

VOLUMETRIC OBJECT RECONSTRUCTION BY MEANS OF PHOTOGRAMMETRY

vorgelegt von

M. Sc. Eng. Yasemin Kuzu

bei der Fakultät VI

Bauingenieurwesen und Angewandte Geowissenschaften
der Technischen Universität Berlin

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

Promotionsausschuss:

Gutachter: Prof. Dr. –Ing. Olaf Hellwich

Gutachter: Prof. Dr. –Ing. Jörg Albertz

Gutachter: Prof. Dr. –Ing. Ayhan Alkis

Vorsitzender: Prof. Dr. –Ing. Lothar Gründig

Tag der wissenschaftlichen Aussprache: 18. Februar 2004

Berlin 2004

D 83

Acknowledgements:

I would like to thank many people who supported and helped me with my research which led to this dissertation. All my doctor fathers have been equally important for developing this thesis. First of all, I would like to say that I owe a great debt to my adviser, Prof. Dr.-Ing. Jörg Albrecht, who suggested and motivated me to work on this interesting research topic. Secondly, I want to thank to Prof. Dr.-Ing. Olaf Hellwich for continuing my supervision, for his invaluable advices and support. I would like to thank Prof. Dr. Ayhan Alkis for giving me the opportunity to study at Berlin Technical University which was a challenging opportunity in my career. I also would like to thank to him for proof reading of this dissertation and his enlightening advices.

There are many people who contributed to this thesis. I should say that working with all these people at the Department of Photogrammetry and Cartography, TU Berlin, was a fruitful and enjoyable experience. I am especially grateful to Olaf Sinram with whom I collaborated throughout my research. His contributions are significant throughout this thesis. I truly appreciate the help of Volker Rodehorst for introducing me the recent developments in computer vision techniques and sharing his ideas. I would like to thank John Moré and Albert Wiedemann for their helpful assistance to the camera calibration and bundle block adjustment tasks. I thank to Iliana Theodoropoulou for checking and correcting the written English of this thesis. I thank to all my colleagues for their support and friendship as well.

I am grateful to Berlin Technical University for financially supporting my studies which made it all possible for me.

Last but surely not the least; I would like to thank my parents and all my family for their support and encouragement.

Berlin, 2003

Yasemin Kuzu

Titel der Dissertation:**Volumetrische Objektrekonstruktion mit Mitteln der Photogrammetrie****ZUSAMMENFASSUNG:**

In vielen Bereichen, wie Virtual Reality, Wissenschaft, Medizin und Unterhaltung wird die automatische Erstellung von 3D Computer Modellen benötigt. In dieser Arbeit behandeln wir das Problem zur Erstellung photorealistischer Modelle aus Einzelbildern und wir präsentieren eine volumetrische Rekonstruktionstechnik, die auf Voxel-Modellen basiert.

Das Ziel war es, die Systemvoraussetzungen so gering wie möglich zu halten, damit der Algorithmus auf möglichst vielen Gebieten Anwendung finden kann. Es wird eine standard CCD Videokamera verwendet, die auch die Möglichkeit besitzt, hochauflösende Standbilder zu erfassen. Ein kreisförmiger Kameraaufbau wurde dadurch realisiert, dass das Objekt auf einem Drehteller gedreht wurde, während die Kamera nicht bewegt wurde. Nachdem der Kamerakalibrierung wurden die Bildorientierungen anhand einer Bündelblockausgleichung berechnet.

Ein etabliertes Verfahren der Computer Vision wird verwendet, um ein genähertes Modell zu erstellen, da der Algorithmus schnell und einfach arbeitet. Für die Verfeinerung des Modells und zur Genauigkeitssteigerung wurden auch Methoden der Photogrammetrie zu Rate gezogen.

Das Objekt wird in mehreren Schritten rekonstruiert. Im ersten Schritt wird ein Volumen definiert, welches den gesamten auszuwertenden Bereich umschließt. Eine erste Näherung des Modells wird durch "Shape from Silhouette" (Form aus Kontur) erreicht, aus dem die sogenannte umgebende Hülle hervorgeht. Leider können in der umgebenden Hülle keine Konkavitäten des Objektes enthalten sein. Es werden mehrere Algorithmen vorgestellt, um diese Darstellung zu verfeinern. Ein wichtiger Faktor ist die Information über Sichtbarkeit eines Voxels in einem bestimmten Bild. Es kann aus mehreren Gründen verdeckt sein. Wir stellen einen Algorithmus vor, der diese Information schnell und zuverlässig anhand einer Linienverfolgung liefert. Darüber hinaus präsentieren wir ein Qualitätsmerkmal für die Sichtbarkeit, indem der Normalenvektor der Tangentialebene mit der Blickrichtung des Bildes in Beziehung gebracht wird. Diese Werkzeuge dienen neben anderen dazu, das Modell zu verfeinern.

Da nur Voxel der wahren Oberfläche auf korrespondierende Bildpunkte abgebildet werden, kann diese Tatsache helfen, um nicht Oberflächen-Voxel auszuschließen. Die Prüfung auf Ähnlichkeit der Projektionen eines Voxels, lässt demnach Entscheidungen zu, ob es sich um einen Oberflächenvoxel handelt. Verschiedene Methoden der Ähnlichkeitsberechnung werden vorgestellt und bewertet. Mit Hilfe der vorgestellten Methoden, gelingt es, die Konkavitäten der Objektoberfläche zu entfernen, so dass das fertige Modell dem Objekt tatsächlich entspricht.

Title of Dissertation:**Volumetric Object Reconstruction By Means of Photogrammetry****ABSTRACT:**

In many fields such as virtual reality, science, medicine and entertainment, an automated generation of 3D computer models is required. In this thesis the problem of creating photorealistic 3D models from still image sequences is addressed and a volumetric object reconstruction technique based on voxel models is presented.

One of the goals of this study is to keep the system requirements as low as possible so that the algorithm can be used in many application areas. A standard CCD video camera is used which can capture still images in high resolution. A circular camera setup was done, which was actually accomplished by rotating the object itself, while the camera position was stationary. After calibrating the camera, the orientation parameters of the acquired images are calculated with a bundle block adjustment.

A popular computer vision technique is used in order to achieve an approximate model since this technique is fast and easy to reconstruct 3D models of real-world objects. In order to refine the model and get more precise results, conventional photogrammetric techniques are often referred.

The object is reconstructed in several steps. The algorithm presented begins by initializing a volume that encloses the 3D objects to be reconstructed. A first approximation of the model is acquired by shape from silhouette which delivers the objects visual hull. Unfortunately, this method does not recover concavities on the object. In order to refine the representation, several tools are described. An important factor is the visibility information of a voxel in a specific image. It can be occluded for several reasons. A fast method is presented to recover this information with a line tracing algorithm. Furthermore, a quality measure of the visibility is introduced, by using the surface normal vector of a voxel in combination with the viewing direction of the image. These tools will help to generate the upgraded model more accurately.

The objects natural texture will serve as criterion to exclude non-surface voxels. Since only surface voxels will be projected into corresponding image points in an oriented bundle, we can make use of a similarity value to distinguish surface from non-surface voxels. Several algorithms to evaluate similarity are introduced and compared.

With the help of the tools introduced, the concave areas on the object's surface are recovered successfully and the final model represents the object correctly.

CONTENTS:

Figure Index

Table Index

Symbol Index

Chapter 1 : Introduction	1
1.1 Introduction	1
1.2 3D Shape Acquisition Techniques	2
1.2.1 Physical Contact	2
1.2.2 Transmissive	3
1.2.3 Reflective Methods	4
1.2.3.1 Active Optical Methods	4
1.2.3.1.1 Active Stereo	4
1.2.3.1.2 Shape from Projected Texture (Interferometry)	4
1.2.3.1.3 Active Depth from Defocus	6
1.2.3.1.4 Time of Flight	6
1.2.3.1.5 Optical Triangulation	6
1.2.3.2 Passive Optical Methods	7
1.2.3.2.1 Shape from Shading	7
1.2.3.2.2 Shape from Motion	8
1.3 Image-Based Modeling and Rendering	8
1.4 Aim of the Study	11
1.5 Problem Statement	11
1.6 Areas of Application	12
1.6.1 Reverse Engineering	12
1.6.2 Medicine	13
1.6.3 Entertainment	13
1.6.4 Virtual Worlds, Virtual Walkthrough, Telepresence	13
1.6.5 Cultural Heritage	14
1.6.6 Home Shopping	14
1.6.7 Industrial Inspection	14
1.6.8 Design	14
1.7 Contents of the Thesis	15
Chapter 2 : 3D Representation Techniques	17
2.1 3D Primitives	17
2.1.1 Surface Graphics	17
2.1.2 Volume Graphics	17
2.1.3 Comparison of Voxel Methods with Polygonal Methods	18
2.1.3.1 Advantages of Voxels	18
2.1.3.2 Disadvantages of Voxels	20
2.2 Volume Data	20
2.2.1 Voxel Definition	21
2.2.2 Voxel Adjacencies	21
2.2.3 Surface Voxel List	22
2.2.4 Vector Line	23

2.2.5	Surface Normal Vector.....	24
2.2.6	Volumetric Data Set Definition.....	24
2.2.6.1	Sources of Volume Data	25
2.2.6.1.1	Sampled Data.....	25
2.2.6.1.2	Modeled (Generated) Data	27
2.2.6.1.3	Simulated Data	28
2.2.7	Storage of Voxels	28
2.2.7.1	Volume Buffer	28
2.2.7.2	Binary Space Partitioning Trees	28
2.2.7.3	Run-Length-Encoding (RLE)	29
2.2.7.4	Octrees	29
2.3	Morphological Operations in 3D.....	30
Chapter 3 : Camera Calibration and Image Orientation		31
3.1	Coordinate Systems.....	31
3.1.1	Image and World Coordinate System	31
3.1.2	Voxel and World Coordinate System.....	33
3.1.3	Pixel and Image Coordinates.....	36
3.2	Camera Calibration	39
3.3	Image Orientation.....	41
Chapter 4 : Volumetric Object Reconstruction Techniques Using Images		43
4.1	Volumetric Intersection (Shape from Silhouettes).....	44
4.1.1	Experimental Setup	45
4.1.2	Automatic Image Segmentation.....	46
4.1.3	The Visual Hull Computation	47
4.1.4	Voting-Based Voxel Carving.....	48
4.1.5	Hierarchical Processing Using Octrees	50
4.1.6	Improved Volume Intersection with Adjusted Image Orientations	51
4.1.7	An Application of the Volume Intersection in a Medical Project.....	53
4.2	Shape from Photo Consistency	55
4.2.1	Comparison of Voxel Coloring Methods with Area-Based Image Matching..	57
4.2.2	Color Consistency	57
4.2.2.1	Color Consistency Measures.....	58
4.2.3	Voxel Coloring.....	59
4.2.4	Space Carving	61
4.2.5	Generalized Voxel Coloring.....	62
4.2.6	Multi-Hypothesis Voxel Coloring.....	64
4.2.7	Voxel Coloring Using Ray Tracing for Visibility Computation.....	65
4.3	Model Refinement by Image Matching	68
4.3.1	Image Matching.....	68
4.3.1.1	Correlation	68
4.3.1.2	Normalized Cross Correlation	69
4.3.1.3	Rank Correlation.....	70
4.3.1.4	Difference Correlation	70
4.3.1.5	Least Squares Matching (LSM).....	71
4.3.1.6	Epipolar Geometry.....	74
4.3.2	Color Image Matching	75
4.3.2.1	Color Theory.....	75

4.3.2.2	Considering Color	76
4.3.2.3	Single Vector Correlation	77
4.3.2.4	Separate Channel Mean Value	77
4.3.2.5	Weighted Channel Mean Value	78
4.3.2.6	Difference Correlation	78
4.3.2.7	RGB-Correlation	79
4.3.2.8	Evaluation	79
4.3.3	A Visual Hull Refinement Algorithm Using Block-Matching	81
4.3.3.1	Polychromatic Block Matching	81
4.3.3.2	Hierarchical Matching Using Image Pyramids	82
4.3.3.3	Integration of Volume Data and Depth Maps	82
Chapter 5 : Proposed Volumetric Reconstruction Algorithm.....		83
5.1	Utilities for Volumetric Processing.....	83
5.1.1	Line Tracing	83
5.1.1.1	18-Connected Line.....	83
5.1.1.2	6-Connected Line.....	84
5.1.1.3	Implementation of Line Tracing	85
5.1.2	Line Segment Limiting.....	88
5.1.2.1	Global Minimum/Maximum Limitation	88
5.1.2.2	Spherical Limitation.....	89
5.1.2.3	Cubical Limitation	90
5.1.3	Line Snapping	91
5.1.3.1	Perpendicular Projection.....	91
5.1.3.2	Point Rotation	92
5.1.4	Recovering Visibility Information	93
5.1.5	Normal Vector on Voxel Surfaces	96
5.2	Model Refinement.....	101
5.2.1	Knowledge-Based Patch Distortion	102
5.2.1.1	Introducing Knowledge about the Approximate Object Shape	107
5.2.2	Shell Carving.....	109
5.2.3	Shape Carving	111
5.2.4	Voting Based Shell Carving.....	115
5.3	Texture Mapping	119
Chapter 6 : Contributions and Future Work.....		123
6.1	Contributions.....	123
6.2	Future Work	124
Chapter 7 : Literature		125
Curriculum Vitae		

FIGURE INDEX:

FIGURE 1-1: 1CMM ‘FAROARM – PLATINUM’ FROM FARO TECHNOLOGIES INC. (WWW.FARO.COM).....	3
FIGURE 1-2: A SCHEME OF OPTICAL SCANNER WHERE 3D POINTS ARE COMPUTED BY TRIANGULATION	7
FIGURE 1-3: IMAGE-BASED RENDERING AND TRADITIONAL MODELING AND RENDERING METHODS.....	10
FIGURE 2-1: A $n \times n \times n$ VOLUME AND A VOXEL WITH COORDINATES (X_v , Y_v , Z_v).....	21
FIGURE 2-2: THE 6- (B), THE 18- (C) AND THE 26-ADJECENIES (D) OF THE VOXEL (A).....	22
FIGURE 2-3: A) 18- AND B) 6-CONNECTED VOXEL LINE.....	23
FIGURE 2-4: LINE INTERSECTING A PLANE WITHOUT DETECTION	24
FIGURE 2-5: TANGENT PLANE AND SURFACE NORMAL VECTOR.....	24
FIGURE 2-6: VOLUME DATASETS: 5 CONSECUTIVE VOLUMINA WITH 50 LAYERS OF 256 BY 256 PIXELS EACH (TOTAL 16 MB = 0.5 SECONDS OF FLOW DATA) [MAAS, 1994].....	25
FIGURE 2-7: UNLABELED, LABELED, MRI AND CT IMAGES RESPECTIVELY “VISIBLE HUMAN PROJECT” [TVHP]	26
FIGURE 2-8: 3D VISUALIZATION OF THE FEMALE CADAVER, FROM THE “VISIBLE HUMAN PROJECT”, EXTRACTED FROM A LOW-RESOLUTION ANIMATED GIF [TVHP]	26
FIGURE 2-9: VIRTUAL SURGERY “VISIBLE HUMAN PROJECT” [TVHP]	26
FIGURE 2-10: GENERATED OBJECTS, BASED ON THE KNOWN GEOMETRY AND TEXTURE	27
FIGURE 2-11: NUMBERING OF OCTANTS IN AN OCTREE AND SUBDIVISION OF THE VOXEL CUBE	30
FIGURE 3-1: RELATION BETWEEN IMAGE AND OBJECT COORDINATES UNDER CENTRAL PROJECTION.....	31
FIGURE 3-2: THREE SEQUENTIAL ROTATIONS.....	32
FIGURE 3-3: RELATION BETWEEN WORLD AND VOXEL COORDINATES.....	35
FIGURE 3-4: RELATIONSHIP BETWEEN PIXEL AND IMAGE COORDINATE SYSTEMS.....	37
FIGURE 3-5: CALIBRATION OBJECT WITH 75 SPATIALLY WELL DISTRIBUTED CONTROL POINTS	39
FIGURE 3-6: RELATION OF DIFFERENT IMAGE SIZES AND REFERRING FOCAL LENGTH.....	40
FIGURE 3-7: DETERMINATION OF OBJECT CONTROL POINTS BY BUNDLE BLOCK ADJUSTMENT ..	42
FIGURE 3-8: THE VIRTUAL CAMERA SETUP, USING VRML-VISUALIZATION.....	42
FIGURE 4-1: VIRTUAL CAMERA POSITIONS ACCORDING TO THE TURNTABLE-FIXED COORDINATE SYSTEM	45
FIGURE 4-2: MATHEMATICAL CAMERA MODEL FOR CENTRAL PROJECTION.....	45
FIGURE 4-3: BACKGROUND ESTIMATION USING AN IHS COLOR SPACE HISTOGRAM (LEFT) AND THE RESULTING SILHOUETTE EXTRACTION (RIGHT)	46
FIGURE 4-4: THE VISUAL HULL COMPUTATION.....	47
FIGURE 4-5: CARVING OF A VOXEL CUBE USING VARIOUS SILHOUETTES UNDER CENTRAL PROJECTION.....	48
FIGURE 4-6: INTERSECTION OF SILHOUETTE CONES USING 1, 3, 5 AND 9 IMAGES.....	48
FIGURE 4-7: VOXEL MODEL OF THE 3-D SHAPE RECONSTRUCTION USING 60 IMAGES AND 256^3 VOXELS WITH TEXTURE MAPPING	49

FIGURE 4-8: CONCAVE AREAS IN SHAPE FROM SILHOUETTE RECONSTRUCTION.....	50
FIGURE 4-9: ITERATIVE GENERATION OF AN OCTREE MODEL USING 4^3 , 16^3 , AND 64^3 VOXELS ..	50
FIGURE 4-10: VOLUMETRIC INTERSECTION WITH ADJUSTED (LEFT) AND ESTIMATED (RIGHT) ORIENTATION PARAMETERS.....	52
FIGURE 4-11: IMAGE ACQUISITION STEP.....	53
FIGURE 4-12: SLICES OF THE RECONSTRUCTION ALONG THREE AXES AND THE HISTOGRAM FOR VOLUME CALCULATION	54
FIGURE 4-13: RECONSTRUCTION RESULTS OF THE LEG WITH 16 IMAGES	55
FIGURE 4-14: COLOR CONSISTENCY IS USED TO DISTINGUISH SURFACE POINTS FROM POINTS NOT ON THE SURFACE	58
FIGURE 4-15: VISIBILITY COMPUTATION IN VOXEL COLORING	60
FIGURE 4-16: VISIBILITY ORDERING IN SPACE CARVING	62
FIGURE 4-17: TWO VARIANTS OF GVC USING DIFFERENT DATA STRUCTURES TO COMPUTE VISIBILITY, ITEM BUFFERS (LEFT) AND LAYERED DEPTH IMAGES (RIGHT).....	63
FIGURE 4-18: WHEN A VOXEL IS CARVED, THE VISIBILITY OF THE ADJACENT INNER VOXELS CHANGE.....	63
FIGURE 4-19: MULTI-HYPOTHESIS VOXEL COLORING.....	64
FIGURE 4-20: TRACING PIXELS BACK INTO THE VOXEL CUBE	66
FIGURE 4-21: RECONSTRUCTION OF A FLOWER BY VOXEL COLORING USING 16 IMAGES.....	67
FIGURE 4-22: AREA-BASED TEMPLATE MATCHING.....	69
FIGURE 4-23: TRANSFORMED SLAVE TEMPLATE AFTER LEAST SQUARES MATCHING.....	74
FIGURE 4-24: EPIPOLAR GEOMETRY	74
FIGURE 4-25: BASIC COLOR THEORY	75
FIGURE 4-26: AN IMAGE SHOWING THE PROPERTIES OF THE HUE-CHANNEL.....	76
FIGURE 4-27: THE IMPORTANCE OF COLOR	77
FIGURE 4-28: TEST IMAGE TO EVALUATE COLOR IMAGE MATCHING	79
FIGURE 4-29: VISUALIZATIONS OF THE COLOR MATCHING ALGORITHMS	80
FIGURE 4-30: COMPUTED STEREO DEPTH MAPS OF THE FLOWER (LEFT, MIDDLE) AND THE COMBINATION OF THREE NEIGHBORED VIEWS (RIGHT)	81
FIGURE 4-31: THE ENHANCED CONCAVE HEAD OF THE FLOWER USING BLOCK MATCHING (RIGHT).....	82
FIGURE 5-1: 18-CONNECTED (DARK PIXELS) AND 6-CONNECTED LINE (LIGHT+DARK PIXELS) IN 2D.....	83
FIGURE 5-2: THE BASIC IDEA OF AN 18-CONNECTED LINE TRACING	84
FIGURE 5-3: 6-CONNECTED LINE TRACING	85
FIGURE 5-4: THE USE OF CLASSES AND INHERITANCE	87
FIGURE 5-5: THE OPERATOR-CLASS FAMILY	87
FIGURE 5-6: MIN/MAX λ LIMITATION OF A LINE	89
FIGURE 5-7: SPHERE-INTERSECTION LIMITATION	89
FIGURE 5-8: CUBICAL INTERSECTION LIMITATION	90
FIGURE 5-9: SNAPPING OF INTEGER POINTS TO THE TRUE LINE.....	91
FIGURE 5-10: PROJECTING CORRECTED AND UNCORRECTED VOXELS.....	93
FIGURE 5-11: RECOVERING VISIBILITY INFORMATION	94

FIGURE 5-12: TOLERANCE FOR RECOVERING VISIBILITY INFORMATION	95
FIGURE 5-13: DIFFERENT RESULTS FOR VOXEL RENDERING	96
FIGURE 5-14: SURFACE VOXELS (GREEN) WITHIN THE RADIUS (RED) AROUND THE VOXEL OF INTEREST (DARK GREY) AND THEIR SURFACE NORMAL VECTOR (BLUE).	98
FIGURE 5-15: SEVERAL SURFACE NORMAL VECTORS SEEN ON A LARGER SURFACE FRAGMENT	98
FIGURE 5-16: ANGLE α BETWEEN THE SURFACE NORMAL \mathbf{n} AND THE PROJECTION VECTOR \mathbf{P} .	100
FIGURE 5-17: VISUALIZATION OF THE ANGLE BETWEEN SURFACE NORMAL AND PROJECTION VECTOR	100
FIGURE 5-18: PROJECTION ERROR DUE TO CONCAVITIES	101
FIGURE 5-19: MASTER PATCH AND DISTORTED SLAVE PATCH FROM TWO VIEWPOINTS	103
FIGURE 5-20: THE EFFECT OF KNOWLEDGE-BASED PATCH DISTORTION	106
FIGURE 5-21: CREATING A TANGENTIAL SURFACE PATCH	107
FIGURE 5-22: THE IMPROVEMENT OF TAKING THE OBJECT SHAPE INTO ACCOUNT	108
FIGURE 5-23: THE ‘MILLING MACHINE’ APPROACH	112
FIGURE 5-24: VOTING-BASED CARVING	116
FIGURE 5-25: DIFFERENT VIEWPOINTS OF THE NEFERTITI CUBE. THE RESULT OF SHAPE FROM SILHOUETTE (LEFT), REFINEMENT BY VOTING BASED CARVING (RIGHT)	118
FIGURE 5-26: THE APPROXIMATE MODEL WITHOUT AND WITH COLOR INFORMATION	119
FIGURE 5-27: CONSIDERING OCCLUDED AREAS	120
FIGURE 5-28: PROJECTING COLOR WITHOUT AND WITH CONSIDERATION OF VISIBILITY	121

TABLE INDEX:

TABLE: 1-1: AUTOMATIC AND SEMI AUTOMATIC SHAPE ACQUISITION SYSTEMS BASED ON THE TAXONOMY PROVIDED BY [CURLESS-II, 1997], [RUSINKIEWICZ-I, 2001]	5
TABLE: 2-1: COMPARISON OF SURFACE AND VOLUME GRAPHICS BASED ON THE TABLE PROVIDED BY ARIE KAUFMAN [KAUFMAN-I, 1993], [KAUFMAN-II, 1996]	18
TABLE 4-1: SPEED OPTIMIZATION USING HIERARCHICAL PROCESSING	51
TABLE 5-1: DIFFERENT ERROR TYPES FOR CARVING	115

LIST OF SYMBOLS:

M	: image center
x_0, y_0	: coordinates of principal point p_0
x, y	: metric coordinates of image point p
x_p, y_p	: left-handed pixel coordinates of point p_p
X_0, Y_0, Z_0	: coordinates of projection center O
X, Y, Z	: metric real-world coordinates of point P
X_v^0, Y_v^0, Z_v^0	: position of voxel cube origin in world coordinates X_v^0
X_v, Y_v, Z_v	: left-handed voxel coordinates of point P_v
X'_v, Y'_v, Z'_v	: cube-centered right-handed voxel coordinates of point P'_v
α, ν, κ	: terrestrial rotation angles
σ	: standard deviation
A, P, l, v, X	: adjustment matrices(design-, weight-, observation-, correction- and unknown matrix)
$R_{av\kappa}$: rotation matrix with angles α, ν, κ
r_{ik}	: elements of rotation matrix
λ	: scaling factor
c	: calibrated focal length
r	: correlation coefficient
w	: weighting factor
g, g_{RGB}	: density (gray value), in one and RGB channels, respectively
\bar{g}	: arithmetic mean of densities (or gray values)
R, G, B	: red, green and blue channel of a pixel
I, H, S	: intensity, hue and saturation channel of a pixel
$rows, cols$	or
n_x, n_y	: dimensions of a digital image
N_x, N_y, N_z	: dimensions of a voxel cube
s_p	: metric size of a pixel
s_v	: metric size of a voxel
$p.x, p.y$: accessing the x- and y-part of a point structure in pseudo code
\vec{n}	: normal vector to a plane
\vec{P}	: projection vector

Chapter 1 : Introduction

1.1 Introduction

Photogrammetry is a science which is occupied with recovering accurate 3D information from several 2D images. The main application areas of photogrammetry are conventionally arial and close range surveying. However, with the emergence of relatively new sciences such as image understanding and computer vision, photogrammetry has broadened its working area through these disciplines in which it has common interests. Computer vision is an image understanding application which deals with automatic extraction of different kinds of information from imagery. The main goal of computer vision is to make a robot system to sense, understand and interpret vision information just like we humans do. One of the purposes of computer vision is the creation of computer models of 3D objects from digital images which overlaps photogrammetric research area and it is also the goal of this dissertation.

