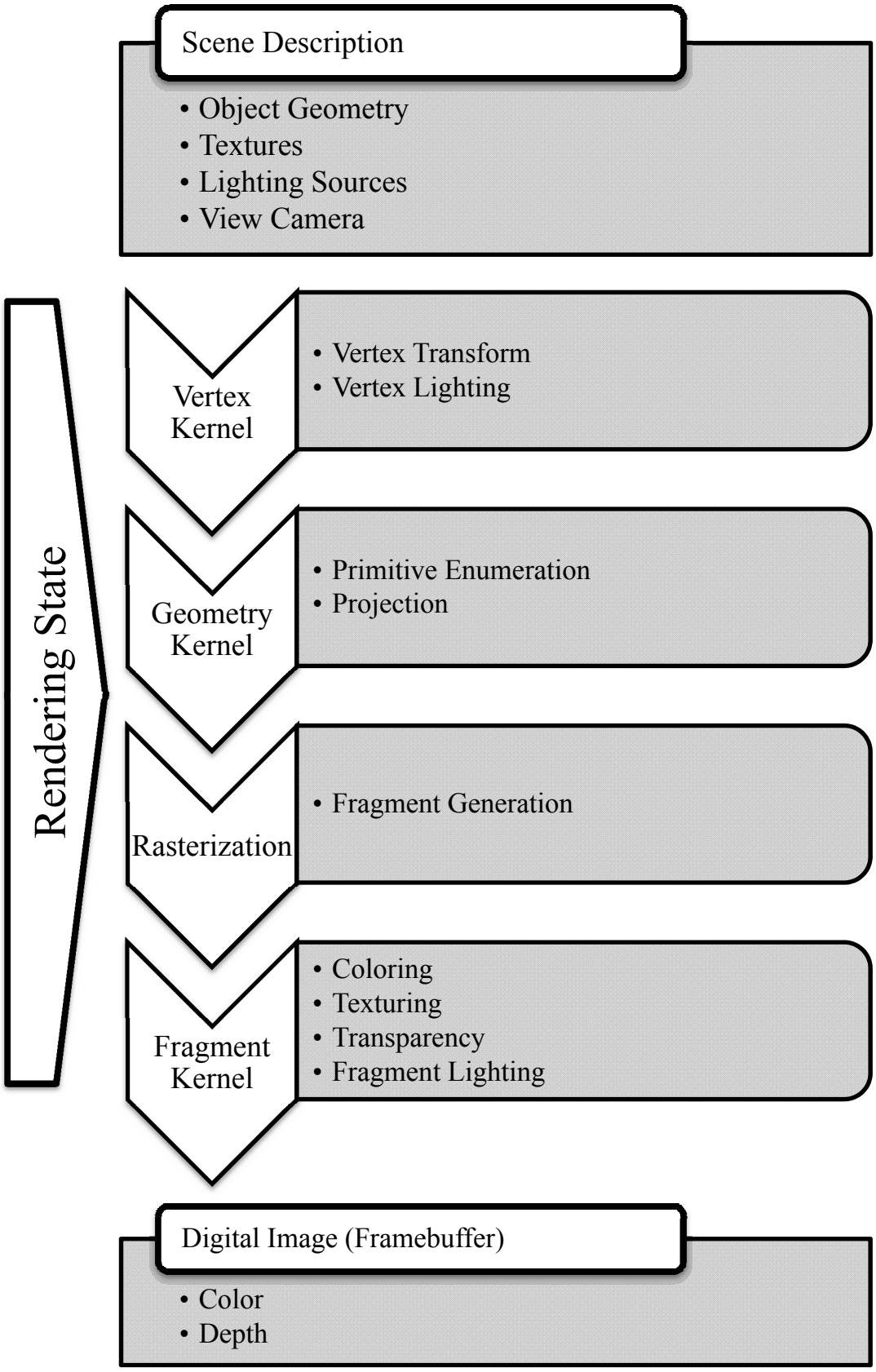


Figure 11 Graphics rendering pipeline.

4.2 Programmable Pipelines

3D graphics pipeline defines a single set of mind concerning rendering of 3D scenes. Its design intent is the realization of real-time computer graphics where speed is the main concern. It is this requirement, which makes the design of the graphics pipeline strictly connected to advances in microelectronics. However, real-time computer graphics becomes possible for different rendering strategies as advances in computing technologies surface.

Backward compatibility concerns limit changes in hardware and software design for 3D rendering, allowing evolution only in smaller steps. On the other hand, advance leaps in software and hardware technologies make certain rendering strategies obsolete and others more optimal. This dilemma is solved by GPU manufacturers with added support for programmability over certain stages of the rendering pipeline.

Programmability for vertex and fragment operations are introduced in 2003 [85]. User defined behavior for rendering is achieved with *shaders*, codes that are sent to GPUs to control rendering. A *vertex shader* is piece of software that controls per vertex operations, such as geometric and projection transformations, vertex coloring or lighting. It can be used to emulate lens distortions and provides faster control for dynamic objects. A *fragment shader* is a code that performs per fragment operations. Realistic rendering techniques such as *per pixel lighting* and *projective texture mapping* [17] are made possible with programmable fragment kernels.

Programmability of geometry kernels is possible with a *geometry shader*, which is introduced in 2007 [85]. Geometry shaders provide support for dynamic generation and suppress of 3D geometry on the GPU.

Other programmable parts of the pipeline are texture lookup and tessellation. The first one is responsible for generating filtered colors from texture images. A *texture shader* stands for user defined behavior for texture lookup, opening possibilities for further image processing support than simple linear filtering. Tessellation, the other programmable stage of the pipeline, performs division of primitives to smaller tiles and transformations on these tiles. A user provided *tessellation shader* provides fine control over the complexity of scenes with respect to other parameters, such as view distance or object size. Both shaders are introduced in 2010 [85].

OpenGL introduces *separate shader* objects in 2010 [85] to combine characteristics of all shaders into a single framework. Separate shaders are software codes that can define different parts of the rendering pipeline and can be injected into any programmable kernel.

Programming shaders is possible with constructs similar to machine language where each command maps to primitive operations on the GPU, although common approach is using high level shading languages, such as Cg, HLSL and GLSL. Idioms and paradigms used in programming shaders are common stream processing fundamentals, such as map, reduce, scatter or gather [28,86]. Owens et al. [28] provide detailed discussions on technologies and techniques related to programming the rendering pipeline.

4.3 Compute Unified Architectures

Programming GPUs provides a highly cost efficient parallel computing framework for many fields as well as increased flexibility for computer graphics applications. On the other hand, the fixed rendering pipeline can perform only a single rendering algorithm with tight limits on configurability and it is replaced by custom algorithms through shaders for real-time computer graphics. Therefore, it is not surprising to find GPUs to be regarded as specialized computing devices rather than programmable graphics accelerators only. Indeed, latest OpenGL standard [87] deprecates fixed pipeline functionality and recommends rendering through user supplied software instead.

Increased demand for graphics processing units for general purpose computing and the blurring of the differences between specialties of the programmable parts of these devices help them evolve into all purpose high performance computing devices. While differences between GPUs and CPUs, standard microprocessors of commodity computers, reduce to the degree of parallelization, a new paradigm called *compute unified programming*, which combines these two processors, surfaces.

A general purpose computing architecture, *Compute Unified Device Architecture* (CUDA) is introduced in 2006 by NVIDIA [88]. CUDA is a programming framework including a programming language, a compiler suite and accompanying hardware support within NVIDIA's GeForce series GPUs. It allows C-like written programs to execute with both CPUs and GPUs; thus, abstracting graphics nature of GPU programming from users.

A vendor independent and open compute unified architecture called *Open Computing Language* (OpenCL) is standardized by Apple Inc. and Khronos Group [89], obtaining full hardware support from vendors in 2008.

Compute unified architectures perform efficient division of work for both GPUs and CPUs providing remarkable acceleration through parallelization [90]. Applications which utilize compute unified architectures and are related to arbitrary view synthesis quickly emerged. Fast depth upsampling [91] and disparity estimation [45] are shown to benefit from availability of compact parallel computing architectures.

CHAPTER 5

PROPOSED METHOD

Arbitrary view rendering is the generation of images of a scene that are not available during capture. Real image or video captures of the scene are used to create virtual realistic views with image based rendering. Generation of these views is a common task for 3DTV applications.

Still photographs lack the explicit information which defines the locations of pixels in 3D scene space. Extracting this information is an ill-posed problem in the sense that no unique 3D scene coordinate exists for any pixel on the photograph. Problems in creating arbitrary views propagate from these simpler computer vision concepts.

Arbitrary views are rendered using depth information accompanying color in source views. Extracting this information can be in the form of depth warping or estimation through stereo vision. A fusion of these provide better results [54].

Although color and depth provide a compact description of the 3D scene, this description is not complete for arbitrary views due to occlusions and dissimilar viewing volumes. Robust measures have to be taken to *fill-in* absent information.

This chapter, step-by-step, builds an arbitrary view algorithm that incorporates time-of-flight depth measurements, stereo depth estimation and rendering from video plus depth data. The first section introduces the *video plus depth* content format which enables a broadcast system for arbitrary view rendering. Following section formulates the problem of generation of video plus depth data and arbitrary view rendering through this content format. The third section presents how depth maps for stereo views are estimated through depth warping, stereo matching and bilateral filtering. The fourth section is the crucial point where arbitrary view rendering from all obtained information is accomplished. The final section mentions implementation details of the whole system in GPU.

5.1 Content Format

Arbitrary view rendering is an important task for *3D television* (3DTV) where viewers receive and perceive 3D content broadcasted in appropriate formats. Numerous methods exist to describe 3D content for 3DTV applications [92]. *Surface based representations* describe 3D data in the form of surface geometry. Volumetric representations provide dense or hierarchical organization of volume pixels, also called *voxels*. *Texture representations* describe content in several texture images which form the scene geometry when warped around 3D objects. *Image based representations* provide compact descriptions of scenes to be broadcasted in the form of images from different views without support for explicit 3D geometry. *Layered depth images* (LDI) [75] is one format with the idea of supplementing color data with depth layer tags. Alatan et al. provides a detailed survey on different representations for 3DTV broadcast and rendering [92]. Schiller and Koch [93] provide a summary of data structures for scene representations obtained with time-of-flight cameras.

Video plus depth defines an image based video format suitable for a variety of 3D applications, specifically the 3DTV [61]. Color video is supported with accompanying depth information through content generation, coding, broadcast and decoding. Since depth maps are technically single channel images as seen in Figure 12, their introduction is analogous to introduction of color to television. Broadcast is still decodable with older televisions where depth information is only interesting for supporting 3D setups.

Video plus depth format also preserves applicability of existing techniques for 2D broadcast. Data hiding, for example, can be accomplished by 2D watermarks added to color maps and methods exists to decode watermarks from arbitrary views [94].