The most important difference between these two disciplines is computer vision desires automation while photogrammetry seeks high geometric accuracy. Not surprisingly, photogrammetry is concerned with high accurate and reliable results; although in computer vision speed is more important which sometimes cause loss in accuracy. However, photogrammetry and computer vision are closely related disciplines and they meet in mutual research areas such as recovering the 3D structure of objects. Therefore, it is beneficial to apply computer vision and photogrammetric techniques in each other. For example when accurate measurements are required in a vision system, it is a good idea to investigate photogrammetric solutions first, since obtaining precise measurements from images has been studied long time in photogrammetry. Accordingly, when automation is desired in photogrammetric tasks computer vision techniques can be used. It is because of this reason; inheriting the well known and efficient techniques developed by computer vision scientist and photogrammetrists rather than re-inventing them would save time and help these two disciplines grow faster. Moreover, a system which brings together the most efficient photogrammetric and computer vision techniques would provide more robust and fast results. In this dissertation issues from both photogrammetry and computer vision are addressed in order to reconstruct volume models of arbitrarily-shaped scene from its color images.

Reconstruction of the shape of 3D objects from a series of digital images is a challenging problem both in photogrammetry and computer vision. When generating 3D models with traditional CAD techniques, the details of 3D objects should be entered manually using graphical interfaces. Furthermore, with traditional CAD techniques objects are modeled with polygon patches and traditional material description which does not give high degree of realism. Since it is a labor intensive and complex process, the efforts concentrated in automatic modeling of 3D objects either with active methods by scanning the objects or with passive methods by using their images taken by a CCD camera. This dissertation presents an efficient image-based approach to compute volume models.

In this dissertation a voxel-based method is used to recover the shape of the 3D objects. Volume graphics is an advancing field in computer graphics. Volumetric data was first introduced in early 70's in medical imaging and now it has been commonly used in scientific visualization, computer graphics and computer vision. Although voxel methods consume large amounts of memory, with the rapid growth of memory and processing requirements in the last decades, it has been gaining importance and it has become a practical alternative to surface based representations. Specifically volume models provide more flexible representations of 3D objects from multiple images of the scene. Furthermore volumetric representations allow us to represent complex objects more effectively. And if desired,

they can be converted into polygonal models in order to be rendered effectively with the use of standard graphics hardware.

The model of the 3D object can be easily acquired by shape from silhouettes technique which is a well known and popular method in computer vision where the shape of the objects is recovered by intersecting the volumes. Shape from silhouette method is fast and robust; however the concave areas on an object cannot be recovered with this method. The later work to recover the shape of the objects from multiple images is concentrated on voxel coloring algorithms. These algorithms use color consistency to distinguish surface points from the other points in the scene. In this dissertation a new method is proposed with several powerful tools to create voxel-based photo realistic volume models from multiple color images.

1.2 3D Shape Acquisition Techniques

The first step of scene reconstruction is sampling the real environment. In this dissertation image-based volumetric modeling is addressed that means the source of data is multiple color images taken by inexpensive CCD video cameras. Before image-based modeling is discussed, an overview of other available 3D shape acquisition methods using different instruments rather than inexpensive cameras are introduced.

When creating 3D models and its textures with classical CAD methods, graphical interfaces are used to enter the details of 3D objects. Since this is a labor intensive process, a great amount of work in this area is focused on automatic modeling of 3D scenes by scanning the objects. There is a great need for acquiring the shape of objects rapidly and without physical contact in many application areas, such as industrial inspection, robot vision, virtual reality, reverse engineering as well as 3D object reconstruction and navigation. Object reconstruction pipeline begins with sampling a real world environment. Generally we can classify the shape acquisition systems into physical contact, transmissive and reflective sensors. In the following different types of 3D acquisition systems are given based on the taxonomy provided by [CURLESS-II, 1997], [RUSINKIEWICZ-I, 2001].

1.2.1 Physical Contact

There are two types of physical contact methods namely mechanical and destructive methods. An example for destructive methods can be given in medicine. After the brain is made solid by freezing, it is sliced by a machine called microtome. The slices are scanned by high resolution instruments like electron microscopes in order to reconstruct brain tissues.

An example of physical contact devices is coordinate measurement machines (CMM) which is illustrated in Figure 1-1. These machines are used in manufacturing industries such as in airplane manufacturing where the wings of the plane are needed to be measured and modeled accurately.

CMM machines have contact sensors such as tactile tips to detect the surface of the object to be measured and these machines give high accurate results. In order to digitize the surface, the tactile tip is placed on surface points of the object, then a contact signal is given and coordinate of the tactile tip is recorded. The other alternative is tactile tip is moved on the object's surface by a user while the machine delivers coordinates in certain intervals. In some cases the contact with the object is not desirable especially when the object is made of a fragile material. However, new developments allow fragile parts to be measured without damage due to stylus touch force.



Figure 1-1: ICMM 'FaroArm – Platinum' from FARO Technologies Inc. (www.faro.com)

There are also photogrammetric solutions to the physical contact 3D shape acquisition. In those approaches solid and calibrated tactile devices are used which have markers on them. In the first photogrammetric approach, several cameras are placed in the environment where 3D shape acquisition is performed. All these cameras are fully calibrated and oriented. The tactile device is interactively placed on the object surface to be reconstructed. Meanwhile, cameras track the markers on the tactile device. Therefore, the orientation of the device can be derived as well as the exact position of the coordinate tip. As a result, the 3D position of the surface point which is detected by the tactile device can be calculated. In the second photogrammetric solution the situation is reversed. Several cameras are placed on the tactile device itself and the control points are applied to the environment and the camera positions are oriented [LUHMANN, 2000].

1.2.2 Transmissive

A non-contact system projects some sorts of energy on the objects surface and records the transmitted or reflected energy. Therefore, non-contact methods can be subdivided into transmissive and reflective methods.

Industrial CT is a transmissive method where a beam of radiation is projected through an object and detectors measure photon interaction between radiation and object matter. The sampled data obtained is straightforward for volumetric reconstruction. Inner parts of the objects can also be acquired by CT devices; however these devices are expensive and might be very hazardous.

Although scanning electron microscopes are mostly used as reflective sensors, they can also work in a high resolution transmissive mode. Electron microscopes work by scanning a specimen surface with an electron beam. In the TEM mode (Transmission Electron Microscope) those electrons are detected which completely penetrate the specimen.

1.2.3 Reflective Methods

Reflective methods are subdivided into two categories, namely optical and non-optical methods. Non-optical methods are microwave radar and sonar. Their main principle is to measure the time required for a pulse of microwave energy or sound reflected from the object. Sonar systems are inexpensive but inaccurate. Microwave radar is generally used for long range remote sensing.

An example for a non-optical reflective sensor is the scanning electron microscope, since it detects backscattered electrons (BSE) or secondary electrons (SE), from the interaction of the electron beam with the specimen's surface.

We can classify optical 3D shape acquisition techniques into two groups: passive and active methods. Image based techniques are passive methods which do not interact with the objects while range scanning techniques are active methods which make contact with the objects by projecting light on them.

1.2.3.1 Active Optical Methods

Active range scanning methods are based on time of flight, depth from defocus [NAYAR, 1995], active stereo and projected light triangulation. Range imaging sensors determine depth either by measuring the time of flight or by triangulating the position of a projected laser pulse. The result is a range image which samples depths on a regular grid.

1.2.3.1.1 Active Stereo

In stereo methods, a feature in one image is searched along the epipolar line in the other image in order to find correspondences. Stereo methods fail on non-textured areas. In the active stereo method, structured light is projected on the objects surface and recorded by two or more cameras. By passive use of structured light the surface of the object is textured with large number of dots and evaluated with correlation techniques. [MAAS, 1992]

1.2.3.1.2 Shape from Projected Texture (Interferometry)

Scanning an object with light is a method to obtain 3D information about its shape. It is also the main principle of depth perception in machine vision. Shape from projected texture (interferometry) is based on active triangulation of projected light. Many techniques are developed based on structured light as Moiré techniques or coded light approaches. Different from active stereo, spatially or temporally varying periodic pattern is projected on the object's surface. In standard stereo vision, triangulation is performed to compute 3D coordinates of the corresponding points. In structured light, the scene is illuminated with a light pattern of known geometry and captured with a single CCD camera to observe how the structured light is distorted by objects. In order to get dense range information either the laser plane is moved in the scene or multiple stripes are projected at once with coded colors. The object surface is then reconstructed by matching projected light features to observed edges in the image. In other words, projector profiles are matched to sensor profiles.

3D Shape Acquisition	Physical Contact	Optical	<ul style="list-style-type: none">• Photogrammetric Measuring Device	
		Mechanical	<ul style="list-style-type: none">• Coordinate Measuring Machine (CCM)• Jointed Arm	
		Destructive	Slicing (Microtome)	
	Transmissive	<ul style="list-style-type: none">• Industrial Computed Tomography (CT)• Ultrasound• Magnetic Resonance Imaging (MRI)• Transmission Electron Microscope		
	Reflective	Non-Optical	<ul style="list-style-type: none">• Sonar• Microwave Radar• Scanning Electron Microscope	
		Optical	Active	Active variants of passive methods <ul style="list-style-type: none">• Active Stereo• Shape from Projected Texture (Interferometry)• Active depth from defocus
				Time of Flight
Triangulation				
		Passive	<ul style="list-style-type: none">• Shape from Silhouettes• Shape from Stereo• Shape from Shading• Shape from Focus/Defocus• Shape from Motion• Photogrammetric triangulation	

Table: 1-1: Automatic and semi automatic shape acquisition systems based on the taxonomy provided by [CURLESS-II, 1997], [RUSINKIEWICZ-I, 2001]

1.2.3.1.3 Active Depth from Defocus

Depth from defocus addresses the reconstruction of the depth information by using the fact that the blur in images increases with the increasing distance from the location of focus. In other words, the objects that are at a specific distance from the camera are focused on the image while the objects at other distances are blurred with respect to their distances. For example when we focus the lens to a close distance in the image the foreground will be well focused but the objects in the background would appear blurred. The basic idea of depth from defocus method is to correlate the grade of the blurring to the distance, by doing so blurring, in other words image defocus is measured in order to compute 3D positions. In passive methods the amount of blurring is determined by surface reflectance. Therefore the depth from defocus method is dependent on the objects texture. Defocus is not the only source of blur but can also be disguised by smooth brightness changes in the scene. Active methods avoid this problem by projecting structured light on the objects surface.

1.2.3.1.4 Time of Flight

In time-of-flight (TOF) methods (imaging radar), a pulse of light is emitted and the round time required for the signal to return from a reflective surface is measured. This is the main principle used in electronic distance measurement instruments (EDM), radar etc. Different from microwave radar, time of flight laser range finders operates in optical frequencies. This method requires precise timing in close range applications where the distance is short and the desired resolution is high.

1.2.3.1.5 Optical Triangulation

Optical triangulation is the most popular range scanning approach. Optical devices consist of an emitter and a sensor, which is typically a CCD sensor, in a well known geometrical context. The laser emits a line or a laser-plane into the scene and the reflected light is detected by a CCD camera. Since we know the exact position of the sensor with respect to the emitter we can obtain the depth value by performing triangulation.

The illuminant and the reflected light intersect the object at one point which gives a depth value. Each range image is the depth information of an object from one point of view. In order to reconstruct the shape of an object we require multiple images. Therefore images should be aligned with respect to each other in the same coordinate system. The surface reconstruction is then obtained by merging aligned multiple images. In his PhD thesis, Brian Curless presents an efficient volumetric method for merging large number of range images which yields high detail models up to 2.6 million triangles [CURLESS, 1997].

Usually, laser triangulation works in combination with a scanning device, which moves the projection over the objects surface, either in one direction, or in a circular motion. It can also be combined with a mechanical arm, which was mentioned in Chapter 1.2.1. Instead of a tactile tip, a laser triangulation device is attached to the mechanical arm. While the user is moving the scanner along the object, the arm delivers its position and orientation. This way, a very dense coverage of the object can be achieved.

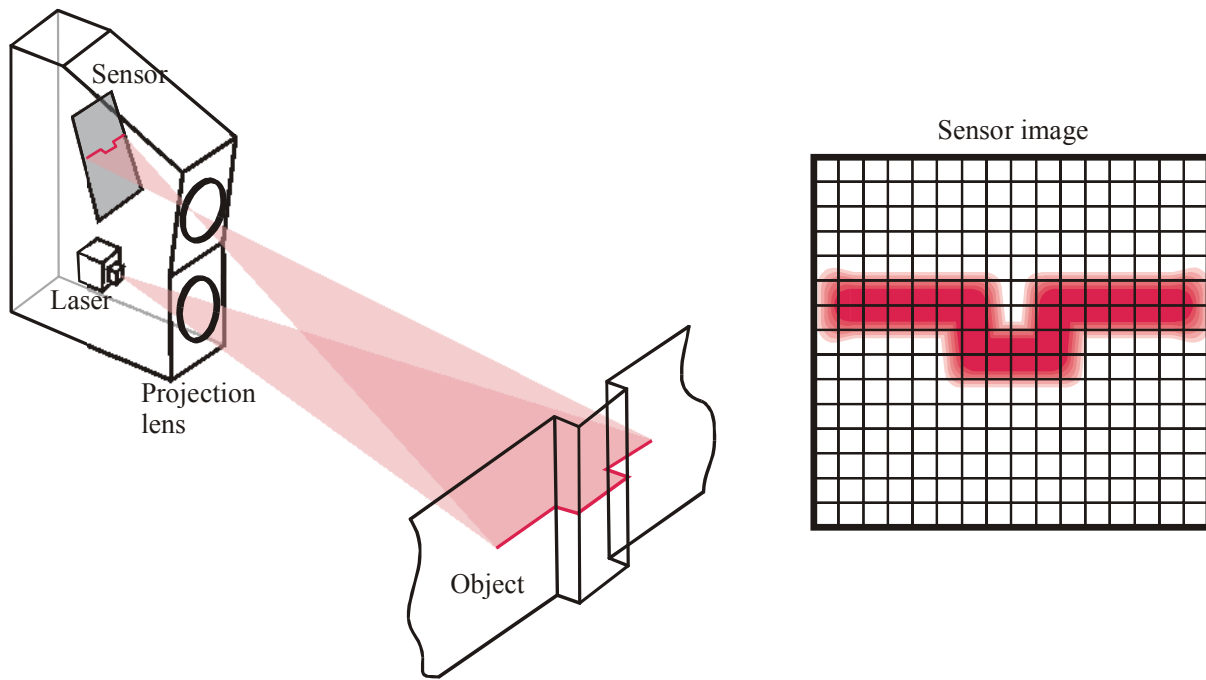


Figure 1-2: A scheme of optical scanner where 3D points are computed by triangulation

1.2.3.2 Passive Optical Methods

Passive methods include computer vision techniques for example shape from shading, shape from motion, shape from focus/defocus and stereo triangulation for image pairs. Passive methods do not contact the object being scanned physically. They require low cost equipment and their general application is object recognition. In this thesis, passive methods are referred, in particular volumetric modeling from multiple images. Since shape from silhouettes and shape from stereo will be discussed in detail in following chapters, it will not be described here.

1.2.3.2.1 Shape from Shading

It is possible to recover the shape of objects from a single image. Different parts of the surface of an object are oriented differently thus they appear with different brightness. Spatial variance of brightness, namely shading, is used to estimate this orientation of the surface patches. Shape from shading [HORN, 1970] methods use these variations in brightness and shading in images in order to recover the shape of the objects. This method delivers 3D information from image irradiance and known positions of the cameras and light sources. In this technique the reflectance of objects is modeled as a function of the angles of incidence i and emergence e . There are different shape from shading techniques like using multiple images with different illumination or using image sequences with moving light sources. Shape from stereo works best with textured images while shape from shading works best with images contains smooth brightness variations. This method is applicable to some tasks where the lighting can be controlled. On the other hand, the shape from shading method is also used in order to generate DEMs when the traditional stereo processing methods are not applicable on areas such as deserts or forests.

1.2.3.2.2 Shape from Motion

Shape from motion is one of the approaches which gained attention by the computer vision scientists during the last decades. Motion can be used as another dimension which is time in computer vision and a great deal of information can be recovered from images over time. In order to recover the shape, we can use the movement of the camera relative to the static scene in the same way as using multiple camera positions. The moving camera captures a succession of images over time that span a baseline similar to stereo images. Therefore it is possible to obtain depth information of the object by tracking object features in multiple images. However we should know the velocities of the camera and the moving shape in the scene. Shape from motion is a technique to recover a set of 3D structure parameters and time varying motion parameters from a set of image features. This method computes the structure from a set of matched points and the performance is significantly improved by using least squares approach. Numerous shape from motion algorithms have been developed. [JEBARA, 1999], [TOMASI, 1992], [SZELISKI-III, 1994]. They are two main variants, shape from motion with calibrated and uncalibrated cameras. In general the scene is captured at video rates therefore the disparities are smaller. In video sequences points are tracked automatically and the matches are validated. Both points and lines can be used for reconstruction. Although the quality of structure and motion is reasonable, the metric accuracy is weak and it needs good feature trackers.

1.3 Image-Based Modeling and Rendering

Generating images of a scene for new viewpoints which can convince the observer that they look photorealistic compared to input images is an important aspect and the basic goal of computer vision. Since real images are being used to model and render the scene therefore photorealistic as a result, computer vision is especially interested in image-based modeling and rendering (IBMR). The main goal of IBMR is to create new views of real environments from a set of its images or of its 3D model and achieve faster and more realistic renderings. With the recent advances in computer vision algorithms in recent years, image-based modeling and rendering has become an alternative to classical computer graphics.

In traditional computer graphics the 3D model of the scene is constructed at first, and then an image of the scene is created by projecting this 3D model onto an image plane using the geometry and camera model and determining the color of image pixels and lighting. This is 3D to 2D transformation. The constructed model includes the geometry of the scene which consists of polygon patches and the surface properties such as reflectance, color or opacity. The realism can be increased using wire frame models, shadow calculation techniques and the realistic renderings can be achieved by tracing the paths that light might take. However, it is hard to model some objects with classical computer graphics techniques such as complicated objects with detailed concave surfaces or fuzzy surfaces with difficult reflectance characteristics. It is because the tools of creating complex models are labor intensive and requires a great deal of talent and when images are rendered complex computations are carried out. In other words, in our world there is an imperfection and we still cannot model these imperfections and real world effects correctly. However, images give us great amount of information about the real world objects since detailed real world affects are captured with images. Moreover taking images is easy, besides it is much more photorealistic.

In computer vision, the images of the scene are given in order to reconstruct the 3D model of the real environment; therefore 2D to 3D transformation is concerned. By image-based modeling algorithms we can model the scene and look at it from different viewpoints with different angles and distances. In IBMR, images are used to determine scene geometry, lighting and reflectance properties of the objects and new rendered images are obtained. In this case, the transformation is from 2D to 2D. This transformation can be achieved either by using computer vision techniques to reconstruct 3D model of the real environment or can also be achieved skipping 3D modeling stage. This is when IBR steps in. An image-based model represents the images of the scene with corresponding 3D locations of the image points and the rendering is obtained by resampling the reference images for a new viewpoint.

IBMR is attractive mainly because of its speed and realism. The modeling of complex scenes is simplified by using images. Furthermore, rendering time is not affected by the scene complexity. Image-based rendering uses large amounts of images in order to provide a high degree of realism. An important characteristic of image-based rendering is that previous modeling is not required. The samples of the environment are captured by photographs or video sequences and they are sampled during the rendering. For example the photographs might be taken in all directions of the object of interest and can be grouped as a panorama so that we can look around the object by displaying different sections of this panorama. In order to move around the scene and look from an arbitrary position we can take as many images as desired from different viewpoints. An early work of the image based rendering approach is Lippman's movie-map. In his work, images of several streets of a city were captured at periodic intervals and simulated by playing back with respect to the closest virtual camera position [LIPPMAN, 1980]. Examples of recent imaged-based rendering systems can be given as; Panoramic Image Mosaics [CHEN-II, 1995], Concentric Mosaics [SHUM, 1999], View Interpolation and View Morphing [CHEN-I, 1993] [SEITZ-I, 1996], Lightfield Rendering and Lumigraphs [LEVOY, 1996] [GORTLER, 1996], Layered Depth Images and Sprites with Depth [SHADE, 1998], Plenoptic Stitching [ALIAGA, 2001]. The main idea of the above image-based rendering systems is sampling the environment with two or more images and generating novel views by using image warping and blending. Since the scope of this dissertation is image-based modeling of real world objects and scenes with voxel based methods, IMR systems will not be mentioned here in detail. You can refer to the papers [SZELISKI-V, 2001] and [OLIVERIA, 2002] for more information about these systems.

In image-based rendering systems, if we want to move around the scene and observe the scene at any direction as we desire we would need a large number of images. If we build accurate full 3D models however, we can reduce the number of required images. This is where we can use image-based modeling. In image-based modeling, 3D model reconstruction is performed from images as opposed to polygons; hence, the modeling step is simplified. Apart from avoiding the labor intensive geometric modeling step, the combination with image-based rendering is very attractive because fine details are hard to render with classical rendering algorithms.

Geometric representations of the scene can be derived by photogrammetric methods, image correlation or by range scanning. The acquired geometry can then be rendered by projecting the colors to the models from photographs. Both range scanning techniques and image-based techniques have advantages and disadvantages in many ways.

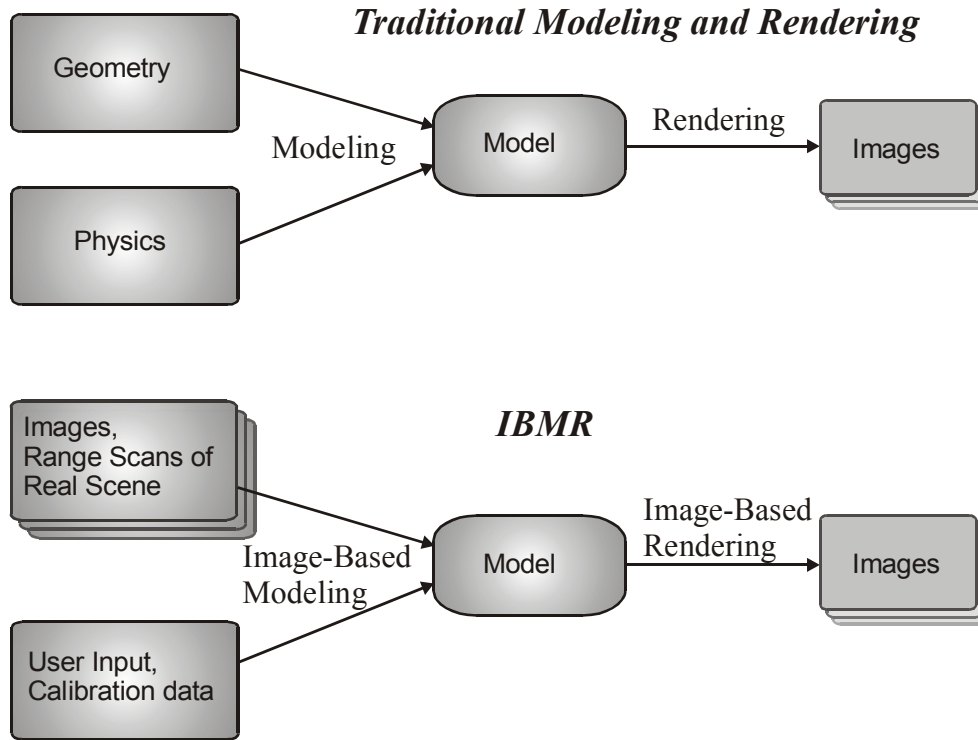


Figure 1-3: Image-based rendering and traditional modeling and rendering methods

The computer vision science is concentrating on image-based techniques where the main goal is to extract useful information from the images automatically. The images are acquired easily with widely available inexpensive cameras. Therefore, image-based techniques are attractive in many applications. In this thesis image sequences acquired by a still video camera are used. Range scanning techniques alone are insufficiently realistic renderings of the object. Therefore the images are required to obtain texture of the object. Besides, some architectural objects which we want to reconstruct might not be existent any longer. In such cases, image-based approaches should be used in order to reconstruct the architecture based on its old, still existing photographs. For the reconstruction of destroyed historic buildings, refer to [WIEDEMANN, 2000]. Debevec presents both geometry based and image-based modeling techniques in order to model and render architectural scenes from photographs [DEBEVEC, 1996]. Debevec recovers the basic geometry of the scene with photogrammetric modeling methods which yield effective and robust results. In order to recover how a real scene deviates from the basic model he presents a model-based stereo algorithm. Finally, Debevec renders the scene with view-dependent texture mapping method. In this method, he uses a combination of views of a scene in order to simulate geometric details better on the models.

Image-based modeling and rendering has become more attractive, because when compared to polygonal based renderings, the number of polygons in the scene is approaching the number of the pixels in the images. However, the disadvantages are large storage requirements and the lack of guarantee to obtain the complete object reconstruction. Even when a large number of images are used, some parts of the objects might not be visible in any of the images captured. Therefore some gaps might appear on the modeled surfaces. Hence, further research is concentrating on methods to determine how many images should be used in order to represent the scene successfully, and if those images contain enough information without artifacts due to visibility or insufficient sampling.

1.4 Aim of the Study

The main goal of this dissertation is to create photorealistic models of the real-world objects. In order to reconstruct object models, it is aimed to use low-cost equipment. In general, the 3D object acquisition can be performed by active methods using range sensors such as laser scanners or by passive methods using CCD cameras. Nowadays laser scanners are used in many applications to acquire the 3D shape of the objects by producing very dense point clouds. Although using cameras to acquire 3D models is cheaper, it is not as commonly used as laser scanners. Since one of the goals of this dissertation is to create 3D models of the objects keeping the system requirements as low as possible, the image sequences of a CCD camera are used which is a low cost alternative and therefore attractive for many applications.

Another aim of this thesis is create photorealistic models of the objects. Photorealistic scene reconstruction is a challenging task in computer graphics. Even after many hours of tedious work with 3D CAD programs, the results are synthetic images which cannot deliver the desired photorealism. Since images contain enormous information about the real-world effects, in this thesis images are used to model the object. The final model is textured using the images also and when it is rendered to arbitrary positions it produces high photorealistic new images.

In this thesis it is aimed to reconstruct complex objects containing concave areas. While man made objects can be easily reconstructed due to their regular and linear shape, many natural objects are harder to model since they contain concave, critical areas. Examples of such objects can be given as flowers or leaves of a tree.

Another aim is to develop efficient voxel processing algorithms. 3D object modeling is traditionally done by polygonal methods. In this thesis however, voxels are used as 3D representation primitives. In voxel based representations the scene is divided into cube-shaped elements, named voxels, which might provide a more flexible representation of 3D surfaces. In this thesis the aim is to create a voxel surface that would correctly represent the 3D geometry of the true shape of the object.

This thesis aims to create a robust algorithm to create 3D voxel-based surfaces. Therefore fast computer vision techniques are used and then the accuracy is improved by photogrammetric techniques.

1.5 Problem Statement

This dissertation presents a reconstruction method for textured objects. The object is reconstructed in several steps and each step has different kind of problems. The first step of the reconstruction pipeline is image acquisition. Hence, we have to clarify which is the most convenient system setup to acquire the images. Thus, the type of the camera and where to place those camera(s) should be considered. Another issue is the illumination source which affects the image quality and color of the images. Therefore, it should be considered what kind of illumination source should be used and where to place it.

The acquired images need to be oriented. The projections can only be performed in a high accurate way when precise image orientation parameters are known. The accuracy of the image orientation parameters derived from the turntable positions should be investigated and if necessary the accuracy should be improved by using bundle block adjustment.

When working with voxels, one of the most important problems encountered is the memory requirements since voxels consume large amounts of memory. Therefore, the voxel cube should be defined so that it can approximate the area of interest in the best way.

Since color images are captured instead of gray images, we should make best use of color information. Therefore existing image matching algorithms can be evaluated for color image matching.

The volume intersection method results in the approximate shape of the object and contains some extra voxels in concave areas. Thus, it is necessary to investigate algorithms to get into these critical areas and remove those extra voxels correctly.

Visibility information recovery is an important problem. It should be known if a voxel of interest is visible from a certain viewpoint or not. This is especially important when we perform image matching, since only voxels on the true surface of the object project into similar regions on the images. Hence, visibility of a voxel should be computed correctly and efficiently. One problem encountered during visibility recovery is determining where a voxel of interest is heading towards. Another problem is tracing a line efficiently.

Line tracing is required in several steps during the reconstruction process. However, tracing a ray of sight is a time consuming process. Therefore a sensible geometric limitation should be defined to trace the line only within the defined voxel space.

Another problem in image matching is caused by perspective distortions from different viewpoints. It is essential to grab the slave patch taking into distortions into account. This will improve the reliability and robustness of image matching. One problem encountered is deriving the tangential surface of a voxel.

The solutions of the above addressed problems will be given in the following chapters of the thesis.

1.6 Areas of Application

Our research is motivated by many applications. Volumetric representations of 3D objects can be applied to variety of fields as described below.

1.6.1 Reverse Engineering

Reverse engineering is a process of taking a part of a product and see how it works and model it precisely in order to duplicate it. The parts of the products nowadays are designed by Computer Aided Design (CAD). However, in some cases, the CAD model of a part does not exist or the CAD model should be updated to reflect the dimensional changes that occurred during prototyping. In this case the part is taken out from a working system and a set of measurements are made by various sensors to digitize it precisely for re-manufacture. Reverse engineering is applied in applications like when a part of a machine is broken or worn out which has been modeled manually therefore no CAD exists.

1.6.2 Medicine

The volume visualization is commonly used in medicine applications in order to visualize the inner parts of the organs.

3D reconstruction of the body models is used in surgical planning laboratories as a studying tool before you go through with the surgery. It will show you where to cut and give you some preparation before you actually do the surgery. In plastic surgery, modeling the topography of the patient's body or face surface enables the doctors decide how much fat to remove or how large the implant should be. 3D images also enable doctors to communicate with their patients before the surgery and simulate the surgery results in a more realistic way.

In radiation treatment in oncology the tumor should be destructed precisely without giving damage to the vital structures near the tumor area. Radiation treatment planning enables irradiating ray from different angles that intersects in the tumor area preventing the side affects in surrounding tissue. Modeling the shape of the patients body help the doctors visualize the radiation distribution.

When 3D reconstruction of the anatomical map of the patient is acquired in high precision, calculations and measurements can be performed in order to design prosthetic implant. This enables the physician high accurately plan the prosthetic and increase the patient comfort.

1.6.3 Entertainment

In entertainment, reconstruction of objects and characters by using CAD software is a labor-intensive process and the result may not look realistic. Computer graphics is increasingly used in special effects for movies. The animated characters in film industry should move and appear realistically in order to satisfy today's viewing public. As a result of high realism demand in computer graphics, film industry in particular has adopted techniques from computer vision because complex scenes would look more realistic if they are generated from its images. Therefore image based scene reconstruction, has attracted significant interests since images provide tremendous details, yet can be acquired fast, cheap and easily even with inexpensive amateur cameras.

Voxel-based graphics is used by a number of computer games. Computer games industry uses voxel techniques as a hybrid system along with polygonal methods. For example polygons are used to represent nearby features and voxels are used to render the terrain.

1.6.4 Virtual Worlds, Virtual Walkthrough, Telepresence

Virtual worlds are computer generated, 3D environments in which virtual world users can participate interactively. Some examples are virtual museums, virtual library and virtual university. Virtual worlds provide users to view virtual objects from different distances and angles. In virtual worlds there are large numbers of dynamic virtual objects which change their position with respect to the viewer. Therefore virtual objects should be modeled effectively. In a virtual walkthrough application, a person can visit a specific place with access to the internet or locally. When viewer walks through the virtual world the information about the virtual objects are rendered into images and viewed by use of monitors or devices like head-mount displays.

Similar to the virtual reality where presence in computer simulation is desired, telepresence is a human/machine system which provides the perception of being present in a remote location. Real environments which are not attainable can be visited by telepresence technologies.

In all these applications, information about virtual objects with their position and shape should be obtained. In order to increase realism of the virtual world, objects can be modeled from their images.

1.6.5 Cultural Heritage

A virtual museum is a virtual environment where the artifacts and information resources of the museum can be viewed locally or on the internet for cultural, research or educational purposes. 3-D reconstruction, modeling, documentation, monitoring, and visualization of cultural heritage artifacts can be done by volumetric displays. Also refer to [KUZU-IV, 2003]. In virtual museums the representation of historical artifacts are enriched by online explanations, animations, music, video and narrations.

A full three-dimensional reconstruction serves as a permanent record of the heritage artifacts in their original position. Such a reconstruction can be used to detect the changes for conservation purposes. The models may also serve as a manufacturing blueprint for machine production of a replica for exhibition purposes. If the object is an historical device for example the visitors might like to see it in action and perform their experiments. High precise reconstruction of the artifacts or replica can be made accessible to scholars and visitors.

1.6.6 Home Shopping

Home shopping is the ability to purchase all kinds of goods by placing orders over a communications network. As a result of the widespread use of internet, commercial vendors are taking advantage of marketing their products on the World Wide Web. By displaying 3D models of the products, the vendors allow their customers to explore the products interactively before purchase.

1.6.7 Industrial Inspection

Industrial inspection is to detect the deviations of the manufactured industrial products with respect to its CAD model for quality control or to calibrate the production process. The real industrial object can be captured by any scanning or imaging system and the geometric model is reconstructed. The captured data and the ideal CAD model are matched and the difference is analyzed to see if the real object differs from the planned CAD model in some regions. Detected errors can be also used for quality control, in order to discard the worn out industrial parts.

1.6.8 Design

Three dimensional shapes are traditionally designed by CAD tools from scratch on the modeling system. Despite the advances in CAD tools, they still don't provide high realistic representations. Since appearance is important for some consumer products with high aesthetic values, cars for example, the combination of digital and physical modeling is becoming an important principle in computer aided industrial design. The artist manually creates a physical model using a material like clay or wood. This hand made model can be photographed in order to create a 3D computer model. 3D reconstruction of the sculpture is used to create a CAD model for manufacture.

1.7 Contents of the Thesis

The following chapters of the thesis are as follows:

Chapter 2 discusses the general 3D representation techniques and introduces the primitives used for 3D reconstructions. Since voxel data and volumetric reconstruction techniques are not so commonly used in photogrammetry, Chapter 2 also introduces the voxel data. Moreover, in Chapter 2, the fundamental concepts of volume data and voxel topology are provided as well as storage of the volume data and morphological operations with voxels. The comparison of surface graphics and volume graphics is also presented in this chapter, the weaknesses and the advantages of voxel data is discussed.

In Chapter 3, the coordinate systems used in this thesis are given and the transformations between them are presented. The calibration and image orientation techniques used are also presented in this chapter.

In Chapter 4, some state of the art volumetric object reconstruction techniques using images are discussed and contributions of this dissertation to these techniques are also explained and experimental results are provided. Furthermore, in Chapter 4, image matching techniques are given and block matching algorithm is proposed to refine the approximate model of the object. Also in this chapter, color image matching algorithms which are used in Chapter 5 are introduced and compared.

In chapter 5, a new volumetric reconstruction algorithm is proposed and some reconstruction results of the new method are presented.

Chapter 6 concludes the thesis. The contributions are summarized and some areas of future work are described in this chapter.

Some material presented in this thesis has been published by Kuzu, Sinram and Rodehorst previously. [KUZU-I, 2001], [KUZU-II, 2002], [KUZU-III, 2002], [KUZU-IV, 2003].

Chapter 2 : 3D Representation Techniques

2.1 3D Primitives

Three dimensional visualization techniques can generally be classified in two methods as surface graphics and volume graphics. Surface graphics use polygons, lines and points as geometric primitives when volume graphics use voxels as unit volume elements. According to the data structures they use we can say that in the same way surface graphics is the 3D counterpart of vector graphics, volume graphics is the 3D counterpart of raster graphics.

We can classify the data structures mainly into two groups as geometry-based models and digital images. The 3D primitives of geometry based models are lines, points, curves, polygons and surfaces. Geometry based models are data structures that represent objects in terms of their geometrical descriptions like size, position and orientation. For example in computer graphics pictures are created from geometry-based models.

In digital images, colors as well as gray values are represented as 2D arrays. However digital images do not contain information about the geometry of the objects that have been imaged. A digital image is a 2D array of data that represents color values or gray values of sample points that lay on a plane area. The unit element of a digital image is called pixel, in other words “picture element” and it is regarded as a small rectangular area. The digital image is obtained by scanning of a photograph or obtained by capturing the real time signal of a digital camera and displayed on a raster output device. Image processing deals with manipulation of digital images and has several kinds of objectives, for example image enhancement and image understanding.

In former computer graphics techniques objects were represented by lines, curves, surfaces and boundary representations with meshes of polygons. With the advent of raster technology, image processing and geometry based computer graphics adopted techniques from each other. For example, remote sensing applications combine satellite images with geographic maps which are based on geometric descriptions of the earth. In computer graphics today texture mapping is used as an image processing technique in order to produce more realistic looking images of the surface.

2.1.1 Surface Graphics

In surface graphics, raster technology is used for display and surface graphics has the geometry-based approach of vector graphics to maintain and manipulate the display list of geometric objects. After every change in scene the frame buffer is regenerated. Polygons require intensive processing, therefore, hardware is supported by powerful geometry engines for hardware acceleration which makes surface graphics state of the art of computer graphics. In surface graphics geometric primitives are kept in the display list and these primitives are transformed and mapped to the screen coordinates and then converted into pixels by scan conversion algorithms, in the end stored in the frame buffer or in a database. This process is called pixelization or rasterization. This process is repeated when there is a change in the scene like when viewing parameters are changed.

2.1.2 Volume Graphics

Volume graphics is an advancing field in computer graphics. Voxels and pixels both represent the sampled data by volumetric data sets and images. In volumetric data, the samples are represented in

3D instead of 2D therefore it is the counterpart of digital images. As scientific visualization becomes more demanding and sophisticated the research, in order to develop better techniques to create 3D representation other than traditional computer graphics techniques gained speed in recent years. Volume visualization is the method to extract information from volumetric data sets for manipulation and rendering. Volume graphics use volume buffer for representations of 3D scenes instead of using a list of geometric objects. The scene is discretized and converted into unit volume elements, voxels. While volume visualization is generally concerned with sampled data sets, volume graphics is concerned with modeled geometric scenes. A geometric model which is created by conventional surface graphics methods can also be converted into volume model. Geometric model which is defined by an analytic formula is voxelized and stored in volume buffer. This process is also used to convert some objects which are not sampled into the volume model. Surgery tools for example can be voxelized and intermixed in the volume buffer with the sampled organ.

2.1.3 Comparison of Voxel Methods with Polygonal Methods

In general, the advantages of voxels are; they are not affected by scene and object complexity, they can represent sampled data and inner information effectively and they support block operations. The main disadvantages are memory and processing power requirements, the loss of geometric information, and its discrete form. In the following section the advantages and weaknesses of volumetric methods will be discussed based on the comparison table presented by Arie Kaufman [KAUFMAN-I, 1993], [KAUFMAN-II, 1996].

	Surface Graphics	Volume Graphics
Rendering Sensitivity to Scene/Object Complexity	-	+
Block Operations	-	+
Sampled Data	-	+
Interior Parts	-	+
Memory and Processing	+	-
Aliasing	+	-
Transformations	+	-
Geometric Information	+	-

Table: 2-1: Comparison of surface and volume graphics based on the table provided by Arie Kaufman [KAUFMAN-I, 1993], [KAUFMAN-II, 1996]

2.1.3.1 Advantages of Voxels

Voxel methods have several advantages over the polygonal methods. Although modeling stage is affected by object and scene complexity, rendering is insensitive to scene and object complexity. Because the rendering stage is only affected by the resolution of the volume buffer not the number of

the objects in the scene or how complex the objects are. Therefore objects can be rendered without increasing the amount of processing power greatly. Anti-aliasing and texture mapping is performed only once and a color value is assigned to each voxel. When an object is hard to render by conventional methods volume graphics is preferable.

Voxel methods seem to be more natural since they can represent inner parts of the objects and represent digital samples directly and easily. In some applications such as computational fluid dynamics and volume microscopy sampled data is already in hand therefore such fields are the main application areas of volume graphics.

In polygonal methods an object is represented by edges and surfaces approximated to polygons. However, in our environment 3D objects might not consist of edges and surfaces and might have critical concavities.

Imagine the image of a tree. To have a realistic representation we would choose to use of a raster image scanned from a photograph or from a digital camera directly. Since this is an image of the real world, it will look accordingly real, depending on the image resolution. If we were to represent a tree (here 2 dimensional) with lines and polygons, we would have tremendous work to do yet it would be incomplete and still look unreal.

There is an irregularity in our nature so real world objects do not have sharp edges. Moreover, they might have information inside them. Not only can the shell of the objects, but also the interior parts also can be represented by voxels. This is very important for applications like medicine when inner organs have to be visualized. On the other hand, objects without surfaces or real phenomenon like cloud, fire and smoke are hard to be represented by polygons. Therefore, polygons might not provide accurate representation of natural objects. However, they can represent many man-made objects very accurately. A table for example can be represented by polygons very easily and quickly since it consists of flat surfaces. Some objects like tree or flower on the other hand, require millions of polygons to create a very approximate representation. In order to reconstruct a complex scene, we can make use of voxel representations. Even when texture mapping is applied, polygonal representations might not provide realistic representations of real world objects. Scientific data like topological data and medical scans can be understood more easily when it is visualized with voxels. Furthermore, voxels can be given an opacity value in addition to their appearance (e.g. color). So by manipulating the opacity of designated voxels, we can look into an object, layer by layer.

Volume graphics uses a 3D volume buffer for representations and manipulation of 3D scenes instead of geometric objects. Rendering process of voxels is affected by the resolution of the voxel cube not by the scene and object complexity since it has been converted into a constant volume size previously. Therefore complex objects with curved surfaces which require large polygon meshes can be rendered easier with voxel methods.

Voxel methods are also viewpoint independent in contrast to polygonal methods. This is because all objects have been converted into voxels and for each voxel a unit memory is allocated where in polygonal methods the memory is assigned to complete surface patches.

Voxels are closely related to pixels due to their discrete form and many of the problems in raster graphics also exist in voxel based methods. Therefore, many of the problems that have been solved in 2D graphics like anti aliasing and block operations can be adapted to voxels methods easily. For example block operations as quadtree in raster graphics is applied to volume graphics as octree. Filter operations like blurring, sharpening or gradient operations (e.g. Sobel filter) can be enhanced to the third dimension and be directly applied to the voxel space.

2.1.3.2 Disadvantages of Voxels

Due to huge memory and capacity requirements, voxel methods are going to be applied more with the advances in computer hardware. Volumetric representations use voxels to model the scene, which consume large amounts of memory. For example 256^3 bytes (16.77 mbytes) of memory is required for a rather small cube (256 units in each direction). When linear resolution is increased, the memory requirements increase in proportion to the cube. For example when 2D image with dimension n has n^2 pixels, 3D cube with dimension n has n^3 voxels. However, the rapid advances in hardware provide faster, larger and cheaper memories and in the near future this advantage will become less of a problem and volumetric representations become more attractive. On the other hand volume engines which will load, store, manipulate and render volumetric scenes in real time are configured as accelerators of existing geometric engines.

There are some disadvantages of voxels due to their discrete form. When low resolution is used it yields high aliasing artifacts. When compared to 2D images, we can not get more information by zooming into the image. At some stage, it will only reveal the pixel structure, or the voxel structure, respectively. When zooming into a polygonal model, some details might be uncovered, that have been invisible before (e.g. two very close parallel lines). We cannot derive more information, but on the other hand, we will not see a pixel structure either.

Although some measurements are easier with voxels, the finite resolution of cube limits the accuracy of the measurements which are dependent on voxel counting like volume or area measurements. When the discrete volume is manipulated or transformed, some information is lost and image quality is decreased since subsequent rotations will distort the image. However, the techniques applied to raster images can be applied to lessen distortion. And also a discrete object does not have geometric information about objects surface description. We could say that one voxel knows nothing about its neighbor. Therefore when exact measurements are required polygonal methods might be the preferable because objects geometric definition is available.

2.2 Volume Data

As mentioned previously, in general, there are two main three dimensional data structure to create and manipulate images; geometry-based models and digital images. In this thesis another type of data structure is used, voxel data, which recently becomes important with the new advances in computer hardware. A voxel is a rectangular cuboid and have 6 faces, 8 corners and 12 edges.

Volume data was first introduced in mid 70's in medicine in order to visualize inner parts of the body and nowadays it is commonly used in research and development. The name voxel stands for volumetric pixel. A voxel is a unit volume and it simply adds another dimension to digital picture element, pixel. If the definition of pixel can be extended to 3D; just as we think of pixels as square unit elements in an image, we think of voxels as cubic elements in 3D voxel space.

Voxels define geometrical, physical and radiometric properties of every point in 3D space and they are stored in a three dimensional matrix. Every voxel has a numeric value which represents some measurable properties or independent variables, for example color, opacity, density, heat of the real phenomenon or object. These variables can also be a vector representing variables like velocity and time.

2.2.1 Voxel Definition

Let volume $V(n \times n \times n)$ is the point samples taken from the real environment in 3D Euclidean space R^3 . V is the subset of natural volumes in R^3 which forms a grid with integer coordinates. The voronoi neighborhood of a grid point P is used to represent its local geometric environment and it is the set of all points that lie nearest to P . The voronoi neighborhood of a 3D grid point is a closed unit cube called as a *voxel*.

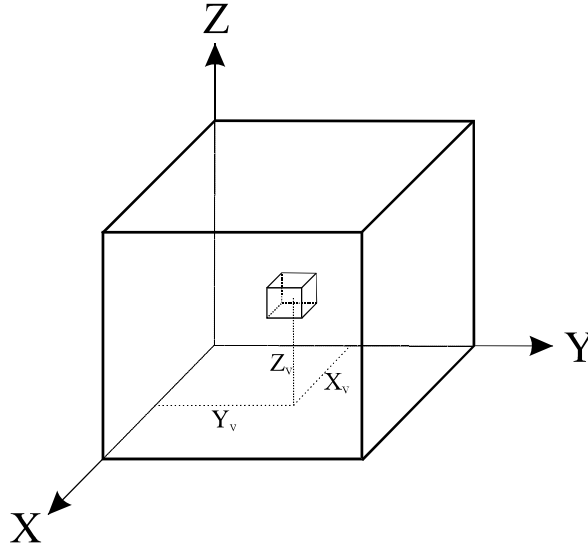


Figure 2-1: A $n \times n \times n$ volume and a voxel with coordinates (X_v, Y_v, Z_v)

In Figure 2-1 voxels form a 3D array “ $n \times n \times n$ ” in real space R^3 and cover the local space defined with coordinates below:

$$X \leq 0 < n_x$$

$$Y \leq 0 < n_y$$

$$Z \leq 0 < n_z$$

2.2.2 Voxel Adjacencies

In digital images each pixel is surrounded by other pixels. We can distinguish those pixels that are directly connected with an edge, from those that are only indirectly connected by a vertex, from the center pixel. We can define these as 4-neighborhood and the 8-neighborhood.

Voxels extend this definition to the third dimension. For every voxel there are 26 adjacent voxels. Hence, 6 voxels share a face, 12 voxels share an edge and 8 voxels share only a vertex with the center voxel. We might call these neighbors of first, second and third order. As seen in Figure 2-2, face sharing voxels are called 6-adjacent, face and edge sharing voxels are called 18-adjacent and face, edge and vertex sharing voxels are called 26-adjacent. This becomes important, when we define the surface voxel list (see Chapter 2.2.3) where its density will depend on which neighborhood we choose.

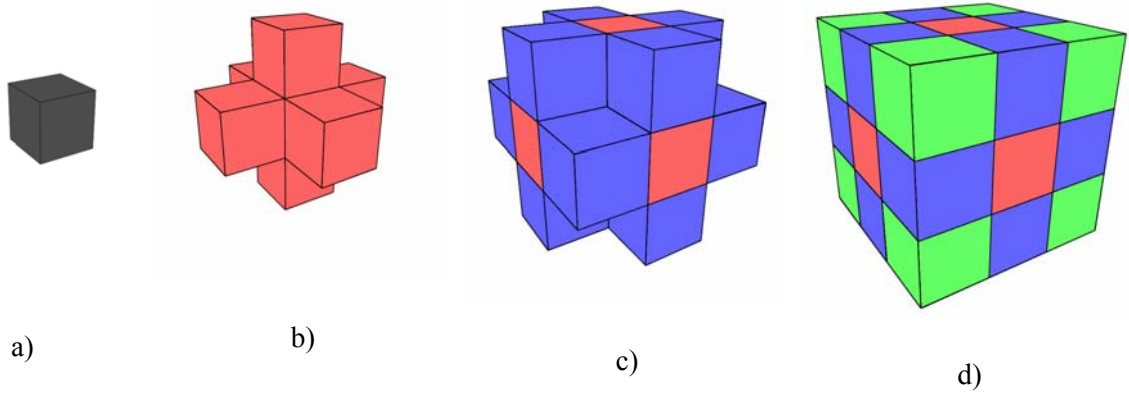


Figure 2-2: the 6- (b), the 18- (c) and the 26-adjacencies (d) of the voxel (a)

2.2.3 Surface Voxel List

The surface voxel list is designed to contain all voxels which lie on the object's surface. While the storage details are a matter of programming and may vary greatly, the definition of this list is quite clear. In general, we should say that a surface voxel is the voxel which lies on the border of two voxel classes. In our case we are dealing with non-transparent objects. Hence, the properties of inner voxels are of no interest. Therefore, we have only two classes: object and non-object. In other cases, like medicine, the inner voxels are of great importance and we might encounter classes like skin, bone, liver, etc. (see Figure 2-8 for Visible Human Project [TVHM]). Here, we could define surface voxels for each class, i.e. those voxels, which for example separate a liver from the rest. For simplicity and because in this work we are mostly dealing with solid voxel models, we will be limited to the classes object and background.

Before we can construct the surface voxel list, we have to define what makes an arbitrary voxel a surface voxel. There are mainly two conditions:

1. The voxel itself is an object voxel.
2. At least one neighbor is not an object voxel.

These conditions are sufficient to distinguish inner voxels from surface voxels. But if we go into detail, we will discover that this might not be an adequate definition. First of all, we would have to specify what kind of neighborhood we want to allow (see section 2.2.1). Depending on the neighborhood order fewer or more voxels are considered to be surface voxels and the whole set becomes more or less dense.

Secondly, with the above definition any isolated voxel or noise voxel will be regarded as surface voxel. If noise is likely to be the case but not desired, the second condition should be restated as:

2. At least one neighbor is not an object voxel, and at least one other neighbor is again an object voxel.

By varying the number of neighbors which also should be object voxels, we can influence the tolerance against noise.

Nevertheless, the set of all surface voxels forms the surface voxel list (SVL). We will be limited to one SVL for the whole voxel cube regardless of the number of separable objects inside the cube. The simplest solution constructs a dynamic linked list where each item contains the voxels coordinates.

The voxels property is implicitly defined by the list itself. In the simplest case they are all surface voxels. In the case of more complex classes, we may have several surface voxel lists, e.g. the liver surface, lung surface, etc., where each list defines the property of its very voxels.