Video plus depth format does not suffer from back-projection problem since its depth component inherently exposes 3D scene coordinate of each pixel. Synthesis of an arbitrary view given a single color and depth image pair can be accomplished by projecting every pixel from their scene coordinates to target image coordinates preserving color information. Occlusion gaps can be prevented by warping a surface mesh instead of a point cloud.

High quality intermediate view synthesis is achieved with *multiview video plus depth*, which describes the scene with color and depth for several views. Smolic et al. [13] provides a framework for correcting artifacts occurring after warping of several source views. Gaps in intermediate views are filled with neighboring pixels and outliers are eliminated with color consistency measures to obtain good synthesis results.

Figure 12 Single frame example for multiview video plus depth content format taken from ballet studio sequence of Zitnick et al. [95]. Each row of the figure shows the color and depth map of a single viewpoint.

5.2 Problem Formulation

A viewpoint is a combination of intrinsic and extrinsic camera parameters as given by the *pinhole camera model* presented in Chapter 2. A viewpoint V, can be mathematically represented by $\boldsymbol{P}_V$, a 4×4 projection matrix of a 3D homogeneous projective transformation as given in (7) and (8).

$$\boldsymbol{P}_V = \boldsymbol{I}_V \boldsymbol{E}_V \tag{7}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \\ z' \end{bmatrix} = \boldsymbol{P}_V \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{8}$$

Extrinsic transformation matrix transforms space coordinates in 3D, from scene space to camera space. Points in front of the camera will have negative z-coordinates due to right handedness of the coordinate system. This z-coordinate z', is made positive and passed as a factor by the intrinsic camera matrix, resulting in an irreversible perspective division as given in (9) and (10).

$$\begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \\ 1 \\ z' \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} \equiv \begin{bmatrix} x' \\ y' \end{bmatrix} / z' \tag{10}$$

Perspective division is irreversible in the sense that depth values of projected points are lost and pixels in image space cannot be transformed back to scene space. This is called the *back-projection problem* and discussed in detail in Chapter 2. However, if depth values of image coordinates are known, inverse of $\boldsymbol{P}_V$, provides a reverse transformation from image space to scene space as given in (11) and (12), thus back-projection problem is solved.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V^{-1} \begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \tag{11}$$

$$\boldsymbol{P}_V^{-1}\begin{bmatrix} u \\ v \\ 1/z' \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V^{-1}\begin{bmatrix} uz' \\ vz' \\ 1 \\ z' \end{bmatrix} \tag{12}$$

The fundamental goal of arbitrary view rendering is to estimate the hypothetical photograph to be taken if an actual camera was used to capture the scene from a specific view geometry. Multiview plus depth content format is a common ground between 3D broadcast applications and arbitrary view rendering is formulated around this approach.

This thesis covers content acquisition with two color cameras and a depth sensing time-of-flight camera. Color images obtained from the stereo pair provides a trivial approach to choosing views for video plus depth format. These color images are augmented with estimated depth maps to generate the transmission format for this arbitrary view rendering system.

Beginning with stereo viewpoints, L and R, their corresponding color images $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$, range sensor viewpoint ToF and its respective depth map $\boldsymbol{D}_{ToF}$, initial aim is to estimate the artificial depth maps, $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ at stereo viewpoints in order to have a complete stereo video plus depth representation of the scene. This stage, which is called depth estimation, is summarized in (13).

$$\boldsymbol{D}_L, \boldsymbol{D}_R = Depth\ Estimation(L, \boldsymbol{C}_L, R, \boldsymbol{C}_R, ToF, \boldsymbol{D}_{ToF}) \tag{13}$$

Estimation of $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ is a compound process in which both views from the stereo pair are utilized with stereo correspondence methods presented in Chapter 2. Elaboration on depth estimation is given in the next section.

Second stage of the algorithm is arbitrary view rendering from multiview video plus depth and this stage is intended to be implemented on display side of the broadcast system. Two views, L and R, along with their color and depth maps are transmitted and intermediate views are generated upon request. Given an intermediate viewpoint V, view rendering stage is formulated in (14).

$$\boldsymbol{C}_V = View\ Rendering(V, L, \boldsymbol{C}_L, \boldsymbol{D}_L, R, \boldsymbol{C}_R, \boldsymbol{D}_R) \tag{14}$$

Depth estimation deals with content acquisition stage of the whole system whereas view rendering is a display algorithm. Inner workings of these two stages are presented in the remaining of this chapter.

5.3 Depth Estimation

Under image based rendering terms, a depth map can be rendered for any view, given other views of the same scene. Stereo correspondence provides an estimate to depth, as given in Chapter 2.

Time-of-flight cameras also provide low resolution depth maps. Since depth maps are inherently 3D descriptions, they do not suffer the back projection problem and can easily be projected into other views. Therefore, depth maps obtained from such range sensors provide reference depth estimates to obtain video plus depth. However, upscaling of obtained depth maps is a problem, as discussed in Chapter 3.

This thesis combines ideas from these passive and active methods in order to obtain auxiliary depth maps for the stereo cameras used in content acquisition. $\boldsymbol{D}_{ToF}$, depth map obtained from time-of-flight camera is *warped* onto views *L* and *R* with measures to overcome occlusions. This provides an initial estimate to $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$. Then, and *bilateral filtering* and *stereo matching* between $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ is performed around this initial estimate to refine obtained depth maps.

5.3.1 Time-of-Flight Depth Warping

Depth map captured with a time-of-flight camera is represented with $\boldsymbol{D}_{ToF}$ and represents the surface of scene content visible from viewpoint ToF. Projection of any point on this surface to viewpoint V is carried out with (15) and (16).

$$\begin{bmatrix} u_V \\ v_V \\ 1/z_V \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V \boldsymbol{P}_{ToF}^{-1} \begin{bmatrix} u_{ToF} \\ v_{ToF} \\ 1/z_{ToF} \\ 1 \end{bmatrix} \tag{15}$$

$$z_{ToF} = \boldsymbol{D}_{ToF}(u_{ToF}, v_{ToF}) \tag{16}$$

Depth values of target viewpoint are given by (17). However, surface projection is not strictly a one-to-one mapping. Some pixels at the target depth map cannot be deduced and some others can have multiple projections due to occlusions. Multiple projection values can be singled out with depth testing and missing depth values can be interpolated from neighboring projections. Both of these solutions are provided by graphics processing units.

$$\boldsymbol{D}_V^{warp}(u_V, v_V) = z_V \tag{17}$$

Depth map projection can be implemented on the GPU by modeling the source depth map as a 3D surface in the form of a triangle mesh. Pixels of the depth map provided by the time-of-flight camera are converted to 3D vertices and triangles are formed in-between as shown in Figure 13 and (18). These triangles are first transformed to scene space and then projected onto target depth map. This operation is called *warping* and covers the entire target image leaving no gaps.

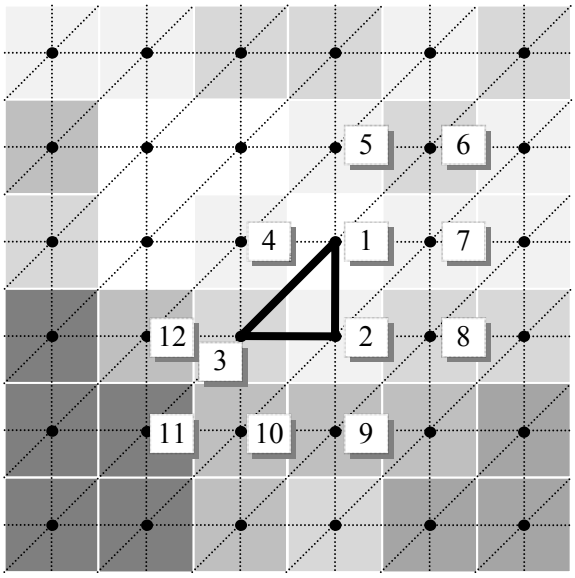


Figure 13 Triangulation of depth map into a surface mesh.

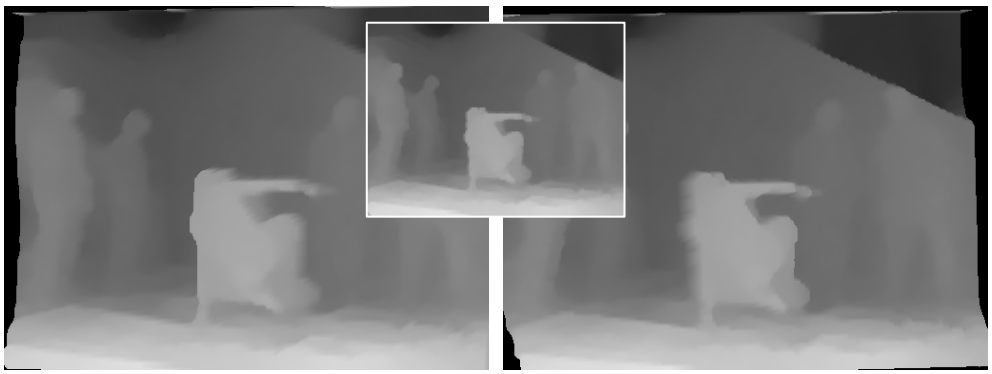


Figure 14 Warping result from a 128x96 depth map (middle) to arbitrary left and right viewpoints with 512x384 pixel resolution. Breakdancers sequence from Zitnick et al. [95].

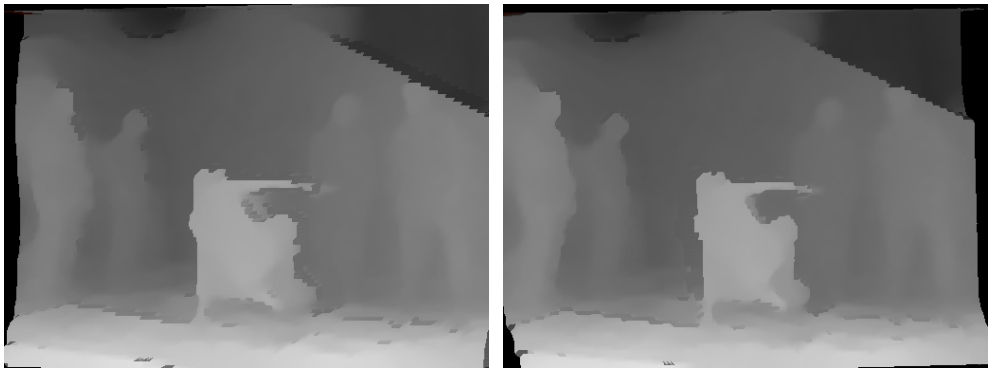


Figure 15 Depth map rendering with triangle suppression.

Projection of the connected surface mesh to the target view is a rendering flow which begins by a vertex shader which transforms all vertices in the mesh according to $\boldsymbol{P}_V \boldsymbol{P}_{ToF}^{-1}$. This operation aligns source mesh onto target viewpoint and rendering with depth test results in an estimated depth which does not have gaps at occlusions.

After mesh warping, discontinuities are covered by larger triangles which usually span from background to foreground resulting in large depth patches. Areas of the target depth map that are not visible to ToF are interpolated between background and foreground, as seen in Figure 14. These *rubber sheet* artifacts remain at discontinuities after warping and they can be removed with mesh segmentation techniques [96]. Problematic triangles are detected inside a geometry shader and they are suppressed to background to limit their favorability over foreground patches from other views. Depth selection process for triangles is given in (19) and (20). Depth of the farthest vertex among neighboring triangles, which are marked in Figure 13, is used as the suppression depth. Comparison area a' is an empirical value to

detect occlusion patches. If depth testing is enabled, using depth and color information from these patches becomes only a last resort, thus occlusions are handled. Figure 15 shows the effects of triangle suppression.

$$T = \left\{ \begin{bmatrix} u_1 \\ v_1 \\ 1/z_1 \\ 1 \end{bmatrix}, \begin{bmatrix} u_2 \\ v_2 \\ 1/z_2 \\ 1 \end{bmatrix}, \begin{bmatrix} u_3 \\ v_3 \\ 1/z_3 \\ 1 \end{bmatrix} \right\} \tag{18}$$

$$T' = \begin{cases} T & ; \text{area}(T) < a' \\ \left\{ \begin{bmatrix} u_1 \\ v_1 \\ 1/z_{max} \\ 1 \end{bmatrix}, \begin{bmatrix} u_2 \\ v_2 \\ 1/z_{max} \\ 1 \end{bmatrix}, \begin{bmatrix} u_3 \\ v_3 \\ 1/z_{max} \\ 1 \end{bmatrix} \right\} & ; \text{area}(T) \geq a' \end{cases} \tag{19}$$

$$z_{max} = \max(z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}, z_{11}, z_{12}) \tag{20}$$

Depth data coming from the time-of-flight sensor is warped onto L and R, resulting in initial estimates to $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$. These maps lack the finer depth details as a result of the low resolution output of the range sensor and they are refined further with stereo matching and bilateral filtering.

Incorporation of depth estimates from stereo matching and bilateral filtering is handled in a Bayesian framework that combines cost functions of different results. A cost function denotes the selection cost for all depth candidates for all pixels. Cost function for depth warping is built in a way that minimizes the cost of warped depth and increases the cost of other depth candidates as they deviate from this center. A quadratic formulation is used as suggested by previous works [54,71] and given in (21). Depth deviation factor (σ^{warp}) is an application specific constant.

$$Q_V^{warp}(u, v, d) = \frac{\left(\boldsymbol{D}_V^{warp}(u, v) - d\right)^2}{\sigma^{warp\,2}} \tag{21}$$

Given a cost function, a depth map is constructed with a winner-takes-all approach in which the depth value with the minimum cost is selected as given in (22).

$$\boldsymbol{D}_V(u,v) = \underset{d}{\operatorname{argmin}}\, Q_V(u,v,d) \tag{22}$$

The cost function constructed for depth warping always yields the original warped map. However this triviality is broken by the introduction of stereo matching and bilateral filtering.

5.3.2 Stereo Matching

In addition to the active range sensing device used, passive methods can also support the intermediate depth maps obtained. Stereo correspondence, presented in Chapter 2, is a set of algorithms for inferring depth from two different image based representations of the same scene. Fusion of depth maps from stereo correspondence and time-of-flight cameras provide even better depth estimates, surpassing both of these approaches [54].

Stereo matching is performed within the Bayesian framework of depth cost functions. Stereo matching cost function for left viewpoint is given by (23) and (24). In these equations, c is the block distance between two coordinates on stereo images, N is the block size parameter, δ is the distance between pixels in image space, D_s is the empirical depth search range and s is the Euclidean distance function for color comparison. All depth and color values are in range [0, 1] and practical values of D_s are around $1/20$.

$$Q_L^{stereo}(u,v,d) = \begin{cases} \min(c,1) & ;\ \left|\boldsymbol{D}_L^{warp}(u,v) - d\right| \le D_s/2 \\ 1 & ;\ \left|\boldsymbol{D}_L^{warp}(u,v) - d\right| > D_s/2 \end{cases} \tag{23}$$

$$c = \frac{1}{(2N+1)^2} \sum_{i=-N}^{N} \sum_{j=-N}^{N} s\big(\boldsymbol{C}_L(u+i\delta, v+j\delta), \boldsymbol{C}_R(u'+i\delta, v'+j\delta)\big) \tag{24}$$

$$s(\boldsymbol{A},\boldsymbol{B}) = \sqrt{(A_{red}-B_{red})^2 + \left(A_{green}-B_{green}\right)^2 + (A_{blue}-B_{blue})^2} \tag{25}$$

Stereo matching is performed on the portions of epipolar lines enforced by the search range by comparing pixel values. This is the *plane sweep* method shown in Figure 8. Pixels are compared with Euclidean distances in RGB color space as given in (25). Candidate depth values are bounded inside a depth search range to eliminate outliers [54]. Image coordinates on the right image (u', v') are calculated from image coordinates on the left image

(u, v) and depth candidate d. Projection matrices of stereo viewpoints are used to relate 2D projection coordinates from the left image to the right one as given in (26). Stereo search for the other direction is similar.

$$\begin{bmatrix} u_R \\ v_R \\ 1/d_R \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_R \boldsymbol{P}_L^{-1} \begin{bmatrix} u_L \\ v_L \\ 1/d_L \\ 1 \end{bmatrix} \tag{26}$$

5.3.3 Bilateral Filtering

Time-of-flight cameras provide real-time capture of depth information, unfortunately at low resolutions and up to a certain precision. Stereo matching is a well studied classical approach to depth estimation problem, which can be supplementary to the initial depth map obtained from range sensors. However, since stereo matching methods employ block matching to relate images, matching and therefore depth estimation performance at object boundaries and color discontinuities is not satisfying.

Bilateral filtering [70] is a low-pass image filtering algorithm which aims preserving edges while smoothing the image. An adaptive pixel filter kernel, which takes high values for similar neighbors, is applied to all pixels and therefore smoothing is prevented at edges. Similarity is defined both in color and spatial domain.

The idea of bilateral filtering can be extended to smoothing depth maps with respect to related color maps to align depth boundaries with color boundaries [71]. Depth values obtained from adaptive filter is found by (27) using similarity measures given in (28) and (29). Among similarity measures, w_c denotes color similarity and w_s denotes spatial similarity. Color similarity is the decay of Euclidean RGB distance and spatial similarity is the decay of pixel coordinate distance.

$$\boldsymbol{D}_V^{bilateral}(u, v) = \frac{\sum_{i=-M}^{M} \sum_{j=-M}^{M} \boldsymbol{D}_V^{warp}(u+i, v+j) w_c(\boldsymbol{C}_V, u, v, i, j) w_s(i, j)}{\sum_{i=-M}^{M} \sum_{j=-M}^{M} w_c(\boldsymbol{C}_V, u, v, i, j) w_s(i, j)} \tag{27}$$

$$w_c(\boldsymbol{C}, u, v, i, j) = e^{-\frac{s(\boldsymbol{C}(u,v), \boldsymbol{C}(u+i\delta, v+j\delta))}{\gamma_d}} \tag{28}$$

$$w_s(i, j) = e^{-\frac{\sqrt{i^2+j^2}}{\gamma_s}} \tag{29}$$

In order to fuse the depth map obtained from bilateral filtering with the results of depth warping and stereo matching, a cost function similar to Q_V^{warp} is constructed as in (30).

$$Q_V^{bilateral}(u,v,d) = \frac{\left(\boldsymbol{D}_V^{bilateral}(u,v) - d\right)^2}{\sigma^{bilateral\,2}} \tag{30}$$

5.3.4 Depth Cost Fusion

Initial depth estimate for stereo viewpoints from the time-of-flight camera leads to three different depth cost functions. These cost functions are summed with different weights as given in (31) and (32) and a fused cost function is obtained for both viewpoints, L and R.

$$Q_L(u,v,d) = Q_L^{warp}(u,v,d) + \alpha Q_L^{stereo}(u,v,d) + \beta Q_L^{biltareal}(u,v,d) \tag{31}$$

$$Q_R(u,v,d) = Q_R^{warp}(u,v,d) + \alpha Q_R^{stereo}(u,v,d) + \beta Q_R^{biltareal}(u,v,d) \tag{32}$$

Overall effect of stereo matching and bilateral filtering on the final depth cost function can be controlled with parameters α and β respectively. Final depth maps for left and right viewpoints are governed by the winner-takes-all approach given in (22). In this respect, depth estimations needed for multiview video plus depth format, $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ are obtained using the fused cost functions as in (33) and (34). It is seen in Figure 16, how utilization of stereo color maps restores finer object details on the estimated depth map and Figure 17 offers a general look at the whole depth estimation process.

$$\boldsymbol{D}_L(u,v) = \underset{d}{\operatorname{argmin}}\, Q_L(u,v,d) \tag{33}$$

$$\boldsymbol{D}_R(u,v) = \underset{d}{\operatorname{argmin}}\, Q_R(u,v,d) \tag{34}$$

Estimations to $\boldsymbol{D}_L(u,v)$ and $\boldsymbol{D}_R(u,v)$ along with $\boldsymbol{C}_L(u,v)$ and $\boldsymbol{C}_R(u,v)$ make up the stereo video plus depth data needed for arbitrary view rendering at the display side of the broadcast system.

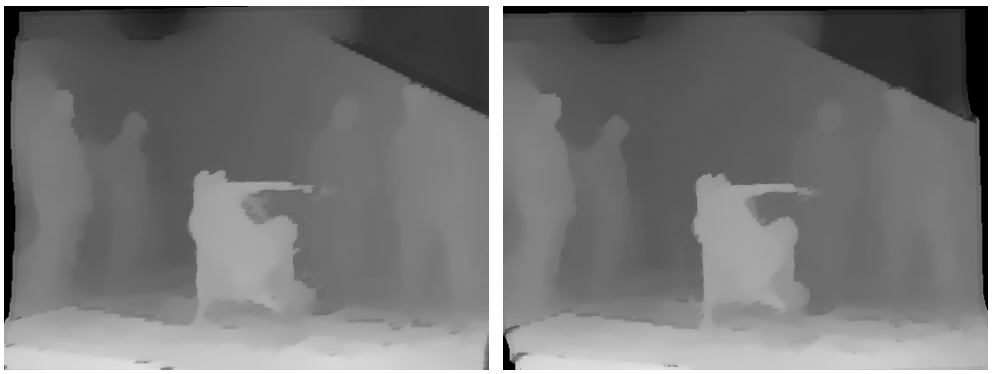


Figure 16 Depth estimation result for a stereo pair after stereo matching and bilateral filtering.