Creating the SVL might speed up some operations tremendously. When applying texture mapping, for instance, it is obvious that inner voxels are completely uninteresting in the case of non-transparent objects. Therefore, only surface voxels have to be considered. Compared to the total amount of object voxels, the SVL contains only a small percentage.

2.2.4 Vector Line

For many tasks it is necessary to trace voxels along a line. In Chapter 5.1.1 some examples are given where it becomes useful and how it can be accomplished. In this chapter, some definitions and properties concerning voxel lines are explained.

As with the surface voxel list, which actually forms a voxel surface, the voxel line is dependent on the degree of neighborhood. The number of voxels varies with the neighborhood. Since by definition all voxels in a line are connected to each other, we can influence the line by defining the connection. Figure 2-3 shows a line with 18-connected voxels and 6-connected voxels.

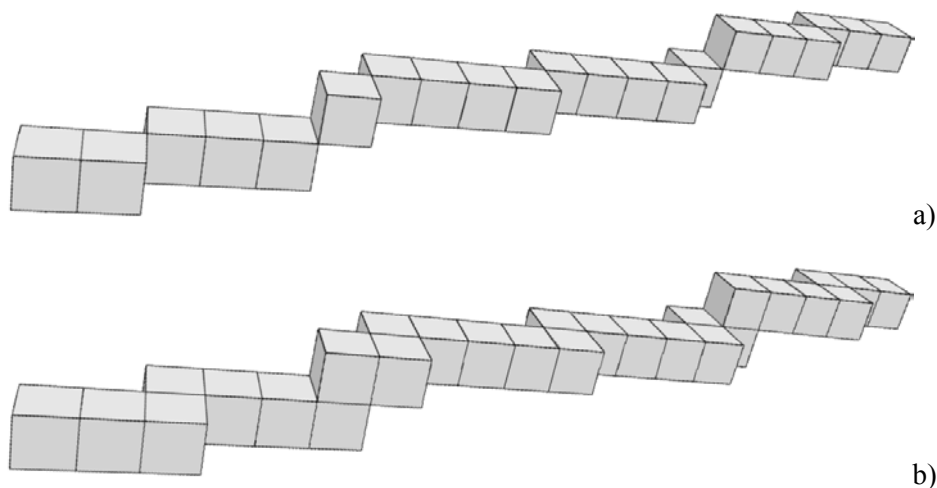


Figure 2-3: a) 18- and b) 6-connected voxel line

In the case of the 6-connected line, only fewer voxels are considered connected. So the line has to make detours to trace from start to end and hence becomes thicker. In the 18-connected case, the line consists of fewer voxels, but the mathematical line may traverse a voxel which is not part of the voxel line. Furthermore, the thin line has the disadvantage of being able to intersect an 18-connected surface without detection (see Figure 2-4). Algorithms and applications of line tracing will be explained in detail in chapter 5.1.1.

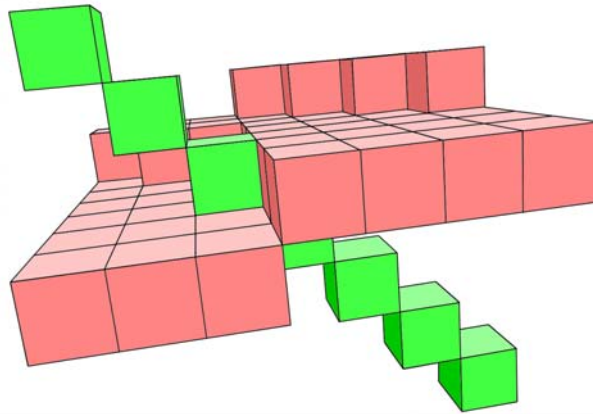


Figure 2-4: Line intersecting a plane without detection

2.2.5 Surface Normal Vector

In some occasions, knowing where a surface is heading towards might be useful information. This is required information, when it comes to photo realistic lighting simulation and shading. But we might use this information to evaluate a usefulness of some selected operations. For example, if we perform image matching, we would like to decide, whether a certain point is a sensible candidate. One criterion is its radiometric properties, described by an interest-operator. Another criterion would be where this part of the surface is heading. If it looks away from the camera, we can expect a highly distorted pattern, but when it looks in the direction of the camera it might yield a good result.

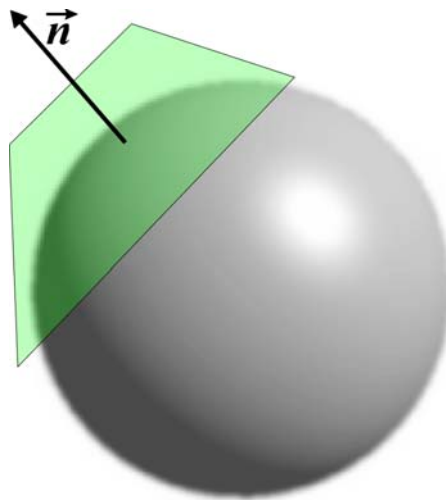


Figure 2-5: Tangent plane and surface normal vector

The derivation for the tangential plane and therefore the surface normal vector is trivial for polygonal models. It is also more or less simple for mathematically described surfaces. But it is a little bit more complicated for voxel models. Here we are dealing with a number of regularly arranged coordinates. A detailed solution for this problem will be introduced in Chapter 5.

2.2.6 Volumetric Data Set Definition

A volume data set in other words, volumetric data is the set of voxels represented as a 3D discrete volume. A volume data set contains sample values associated with the points of a regular grid in 3D

space. Therefore, volume data sets can be treated as 3D images with optical properties, like color and opacity. For example for 8 bit images, voxel gray values are derived from the corresponding pixel gray values and each voxel V has a value ranging from 0 to 255. The color value can be written as the function of the point coordinates as; $V = g(x, y, z)$. If the volumetric data set contains binary data, the black voxels with the value “0” represent transparent regions and the white voxels with the value “1” represent the object.

Figure 2-6 is an example of a volume data set in a technical chemistry experiment. It shows an example of a flow tomography sequence in a turbulent mixing process [MAAS, 1994].

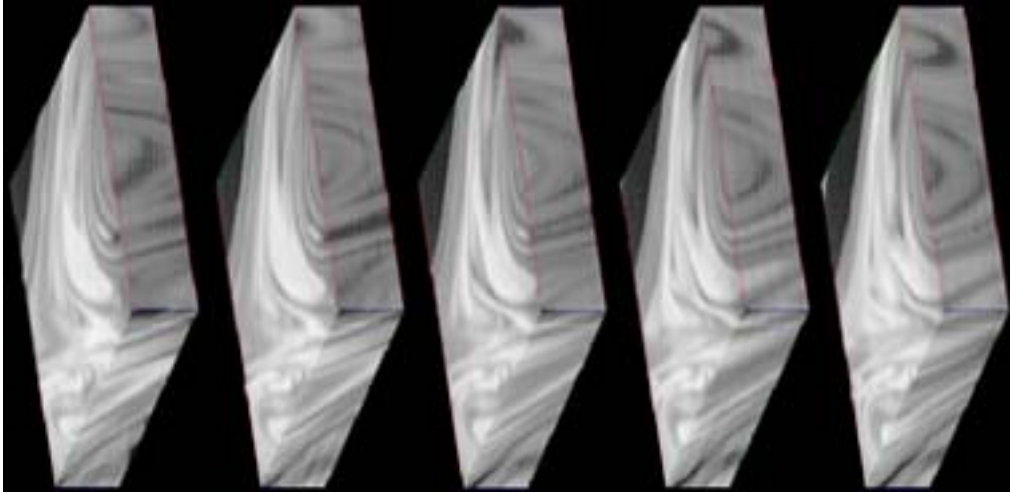


Figure 2-6: Volume datasets: 5 consecutive volumina with 50 layers of 256 by 256 pixels each (total 16 MB = 0.5 seconds of flow data) [MAAS, 1994]

2.2.6.1 Sources of Volume Data

Since everything in the real world has three spatial dimensions, any application that produces image data can be the source of volume data set. This thesis specifically focuses on image sequences acquired by a video camera. In general, source of volume data are sampled data of real world objects and phenomenon, modeled data generated from a geometric model and simulated data acquired by computer simulation [KAUFMAN-I, 1993], [KAUFMAN-II, 1996].

2.2.6.1.1 Sampled Data

In medicine, when we want to study an organ, CT (computed tomography) exposures are taken through several parallel slices. These 2D slices can then be stacked into a volume data set and reconstructed into a volume model in order to study 3D structure of an organ. In a similar way, other methods to obtain volume datasets are magnetic resonance imaging (MRI), ultrasound imaging, the positron emission tomography (PET) and single-photon emission computed tomography (SPECT) methods of nuclear medicine all can similarly produce volume data sets. The “Visible Human Project” can be given as a good example of sampled data [TVHP]. This project was founded by the National Library of Medicine, USA in 1991 for developing a digital database of volumetric data representing a complete adult male and female for teaching and research purposes in applications as virtual surgery. The goal of the project was to provide radiological and photographic representation of a complete, human male cadaver at a resolution of 1mm in all three dimensions. Imaging of the cadaver was performed using the magnetic resonance imaging (MRI) and computed tomography (CT) equipments. The photographs were obtained by sectioning the body at 1mm increments and capturing a 2048 x

1216 digital image in 24-bit color at every level. These images have been registered and therefore, can be restacked to define the human body at every location in space with 1mm voxels. The voxels have attributes of RGB color in addition to the electron density provided by the CT images.

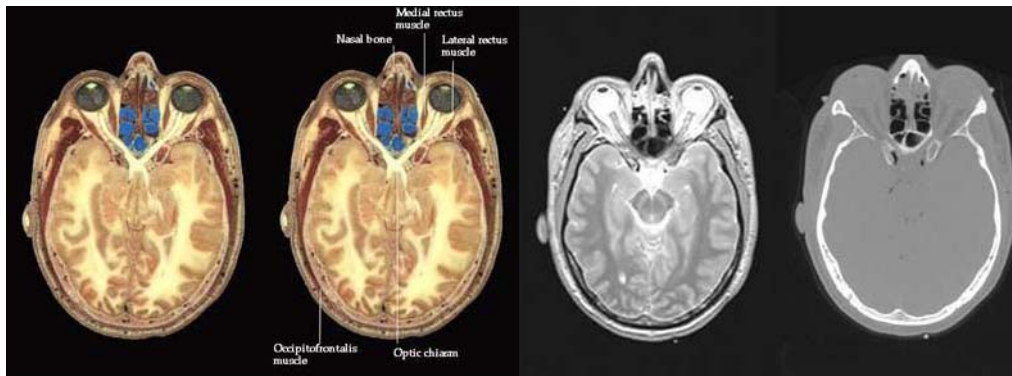


Figure 2-7: Unlabeled, labeled, MRI and CT images respectively “Visible Human Project” [TVHP]

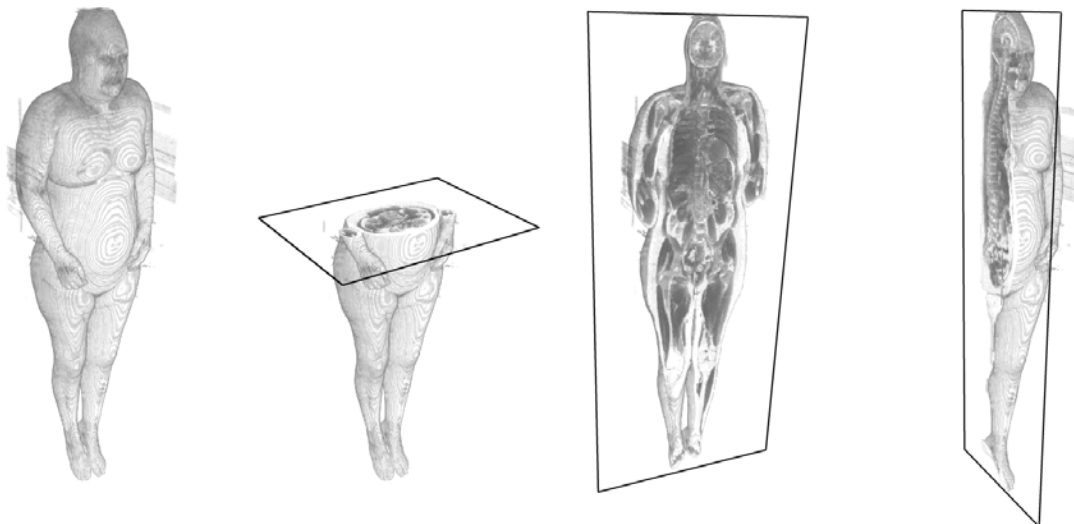


Figure 2-8: 3D Visualization of the female cadaver, from the “Visible Human Project”, extracted from a low-resolution animated gif [TVHP]



Figure 2-9: Virtual surgery “Visible Human Project” [TVHP]

Seismic exploration is another application area which results in sampled volume data sets for exploration of petroleum and mineral resources. The basic technique of seismic exploration is to generate artificial elastic waves by means of an acoustic energy source and to detect the energy reflected from geological interfaces within the earth. The receivers are generally located on the earth forming 2D grids. And each receiver gets the data in terms of time series. The amplitude of the reflected pulse carries information about the rock layers and the arrival time is directly related to the depth of these layers. Seismic data is a large volume set with voxels representing the acoustic amplitude and the time dimension is the depth of the layer.

2.2.6.1.2 Modeled (Generated) Data

When the shape and dimension of an object is known, it can easily be generated with voxels. This is especially the case, when it is a regular object, like a sphere, a box or a cylinder. Figure 2-10 shows voxel models of consumer products, as they could be placed in a virtual shop. If the visualization software allows, they can be zoomed, rotated and investigated. In the following case, the objects are very much simplified, but the natural texture, plus any possible software shading, gives them a very realistic appearance.



Figure 2-10: Generated objects, based on the known geometry and texture

Although volumetric representations can be seen more advantageous for sampled data, because they represent the digital samples and the interior parts of the objects, it is also commonly used in applications that deal with modeling and rendering of the synthetic scenes represented by geometric models. Surface representations are converted to volume data using a method called voxelization. Some examples are rendering of fur gases and CAD models and terrain models for flight simulators. Photorealistic volume based modeling of real world terrains are created from a single elevation map and a satellite or aerial photograph of the terrain. This is achieved using voxelization, 3D texture splatting and ray casting. However these systems can only handle 2.5D voxels which causes 3D aliasing, when the viewpoint is changed or when viewing a region of sharp boundaries.

2.2.6.1.3 Simulated Data

The simulations of complex physical phenomena present new challenges for computational scientists including analyzing and visualizing complex three dimensional data. In many computational fields like computational fluid dynamics, volume data sets arise by running such a simulation on a supercomputer. Three dimensional fields are quantities associated with the points in defined region of 3D space and these quantities differ in time from point to point. The simulations sample 3D fields numerically on finite sets of sample points. These points are arranged in regular grids or can be mapped to regular grids therefore can be visualized as volume data. Other examples of computational simulations can be given as storm prediction in meteorology, oil reservoir simulation, stress and thermal studies in engineering as well as computational chemistry and quantum physics. In computational fields applications there is no limit to the size of the volume because the study is continuous. As in a 3D simulation, millions of sample points might be used, the resolution is dependent on the power of the machine.

2.2.7 Storage of Voxels

Memory requirements are the biggest disadvantage of voxels this is because the scene is divided into cubes and even the empty voxels need to be represented in the memory. When we increase the resolution, the memory requirements also increases by power of 3. The amount of memory to store voxelized scene is large but we can keep it to a minimum by a number of techniques.

Some storage techniques include volume buffer, octrees, binary space-partitioning trees and run-length-encoding (RLE). In this thesis run-length encoding and octrees are referred as compression methods. Some methods which are the variations of octrees i.e. linear octrees, PM Octrees are not mentioned.

2.2.7.1 Volume Buffer

The simplest way of storing volumetric data set is the volume buffer. This is the large 3D array of voxels where each voxel represent either a value or a blank space. This method requires the most memory because it does not provide data compression. For smaller objects it might be the preferable method.

2.2.7.2 Binary Space Partitioning Trees

Binary space partitioning trees recursively divide the space by a plane into two subspaces. In contrast to octrees subdivision is dimension independent because separating plane has arbitrary orientation and position. Each node has two pointers on each side of the plane. The elements are classified according

to which subspace they belong to. Assuming the normal points out of the object, the left subspace is defined as “in” the plane and the right subspace is defined as “out” the plane. Each node also has a thickness and any point lying within this distance is defined as “on” the plane. If a side of the plane is homogeneous, it has no child; otherwise it is further divided just as in octree method. Although BSP has the advantage of dimension independence, it is computationally more expensive.

2.2.7.3 Run-Length-Encoding (RLE)

A quite old and simple, yet efficient method of data compression may be achieved by run-length-encoding. Instead of writing an exact copy of the data into the file, the algorithm takes into account, that data might be homogeneous and repetitious. The simplest way to store data is to let each sample be represented by one byte and write it. Run-length-encoding would count the number of repetitive data and store the value and the number of occurrences. The more homogeneous the data is, the more efficient the algorithm will work. In the case of voxel cubes, we will encounter many empty voxels, as well as many occluded inside voxels, meaning; there are wide areas with homogeneous data.

An advantage is, that this algorithm can be very powerful, but is easy to implement. Furthermore it is a lossless compression technique, i.e. the data will be restored without information being lost.

The above will be valid for data stored with 8 bit per sample or less. If the voxels are stored in true color (24 bit) run-length-encoding is unlikely to be helpful. Instead, lossy compression techniques would be the choice, i.e. unnecessary information is deliberately suppressed, as the JPEG-compression does, in the case of 2D-images.

2.2.7.4 Octrees

Many researchers have dealt with octrees since it is discovered in late 70's [POTMESIL, 1987], [SRIVASTAVA, 1990], [SZELISKI-I, 1990]. The octree structure is a hierarchical representation of a $2^n \times 2^n \times 2^n$ voxel array. Octrees are the most common method to store voxels because of good data compression and ease of implementation. A typical voxel raster contains huge homogenous regions. It is possible to reduce the memory requirements if these homogenous regions are not represented. Octree method derives from quadrees which is a well known method to store 2D raster image. Quadrees work successively dividing the area of the image into equal quadrants. The division is made in three dimensions in octrees and the space is divided into 8 equal octants at each division. Each octant may be full, partially full or empty. Octants are referred by numbers beginning from 0 to 7. When eight sibling octants are homogenous, either full or empty, they are deleted and partially full parent is replaced with full or empty node. This process continues until a proposed tree depth is reached or all octants are homogenous. A maximum tree depth is set by user to limit the memory requirements and to define the resolution of the voxel cube. Figure 2-11 shows the subdivision of a $2^3 \times 2^3 \times 2^3$ voxel array with resolution $8 \times 8 \times 8$. Most of volume data sets contain large areas of empty space. In this case, it can be compressed and stored in an octree because it is very economic and provides high resolution and low memory consumption.

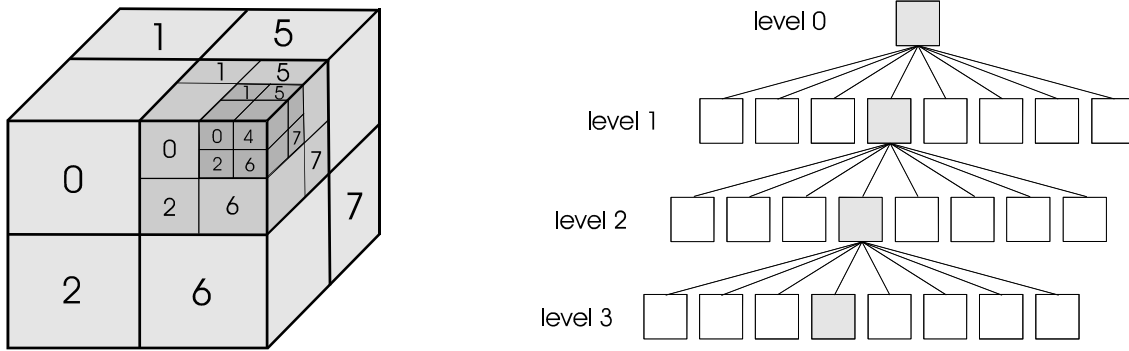


Figure 2-11: Numbering of octants in an octree and subdivision of the voxel cube

2.3 Morphological Operations in 3D

Mathematical morphology is a method to extract image components which are useful for representation and description. Mathematical morphology is proposed by Matheron and Serra in the sixties and has been used in image and signal analysis to provide quantitative description of geometrical structures [MATHERON, 1975], [SERRA, 1982]. In general morphological operators are based on expanding and shrinking operations. Morphology can be used to extract object boundaries, skeletons or convex hulls as well as they can be used in post processing techniques like edge thinning.

Primary usage of morphology occurs in binary and gray level images. By use of structuring elements (SE) many deformation effects can be generated by adding or removing pixels according to the pattern of neighboring pixels. There are two basic types of mathematical morphology operations which are dilation and erosion. Erosion, dilation and their combined usage add or remove pixels from the boundaries of features, join separated portions of features or remove isolated noise from the images.

As volumetric objects are presented with a set of binary voxels, mathematical morphology can be easily adapted to 3D. The main goal is to analyze volume for noise elimination.

3D dilation of the object A, by 3D binary structuring element SE, can be provided as follows:

$$dilation(A, SE) = \bigcup_{(i_n, j_n, k_n) \in SE} trans(A; i_n, j_n, k_n) \quad (2.1)$$

where: trans is the translation operator which is defined as follows;

$$trans(A; i_n, j_n, k_n) = \{(l_t, m_t, n_t) | l_t = l + i, m_t = m + j, n_t = n + k, (l, m, n) \in A\} \quad (2.2)$$

The erosion operator is defined as the intersection of each voxel of A which is translated with a structuring element SE in the negative direction.

$$erosion(A, SE) = \bigcap_{(i_n, j_n, k_n) \in SE} trans(A; -i_n, -j_n, -k_n) \quad (2.3)$$

As it is in 2D an opening is an erosion followed by a dilation. An opening provides smoothing of the surface and removing irrelative floating volumes.

$$opening(A, SE) = dilation[erosion(A, SE), SE] \quad (2.4)$$

A closing is a dilation followed by an erosion. A closing provides filling the holes in the surface and combining neighbor voxels.

$$closing(A, SE) = erosion[dilation(A, SE), SE] \quad (2.5)$$

Chapter 3 : Camera Calibration and Image Orientation

3.1 Coordinate Systems

In order to reconstruct the shape from images we should know the relation between image and world coordinate systems. There is a similar relation between digital and photogrammetric coordinate systems for images and for voxel cubes. The digital coordinate system is closely related to the memory structure of the computer (left-handed and the origin is in the corner), while the photogrammetric coordinate system is right-handed and the origin is mostly found in the center. The following chapters will introduce the coordinate systems of the digital and photogrammetric image coordinates and the world and voxel coordinates. The relation between these systems will be mathematically explained.

3.1.1 Image and World Coordinate System

It is well known that in order to reconstruct the geometric situation of the image acquisition, we need to have knowledge about the cameras interior and exterior orientation. In other disciplines, we may encounter terms like intrinsic and extrinsic parameters. Regardless of the title, they describe the general properties of the camera (interior) and its position at the time of the image acquisition (exterior). In simple words, we have to know what kind of camera we were using and where it was. This is expressed by the cameras internal parameters, the calibrated focal length, the position of the principal point and eventually advanced parameters to describe lens distortion. While these properties are more or less constant, the external parameters are highly variable. Any picture taken with a camera however stores these parameters at the precise moment of acquisition. They consist of the position (projection center) and the orientation (rotation) of the camera. So regardless of lens correction terms, the geometric situation is described by nine parameters. It is described later, how these parameters can be derived.

The relation between image coordinates (x, y) of an image point p and the coordinates X, Y, Z of an object point P is shown in Figure 3-1.

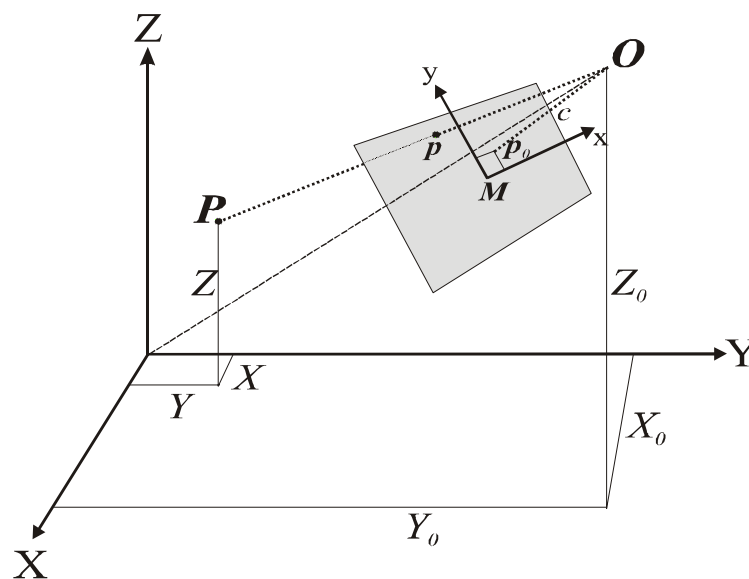


Figure 3-1: Relation between image and object coordinates under central projection

Where:

O : Projection center with the coordinates X_0, Y_0, Z_0 .

p_0 : Principal point with the coordinates x_0, y_0 .

M : Image center, or fiducial center respectively

P : Real world point with the coordinates X, Y, Z

p : Image point with the coordinates x, y .

The transformation from world coordinates into image coordinates is defined by the collinearity equations:

$$x = x_0 - c \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) - r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) - r_{33}(Z - Z_0)} \quad (3.1)$$

$$y = y_0 - c \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) - r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) - r_{33}(Z - Z_0)} \quad (3.2)$$

Where, parameters r_{ik} are the elements of the rotation matrix \mathbf{R} , with three rotations α, ν, κ . Since we are assuming the terrestrial case, we can use Krauss equations for terrestrial photogrammetry in construction of the rotation matrix [KRAUS, 1997].