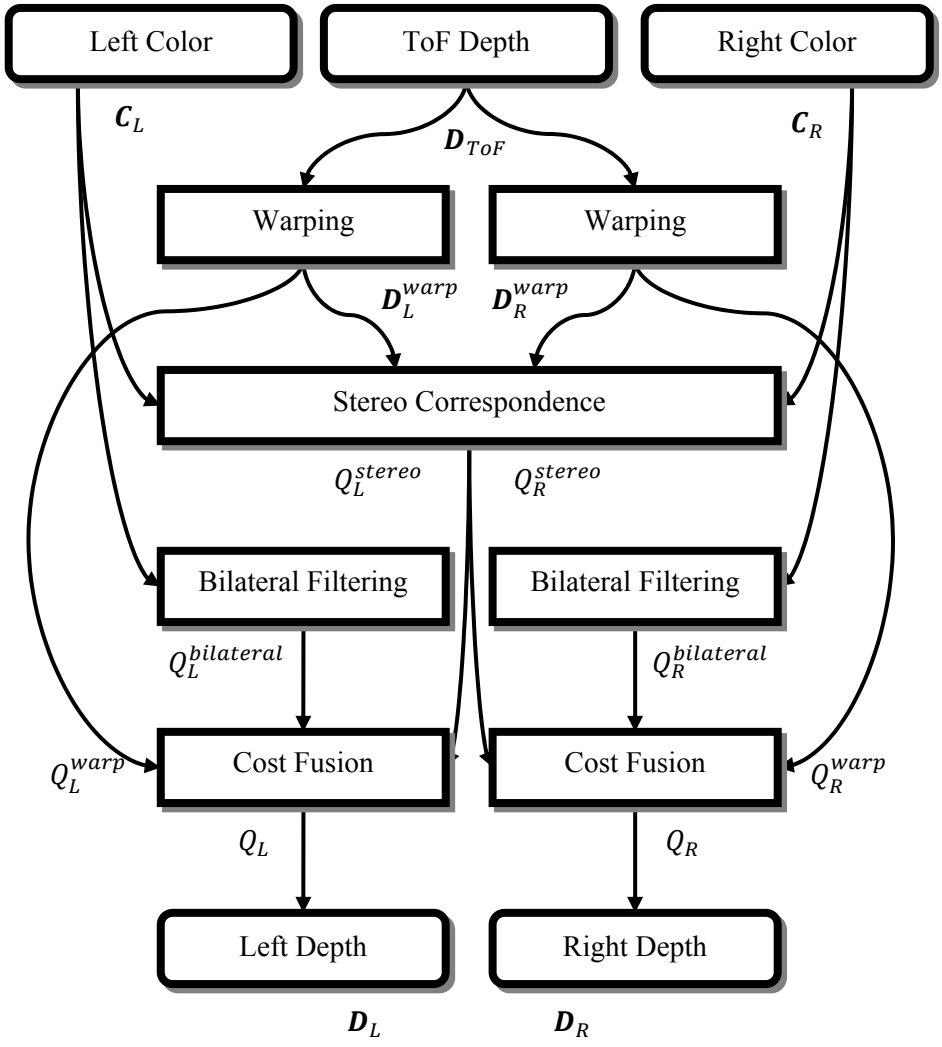


Figure 17 Flowchart for depth estimation stage.

5.4 Arbitrary View Rendering

Image based rendering, as mentioned in Chapter 2, generates the final image by reverse projection of every target pixel to each of the source images supplied to the system. Analyzing pinhole camera model reveals that back projection of 2D image coordinates is an ill-posed problem.

Given a supplementary depth map to any color image, its 3D characteristics are trivially revealed and the image can be projected back into the scene space. Similar to warping of time-of-flight depth images onto arbitrary viewpoints, a color surface represented in scene space can be warped onto any viewpoint of choice. Several image and depth based representations of the same scene help eliminate rubber sheet artifacts occurring at discontinuities of the warped image.

Color image $\boldsymbol{C}_L^{warp}$ or $\boldsymbol{C}_R^{warp}$ are warped from two different color sources. These images provide an initial working ground for the final result of this thesis, an arbitrary rendered view. Selective median filtering and smoothing operations on this initial image leads to a natural looking estimation to the missing intermediate view representation.

5.4.1 Video Plus Depth Warping

Given any viewpoint V and stereo video plus depth data, arbitrary view rendering for V is governed by the input-output relation given in (14). $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ store sufficient information to relate pixels in $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ to 3D scene space coordinates. Much like the depth warping process to transform time-of-flight output to other viewpoints; $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ can be back-projected to scene space and then projected onto target viewpoint V.

Pixels on the image space of V are related to the pixels of the image spaces of L and R as given in (35) and (36).

$$\begin{bmatrix} u_V \\ v_V \\ 1/z_V \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V \boldsymbol{P}_L^{-1} \begin{bmatrix} u_L \\ v_L \\ 1/\boldsymbol{D}_L(u_L, v_L) \\ 1 \end{bmatrix} \tag{35}$$

$$\begin{bmatrix} u_V \\ v_V \\ 1/z_V \\ 1 \end{bmatrix} \equiv \boldsymbol{P}_V \boldsymbol{P}_R^{-1} \begin{bmatrix} u_R \\ v_R \\ 1/\boldsymbol{D}_R(u_R, v_R) \\ 1 \end{bmatrix} \tag{36}$$

Similar to depth warping, these projections are not one-to-one mappings. Therefore, the same triangulation approach is used to convert color and depth maps of stereo view into polygonal meshes. A vertex shader transforms incoming vertices according to $\boldsymbol{P}_V \boldsymbol{P}_L^{-1}$ and a geometry shader performs triangle transformation according to (19).

Rendering of generated fragments through a fragment kernel with depth testing result in two alternative arbitrary view images, $\boldsymbol{C}_V^L$ and $\boldsymbol{C}_V^R$. Two separate estimations to the depth map at arbitrary viewpoint, $\boldsymbol{D}_V^L$ and $\boldsymbol{D}_V^R$, can be obtained in a similar fashion. Fusion of two color maps is performed through a selective blending scheme given in (37).

$$\boldsymbol{C}_V^{combined}(u,v) = \begin{cases} \boldsymbol{C}_V^L(u,v) & ; \quad \boldsymbol{D}_V^L(u,v) - \boldsymbol{D}_V^R(u,v) \leq -\boldsymbol{D}_t \\ \dfrac{\alpha_L \boldsymbol{C}_V^L(u,v) + \alpha_R \boldsymbol{C}_V^R(u,v)}{\alpha_L + \alpha_R} & ; \quad \left|\boldsymbol{D}_V^L(u,v) - \boldsymbol{D}_V^R(u,v)\right| < \boldsymbol{D}_t \\ \boldsymbol{C}_V^L(u,v) & ; \quad \boldsymbol{D}_V^L(u,v) - \boldsymbol{D}_V^R(u,v) \geq \boldsymbol{D}_t \end{cases} \tag{37}$$

$$\alpha_L = 1 - \frac{\text{dist}(L,V)}{\text{dist}(L,V) + \text{dist}(R,V)} \tag{38}$$

$$\alpha_R = 1 - \frac{\text{dist}(R,V)}{\text{dist}(L,V) + \text{dist}(R,V)} \tag{39}$$

$$\text{dist}(V_A, V_B) = \|\boldsymbol{T}_A - \boldsymbol{T}_B\| \tag{40}$$

Color images are blended primarily according to their depth counterparts. If a pixel from a view is nearer than the respective pixel from the other view at least by $\boldsymbol{D}_t$, it is directly copied to the target color map. If corresponding depth values are similar, a weighted color is produced with weights calculated from the Euclidean distances between viewpoints inside the scene space, with $\boldsymbol{T}_V$ being the translation vector of viewpoint V defined by the pinhole camera model. This interpolation ensures smooth transition as V moves from one viewpoint to another [13].

After combining two warped color maps to obtain an initial intermediate view, post-processing measures, such as image filtering, are taken to eliminate pixel artifacts.

5.4.2 Post-Processing

Single pixel artifacts remain on $\boldsymbol{C}_V^{combined}$ due to imperfect alignment of color and depth images during previous stages of the algorithm. These artifacts are corrected with 2D

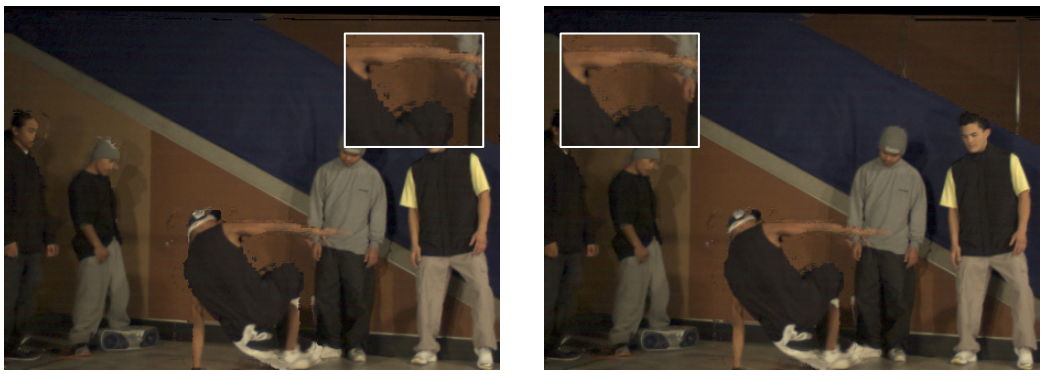
selective median filtering which preserves pixels which are similar to their neighbors and replaces others with the local median. Selective median operation is given in (41) where s is the Euclidean distance function given in (25), s' is the selection threshold and K is the median filter size parameter. Median is chosen according to the green channel of pixel colors as given in (42).

$$\boldsymbol{C}_V^{median}(u,v) = \begin{cases} \boldsymbol{C}_V^{combined}(u,v) & ; \quad s\left(\boldsymbol{C}_V^{combined}(u,v), m\right) \leq s' \\ m & ; \quad s\left(\boldsymbol{C}_V^{combined}(u,v), m\right) > s' \end{cases} \tag{41}$$

$$m = \text{median}_{\text{green}}\left(\left\{\boldsymbol{C}_V^{combined}(u',v') \;\; and \;\; -K \leq (u'-u)\delta, (v'-v)\delta \leq K\right\}\right) \tag{42}$$

Mesh warping, selective color map fusion and selective median filtering add up to an unintentional sharpness over the generated arbitrary view. A final touch of selective low pass filtering at edges provides a more natural image. This low pass filter is explained by (43). In this equation, Sobel is the Sobel filtering response of the image, e' is the edge threshold and Box is a linear smoothing filter.

$$\boldsymbol{C}_V(u,v) = \begin{cases} \boldsymbol{C}_V^{median}(u,v) & ; \quad \text{Sobel}\left(\boldsymbol{C}_V^{median}\right)(u,v) \leq e' \\ \text{Box}\left(\boldsymbol{C}_V^{median}\right)(u,v) & ; \quad \text{Sobel}\left(\boldsymbol{C}_V^{median}\right)(u,v) > e' \end{cases} \tag{43}$$



(a) (b)

Figure 18 Arbitrary view generation after (a) warp combining and (b) post-processing from breakdancers sequence from Zitnick et al. [95].