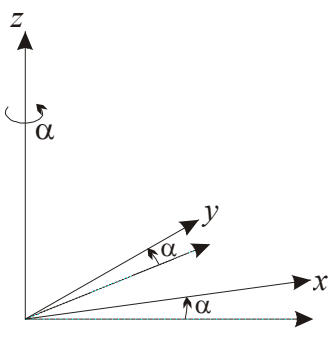
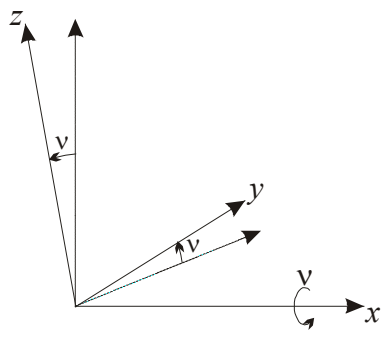
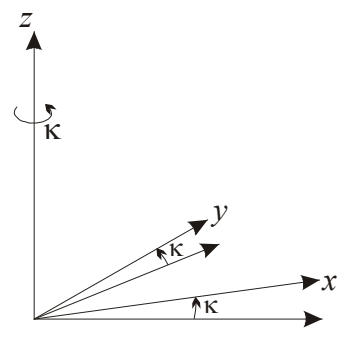
1st rotation α along Z axis:	2nd rotation ν along X axis:	3rd rotation κ along Z axis:
		
$R_\alpha = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$R_\nu = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \nu & -\sin \nu \\ 0 & \sin \nu & \cos \nu \end{pmatrix}$	$R_\kappa = \begin{pmatrix} \cos \kappa & -\sin \kappa & 0 \\ \sin \kappa & \cos \kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Figure 3-2: Three sequential rotations

Multiplication of the three matrices gives the complete rotation matrix:

$$R_{\alpha\nu\kappa} = \begin{pmatrix} \cos \alpha \cdot \cos \kappa - \sin \alpha \cdot \cos \nu \cdot \sin \kappa & -\cos \alpha \cdot \sin \kappa - \sin \alpha \cdot \cos \nu \cdot \cos \kappa & \sin \alpha \cdot \sin \nu \\ \sin \alpha \cdot \cos \kappa + \cos \alpha \cdot \cos \nu \cdot \sin \kappa & -\sin \alpha \cdot \sin \kappa + \cos \alpha \cdot \cos \nu \cdot \cos \kappa & -\cos \alpha \cdot \sin \nu \\ \sin \nu \cdot \sin \kappa & \sin \nu \cdot \cos \kappa & \cos \nu \end{pmatrix} \quad (3.3)$$

Note that a change in the order of rotation will result in a different rotation matrix. Depending on the task at hand the one or the other way is more efficient.

Every object point projects into exactly one image point. But the converse transformation is not possible because for every image point there are infinite object points, along the projection line. Therefore it is impossible to reconstruct an object from a single image. By using a second image, we are able to derive an object point in space. Otherwise, we need extra information about the Z coordinates of the object, e.g. all object points lie in a horizontal plane of known height.

Every object point along a line of sight may be projected into the same pixel. Therefore, according to the rules of central projection, each pixel has its individual scale factor λ , depending on the point's distance from the projection centre. This scaling factor, however, is normally unknown, as well as not required.

Nevertheless, if we happen to know this factor, or if we just define it, we are able to refer to a specific point in space. Depending on the scale factor λ , the transformation equations from image coordinates into world coordinates are defined as follows:

$$\lambda \begin{pmatrix} x - x_0 \\ y - y_0 \\ -c \end{pmatrix} = R \cdot \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R^{-1} \cdot \lambda \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \\ -c \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \quad (3.4)$$

from the equations above we can derive the formulas:

$$\begin{aligned} X &= \lambda r_{11}(x - x_0) + \lambda r_{12}(y - y_0) - \lambda r_{13}c + X_0 \\ Y &= \lambda r_{21}(x - x_0) + \lambda r_{22}(y - y_0) - \lambda r_{23}c + Y_0 \\ Z &= \lambda r_{31}(x - x_0) + \lambda r_{32}(y - y_0) - \lambda r_{33}c + Z_0 \end{aligned} \quad (3.5)$$

where:

c : calibrated focal length

X, Y, Z : object point coordinates

X_0, Y_0, Z_0 : projection center

x, y : image point

x_0, y_0 : principal point

r_{ik} : elements of the rotation matrix

λ : scaling factor

3.1.2 Voxel and World Coordinate System

Voxel coordinates differ from world coordinates in a way that they only exist as integer numbers in a predefined three dimensional array of samples.

$$\begin{aligned}
g = g(x, y, z) \quad & 0 \leq x < N_x \\
& 0 \leq y < N_y \\
& 0 \leq z < N_z
\end{aligned}$$

where: N_i denotes the number of voxels along the three major axes. There can be no information outside these boundaries. As in digital images, the axes form a left-handed system. The coordinates refer to the voxels center. Information “between” voxels is not available and could only be guessed by suitable interpolation algorithms.

However, we want these voxels to represent very specific points in space, so we will have to supply means of transformation from voxel coordinates into world coordinates, and vice versa. This transformation will have to include an individual shift along the three major axes, and a scaling factor, which represents the size of a voxel. If for example, we choose a voxel size of 2mm, we simply have to multiply this value with the number of voxels along the major axes, let's say 256x256x128, to learn about the cubes dimension in the world coordinate system. It would correspond to an area of 512x512x256mm in world coordinates.

Furthermore, we would like to strictly limit the voxel cube to the area of interest in object space, we can think of it as a bounding box, so it might become handy to be able to rotate the cube about the z-axis. The transformation routine will be designed accordingly. We will end up with a similarity transformation for x and y, and a simple scale and shift in z, see equation below. The idea is depicted in Figure 3-3. This transformation could easily be extended to a full three dimensional rotation, by simply using a full rotation matrix (see equation 3.3), but it seems a sufficiently approximated area of interest, when considering one rotation only. After all, a full rotation matrix would also slow down the computers performance.

Prior to the transformation into world coordinates, we will shift the X- and Y-coordinate to the center of the cube. We will furthermore swap the X- and Y-axis into a right-handed system,

so that:

$$\begin{pmatrix} X'_V \\ Y'_V \\ Z'_V \end{pmatrix} = \begin{pmatrix} Y_V - N_y/2 \\ X_V - N_x/2 \\ Z_V \end{pmatrix} \quad (3.6)$$

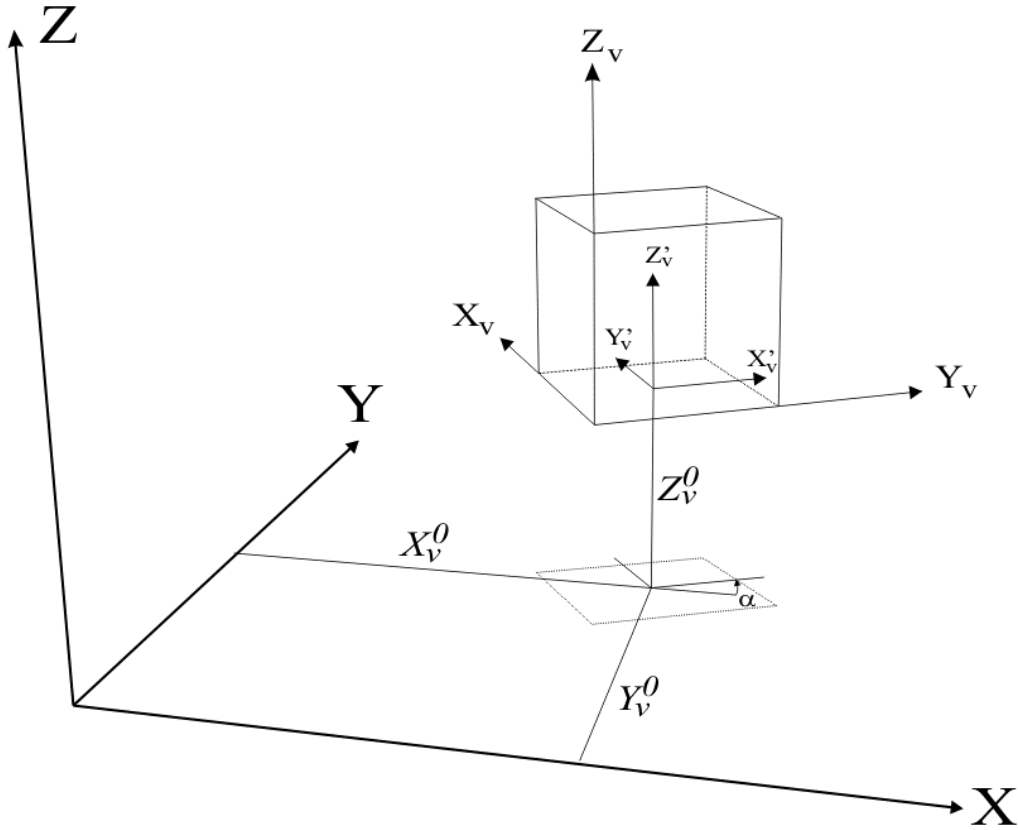


Figure 3-3: Relation between world and voxel coordinates

The transformation from voxel coordinates to world coordinates can finally be given as:

$$\begin{aligned}
 X &= s_v \cdot \cos \alpha \cdot X'_v - s_v \cdot \sin \alpha \cdot Y'_v + X_v^0 \\
 Y &= s_v \cdot \sin \alpha \cdot X'_v + s_v \cdot \cos \alpha \cdot Y'_v + Y_v^0 \\
 Z &= s_v \cdot Z'_v + Z_v^0
 \end{aligned} \tag{3.7}$$

or,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = s_v \cdot \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X'_v \\ Y'_v \\ Z'_v \end{pmatrix} + \begin{pmatrix} X_v^0 \\ Y_v^0 \\ Z_v^0 \end{pmatrix} = s_v \cdot \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 & X_v^0 \\ \sin \alpha & \cos \alpha & 0 & Y_v^0 \\ 0 & 0 & 1 & Z_v^0 \end{pmatrix} \cdot \begin{pmatrix} X'_v \\ Y'_v \\ Z'_v \\ 1 \end{pmatrix} \tag{3.8}$$

where:

s_v : metric size of a voxel

X, Y, Z : World coordinates

X'_v, Y'_v, Z'_v : Voxel coordinates (cube-centered and right-handed)

X_v^0, Y_v^0, Z_v^0 : voxel cube origin in world coordinates

α : rotation angle about Z

The inversion describes the transformation from object space to voxel coordinates:

$$\begin{pmatrix} X'_v \\ Y'_v \\ Z'_v \end{pmatrix} = \frac{1}{s_v} \begin{pmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X - X_v^0 \\ Y - Y_v^0 \\ Z - Z_v^0 \end{pmatrix} \quad (3.9)$$

And decentered, left-handed voxel coordinates:

$$\begin{pmatrix} X_v \\ Y_v \\ Z_v \end{pmatrix} = \begin{pmatrix} Y'_v + N_y/2 \\ X'_v + N_x/2 \\ Z'_v \end{pmatrix} \quad (3.10)$$

When we are transforming world coordinates into voxel coordinates, we have to keep in mind that information is only available within the cubes dimension. Hence, the result of the transformation into voxel coordinates should always be boundary checked. Furthermore, true information exists only for integer coordinates, which will be the exceptional result from the above transformation. The simplest way is to round the coordinates to the closest integer number, corresponding to the nearest neighbor algorithm. Other approaches could take the neighboring voxels into consideration and perform an interpolation in between.

3.1.3 Pixel and Image Coordinates

As voxel coordinates differ from world coordinates in a way that they are integer numbers defining 3D array in space, in the same way pixel coordinates differ from image coordinates as they are integer numbers defining a 2D array in a CCD plane. Therefore, it is important to understand photogrammetric and pixel coordinate systems in order to perform transformations.

A digital image is actually a 2D array of pixels containing color values. Locations in the image grid are defined by rows and columns, where the horizontal axis is called x axis, and the vertical axis is called y axis. Each pixel covers the same amount of area in which the points share the same attribute, mostly brightness in one or more channels. Mainly we can distinguish between 8 bit gray, 8 bit color and 24 bit true color images. The first describes an image of 256 possible gray values, with $g = g(x, y)$.

The second uses a look-up-table to provide color information in a limited palette of 256 entries, from a total choice of usually $256^3 = 16.77\text{mio.}$ colors. The table holds arbitrary RGB-triplets, so that $g_{RGB} = LUT(i)$. The color of a pixel can be obtained with $g_{RGB} = LUT(g(x, y))$. In the last case, every pixel stores three byte of information, allowing the image to hold the full information of the 256^3 possible colors, hence $g_{RGB} = g_{RGB}(x, y)$.

The width and the height of the digital image may differ and it is called x- and y-resolution. It is a most sensible assumption that integer image coordinates refer to a pixels center. To be compatible with zero based arrays in our source code, we will assign the origin (0, 0) to the upper-left pixel. This will result in negative coordinates (-0.5, -0.5) for the very upper-left image corner, however this will not cause any problems. Hence, the lower-right pixel of the image will have the coordinate $(cols-1, rows-1)$. Figure 3-4 shows the origin of the pixel coordinate system. Note that the digital coordinates are part of a left-handed system.

The digital image exists purely virtual, unless printed. So its metric size depends on the means of display and will vary on each monitor. However, the elements of the CCD chip, or more generally the pixels of a digital image, represent a very specific size. Multiplying the size with the number of rows and columns, we can calculate the true image size in metric values. Furthermore, the size information is required when we want to transform digital into metric image coordinates. It is a sensible assumption that the pixels are squared. There are exceptions, but we will neglect them and only consider square pixels. Nevertheless, non-squared, rectangular pixels can be considered with additional camera parameters, like affine distortion.

Metric image coordinate systems define spatial positions in image space. Its origin is in the projection center and the principal point on the image plane, so that image points have coordinates $[x, y, -c]$. The principal point however, is not marked and can only refer to another origin that can be reconstructed. In metric cameras, fiducial marks define the fiducial center, which we will just title the images middle point M . In non-metric cameras the image center may serve as a restorable origin. The offset between the fiducial center (or the image center) and the principal point, P_0 , can be derived from the camera calibration process.

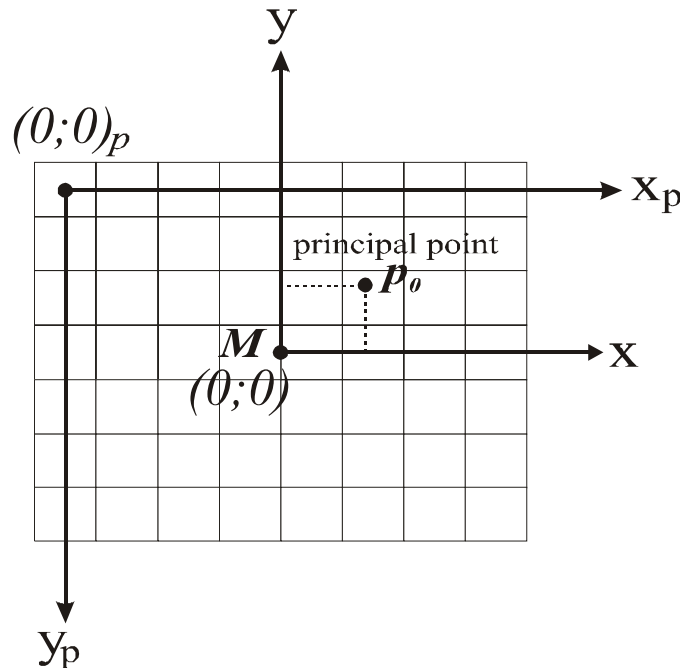


Figure 3-4: Relationship between pixel and image coordinate systems

In order to transform pixel into metric coordinates (vice versa accordingly) there are three steps to consider:

1. Inverting the y-axis (left-hand to right-hand).
2. Translating the upper-left pixel (-center) to the image center.
3. Dividing the pixel coordinate by the pixel size.

In mathematic terms, the transformation of pixel into metric coordinates is described as:

$$x = s_p \cdot \left[(x_p + 0.5) - \frac{n_x}{2} \right] \quad (3.11)$$

$$y = s_p \cdot \left[\frac{n_y}{2} - (y_p + 0.5) \right] \quad (3.12)$$

And the inversion:

$$x_p = \frac{x}{s_p} + \frac{n_x}{2} - 0.5 \quad (3.13)$$

$$y_p = \frac{n_y}{2} - \frac{y}{s_p} - 0.5 \quad (3.14)$$

with:

x_p, y_p : pixel coordinates

x, y : metric image coordinates

s_p : size of a square pixel, e.g. in *mm* or μm

n_x, n_y : dimension of the digital image in pixels

If true metric cameras are available, we can take fiducial marks into consideration, and include a rotation a skew and individual scaling to the transformation, hence an affine transformation. Since we are limited to digital CCD cameras, we will use the simpler solution, as shown above.

3.2 Camera Calibration

The system used in this thesis consists of a simple CCD video-camera with the ability to acquire still images of higher resolution, 1020 by 1360 pixels to be exact. Furthermore, a calibration object is required to compute the interior orientation parameters of the camera.

As a very basic precondition to any subsequent spatial object reconstruction, before starting the image acquisition, the sensor has to be calibrated. The interior and, as a byproduct, the exterior parameters of the camera must be computed, i.e. the position and orientation of the camera and the focal length and principal point. Assuming, that our camera does not change during the image acquisition process, the derivation of the interior orientation can be seen as a first step, while the exterior orientation gives individual results for each image. It cannot be pre-calculated, but will be evaluated in a bundle-block-adjustment. So we will assume one camera, with constant interior orientation for the time of image acquisition.

In our experiment, there is no way of using a calibration certificate for the sensor, since we are using a standard non-metric video camera with auto-focus. Although the focus can be fixed, we cannot assure that it has been unchanged since the last use. So we have to calibrate it anew, using several images with a special calibration object, as shown in Figure 3-5. The calibration object consists of three square perpendicular faces containing 25 control points on each side.

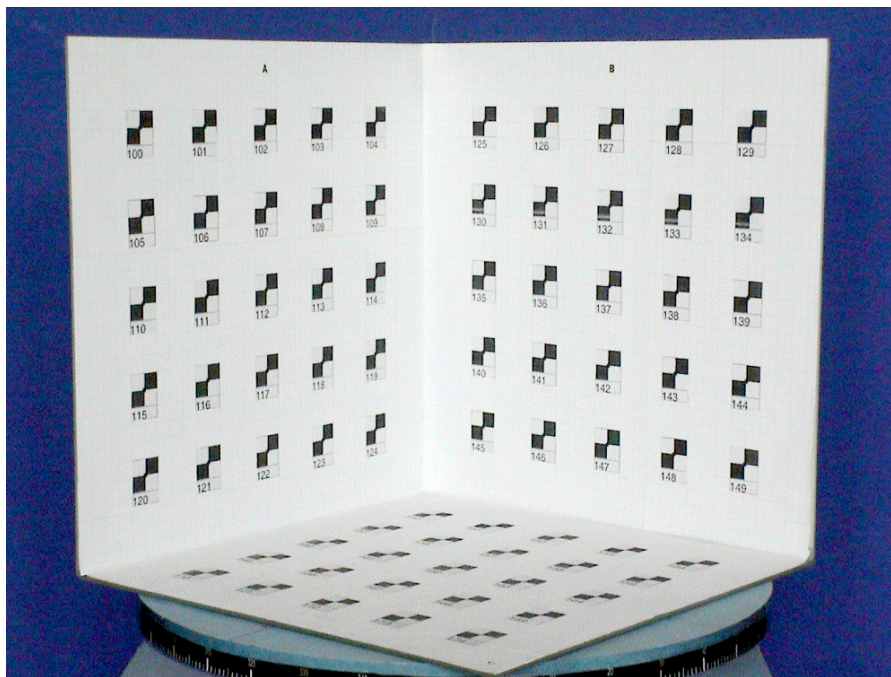


Figure 3-5: Calibration object with 75 spatially well distributed control points

In this example, the camera parameters were calibrated in a bundle block adjustment with self calibration, using five images and 75 control points each. The system had approximately 750 observations and 33 unknowns, 6 for each image and three for the camera. Additional parameters, like lens distortion or affine distortion, were intentionally ignored, since previous calibrations have shown that they are insignificant.

The resulting parameters for the camera were as follows:

Calibrated focal length: $c = 34.133 \pm 0.185 \text{ mm}$

Principal point: $x_p = 0.211 \pm 0.146 \text{ mm}$

$y_p = 0.050 \pm 0.137 \text{ mm}$

As mentioned above, the focus remained fixed throughout the subsequent processes.

There is another challenge to the calibration. It is the unknown metric size of the image, or the CCD-array respectively. In order to obtain correct metric values for the interior orientation, we have to use metric image coordinates, hence the pixel size must be known. Unfortunately it has to be considered unknown, since it is not, or insufficiently documented. Eventually the size of the chip is stated in the manual, but again we might not know, whether this refers to the whole chip, or the active part. Consequently, the pixel size is unknown.

The best we can do is to just define a sensible size. The interior parameters will be calculated according to our definition. They will represent correct values, referring to our defined pixel size, only. The error that we make, estimating the pixel size will be the same in the resulting parameters, so that finally we will end up with internally correct camera parameters. In simple words, an enlargement of the image (by varying the pixel size) causes an equally enlarged focal length, so that: $\frac{c_1}{s_{p1}} = \frac{c_2}{s_{p2}}$

Figure 3-6 shows the relation between different image sizes and their according focal length.

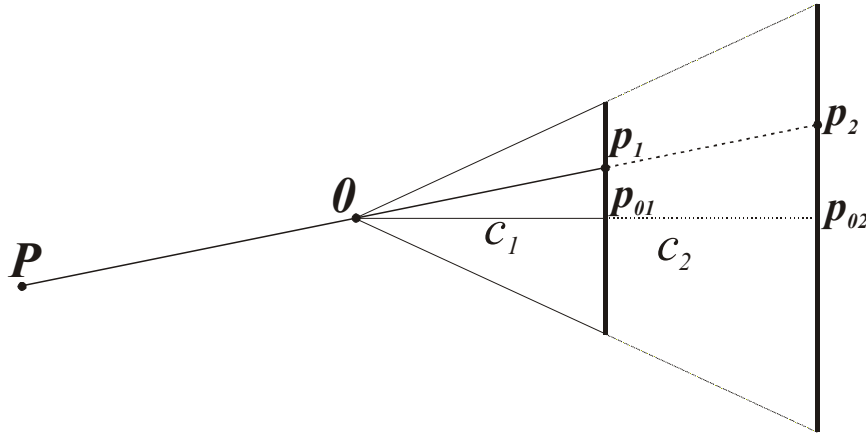


Figure 3-6: Relation of different image sizes and referring focal length

3.3 Image Orientation

The reconstruction of spatial objects requires knowledge of the original situation. The camera can be assumed to be well known due to the previous calibration (see above). We have to learn about the location and the rotation of the images. Reminding Figure 3-6, we have 6 unknown parameters per image:

X_0, Y_0, Z_0 : Projection center in world coordinates

α, ν, κ : three dimensional rotation angles

The derivation of these parameters can be achieved most conveniently with the use of control points. The most common approach is the bundle block adjustment. Every control point here will give three observations, but will cause no unknowns.

Besides the correspondence of control points in world- and image coordinates, we can introduce tie points between images, which have no immediate correspondence in space, but can strengthen the relation of several images. The introduction of tie points gives two observation for each image in which they are observed, plus 3 unknowns for the according new point in space. So a tie point in two images already introduces an additional observation. Any further image in which this tie point is measured introduces two more observations.

For the whole bundle we will have the following conditions:

- 6 unknowns per image
- 2 observations per measured control point
- 3 unknowns per new point (tie point)
- 2 observations per measured tie point

If control points are available, we can perform an absolute orientation, i.e. orientation parameters, that correspond to our world coordinate system. Without control points, we would only be able to achieve a relative orientation. All newly produced object points, would correspond to a local coordinate system, which, for example, we can define, by setting one images projection center to (0, 0, 0). This system would be of undefined scale, which we have to define, e.g. by a known distance in object space.

However, the bundle adjustment is a non linear problem. It follows, that we require approximation values for the unknowns. In an iterative process, we will try to correct the unknowns step by step. The delivery of approximation values for the unknowns, in this case for the image orientations, can be a problem, but it certainly is a limitation towards a more automatic approach.

During our work, we had a circular camera setup, by rotating the object on a turntable, so the derivation of estimated orientation was quite simple. Nevertheless, it mostly is a manual process that can be quite time consuming.

Another way to achieve approximation can be done with the direct linear transformation (DLT). This approach requires no less than six control points per image, however. We will not go into detail here, additional information can be learned from a great variety of literature. It will be shown later, that the DLT does not solve all our problems, last not least because we would like to use as few control points as possible.

Prior to the image orientation, we had to apply some control points to the object itself. In this experiment, it was out of question to mark points artificially, so we had to choose ‘natural’ textures, instead.

We used the coordinates of the calibration object to define a local coordinate system, in which we derived the control points on the object. This is an arbitrary system, without relation to a geodetic reference system. Figure 3-7 illustrates the control points on the object, which served as reference for the subsequent image orientation.

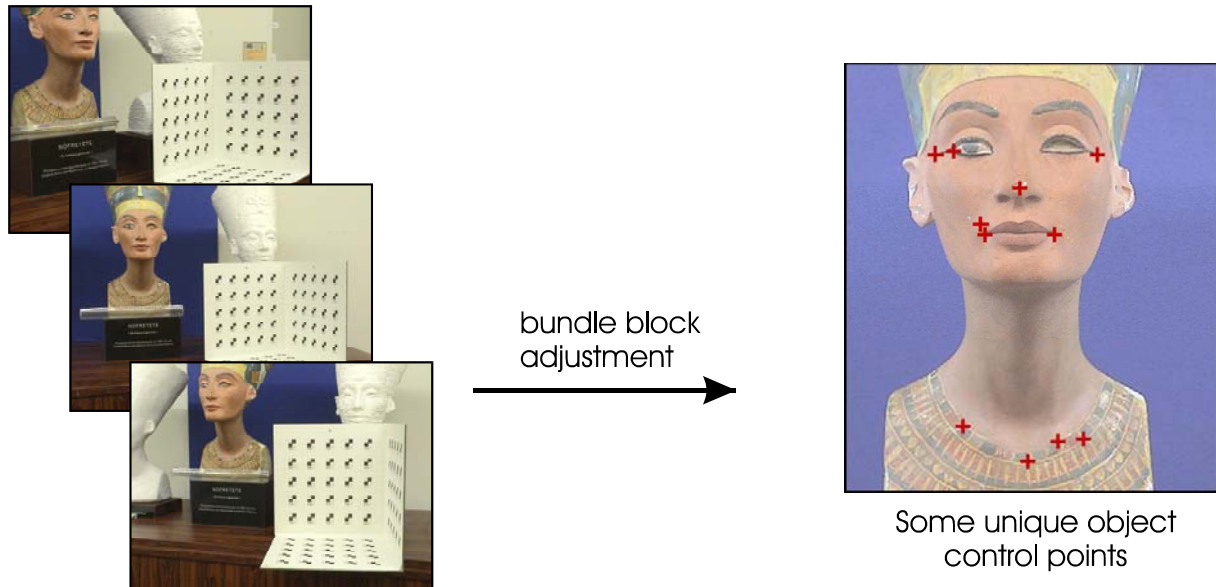


Figure 3-7: Determination of object control points by bundle block adjustment

For an accurate object reconstruction the exact image orientation must be known. Consequently, the images were adjusted in a bundle block adjustment, in this experiment for example, using 22 images in a circular setup. Figure 3-8 shows a visualization of the setup situation. Tie points are measured in every image which stabilizes the whole system and is a requirement for those images in which the control points are not visible. As depicted in Figure 3-7, we were able to use control points on the front part of the object. Using enough tie points in all the images, it was possible to perform a bundle block adjustment with all the surrounding images. With the previously calibrated camera, we managed to achieve very accurate results. The image projection centers have accuracies of 1-2 *mm*, the rotations were determined with 0.05-0.1 *gon*.

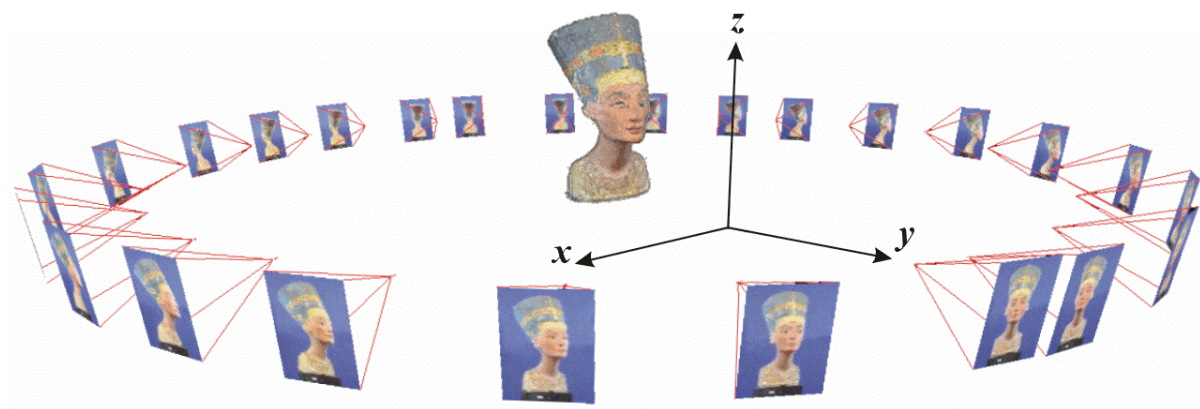


Figure 3-8: The virtual camera setup, using VRML-visualization

Chapter 4 : Volumetric Object Reconstruction Techniques Using Images

This chapter focuses on the present techniques to compute full 3D representations of real world scene from its multiple 2D images. Before the proposed algorithm is introduced in chapter 5, an overview of other possible scene reconstruction algorithms is given here. This chapter also includes the contributions of this thesis to current algorithms. And some reconstruction results we acquired are also presented here. In this chapter only passive reconstruction methods are presented, which means scene is captured using inexpensive cameras. Therefore, these techniques are often referred to image based modeling. Active methods mentioned in chapter 1.2.3.1 that project structured light on the scene or laser range imaging are out of our scope.

The goal of all the methods presented here to reconstruct the outer surfaces of objects. The reconstruction of inner parts that requires special purpose equipments such as in medical imaging is not our interest. All algorithms mentioned in this chapter require full calibration data, so to say, a voxels projection in an image is known.

In many reconstruction algorithms more than one camera is used or the scene is captured from several viewpoints. Because of this, the surface of the object is not occluded on all images. Visibility computation, in other words, occlusion determination is the main problem in volumetric reconstructions. Visibility of a voxel is determined with different algorithms in all methods described here. As a result we know, on which image a specific voxel is visible.

The volumetric approaches in this chapter model the scene in 3D discrete space. This space is defined as a voxel cube so that it encloses the scene to be modeled. Many algorithms start with an opaque volume and true shape of the objects is computed by occupancy classification. Voxels are classified with binary decision either as *opaque* representing the object or *transparent* representing empty regions. In some methods voxels store color information. In some algorithms voxels are classified into three groups as *surface voxels* representing outer shell of the objects, *interior voxels* representing voxels that occluded by surface voxels and *empty voxels* representing empty space. Some methods use octrees to carve away large empty spaces to accelerate the algorithm.

Stereo vision is the traditional process for scene reconstruction. The correspondences are computed and then converted into the object space as 3D points by triangulation. Traditional stereo techniques alone are limited for several reasons which are views must be close to each other and occlusions are hard to compute. When tracking techniques are used in video sequences however, stereo vision is more effective. In contrast to traditional stereo techniques, when voxel based methods are used, occlusions are modeled correctly.

The earliest attempts of volumetric scene representations from photographs are based on volume intersection methods. These methods are often referred to as shape from silhouette algorithms. As a precondition of volume intersection methods images should be segmented in order to distinguish object pixels from background pixels. Although segmentation is a limitation, this is an attractive method because of its effectiveness. The intersection of silhouette cones from multiple images defines estimate geometry of the object called *the visual hull*. This technique is fast and easy to compute and gives a good approximation of the model. However, the concavities on an object cannot be recovered since the viewing region doesn't completely surround the object.

In many applications like automatic 3D model reconstruction, telepresence, virtual walkthroughs there is a need for developing photorealistic 3D models of real environments from their images that look realistically. The recent attempts are based on voxel coloring algorithms where the main goal is to reconstruct the volumes or surfaces that are color consistent with the input images. Voxel Coloring techniques assume the surfaces are Lambertian which means they reflect light equally in all directions. You can also refer to survey papers [DYER, 2001] and [SLABAUGH-III, 2001] for more information about the present volumetric scene reconstruction techniques.

4.1 Volumetric Intersection (Shape from Silhouettes)

One of the well-known approaches for 3-D modeling is shape from silhouette, which recovers the shape of the objects from their contours. A silhouette image is a binary image and easily obtained by image segmentation algorithms. Image pixels indicate if they represent an object point or background point. These algorithms intersect all silhouette cones from multiple images to achieve the estimate geometry of the object which is called the objects visual hull. Shape from silhouettes algorithms used in many applications such as 3D object recognition, 3D object reconstruction and human motion tracking.

Voxel-based visual hull reconstruction is the most popular shape from silhouette method in computer vision and in computer graphics due to its fast computation and robustness. Besides, silhouettes can be obtained very easily especially in indoor environments where a monochromatic background can easily be applied during image acquisition step and where the cameras can be stationary.

When the greater number of views is used, this technique progressively refines the object model. If we could use infinite number of silhouette images, this method would model the object's true shape. However, if only a few views are used, the recovered shape can be very coarse. Shape from silhouette method is sensitive to camera calibration errors and silhouette quality. Noisy silhouettes might cause voxel misclassification. And if we don't know the precise camera calibration and image orientation data the carving of the volume might go too far. Silhouette based constructions have the principal inability to detect object concavities. Therefore, silhouette contours alone cannot recover the object model geometry precisely and should be supported by another technique to get into the critical, concave areas. However, since the result encloses the largest possible volume where the true shape lies and the implementation is straightforward and easy, it is an attractive method for applications where approximate shape is required. It can also be used as the first step of shape reconstruction.

The first work on the construction of volumetric models from multiple views has been done by Martin and Aggarwal [MARTIN, 1983]. Chien and Aggarwal used orthographic projection for the construction of volume models [CHIEN, 1986]. [POTMESIL, 1987] and [SRIVASTAVA, 1990] used arbitrary views and perspective projection. Numerous researchers have dealt with the shape from silhouette technique to convert visible contours into a visual hull, i.e. [SZELISKI-II, 1991], [NIEM, 1994] and [VENDULA, 1998]. Some researchers applied voxel-based shape from silhouette technique to human motion capture [BOTTINO, 2001].

In the next section the volumetric intersection method we used will be explained. Since it is a good approximation of the object, we begin our volumetric reconstruction algorithm with the model acquired by shape from silhouette algorithm and we refine this model using several tools described in chapter 5. We also applied polychromatic block matching as a refinement algorithm in chapter 4.3.3 in order to get into visual hull.

4.1.1 Experimental Setup

In this experiment to construct visual hull, we use a system configuration which consists of a single stationary video CCD camera and a turntable as a controlled motion platform on which the object is placed. Moreover, we use a calibration object to compute the interior orientation parameters of the camera. In Figure 4-1 the virtual camera positions are shown for the calibration (left) and the object reconstruction of the sample (right). We capture multiple views of the object simply by rotating the turntable. Since the camera is fixed and the coordinate system is tied to the turntable and therefore rotating itself, we show virtual camera positions.

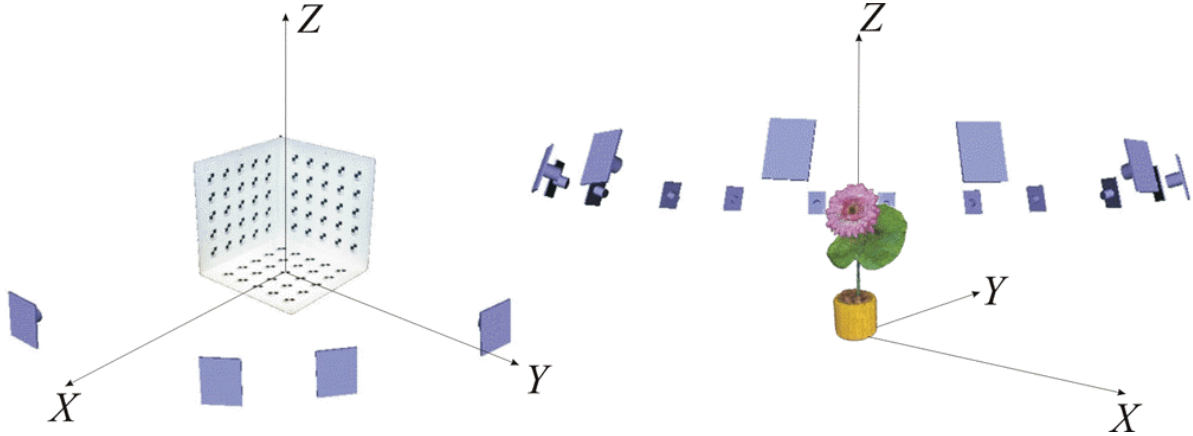


Figure 4-1: Virtual camera positions according to the turntable-fixed coordinate system

Before starting the image acquisition, the system must be calibrated. The interior and exterior parameters of the camera such as the position and orientation of the camera and the principal distance must be known. We perform a system calibration by using a special calibration object shown in Figure 4-1 (left). Also refer to Figure 3-5 in Chapter 3.2. The calibration object provides a good coverage of the object with three square faces containing 25 control points on each face. Our mathematical camera model uses central projection without a lens distortion as shown in Figure 4-2. For detailed explanation please refer to chapter 3.1.1.

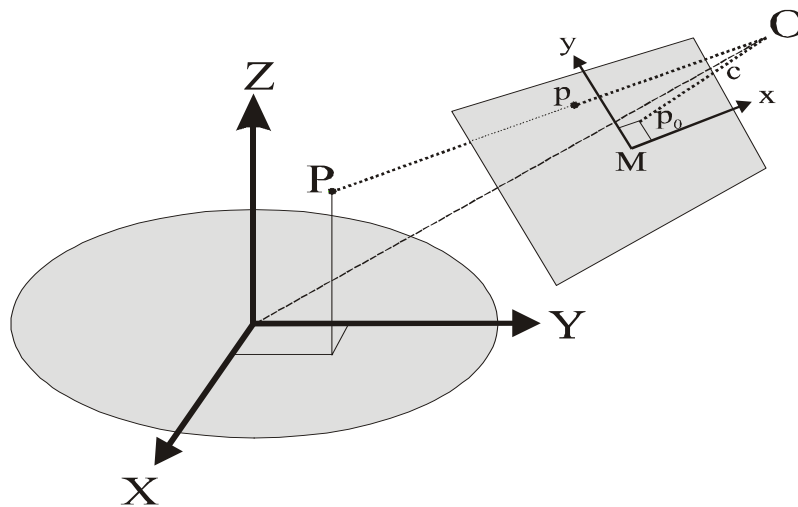


Figure 4-2: Mathematical camera model for central projection

4.1.2 Automatic Image Segmentation

Shape from Silhouette method is sensitive to silhouette noise. When an object pixel is misclassified as background pixel, object voxels which project into them might be carved from the volume. As a result, there might be gaps in the objects model. Accordingly, when a background pixel is classified as object pixel there might be noise in the object's model. Therefore image segmentation should be performed precisely. In our experiment, the turntable is rotated in controlled steps and the video images are captured and the contour of the real object is extracted. For this purpose a monochromatic blue background was used to distinguish the object from the environment. Since blue background is homogenous enough it is easy to define a hue domain representing the background. For a small frame (see dotted line in Figure 4-3) the background color is estimated using a histogram of the hue values.

For every color pixel (R, G, B) a transformation into the IHS color space is computed:

$$I = \frac{R + G + B}{3} \quad S = 1 - \frac{\min(R, G, B)}{I}, \quad I > 0.05 \quad (4.1)$$

$$H = \arccos\left(\frac{(R - G) + (R - B)}{\sqrt{(R - G)^2 + (R - B)(G - B)}}\right), \quad B < G \quad (4.2)$$



Figure 4-3: Background estimation using an IHS color space histogram (left) and the resulting silhouette extraction (right)

The automatic segmentation algorithm compares all the hue data with the estimated background information using a simple threshold and a minimum saturation value. Finally, a morphological operation ('opening') eliminates isolated pixel.

The automatically derived results were then checked with the academic software HsbVis (HSB-Visualization) that allows interactive segmentation and color channel splitting and merging on a graphical user interface [HSBVIS].

4.1.3 The Visual Hull Computation

If we know the calibration information and image orientation data, we can construct a bounding pyramid for each silhouette image. This is performed by the lines of sight from the camera focal point through all contour points of the object silhouette. Each point in silhouette defines a line in object space that intersects the object at some depth. The combination of all these rays for all foreground silhouette points defines a cone in which the objects should lie. In Figure 4-4, for simplicity, the constructed volume is shown in 2D with its input images in 1D. In the figure, black lines on the images represent foreground pixels. Foreground pixels on each image are back projected into the voxel space and intersected. The result is shown with black polygon which is the visual hull of the object.

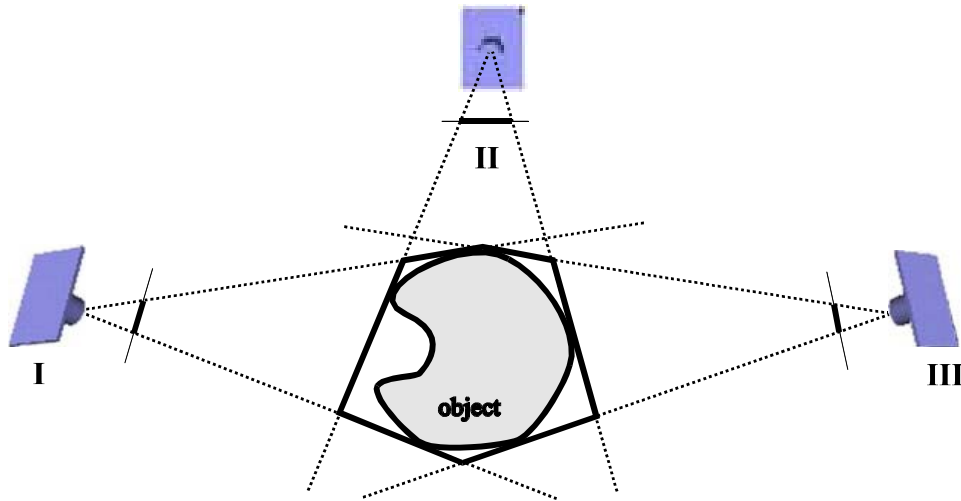


Figure 4-4: The visual hull computation

Shape from silhouette algorithms start by initializing a voxel cube that encloses the entire scene which is desired to be reconstructed. This volume is discretized into voxels. To begin with, all the voxels in the cube is set to 255 (white) which means the starting cube is fully opaque. The shape is computed volumetrically by carving away all voxels outside the projected silhouette cone. We mean with carving to label the voxels as transparent. When a voxel is carved the value is set to 0 (black) since it represents the empty space in the cube.

In Figure 4-5, silhouette cones are represented for two images. The intersection of all silhouette cones from multiple images defines estimate geometry of the object called visual hull [LAURENTINI, 1995], [MATUSIK, 2000].

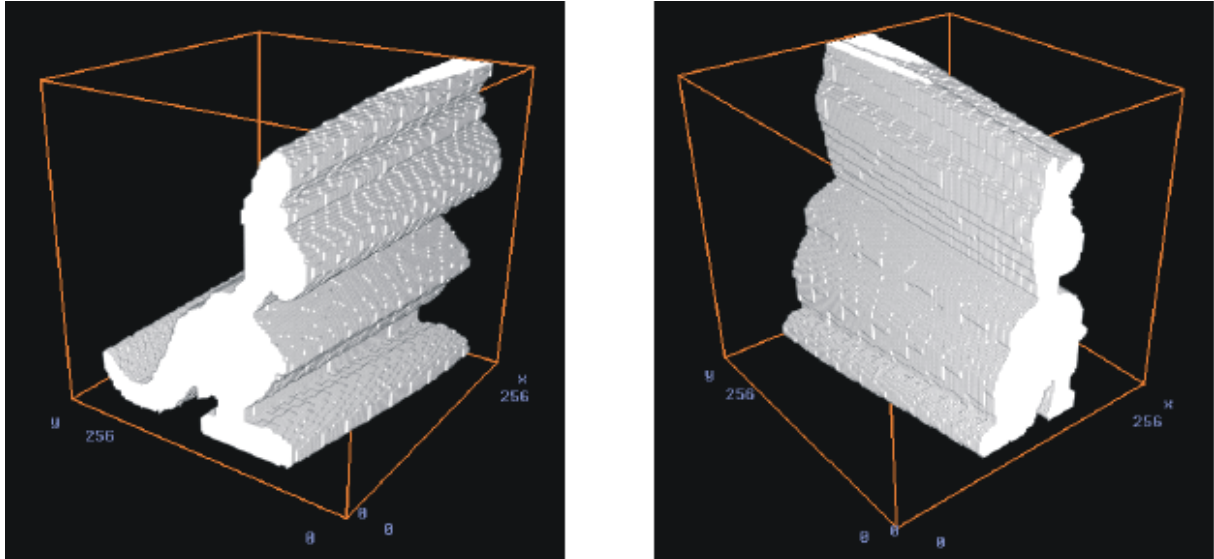


Figure 4-5: Carving of a voxel cube using various silhouettes under central projection.

In Figure 4-6 the intersection of the silhouette cones are depicted using 1, 3, 5 and 9 images. As seen in the figure with the increasing number of images, object's model is recovered successfully with the exception of concave areas.

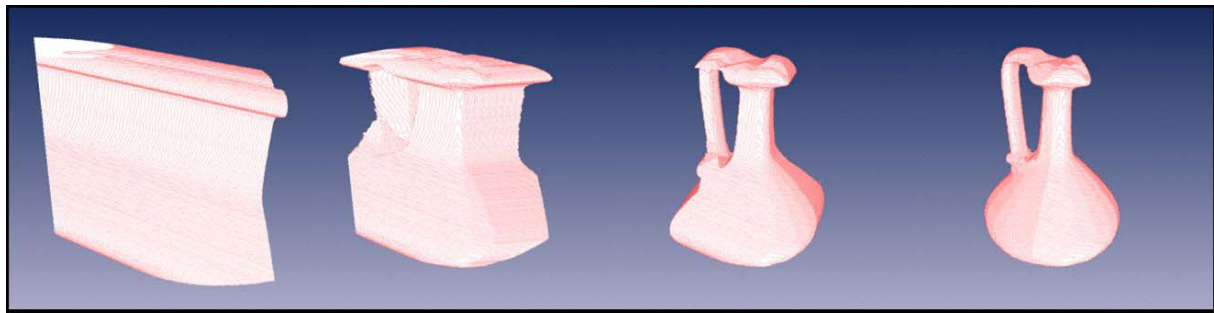


Figure 4-6: Intersection of silhouette cones using 1, 3, 5 and 9 images

4.1.4 Voting-Based Voxel Carving

The visual hull is the conservative approximation of the true shape. Shape from silhouette algorithms should not carve the voxels which are on the object surface since the visual hull should contain the objects true shape. One way to find the projected voxel on the images is projecting centroid of the voxel. Another way is to project eight corners and compute the convex hull of the projected points. If the projected points define background pixels we say the voxel is transparent and should be carved away. This implementation might cause misclassification of the voxel due to silhouette noise or at border voxels where it projects partly background pixels. In order to prevent misclassification of the voxel we implemented voting based carving. Voxels are not immediately carved when they project into background pixels on only one image, but carving of the voxel is confirmed at least by another image. Our carving algorithm use the position for each voxel in real-world coordinates (X, Y, Z) and perform a projection into image coordinates (x_i, y_i) using the well known collinearity equations (3.1) and (3.2) which were given in Chapter 3.1.1.

If the image coordinate defines a background pixel, the voxel is marked to be deleted (voting). After the processing of all voxels, the algorithm continues with the next image. Finally, the voxels are purged using a threshold for the number of votes.

The pseudo-code of the voxel-based shape from silhouettes algorithm we implemented is as follows:

```
#initialization
define cube dimensions
initialize all voxels as opaque

#voting
for every image I
{
    for every voxel V
    {
        project V into I
        if projected into background pixel
            mark V
    }
}

#thresholding
for every voxel V
{
    if V > threshold
        set V as Background
    else
        set V as Object
}
```

Figure 4-7 shows an experimental result for the visual hull with a real toy kangaroo using the presented volume intersection algorithm.

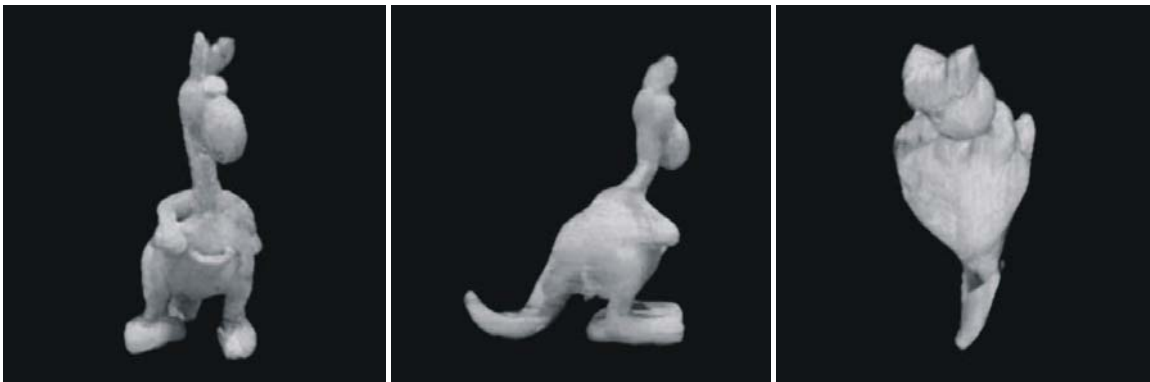


Figure 4-7: Voxel model of the 3-D shape reconstruction using 60 images and 256^3 voxels with texture mapping

As seen in Figure 4-8, the visual hull is only the approximation of the 3D shape of the objects which surrounds the true volume. In particular, concavities of the object are not visible on the images since the viewing region doesn't completely surround the object. The accuracy of the visual hull depends on the number of the images and the complexity of the object. Although visual hull is only the approximation of the true shape of the object, it has an important characteristic which guarantees to

contain the true shape of the object. Which means visual hull is the maximal shape of the object. If more precise reconstruction is desired, this method should be supported by another technique to get into the concave areas.

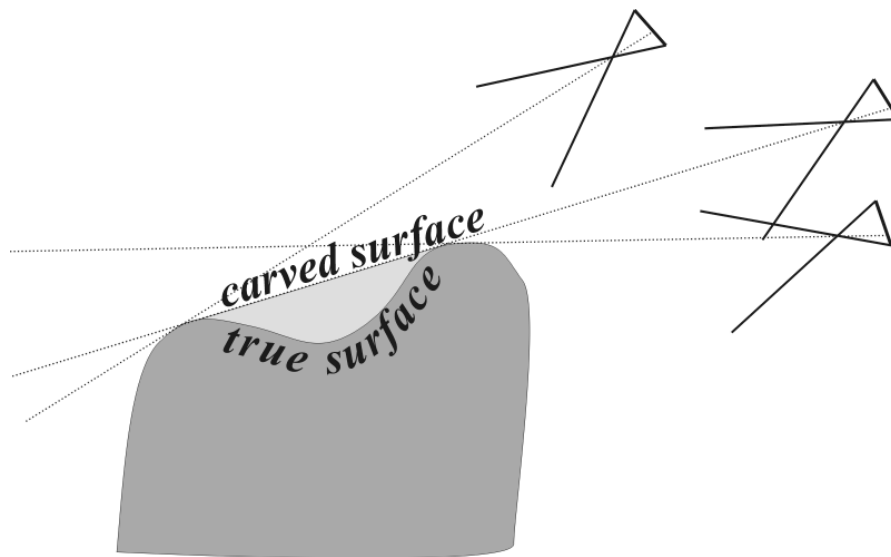


Figure 4-8: Concave areas in shape from silhouette reconstruction.