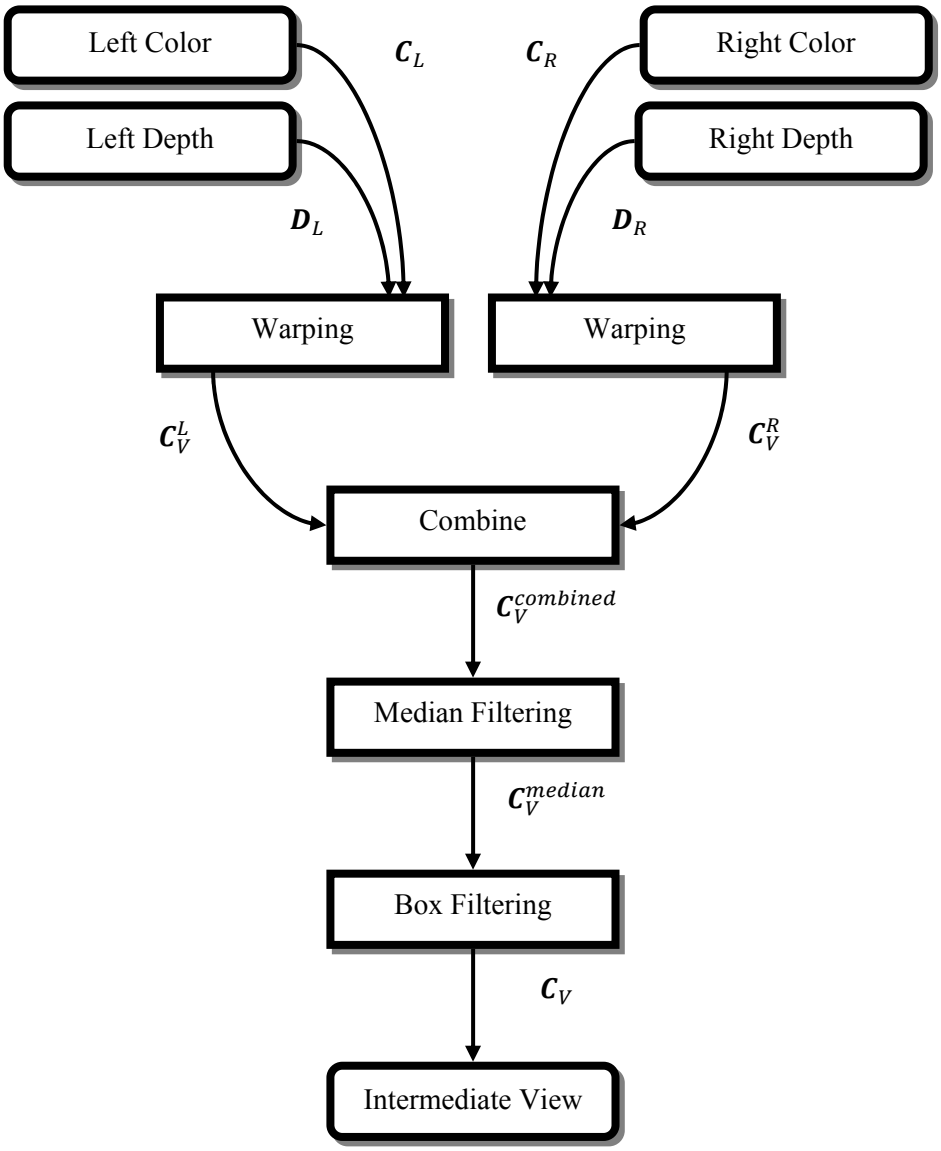


Figure 19 Flowchart for view rendering stage.

The effect of post-processing on the final generated image is best seen with an example. Figure 18 shows a generated view without and with post-processing. In the separate smaller boxes of content detail, loss of extraneous patches is due to selective median filtering and smoothing of jagged edges is achieved with box filtering at edges. A general overview of the display stage is found in Figure 19.

A freeview display system applies the procedure to generate $\boldsymbol{C}_V$ from $\boldsymbol{C}_L$, $\boldsymbol{C}_R$, $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ whenever broadcast content or viewpoint V changes. The result is an additive cue to the viewer and an enriched user experience compared to traditional 2D displays.

5.5 Implementation with Graphics Processing Units

Graphics processing units are hardware devices that perform fast model based rendering as presented in Chapter 4. Arbitrary view rendering framework presented in this chapter has two important image based rendering stages to be performed in real-time. The first stage is depth estimation, which is rendering from raw camera inputs to obtain video plus depth 3D transmission format. The second stage is view rendering, which is rendering from video plus depth for arbitrary view display.

Both color and depth maps are stored on the GPU memory in 2D *texture* format, a special data structure for coloring vertices and fragments. Depth estimation and intermediate view rendering are performed with *shaders* presented in Chapter 4. Since vector 3D modeling information is not available during both stages, dummy triangles are sent to the GPU to trigger the graphics rendering pipeline at various steps of the algorithm.

Depth estimation performed by first warping an initial depth map from one view to another. This operation is carried out in a single rendering pass with twice as many triangles as there are pixels. These triangles are projected from one viewpoint to another using the vertex kernel of the programmable pipeline. Vertex shader is also programmed to trace neighbors of each vertex to find the farthest depth in (20). After triangles are aligned to target viewpoint, a shader in geometry kernel finds and modifies triangles over occlusion regions. Since geometry kernels operate on a primitive level, triangles are filtered one-by-one, and vertices of larger triangles are suppressed to background. After these triangles are decomposed into fragments, warping operation continues with a trivial fragment kernel which passes incoming fragments to target buffers for depth testing.

Cost fusion is a single rendering pass, which is triggered with a single quadrilateral covering the whole target rendering canvas. Inside the GPU stream, default vertex and geometry kernels generate all of the pixels on the target viewpoint. A fragment shader accesses depth and color maps of both stereo views. In this kernel; bilateral filtering is performed, matching pixels are traced inside opposite views with plane sweeping and three different cost functions per each view are constructed. Cost fusion is performed for pixels on both left and right viewpoints and the winning depth candidate is passed to the framebuffer.

Depth estimation stage, which is performed on content acquisition phase, is completed in three rendering passes on the GPU with the following order:

1. Warp $\boldsymbol{D}_{ToF}$ onto L and obtain $\boldsymbol{D}_{L}^{warp}$.
2. Warp $\boldsymbol{D}_{ToF}$ onto R and obtain $\boldsymbol{D}_{R}^{warp}$.
3. Construct Q_L and Q_R by cost fusion to obtain $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$.

On the display side, projection of color maps with depth information onto target viewpoint is accomplished with the same warping pass used in depth projection. Thus, this pass performs the same division of work among different parts of the GPU. A per pixel fragment shader is responsible for another rendering pass to combine these two color maps.

Post-processing is accomplished with a separate rendering pass for each filter used. Both selective median and selective box filtering are non-linear per pixel operations which are suitable for implementation with fragment shaders.

On the display side, arbitrary view rendering is accomplished with the following five rendering passes:

1. Warp $\boldsymbol{C}_L$ onto V and obtain $\boldsymbol{C}_V^L$.
2. Warp $\boldsymbol{C}_R$ onto V and obtain $\boldsymbol{C}_V^R$.
3. Combine $\boldsymbol{C}_V^L$ and $\boldsymbol{C}_V^R$ to obtain $\boldsymbol{C}_V^{combined}$.
4. Selective median filtering over $\boldsymbol{C}_V^{combined}$ to yield $\boldsymbol{C}_V^{median}$.
5. Selective box filtering over $\boldsymbol{C}_V^{median}$ to yield $\boldsymbol{C}_V$.

For warping passes of both stages, depth testing is enabled to handle overlapping triangles, thus target buffers are cleared before rendering. All other passes work on pixel-by-pixel basis and intermediate buffers can be reused by overwriting.

CHAPTER 6

EXPERIMENTS

This chapter presents the experiments conducted to test the algorithm presented throughout this thesis. Experimental setup used to obtain video data is explained and empirical and numerical results are given.

6.1 Experimental Setup

Proposed algorithm is tested with color and time-of-flight cameras and a real-time application is developed. Data acquisition from the cameras and display of arbitrary view generation results take stage together in a single architecture. Data acquisition setup is seen in Figure 20.

Two LightWise LW-3-S-1394 FireWire cameras are used for obtaining color images for stereo vision. Histogram equalization [97] between cameras is performed in software to match their colors. This step ensures correct results in block matching of stereo correspondence and consistent color output in view rendering stage.

SwissRanger SR-3000 time-of-flight camera is used for range measurements. Systematic error, which is mentioned in Chapter 3, in raw depth measurements from time-of-flight camera are corrected with software using SDK of SR-3000. This correction includes both *denoising* and *median filtering*.

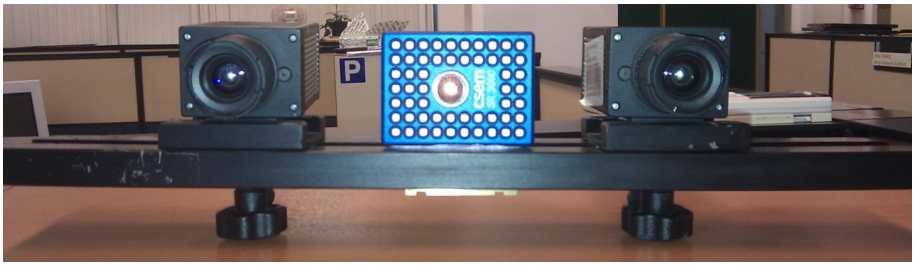


Figure 20 Data acquisition setup used in the experiments.

Camera calibration is accomplished with OpenCV library functions, linked into the real-time demo application. A flat board with a checkerboard pattern is shown to all cameras and capture command is sent to align intrinsic and extrinsic properties of all cameras. Several variations of the checkerboard pattern is needed to obtain robust values for projection matrices $\boldsymbol{P}_L$, $\boldsymbol{P}_R$ and $\boldsymbol{P}_{ToF}$.