4.1.5 Hierarchical Processing Using Octrees

In order to make voxel carving more efficient most methods use an octree representation. Also refer to Chapter 2.2.7.4 for more information about octrees. Voxels are projected into images and checked if they intersect the silhouette on each image. If a voxel does not intersect a silhouette in any of the images, it is removed from the voxel space. If a projected voxel intersects only silhouette pixels at every image it is marked opaque. Otherwise, the voxel intersects both silhouette and background pixels in the images therefore it is subdivided into octants and each sub-voxel is processed recursively.

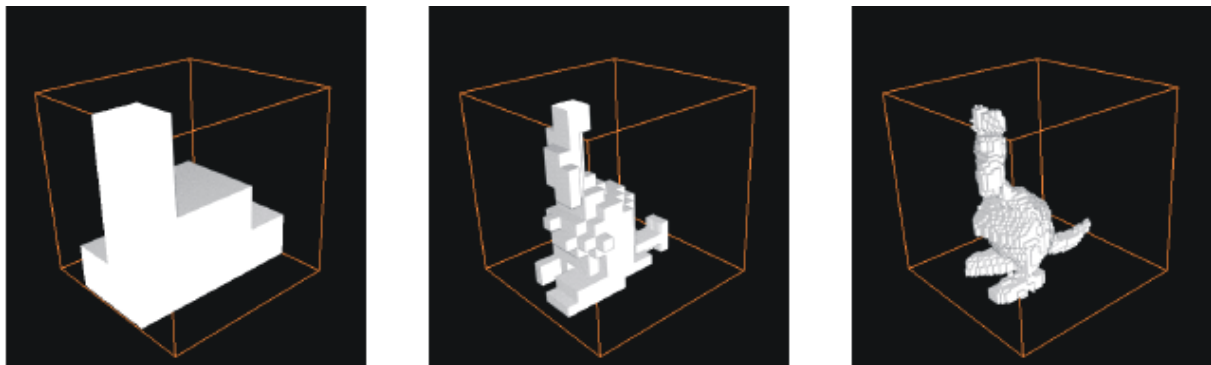


Figure 4-9: Iterative generation of an octree model using 4^3 , 16^3 , and 64^3 voxels

We enhanced the voted-based voxel carving by using a simple octree structure [POTMESIL, 1987] and [SRIVASTAVA, 1990]. The voxels at one level can be derived from the results of the preceding level by applying a modified algorithm. Some examples of the iterative generation are shown in Figure 4-9. The restrictions to areas, which contain relevant object information, accelerate the computation time significantly. (see Table 4-1)

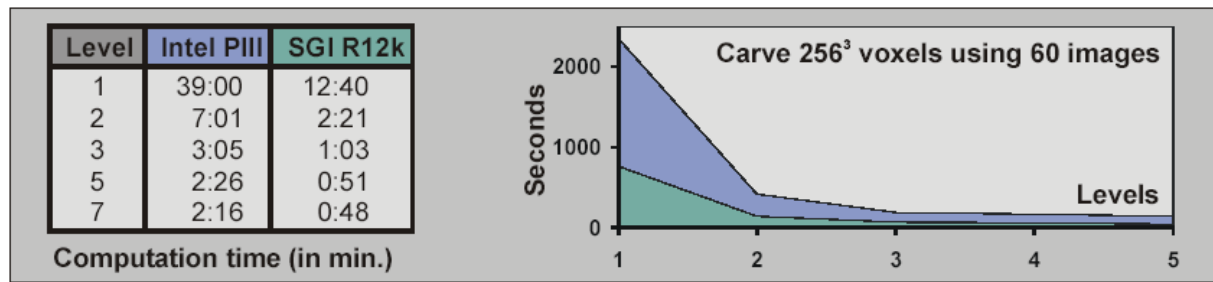


Table 4-1: Speed optimization using hierarchical processing

4.1.6 Improved Volume Intersection with Adjusted Image Orientations

Shape from Silhouette methods is sensitive to camera calibration errors. The carving may go too far if the exact image orientation data is not known. In order to project voxels to images the interior and exterior orientation must be known. The interior orientation, meaning the focal length and the principal point of the camera, is calculated in a separate camera calibration process as seen in Figure 4-1 (left). For the exterior orientation parameters (the location and rotation of the projection center), we are able to get very good estimation values, due to the scene setup, which allows the turntable to be rotated in controlled steps. However, since we use a non-calibrated turntable, the orientation parameters turned out to be quite noisy. Therefore the projections of the voxels would not yield the true pixel, instead the projection spread around the true position.

Theoretically the image orientations should only be dependent on the rotation angle of the turntable. Two rotations (ν, κ), Z_0 and the distance of the projection center to the coordinate system's origin should be constant. The third rotation (α) is a direct function of the turntable position. But since the setup is not high accurate, for example a small eccentricity of the coordinate system's origin, would result in changing parameters.

The introduction of independent, individual orientation parameters for the images, removes the limitations concerning the scene setup. The images can be taken from arbitrary positions.

Figure 4-10 shows the consequences of inaccurate parameters to the volume intersection. On the left image adjusted parameters were used, minimizing the projection error.

On the right image we see the result when using the estimated orientation parameters. Due to the errors in projections some voxels might be falsely considered as background, which will delete too many voxels. In Figure 4-10 it is clearly visible that the stem disappeared almost completely. As seen on the left image however, it remains when projected with adjusted parameters.

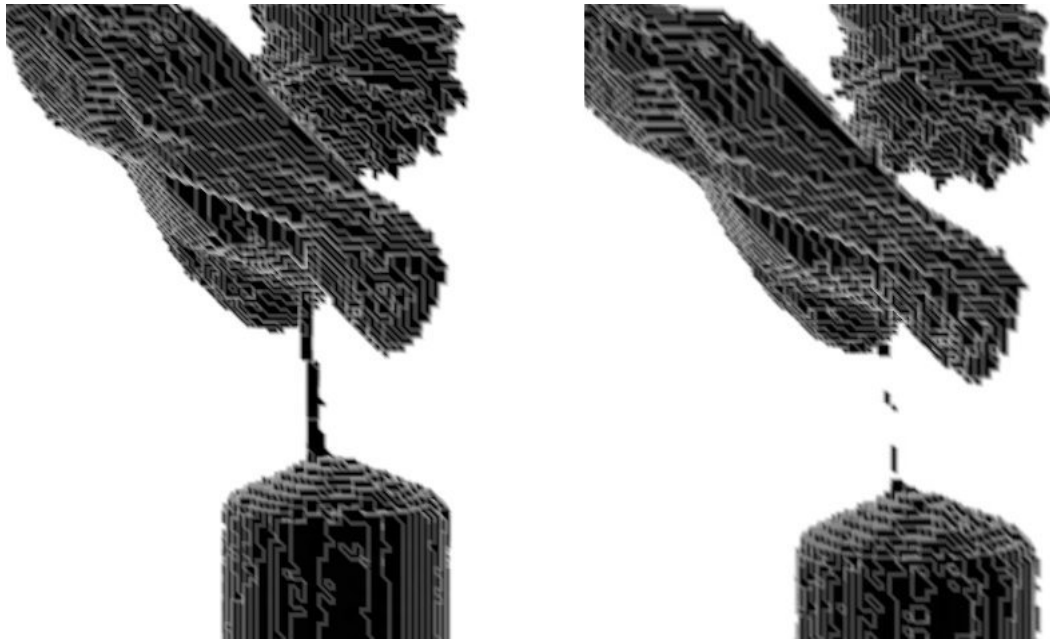


Figure 4-10: Volumetric intersection with adjusted (left) and estimated (right) orientation parameters

The orientation was done in two steps. Before we make the bundle block adjustment which delivers the final orientation parameters, we first perform camera calibration and calculate the interior orientation parameters. This process also yields one camera position when replacing the calibration object with the actual object. With this first image and a number of tie points the camera setup can be reconstructed except for the scale. This can be eliminated using one single control point. As an alternative, a distance between two arbitrary points can be taken as well, if no control points exist at all.

In this experiment, the whole bundle was adjusted with the following parameters:

12 known parameters	($c, x_0, y_0, \alpha, v, \kappa, X_0, Y_0, Z_0, X_P, Y_P, Z_P$ for camera, one image and one control point)
267 unknowns	(15.6 for the images, 59.3 for object points)
638 observations	(319 observed tie points)

It yielded the image orientations for all 16 images, see Figure 4-1 (right), as well as object coordinates for the observed tie points. Those object points were not the goal but just a byproduct, because the very object will be reconstructed by the techniques described in the following chapters. After the adjustment the image orientations were calculated with an accuracy of 5-7 mm in position and 0.3 and 0.5 gon in rotation. However, the introduction of control points can only improve the results.

4.1.7 An Application of the Volume Intersection in a Medical Project

In a student project at our department, we have adapted the volume intersection method to a medical project. The goal of the experiment is to measure deformations in the human leg due to long distance flights. Therefore, the volume of the leg before and after the flight should be computed. With volumetric methods volume calculations are quite easy since it is only a matter of voxel counting. In the system setup, we used a blue homogenous background to segment the leg from empty space. The images are captured by moving the camera around the leg to be reconstructed. Figure 4-11 shows the image acquisition step.

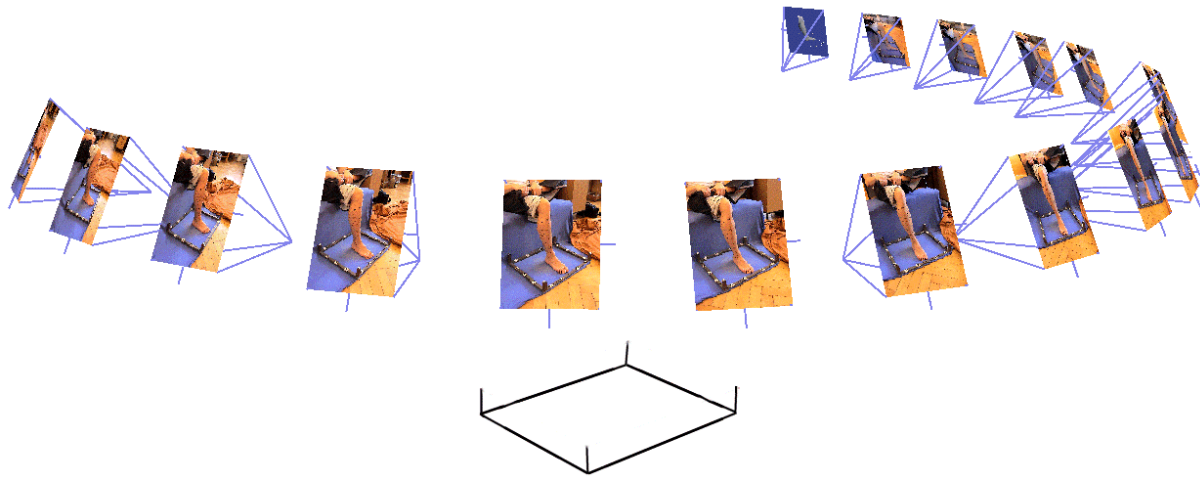


Figure 4-11: Image acquisition step

Figure 4-12 shows some slices of the reconstruction acquired by using the academic program “Slicer” which was developed at our department. As seen in the histogram on the right, the volume of the leg is calculated simply by counting the voxels of the final reconstruction, then the volume is computed considering the resolution of the voxel cube, in this experiment for example, $1 \text{ voxel} = 1.5 \text{ mm}^3$.

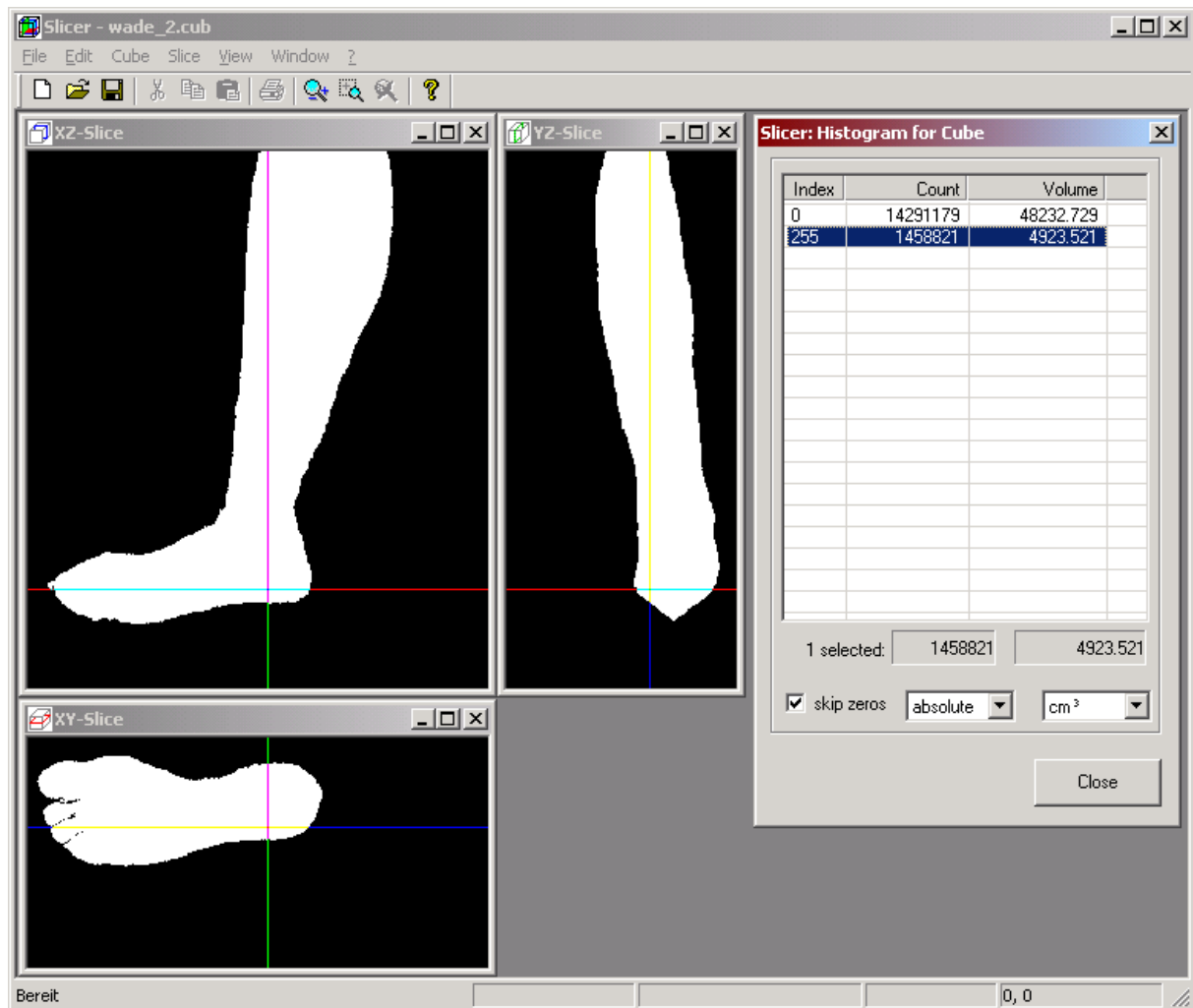


Figure 4-12: Slices of the reconstruction along three axes and the histogram for volume calculation

In Figure 4-13 the reconstruction results are presented from different views. As mentioned previously, in this method the accuracy is dependent on the number of the images, hence, as it can be observed in the Figure 4-13 there are flattened areas where the adequate images are not provided. Therefore a large number of images from a widely spread viewpoints should be used in order to achieve the desired accuracy of volume computation.



Figure 4-13: Reconstruction results of the leg with 16 images

4.2 Shape from Photo Consistency

In Chapter 4.1, performing image segmentation, binary images are obtained and these images are used to compute the visual hull of the object. However, when color images are captured instead of binary images, the photometric information can be used to reconstruct a 3D scene that is color consistent with input images. In Chapter 4.3.3 polychromatic block matching method is used to refine the visual hull which makes use of color images. In this chapter, instead of stereo algorithms, voxel coloring methods are examined for reconstructing photorealistic models of the objects. The main task of voxel coloring algorithms is to determine surface voxels of the objects by using a color consistency test and assign color values to those which pass the test. Thus, when we render the model from different viewpoints we should obtain synthetic images as realistic as the real-world scene. Therefore, the final reconstruction should be accurate and dense.

Voxel Coloring [SEITZ-II, 1997], Space Carving [KUTULAKOS-I, 1998] and Generalized Voxel Coloring [CULBERTSON, 1999] have been relatively recent solutions to the 3D scene reconstruction problem. Seitz and Dyer's voxel coloring algorithm restricts the position of the cameras in order to simplify visibility information. Their algorithm works coloring the voxels in a special order similar to Collins' Space-Sweep approach [COLLINS, 1996], which performs an analogous scene traversal.

Kutakulos and Seitz's space carving method removes the constraint of camera placement. However, space carving algorithm does not use the full visibility of the scene. Both voxel coloring and space carving scans voxels for color consistency by evaluating a plane of voxels at a time.

Culbertson *et al.* introduced an efficient voxel coloring algorithm called Generalized Voxel Coloring (GVC) that computes visibility correctly and the result is a color consisting model [CULBERTSON, 1999]. Unlike Voxel Coloring, GVC removes the constraint on camera positions. On the other hand, it uses all images to check voxel consistency unlike Space Carving, which uses only a subset of the images. There are two variants of the algorithm called GVC and GVC-LDI which will be explained in detail in Chapter 4.2.5.

Volumetric reconstruction algorithms in general assume that the objects are opaque therefore transparency and aliasing effects can be ignored. Even though the object in the scene does not contain transparent regions and completely opaque, the boundary regions might require transparency since they are partially filled. In their approach, Szeliski and Golland [SZELISKI-IV, 1998] consider to represent partially transparent regions by using real valued transparencies. An accurate model of partially occupied voxels is useful to represent semi transparent materials like colored glass.

De Bonet and Viola [DE BONET, 1999] proposed an optimization method for voxel coloring that represents transparency and color of 3D voxel space. They call their approach Responsibility Weighted 3D Volume Reconstruction (Roxel) algorithm. Roxel algorithm is able to reconstruct volumes from images which contain both opaque and partially transparent objects. In their approach opacities are assigned to voxels as well as color values. Roxels also attempt to minimize reprojection error. They observe that image pixels are linear combinations of the colors of the voxels along the each visual ray through the volume. And when projected into the images it minimizes the errors of the input images. The coefficients of the linear combinations are called responsibilities of the voxels. Roxel algorithm is a multi step procedure which alternates between estimation of the colors, responsibilities and opacities. They use an iterative algorithm that solves the responsibilities and then uses the responsibilities to estimate the opacities.

Voxel coloring algorithms reconstruct small scale scenes successfully but it is not well suited to capture the environment like cloud or background objects. It is hard to reconstruct large scale scenes with voxel coloring algorithms, since it requires extensive number of voxels. Furthermore, it is unnecessary to present the further objects with high resolution voxels. Slabaugh *et al.* presented volumetric warping method using a frustum warping function to reconstruct large scale scenes with spatially adaptive voxel size that increases away from the cameras [SLABAUGH-II, 2000]. In this method a hybrid voxel space is developed consisting of two regions: interior space to reconstruct the objects and the exterior space to reconstruct the background surfaces. The interior space is modeled with a fixed voxel resolution, thus, volumetric warping technique does not affect the interior space and it provides compatibility to other volumetric reconstruction techniques.

Voxel coloring algorithms might result in thicker reconstruction than the true object. Level set methods, however, provides thinner reconstruction and avoids the floating extraneous geometry. Level set theory is developed by Osher and Sethian [OSHER, 1988]. The method considers tracking of

motion of a surface whose speed depends on the local curvature. The algorithm has many application areas including computer vision and image processing. Faugeras used level set method in scene reconstruction problem [FAUGERAS, 1998]. Slaubaugh *et al.* combined the method with space carving algorithm [SLABAUGH-IV, 2002]. In their level set method, they produce a smooth reconstruction composed of manifold surfaces. The initial surface is embedded as the zero level set of the volumetric function. This surface then moves along its inwardly pointing normal through the scene. Level set theory solves the partial differential equations that characterize the motion of the surface. The speed of the motion is controlled by photo consistency check similar to voxel coloring algorithms. The initial surface is deformed until it forms the desired shape. The authors provide a multi resolution approach, when the resolution is increased; they dilate the surface and re execute the surface evolution with higher resolution. The visibility is calculated by using item buffers. The result of the algorithm is a polygonal model which is later textured mapped using the images.

In the following chapter some of the voxel coloring algorithms will be introduced. These algorithms mainly differ in the way they compute the visibility information.

4.2.1 Comparison of Voxel Coloring Methods with Area-Based Image Matching

Voxel coloring algorithms do not perform area-based image matching to compute correspondences. Different from stereo methods, they project voxels into the images and compare the color values of the pixel sets on which the voxel's projection overlaps. Voxel coloring algorithms assume that the surfaces are Lambertian, in other words surface points project onto the similar colors in the images where they are visible. However, the normalized cross-correlation takes differences in illumination into account. Therefore, it is less sensitive to image illumination. Another difference of voxel coloring algorithms from traditional stereo methods is that the occlusions are taken into consideration and fully modeled during the reconstruction. Cameras can be positioned far from each other, in other words a small base line is not required and it would not cause inaccuracies. In area-based image matching, large base line would cause high patch deformations due to perspective distortions; as a result image matching would fail. However, in least squares matching method mentioned in Chapter 4.3.1.5 or the proposed knowledge based patch distortion algorithm explained in Chapter 5.2.1, this effect is reduced since these deformations are considered and accordingly transformed patches are grabbed for image matching. Photo consistency algorithms use large number of images to reconstruct a dense model and when this model is rendered, they produce high quality synthetic views. The main disadvantage of these algorithms is they need very accurate calibration data in order to achieve high accuracy of stereo methods. They are especially sensitive to calibration accuracy at highly textured regions like image edges. Moreover, smaller voxels are more sensitive to calibrations errors than larger voxels since photo consistency is checked with fewer pixels.

4.2.2 Color Consistency

Color consistency is introduced by Seitz to distinguish surface points from the other points in the scene [SEITZ-II, 1997]. The final model of the scene includes the geometric information as well as surface reflectance model and scene illumination. Color-consistency checks the consistency of a real world point projecting it into given set of images in order to verify if it's a surface voxel or not. This is performed by making use of the fact that only surface points in a scene project into equal (consistent) colors in the input images. In other words, the irradiance of the surface point is equal to the intensity of the pixels which the voxel project into. Voxels that are color consistent with the images on which they are visible are assumed to be on the surface of the object and given their corresponding color

value. However, inconsistent voxels are assumed to represent empty space and are removed from the voxel space. Voxel coloring algorithms continue until no non-photo consistent voxels exist in the model and the final set of voxels contain sufficient color and texture information.

In photo-consistent reconstructions, all surfaces are considered Lambertian. This means surfaces are equally bright in all directions regardless of the illumination.

In Figure 4-14, the surface voxel A is projected into the same colors in the images while the voxel B that is not on the surface is projected into different colors. Voxel A is seen red on both cameras while voxel B is seen green on camera I and transparent (background) on camera II.

In the presence of noise and quantization effects the consistency of a set of pixels can be defined as their standard deviation. The voxel is considered to be on the surface if it is less than a sensible threshold, which varies according to the object, the sensor and lighting conditions. The voxels that exceed this threshold are considered non-surface voxels and discarded or carved. Color consistency should be accurate not to carve consistent voxels. Because carving a voxel falsely might affect the other voxels visibility as well.

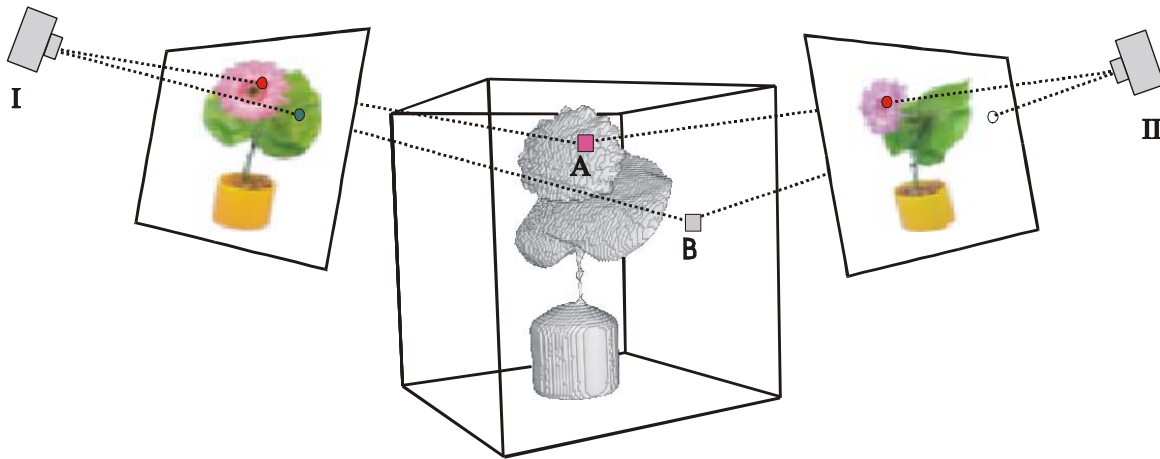


Figure 4-14: Color consistency is used to distinguish surface points from points not on the surface

4.2.2.1 Color Consistency Measures

The accuracy of the photo consistent reconstructions, in other words how well they produce the true scene, depends on the color consistency test. The simplest way to compute photo consistency is to threshold the colors of the pixels where the center of the voxel projects into the images it is visible. Seitz and Dyer [SEITZ-II, 1997] compute photo consistency from the set of all pixels π_j from the given m images in which the voxel projects onto:

$$\bigcup_{j=1}^m \pi_j$$

In order to determine the voxels consistency, the standard deviation σ , of the pixels colors is computed and thresholded by using a specified threshold λ .

$$\sigma \leq \lambda \Rightarrow \text{voxel is consistent.}$$

$$\sigma > \lambda \Rightarrow \text{voxel is inconsistent.}$$

Photo consistency check can be easily adapted to RGB channels. Thresholding the standard deviation also works with other color spaces rather than RGB. In general there is no single color threshold that is ideal for reconstruction of all surfaces in the scene. Some surfaces might be better reconstructed with lower threshold but using a lower threshold might cause other surfaces carved which are consistent.