6.2 Visual Results

Typical captures obtained with the acquisition setup are given in Figure 21. The SwissRanger SR-3000 sensor emits 850nm infrared light ray, which FireWire cameras cannot completely filter out, resulting in directed red illumination on color images obtained from the stereo pair.

(a) (b) (c)

Figure 21 Several (a) left, (b) time-of-flight and (c) right frame groups.

Depth warping and estimation results are given in Figure 22. Rubber sheet artifacts introduced by occlusions are almost eliminated with triangle suppression and bilateral filtering. However, jagged depth boundaries appear because of false alarms occurring at near occlusion areas.

Several intermediate views generated at the display stage are given in Figure 23. Intermediate views from ballet studio and breakdancers sequences are obtained by using two high resolution color maps from these sets and a single downscaled depth map in between stereo viewpoints. Numerical results for these sets are given in the next section.

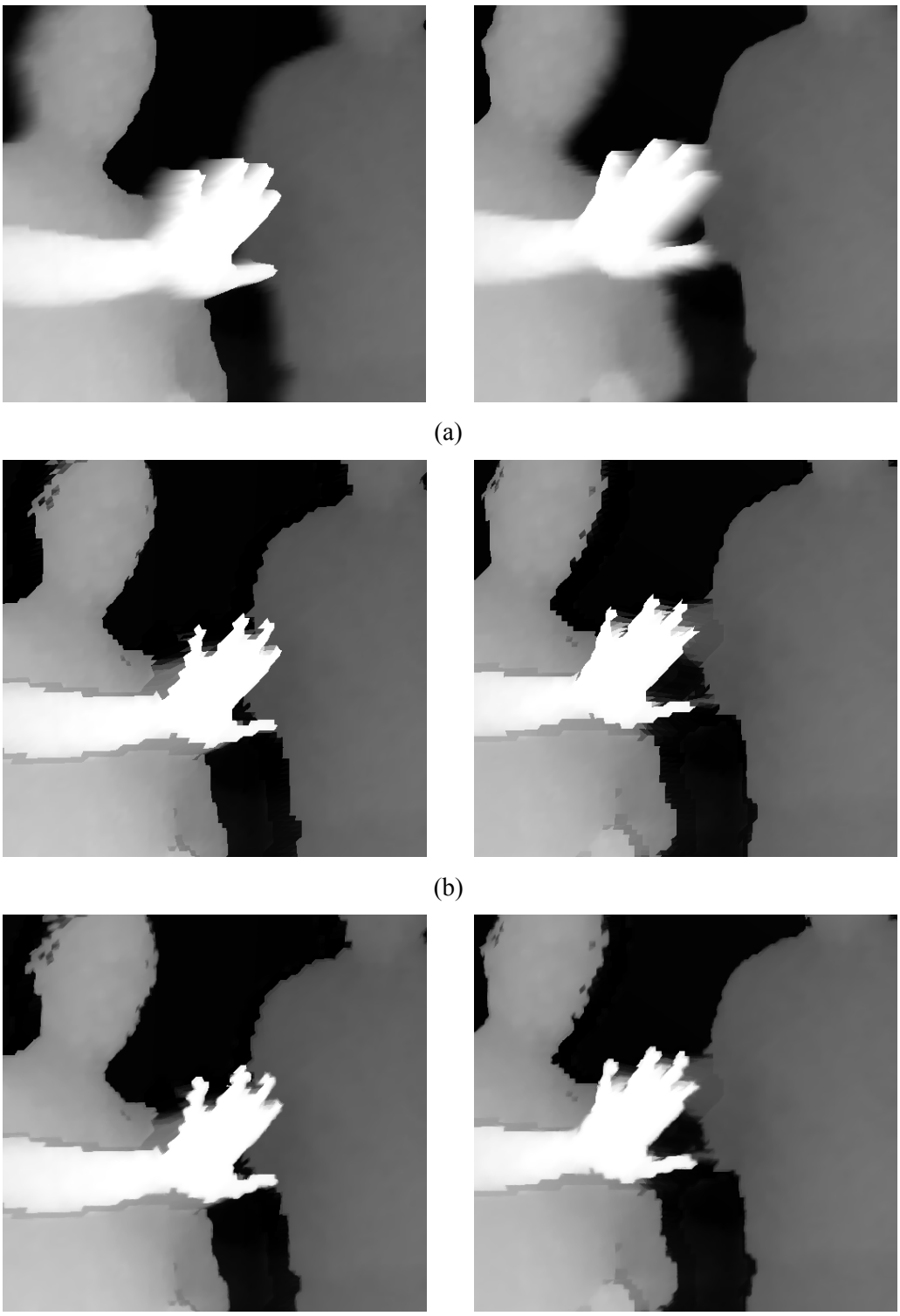


Figure 22 (a) Depth warping without triangle suppression, (b) depth warping with triangle suppression and (c) depth estimation through fusion of cost functions.

(a) (b) (c) (d)

Figure 23 Generated intermediate views for (a) (b) capture obtained from data acquisition setup, (c) ballet studio sequence and (d) breakdancers sequence.

6.3 *Performance of the Algorithm*

Primary objective of arbitrary view rendering is to imitate real cameras in creating representations of 3D scenes from a requested viewpoint. Success of a specific method is assessed by human perception. In other words, subjective quality of the rendered output determines performance of the algorithm.

Quality of the rendered color maps can be estimated with the *peak signal-to-noise ratio* (PSNR) with respect to a reference color map. Breakdancers and ballet studio datasets, which are multiview video plus depth frame sequences, are used to build a testing ground for the algorithm presented in this thesis. Figure 24 shows the relative configuration of eight different color cameras used in obtaining the sequence and the estimated depth data.

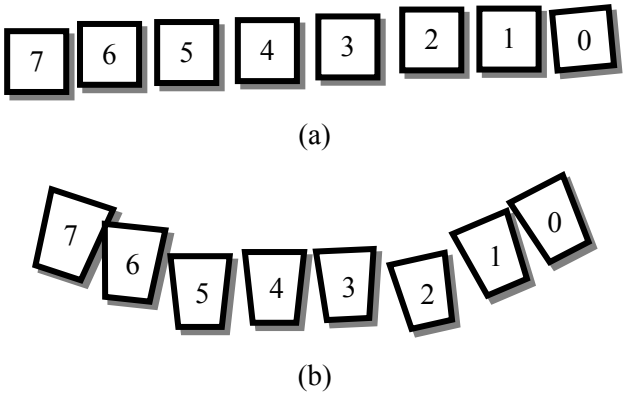


Figure 24 Camera alignments used in breakdancers and ballet studio sequences. (a) Behind the camera and (b) top view.

The arbitrary view rendering algorithm is tested in different configurations. Camera 4 is selected as the reference viewpoint and its color map is estimated through neighboring color maps.

Low resolution output of SR-3000 and its systematic errors are simulated by downscaling the depth map of Camera 4 and introduction of white Gaussian depth noise. An average PSNR value is calculated between the actual color map and its estimation over all frames and for both datasets. Obtained quality results are given in Table 1 with respect to the standard deviation of Gaussian noise added to the initial depth map. PSNR values obtained from high resolution depth maps without time-of-flight camera simulation is also given.

Table 1 Quality of the view rendering method in PSNR for different stereo setups and noise levels.

Dataset	Stereo Pair	High Res.	$\sigma_E = \frac{0.01}{255}$	$\sigma_E = \frac{0.1}{255}$	$\sigma_E = \frac{1}{255}$	$\sigma_E = \frac{10}{255}$
breakdancers	3 and 5	30.36	29.10	29.10	29.13	26.47
breakdancers	2 and 6	29.58	28.17	28.17	28.11	24.08
breakdancers	1 and 7	27.29	26.14	26.14	26.07	22.42
ballet studio	3 and 5	27.19	25.37	25.36	25.32	22.19
ballet studio	2 and 6	25.70	24.24	24.24	24.10	21.19
ballet studio	1 and 7	21.51	20.68	20.68	20.69	18.80

Individual performance of depth warping algorithm is calculated by comparing warping stereo depth views onto target view and warping target depth map onto its original viewpoint. Table 2 lists average depth PSNR values between source depth images and their back-projected counterparts for left and right views

Table 2 Average signal preservation in PSNR due to depth warping from source view to target view and backwards.

Dataset	3 and 5	2 and 6	1 and 7
breakdancers	32.36	32.07	31.85
ballet studio	24.20	23.07	22.72

Quality comparison tests for both final rendered images and intermediate depth warping stage show that the algorithm performance is very susceptible to the stereo configuration used. Depth warping onto stereo viewpoints perform best when these camera position are nearer to the depth sensor. Furthermore, the algorithm is robust against depth distortions up to a certain noise level where performance decays beyond.

The algorithm causes erosion of small object regions at both depth estimation and view rendering stages. This weakness is most exposed within both subjective and numerical results of the ballet studio dataset.

6.4 *Software Benchmark*

Ideas discussed in this thesis are implemented and tested under an arbitrary view synthesis framework. A benchmark application is developed, which users can choose data, resolution and algorithm to test. Source views can be switched through panels and target view camera can be freely moved around. A screenshot is provided in Figure 25.

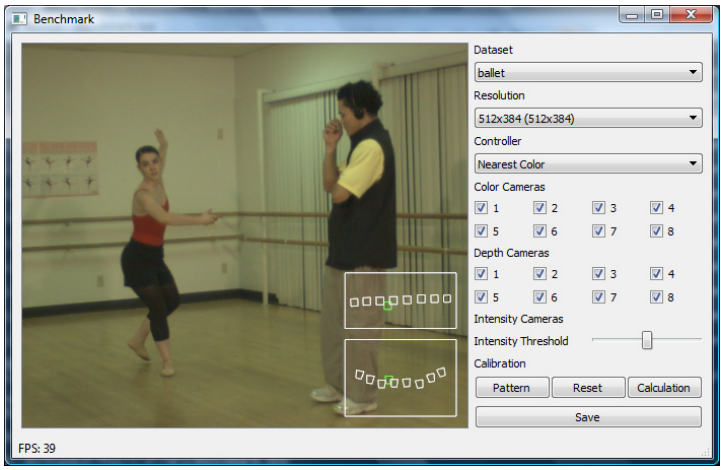


Figure 25 Screenshot from the experimental benchmark.

The benchmark developed offers navigation inside a view surface extracted from the geometry of source cameras. The user is allowed to navigate the target camera inside a region on this surface near source cameras and the target camera always looks at a certain point in scene space. In this manner, two spherical directions are used for navigating camera around the scene and one radial parameter controls zoom.

Real data capture is available through a special dataset inside the benchmark.

CHAPTER 7

CONCLUSIONS

This chapter begins with a summary of the work presented in this thesis. Second section provides remarks on the experimental results obtained and the third and final section provides a discussion of possible improvements to the methods presented.

7.1 Summary

Arbitrary view rendering is the problem of generating missing views of real world scenes from actual views. Generation of intermediate views is a primary concern for 3DTV systems. This thesis provides an intermediate viewing framework from content acquisition to display front end by using a high resolution camera pair and a single low resolution time-of-flight sensor.

Arbitrary view rendering is explained in *image based rendering* context, which encapsulates special rendering techniques for creating digital images from photographic representations of scenes. Image based rendering is a step forward from model based rendering methods to achieve artificial photorealism.

Time-of-flight cameras are relatively new range sensing devices which provide planar depth maps from real world scenes. Despite their low resolution outputs, they provide valuable assistance to passive methods for depth estimation.

Arbitrary view rendering is suitable for 3DTV applications only if achieved in real-time rates. *Graphics processing units* provide a sensible alternative to standard microprocessors by supporting hardwired rendering phases and programmability with high throughput.

An arbitrary view rendering framework which consists of data acquisition and view display stages is presented in this thesis. Raw data obtained from stereo camera pair and ToF sensors are converted into *multiview plus depth* data format which is suitable for current

transmission and broadcast infrastructures. Several alternative approaches for depth and arbitrary view estimation are compared.

7.2 Discussions

Depth sensing is strictly fundamental yet a challenging task for computer vision and computer graphics applications. Passive methods like stereo matching have limitations mainly in textureless regions and discontinuities. Active range sensing devices, on the other hand, provide accurate depth information for large flat regions but they fail at regions with high texturing. Both low resolution output and intensity related errors of range sensing devices lead to erroneous results for non-flat image portions.

Overcoming limitations of both passive and active methods for depth estimation is possible with fusion of these methods. General layout of the scene in 3D space can be extracted with time-of-flight sensors, smaller disparity details can be corrected with stereo matching and boundaries can be aligned to accompanying color maps by bilateral filtering leading to depth maps more accurate than a single approach can achieve.

After per pixel estimation of depth maps, global optimization methods can be utilized. Error minimization oriented approaches can be time consuming but real-time rates can be achieved with compute unified programming [98]. Easy access to high performance parallelization on GPUs for general purpose tasks, as discussed in Chapter 4, is helpful when computing capabilities of CPUs are left alone for other tasks.

Arbitrary view rendering is possible with accurate extraction of depth information. Although multiview video plus depth data transmission format is a viable choice for 3DTV applications, data acquisition for obtaining aligned depth maps for color images is a problem. Stereo plus ToF camera structure provides a reliable acquisition system for 3D if it is followed by accurate data transformation.

7.3 Future Work

Dense depth estimation is an active research area and a significant portion of literature on this topic focuses on global optimization methods as mentioned in Chapter 5. Cost minimization methods help align color and depth maps better and eliminate small artifacts occurring in depth maps. Compute unified architectures presented in Chapter 4 provide computing power through parallelization beyond stream processing paradigms, thus

providing the assistance of GPU acceleration for non-rendering related problems. Global depth approximation methods can be augmented inside the framework with compute unified architectures to eliminate the need to switch between devices.

Arbitrary view rendering systems commonly employ post-processing steps that correct artifacts occurring after color warping. Smolic et al. [13] provide a set of correction techniques which are not trivial to parallelize with stream processing paradigms. Compute unified architectures, again, can be useful for extra view correction measures after rendering passes are completed.

REFERENCES

[1] Akira Kubota, Aljoscha Smolic, Marcus Magnor, Masayuki Tanimoto, and Tsuhan Chen, "Multiview imaging and 3dtv," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10-21, November 2007.

[2] Robert A. Weale, "Brewster and Wheatstone on vision," *Journal of Modern Optics*, vol. 31, no. 3, p. 274, March 1984.

[3] Janusz Konrad and Michael Halle, "3-d displays and signal processing: an answer to 3-d ills?," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 97-111, November 2007.

[4] Mürsel Yıldız and Gözde Bozdağı Akar, "User directed view synthesis on omap processors," in *Proceedings of 3DTV Conference (3DTV)*, Tampere, Finland, 2010.

[5] Leonard McMillan and Gary Bishop, "Plenoptic modeling: an image-based rendering system," in *Proceedings of the 22nd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Los Angeles, California, U.S.A., 1995, pp. 29-46.

[6] Edward H. Adelson and James R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, Michael Landy and J. Anthony Movshon, Eds. Cambridge, Massachusetts, United States: The MIT Press, 1991, ch. 1, pp. 3-20.

[7] Marc Levoy and Pat Hanrahan, "Light field rendering," in *Proceedings of the 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New Orleans, Louisiana, U.S.A., 1996, pp. 31-42.

[8] Ruigang Yang, Marc Pollefeys, Hua Yang, and Greg Welch, "A unified approach to real-time, multi-resolution, multi-baseline 2d view synthesis and 3d depth estimation using commodity graphics hardware," *International Journal of Image and Graphics*

(IJIG), vol. 4, pp. 627-651, October 2004.

[9] Ruigang Yang and Marc Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Chapel Hill, North Carolina, U.S.A., 2003, pp. 211-217.

[10] Jan Woetzel and Reinhard Koch, "Real-time multi-stereo depth estimation on gpu with approximative discontinuity handling," in *Proceedings of the 1st European Conference on Visual Media Production (CVMP)*, London, United Kingdom, 2004, pp. 245-254.

[11] Satoshi Yaguchi and Hideo Saito, "Arbitrary viewpoint video synthesis from multiple uncalibrated cameras," *IEEE Transactions on Systems, Man, and Cybernetics (SMC), Part B: Cybernetics*, vol. 34, no. 1, pp. 430-439, February 2004.

[12] Yosuke Ito and Hideo Saito, "Free-viewpoint image synthesis from multiple-view images taken with uncalibrated moving cameras," in *Proceedings of the IEEE International Conference On Image Processing (ICIP)*, Genoa, Italy, 2005, pp. 29-32.

[13] Aljoscha Smolic et al., "Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, San Diego, California, U.S.A., 2008, pp. 2448-2451.

[14] Daniel Jung and Reinhard Koch, "Efficient depth-compensated interpolation for full parallax displays," in *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Paris, France, 2010.

[15] Richard Szeliski, "Image mosaicing for tele-reality applications," in *Proceedings of the Second IEEE Workshop on Applications of Computer Vision (WACV)*, Sarasota, Florida, U.S.A., 1994, pp. 44-53.

[16] Takeo Kanade, P. J. Narayanan, and Peter W. Rander, "Virtualized reality: concepts and early results," in *Proceedings of the IEEE Workshop on Representation of Visual Scenes*, Cambridge, Massachusetts, U.S.A., 1995, pp. 69-76.

[17] Paul E. Debevec, Yizhou Yu, and George Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," University of California at

Berkeley, Berkeley, California, U.S.A., Technical Report CSD-98-1003, 1998.

[18] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proceedings of the 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New Orleans, Louisiana, U.S.A., 1996, pp. 11-20.

[19] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New York City, New York, U.S.A., 2001, pp. 425-432.

[20] Matthew Uyttendaele et al., "High-quality image-based interactive exploration of real-world environments," *IEEE Computer Graphics and Applications (CG&A)*, vol. 24, no. 3, pp. 52-63, May/June 2004.

[21] Kanji Tanaka, Kuniaki Otsuka, Mitsuru Hirayama, and Eiji Kondo, "View synthesis on mobile robot image database," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Hong Kong SAR and Macau SAR, P. R. China, 2005, pp. 455-461.

[22] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen, "Time-of-flight cameras in computer graphics," *Computer Graphics Forum*, vol. 29, no. 1, pp. 141-159, February 2010.

[23] Marc Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29-37, May 1988.

[24] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan, "Volume rendering," *ACM SIGGRAPH Computer Graphics*, vol. 22, no. 4, pp. 65-74, August 1988.

[25] Rick Parent, *Computer animation: algorithms and techniques*, 2nd ed.: Morgan Kaufmann, 2007.

[26] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips, *Introduction to computer graphics*.: Addison-Wesley Longman Publishing

Co., Inc., 1994.

[27] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer graphics: principles and practice*, 2nd ed.: Addison-Wesley Longman Publishing Co., Inc., 1995.

[28] John D. Owens et al., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80-113, March 2007.

[29] H. Christopher Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133-135, September 1981.

[30] Richard I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580-593, June 1997.

[31] Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Lexington, Kentucky, U.S.A., 2008.

[32] Ingo Schiller, Christian Beder, and Reinhard Koch, "Calibration of a pmd-camera using a planar calibration pattern together with a multi-camera setup," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. XXXVII, no. B5, pp. 297-302, July 2008.

[33] Carlo Dal Mutto, Pietro Zanuttigh, and Guido M. Cortelazzo, "A probabilistic approach to tof and stereo data fusion," in *Proceedings of the 5th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Paris, France, 2010.

[34] Daniel N. Lapedes, *Dictionary of scientific and technical terms*, 4th ed.: McGraw-Hill, 1974.

[35] Jacques Hadamard, *Sur les problèmes aux dérivés partielles et leur signification physique*., 1902.

[36] Jose Marroquin, Sanjoy Mitter, and Tomaso Poggio, "Probabilistic solution of ill-

posed problems in computational vision," Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, Technical Report AIM-897, 1987.

[37] H. Harlyn Baker and Thomas O. Binford, "Depth from edge and intensity based stereo," in *Proceedings of the 7th International Joint conference on Artificial Intelligence (IJCAI)*, Vancouver, British Columbia, Canada, 1981, pp. 631-636.

[38] David J. Fleet, Allan D. Jepson, and Michael R. M. Jenkin, "Phase-based disparity measurement," *Computer Vision Graphics and Image Processing (CVGIP): Image Understanding*, vol. 53, no. 2, pp. 198-210, March 1991.

[39] David G. Jones and Jitendra Malik, "Computational framework for determining stereo correspondence from a set of linear spatial filters," *Image and Vision Computing*, vol. 10, no. 10, pp. 699-708, December 1992.

[40] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," in *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*, Kauai, Hawaii, U.S.A., 2001, pp. 30-36.

[41] Geoffrey Egnal and Richard P. Wildes, "Detecting binocular half-occlusions: empirical comparisons of five approaches," *IEEE Transactions on Pattern Analysis and Machine Inteligence (PAMI)*, vol. 22, no. 7, pp. 1127-1133, July 2002.

[42] Christopher Zach, Konrad Karner, and Horst Bischof, "Hierarchical disparity estimation with programmable 3D hardware," in *Proceedings of the 12th International Conference in Central Europe on Computer Graphics (WSCG)*, Plzeň, Czech Republic, 2004, pp. 275-282.

[43] Kuk-Jin Yoon and In So Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Transactions on Pattern Analysis and Machine Inteligence (PAMI)*, vol. 28, no. 4, pp. 650-656, January 2006.

[44] Sigurjón Árni Guðmundsson, Henrik Aanæs, and Rasmus Larsen, "Fusion of stereo vision and time-of-flight imaging for improved 3d estimation," *International Journal of Intelligent Systems Technologies and Applications (IJISTA)*, vol. 5, no. 3, pp. 425-

433, November 2008.

[45] Bogumil Bartczak, Daniel Jung, and and Reinhard Koch, "Real-time neighborhood based disparity estimation incorporating temporal evidence," *Lecture Notes in Computer Science (LNCS): Pattern Recognition*, vol. 5096, pp. 153-162, June 2008.

[46] Carlo Dal Mutto, Pietro Zanuttigh, and Guido M. Cortelazzo, "Accurate 3d reconstruction by stereo and tof data fusion," in *Proceedings of the Italian Academic Association of Telecommunications Meeting (GTTI)*, Brescia, Italy, 2010.

[47] Geert Van Meerbergen, Maarten Vergauwen, Marc Pollefeys, and Luc Van Gool, "A hierarchical symmetric stereo algorithm using dynamic programming," *International Journal of Computer Vision (IJCV)*, vol. 47, no. 3, pp. 275-285, June 2002.

[48] Yasutaka Furukawa and Jean Ponce, "Accurate, dense, and robust multi-view stereopsis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, Minnesota, U.S.A., 2007.

[49] Daniel Scharstein and Richard Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal on Computer Vision (IJCV)*, vol. 47, pp. 7-42, April 2002.

[50] Philip H. S. Torr, Richard Szeliski, and P. Anandan, "An integrated bayesian approach to layer extraction from image sequences," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV)*, Kerkyra, Corfu, Greece, 1999, pp. 983-990.

[51] Daniel Scharstein and Richard Szeliski, "Stereo matching with nonlinear diffusion," *International Journal on Computer Vision (IJCV)*, vol. 28, no. 2, pp. 155-174, June 1998.

[52] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 11, pp. 1222-1239, November 2001.

[53] Marshall F. Tappen and William T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters," in *Proceedings of the 9th*

International Conference on Computer Vision (ICCV), Nice, France, 2003, pp. 900-906.

[54] Bogumil Bartczak and Reinhard Koch, "Dense depth maps from low resolution time-of-flight depth and high resolution color views," in *Proceedings of the International Symposium on Visual Computing (ISVC)*, Las Vegas, Nevada, U.S.A., 2009, pp. 228-239.

[55] Vincent Nozick, François de Sorbier, and Hideo Saito, "Plane-sweep algorithm: various tools for computer vision," The Institute of Electronics, Information and Communication Engineers (IEICE), Technical Report PRMU2007-259, 2008.

[56] Katrien Jacobs and Céline Loscos, "Classification of illumination methods for mixed reality," *Computer Graphics Forum*, vol. 25, no. 1, pp. 29-51, March 2006.

[57] Naho Inamoto and Hideo Saito, "Free viewpoint video synthesis and presentation of sporting events for mixed reality entertainment," in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE)*, Singapore, 2004, pp. 42-50.

[58] Michael Goesele, Brian Curless, and Steven M. Seitz, "Multi-view stereo revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York City, New York, U.S.A., 2006, pp. 2402-2409.

[59] Christopher Zach, Mario Sormann, and and Konrad Karner, "High-performance multi-view reconstruction," in *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Chapel Hill, North Carolina, U.S.A., 2006, pp. 113-120.

[60] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, and Paul Haeberli, "Fast shadows and lighting effects using texture mapping," *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 249-252, July 1992.

[61] Christoph Fehn et al., "An evolutionary and optimised approach on 3d-tv," in *Proceedings of the International Broadcast Conference (IBC)*, Amsterdam, Netherlands, 2002, pp. 357-365.

[62] Aljoscha Smolic et al., "3d video and free viewpoint video - technologies, applications and mpeg standards," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Toronto, Ontario, Canada, 2006, pp. 2161-2164.

[63] Philipp Merkle, Aljoscha Smolic, Karsten Müller, and Thomas Wiegand, "Multi-view video plus depth representation and coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas, U.S.A., 2007, pp. 201-204.

[64] Thorsten Ringbeck, Tobias Möller, and Bianca Hagebeuker, "Multidimensional measurement by using 3-d pmd sensors," *Advances in Radio Science (ARS)*, vol. 5, pp. 135-146, June 2007.

[65] Thierry Oggier et al., "An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRanger™)," in *Proceedings of SPIE: Optical Design and Engineering, vol. 5249, doi. 10.1117/12.513307*, St. Etienne, France, 2003, pp. 534-545.

[66] MESA Imaging. (2006, October) Swissranger sr-3000 manual, version 1.03. SR-3000 Delivery Package.

[67] Willard S. Boyle and George E. Smith, "Charge coupled semiconductor devices," *Bell System Technical Journal*, vol. 49, no. 4, pp. 587-593, April 1970.

[68] Robert Lange and Peter Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390-397, March 2001.

[69] Antonio Medina, Francisco Gayá, and Francisco del Pozo, "Compact laser radar and three-dimensional camera," *Journal of the Optical Society of America (JOSA) A: Optics, Image Science, and Vision*, vol. 23, no. 4, pp. 800-805, April 2006.

[70] Carlo Tomasi and Roberto Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the 6th International Conference on Computer Vision (ICCV)*, Bombay, India, 1998, pp. 839-846.

[71] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér, "Spatial-depth super resolution for range images," in *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), Minneapolis, Minnesota, U.S.A., 2007.

[72] Andreas Kolb, Erhardt Barth, and Reinhard Koch, "ToF-sensors: new dimensions for realism and interactivity," in *Workshop on ToF-Camera based Computer Vision with IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, U.S.A., 2008.

[73] Timo Kahlmann, Fabio Remondino, and Sébastien Guillaume, "Range imaging technology: new developments and applications for people identification and tracking," in *Proceedings of SPIE: Videometrics IX, vol. 6491, doi. 10.1117/12.702512*, San Jose, California, USA, 2007.

[74] Marvin Lindner and Andreas Kolb, "Lateral and depth calibration of pmd-distance sensors," *Lecture Notes in Computer Science (LNCS): Advances in Visual Computing*, vol. 4292, pp. 524-533, November 2006.

[75] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski, "Layered depth images," in *Proceedings of the 25th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New York City, New York, U.S.A., 1998, pp. 231-242.

[76] Anatol Frick, Falko Kellner, Bogumil Bartczak, and Reinhard Koch, "Generation of 3d-tv ldv-content with time of flight camera," in *Proceedings of 3DTV Conference (3DTV)*, Potsdam, Germany, 2009.

[77] Christian Beder, Bogumil Bartczak, and Reinhard Koch, "A combined approach for estimating patchlets from pmd depth images and stereo intensity images," *Lecture Notes in Computer Science (LNCS): Pattern Recognition*, vol. 4713, pp. 11-20, November 2007.

[78] Bogumil Bartczak, Ingo Schiller, Christian Beder, and Reinhard Koch, "Integration of a time-of-flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time," in *Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, Atlanta, Georgia, U.S.A., 2008.

[79] Ingo Schiller, Bogumil Bartczak, Falko Kellner, and Reinhard Koch, "Increasing realism and supporting content planning for dynamic scenes in a mixed reality system incorporating a time-of-flight camera," in *Proceedings of the 5th European Conference on Visual Media Production (CVMP)*, London, United Kingdom, 2008.

[80] Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth, "Geometric invariants for facial feature tracking with 3d tof cameras," in *IEEE International Symposium on Signals, Circuits and Systems (ISSCS)*, Iaşi, Romania, 2007, pp. 109-112.

[81] Michael B. Holte, Thomas B. Moeslund, and Preben Fihl, "View invariant gesture recognition using the csem swissranger sr-2 camera," *International Journal of Intelligent Systems Technologies and Applications (IJISTA)*, vol. 5, no. 3/4, pp. 295-303, November 2008.

[82] Steven Collins, "Game graphics during the 8-bit computer era," *ACM SIGGRAPH Computer Graphics*, vol. 32, no. 2, pp. 47-51, May 1998.

[83] Julio Sanchez and Maria P. Canton, "Displaying bit-mapped images," in *Software Solutions for Engineers and Scientists*.: CRC Press, 2007, p. 690.

[84] NVIDA Corporation. (retrieved in 2010, December) Geforce 256: the world's first gpu. [Online]. http://www.nvidia.com/page/geforce256.html

[85] Khronos Group. (retrieved in 2010, December) Opengl registry. [Online]. http://www.opengl.org/registry/

[86] John D. Owens et al., "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879-899, May 2008.

[87] Khronos Group. (2010, July) Opengl 4.1 core profile specification.

[88] NVIDA Corporation. (2010, June) Nvidia cuda c programming guide, version 3.1.1. CUDA Toolkit 3.1.

[89] Khronos Group. (retrieved in 2010, December) Opencl overview. [Online]. http://www.khronos.org/opencl/

[90] K. Berker Loğoğlu and Tuğrul K. Ateş, "Speeding-up pearson correlation coefficient calculation on graphical processing units," in *Proceedings of the IEEE 18th Signal Processing, Communication and Applications Conference (SIU)*, Diyarbakır, Turkey, 2010.

[91] Derek Chan, Hylke Buisman, Christian Theobalt, and Sebastian Thrun, "A noise-aware filter for real-time depth upsampling," in *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications with Tenth European Conference on Computer Vision (ECCV)*, Marseille, France, 2008.

[92] A. Aydın Alatan et al., "Scene representation technologies for 3dtv - a survey," *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)*, vol. 17, no. 11, pp. 1587-1605, November 2007.

[93] Ingo Schiller and Reinhard Koch, "Datastructures for capturing dynamic scenes with a time-of-flight camera," *Lecture Notes In Computer Science (LNCS): Dynamic 3D Imaging*, vol. 5742, pp. 42-57, October 2009.

[94] Eren Halici and A. Aydin Alatan, "Watermarking for depth image-based rendering," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Cairo, Egypt, 2009, pp. 4217-4220.

[95] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski, "High-quality video view interpolation using a layered representation," in *Proceedings of the 31st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Los Angeles, California, U.S.A., 2004, pp. 600-608.

[96] Renato Pajarola, Miguel Sainz, and Yu Meng, "Dmesh: fast depth-image meshing and warping," *International Journal on Image Graphics (IJIG)*, vol. 4, no. 4, pp. 653-681, October 2004.

[97] Rafael C. Gonzalez and Richard E. Woods, *Digital image processing*, 3rd ed.: Pearson Education, Inc., 2008.

[98] John Congote, Javier Barandiaran, Iñigo Barandiaran, and Oscar Ruiz, "Realtime

dense stereo matching with dynamic programming in CUDA," in *Proceedings of the 19th Spanish Congress of Graphical Informatics (CEIG)*, San Sebastián, Spain, 2009, pp. 231-234.