The errors of photo consistency may cause fatter surfaces than true scene or false concavities, holes or cusps. When the chosen threshold is too high the carving may go deeper and when it is too small the concavities cannot be refined. The holes might be filled by hole-filling methods [CURLESS-I, 1996]. The fattening occurs due to homogenous surfaces. In the homogenous regions, the voxels in front of the true scene pass the consistency test since they have low color variance therefore not carved.

Color consistency check fails on the surfaces with abrupt color boundaries like textured regions and edges. Voxels that project on such boundaries might be visible from a set of pixels but not color consistent. This problem can be solved using an adaptive threshold.

[STEVENSON, 2002] computes the photo consistency of a set of pixels by comparing their histograms. They use histogram intersection to compare the histograms. Unlike the methods mentioned above, they do not pool all the pixels a voxel projects onto in all images it is visible; instead they make series of test on image pairs. They first find the set of pixels, π_j^i from the image j , a voxel i is visible. They test the consistency of a voxel by comparing all pairs of such pixel sets:

$$\forall_{k,l} \pi_k^i \approx \pi_l^i \quad k \neq l$$

where π_k^i and π_l^i are not empty.

They use a 3D histogram over the complete RGB color space. They divide the color space into 512 bins (8x8x8) for each image of each voxel. A sub-space is defined as a bin and there are 8 bins per channel. They construct histograms for each image that can see a voxel and the colors of the pixels are inserted into the histogram. All pairs of histograms are compared by intersecting their corresponding bins.

$$\forall_{k,l} \text{Hist}(\pi_k^i) \cap \text{Hist}(\pi_l^i) \neq \emptyset \quad k \neq l$$

Pairs of histograms intersect if at least one pair of bins has a non-zero count. The voxel is found inconsistent if any pair has no corresponding bins.

4.2.3 Voxel Coloring

Voxel coloring algorithm is introduced by Seitz and Dyer [SEITZ-I, 1997]. This algorithm constructs a colorful scene using coloring consistency check. Voxel coloring algorithm does not begin with the model acquired by volume intersection method but with initially opaque voxels. However, the authors segment the background from object in order not to process most of the background pixels.

During the algorithm, opaque voxels are tested for color consistency and the voxels which pass a certain threshold is given their corresponding color value. The voxels which are below the given threshold are carved away from the voxel space. If all remaining voxels are consistent, in other words, if they represent the scene correctly the algorithm ends. In order to evaluate color consistency test, first of all, the voxels projection on an image should be computed. The authors compute the footprint of a voxel based on voxels shape and the known calibration data. Footprint refers to the set of pixels that a

voxel's projection overlaps on an image. In order to approximate voxels footprint the authors use a square mask.

Since opaque voxels occlude each other, visibility information must be calculated previously, that means the set of pixels that see a voxel should be determined. For proper handling of visibility, the occluded voxels should be visited before the surface voxels. To simplify the voxel visibility calculation and allow reconstruction of the scene with single scan of voxels, Seitz and Dyer introduced ordinal *visibility constraint* on camera locations. They restrict the camera placement so that no scene point is within the convex hull of the camera centers. This constraint requires the cameras should be placed in a way that it can be swept with a single pass through the volume. This is done by placing all cameras at one side of the scene and sweeping the voxel in near to far order away from the cameras. Which means that voxel P occludes voxel Q in image I only if the distance of P to the cameras convex hull is less than the distance of Q. Voxels are swept one layer at a time. The authors define two camera placements that satisfy ordinal visibility constraint. One of which is an inward facing camera rotating around an object. Figure 4-15 shows such a camera geometry that satisfy ordinal visibility constraint and enables the voxels that occlude a surface voxel are visited before the surface voxel is checked for color consistency. As seen in Figure 4-15, cameras are positioned in a plane and scene is slightly below that plane so that no scene point is within the convex hull of the camera centers. The second camera placement is for panoramic configurations with an outward facing camera.

When a voxel is found color consistent, the pixels that overlap voxels projection on the images are marked. If subsequent voxels project to those marked pixels, they are not visible on those cameras thus they are not considered. Since ordinal visibility constraint provides voxel evaluation in occlusion compatible order, marking strategy is sufficient to ensure that pixels are processed only for the voxels visible on the images. In Figure 4-15, voxel shown in gray is a photo-consistent surface voxel. According to sweeping direction, voxel shown in gray is visited first and corresponding pixels are marked on concerning images. Voxel shown in black projects into previously marked pixels on camera II therefore it is not visible on camera II.

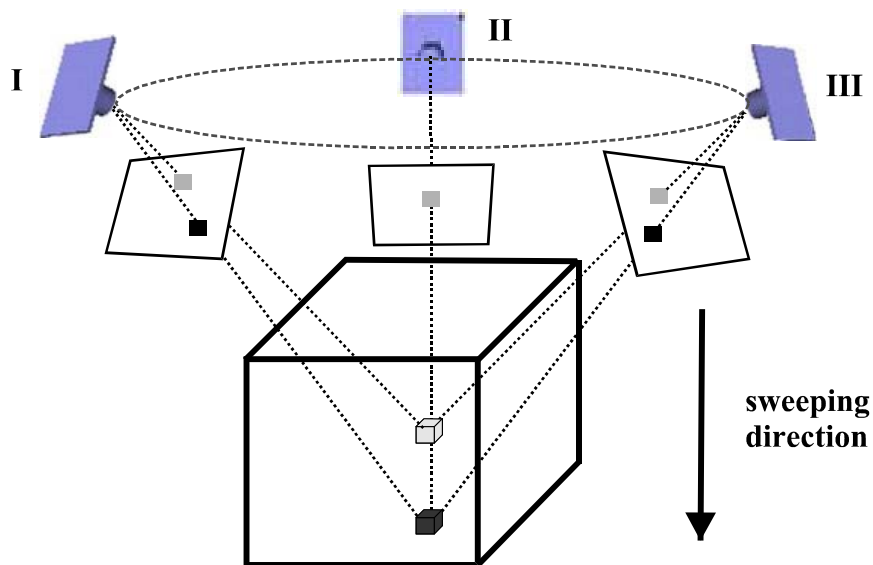


Figure 4-15: Visibility computation in voxel coloring

The pseudo-code of the algorithm is as follows:

```

initialize all voxels as opaque
for every layer l
{
    for every opaque voxel V in the layer l
    {
        for every image I
        {
            compute the voxels footprint p into image I
        }
        compute correlation  $\lambda V$  of pixel colors to evaluate voxel consistency
        if  $\lambda V < \text{threshold}$  {
            color the voxel
        }
        mark image pixels
    }
}

```

In order to speed up the voxel coloring process Prock and Dyer [PROCK, 1998] used coarse to fine fashion voxel coloring. Coarse to fine methods allow processing to be focused on the regions where there are higher details. It starts with low resolution as input, and then resolution is increased to create better reconstructions. Some voxels which actually contain small opaque sub-regions remain uncolored at lower resolution. Therefore voxels with smaller occupied sub regions will be missing which causes gaps in reconstruction. In order to compensate those missing voxels authors has chosen nearest neighbor search strategy. All voxels within 1 norm neighborhood of the low resolution set is added then subdivided into octants. They have also adapted voxel coloring to dynamic scenes assuming temporal coherence. The scene at successive points in time is similar therefore the previous voxel coloring is used as the next time voxel coloring. By doing so, the regions which contain empty space are not analyzed again. This works when the scene is not changed too quickly. Rapid movements will cause surfaces to be missed however they will be recovered at the next step.

4.2.4 Space Carving

Voxel coloring algorithm described previously is simple and computationally inexpensive since every voxel is visited only once, but the restriction on camera placement is a significant limitation. Kutulakos and Seitz remove camera placement restriction in voxel coloring by implementing a multi-sweep procedure [KUTULAKOS-I, 1998]. Unlike Voxel Coloring which uses a single scan, space carving evaluates a plane of voxels at a time that are in front of a subset of cameras. Each scan sweeps the voxels in positive and negative directions of each three axis through the volume and the photo consistency is checked only with the subset of cameras that are behind the plane. Each sweep guarantees that occluded voxels are visited before. This is performed by choosing the cameras that lie on the side of the plane at each sweep. In Figure 4-16, for all the cameras whose x coordinate is less than x_0 , the plane is swept in the direction of increasing x coordinate to make sure that the voxel p which occludes voxel q is visited previously. The cameras become active when the sweeping plane passes over them. The visibility is maintained by marking pixels as it is in Voxel Coloring. Each scan might change the visibility of other voxels since some voxels are carved each time, therefore sweeping process continues until no change occurs and the resulting shape is photo hull.

In this method the input images are also segmented to remove background pixels. In one experiment the authors change the background manually during the image acquisition step in order to avoid the image segmentation process. Using different backgrounds ensures that non-object voxels are carved from the voxel space since they project varying colors thus they are not photo consistent. So to say, segmentation is achieved in object space not in image space. Since the background is outside of the initial volume, it is not reconstructed. Space carving provides arbitrary camera placement and never carves the voxels it should not.

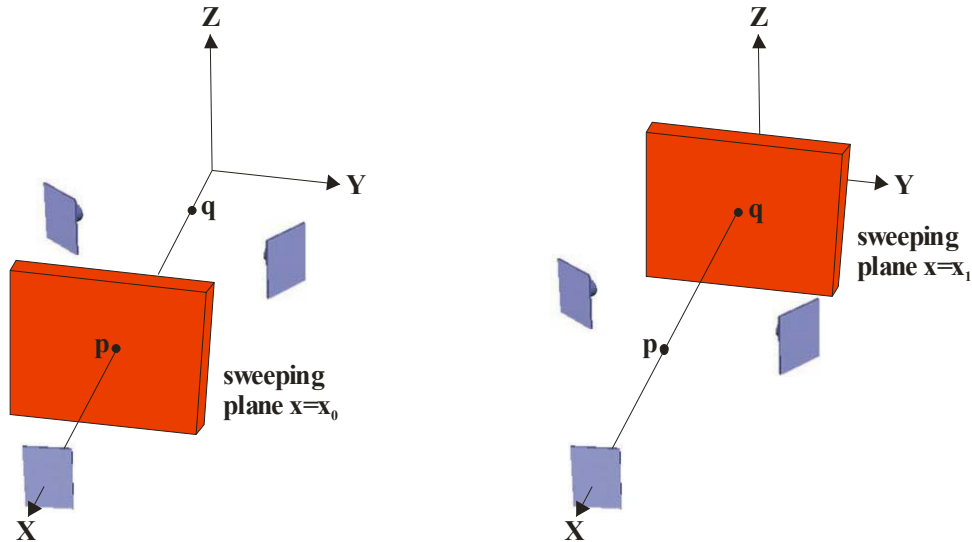


Figure 4-16: Visibility ordering in space carving

4.2.5 Generalized Voxel Coloring

In order to improve efficiency of space carving algorithm, generalized voxel coloring algorithm is developed by Culbertson, Malzbender and Slabaugh which maintains a structure that each pixel indicates the closest opaque voxel along the line of sight of the pixel [CULBERTSON, 1999]. It uses different data structures to compute visibility which are more sophisticated than pixel marking in Voxel Coloring and Space Carving algorithms. There are two variants of the algorithm to compute visibility namely Generalized Voxel Coloring with Item buffers (GVC-IB) and Generalized Voxel Coloring with Layered depth images (GVC-LDI).

In (GVC-IB), item buffers are created for each image in order to get the full visibility information. In the item buffers each pixel is assigned the ID of the closest visible surface voxel it corresponds to. This way, only visible (non-occluded) voxels are considered. In (GVC-LDI), layered depth images record a depth sorted list of all voxels that projects into the pixels in an image. GVC-LDI exactly determines which voxels have their visibility changed when a voxel is carved while GVC re-evaluates all voxels in the current model. Therefore GVC-LDI algorithm requires fewer color consistency evaluations however GVC-LDI consumes more memory since it is the superset of the information in an item buffer.

When a voxel is carved, the visibility of the remaining interior voxels might change. Thus, the item buffers should be updated periodically. The authors keep track of the voxels which are on the surface of the object by using the list of surface voxels. At each iteration surface voxels list is updated because when the voxels are carved, the neighbor voxels become surface voxels. Item buffers are computed by projecting the voxels on the surface voxel list on them. In GVC, item buffers are computed from

scratch. In Figure 4-17 (left) both voxel A and voxel B project into the same pixel. Since voxel A occludes voxel B, in the item buffer the ID of voxel A is recorded in the corresponding pixel.

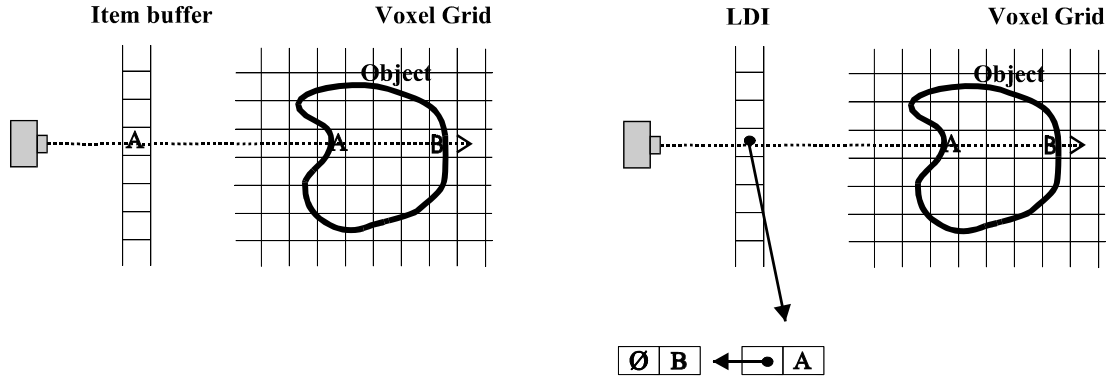


Figure 4-17: Two variants of GVC using different data structures to compute visibility, item buffers (left) and layered depth images (right)

Different from GVC, LDI store all the surface voxels that project on each pixel. In LDI the voxels are sorted according to their depth from the image's camera so that the closest voxel is processed first. When a voxel is carved, LDI can be updated immediately and easily by deleting the carved voxel from LDI and the voxel whose visibility is changed moves to the head of LDI.

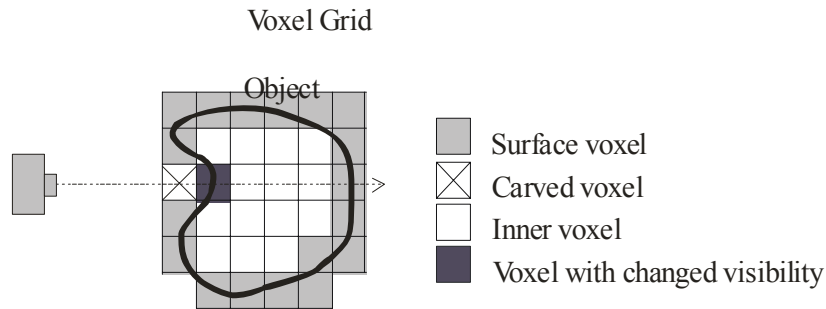


Figure 4-18: When a voxel is carved, the visibility of the adjacent inner voxels change

Once the item buffers are calculated for every image, each voxel on the surface voxel list is projected into all the images. The ID of the projected voxel is compared to the ID stored in the image's ID buffer. If they are equal the pixel is added to the visibility list and the colors are recorded. Then all the visible pixels are checked for color consistency. If the standard deviation for these pixels exceeds a predefined threshold it is removed or carved from the volume. Otherwise it is given the average color value of the visible pixels.

The algorithm continues until no carving occurs at the final iteration in other words, no non-photo consistent voxels exist and the final set of voxels contains sufficient color and texture information to accurately represent the original scene.

[SLABAUGH-I, 2000] post processed the photo hull in order to refine the reconstruction. They considered this approach as an optimization problem. They examine the methods in order to minimize the projection errors by using simulated annealing and greedy methods. They iteratively add or remove border voxels until the sum of the squared differences between the input images and scene model rendered in each image is minimized.

4.2.6 Multi-Hypothesis Voxel Coloring

Eisert and Steinbach presented a multi-hypotheses voxel coloring technique based color hypothesis tests of the voxel model back projected into the images [EISERT, 1999], [STEINBACH, 2000]. Their algorithm begins with hypothesis assignment step where each voxel is filled with several color hypotheses from different camera views. In the following hypotheses removal step, by iterating several times over all views, the assigned hypotheses are checked for consistency and only the hypothesis remain in the voxels which are consistent with all camera views where the voxel is visible. Voxels without a valid hypothesis are considered to be empty voxels and they are carved from the voxel volume till correct shape of the object is recovered.

In hypothesis extraction steps, voxels center is projected into the images and if at least in 2 images voxel projects into the consistent colors, a hypothesis is assigned to the voxel. Due to occlusions, a voxel need not to be visible in all views and if it is an interior voxel it might not be visible in any views at all. In the hypothesis extraction step occlusions are not taken into consideration since the geometry of the object is not known yet. Therefore during the hypothesis extraction step, a voxel might not be assigned correct color hypothesis.

In hypothesis removal step, for each camera view, voxels are traversed into the image in the order of increasing depth from the camera view. Therefore voxels are projected into the image in occlusion compatible order and the pixels they project is compared with voxels hypothesizes. The inconsistent hypotheses are removed. This process continues with all viewpoints. If a voxel's all hypotheses are removed, it is carved from the voxel space since it is an empty background voxel. When a voxel is set to be transparent in other words carved from the volume, the visibility of other voxel is changed. Therefore the algorithm iterates multiple times until no more hypotheses are removed.

The advantage of multi hypothesis voxel coloring is carving process is performed one image at a time which simplifies the visibility process. However, all voxels including the interior voxels are assigned hypotheses which causes extra computation.

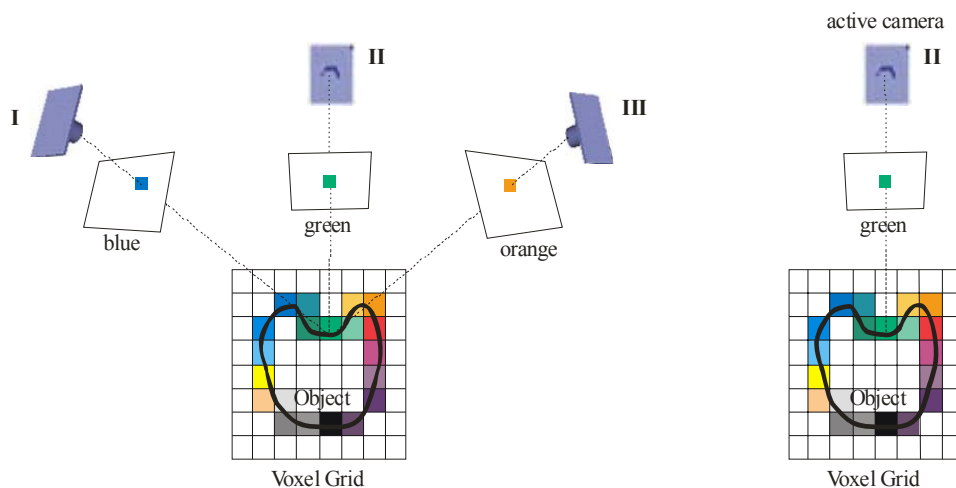


Figure 4-19: Multi-hypothesis voxel coloring

In Figure 4-19 on the left, hypothesizes are assigned to the voxel of interest by projecting it on the images. On the right, hypothesizes removal step is shown which considers the visibility of the voxel. The cameras I and II which observe the voxel as blue and orange are not considered.

4.2.7 Voxel Coloring Using Ray Tracing for Visibility Computation

In GVC, an item buffer is constructed for each image in order to obtain the full visibility information. Item buffers are used to store the visible voxels in the scene from a given viewpoint. In the item buffer, each pixel has the ID of the nearest voxel it projects into. Our method differs from standard GVC method in the way it computes item buffers. We do not project the voxels into the images instead; we trace the pixels into the voxel cube in order to calculate item buffers. Also refer to [KUZU-III, 2002]

Projection of a voxel into the image plane covers more than one pixel due to its cubic shape. Depending on the scale of the whole scene a single voxel may project onto a footprint of pixels (for large images) or several voxels merge into one pixel (for small images). If the exact projection of voxels is not found, there might be gaps in the image's item buffer. Through these holes far away voxels may shine through, storing their ID in the item buffer, although they should not be visible. This condition might lead to false visibility information and failure of color consistency check. Therefore the item buffer should be dense, meaning that each pixel should contain the nearest voxel's ID.

Hence, instead of taking the footprint of voxels, in this method we traced the pixels back into the scene to find the first opaque voxel along the line of sight in the cube. To find the minimum and maximum factors in the loop boundaries, we derived the lambda factor for each corner of the entire voxel cube. This is the factor by which the distance of the image point to the projection center must be multiplied to obtain the distance from the object point to the projection center.

The loop starts with greater step size, which decreases as soon as the cube is entered therefore the process is accelerated. The loop ends when either a surface voxel is encountered, or the ray exits the cube, not encountering any opaque voxel at all. This way each pixel, which does not have a background value, is assigned the ID of a voxel, without leaving gaps in the image item buffer. This approach would correspond to the indirect image resampling methods, which does a geometric transformation indirectly from the destination back to the source.

We begin the algorithm by initializing a volume that encloses the 3D object to be reconstructed. In order to accelerate the algorithm and carve the empty spaces easily we used a rough bounding volume found by the volume intersection method. This bounding volume contains the true shape of the object and it is calculated easily as described in detail in Chapter 4.1. As mentioned previously, although this volume is a good approximation of the model, it does not fully cover the concavities. According to this volume, all boundary voxels that completely separate empty regions of space from opaque regions are created and defined in a list as surface voxel list (SVL) as it is in GVC. The SVL should be updated during the reconstruction iterations to make sure that only a minimum number of voxels are processed. We used item buffers in order to compute the visibility as it is in GVC.

The algorithm uses the image coordinates and performs a projection into voxel coordinates using the equations (3.5) which was derived from equation (3.4) in Chapter 3.1.1.

Once the item buffers are calculated for each image, each voxel on the surface voxel list is projected into each image. This is the inverse transformation compared to the creation of the ID buffers.

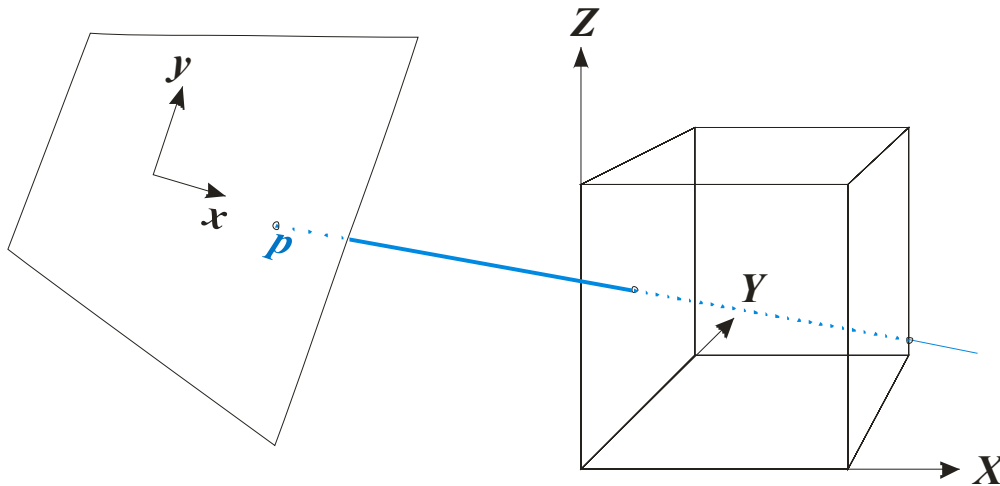


Figure 4-20: Tracing pixels back into the voxel cube

The ID of the projected voxel is compared to the ID stored in the image's ID buffer. If they are equal the pixel is added to the visibility list and the colors are recorded. Then all the visible pixels are checked for color consistency. If the standard deviation for these pixels exceeds a predefined threshold it is removed or carved from the volume. Otherwise it is given the average color value of the visible pixels.

When a voxel is carved the adjacent opaque voxels become surface voxels. The previously derived SVL becomes invalid and has to be re-evaluated in the next iteration. The algorithm stops when all the inconsistent voxels are removed. The remaining surface voxels are photo consistent with the input images and form the photorealistic 3D model of the scene. The final result can be seen in Figure 4-21. It has been computed as a 256^3 voxel cube, using 16 images, with a resolution of 1020 by 1360 pixels, each. Pseudo-code of the algorithm is as follows:

```

Loop {
    initialize item buffers
    initialize SVL
    for every voxel V on SVL {
        find the set S of image pixels from which V is visible
        if V is consistent {
            color V
        }
        else {
            carve V
        }
    }
    if all voxels are consistent
        quit
}

```

Line tracing with the above explained method is time consuming. In Chapter 5.1 an improved line tracing algorithm is introduced.

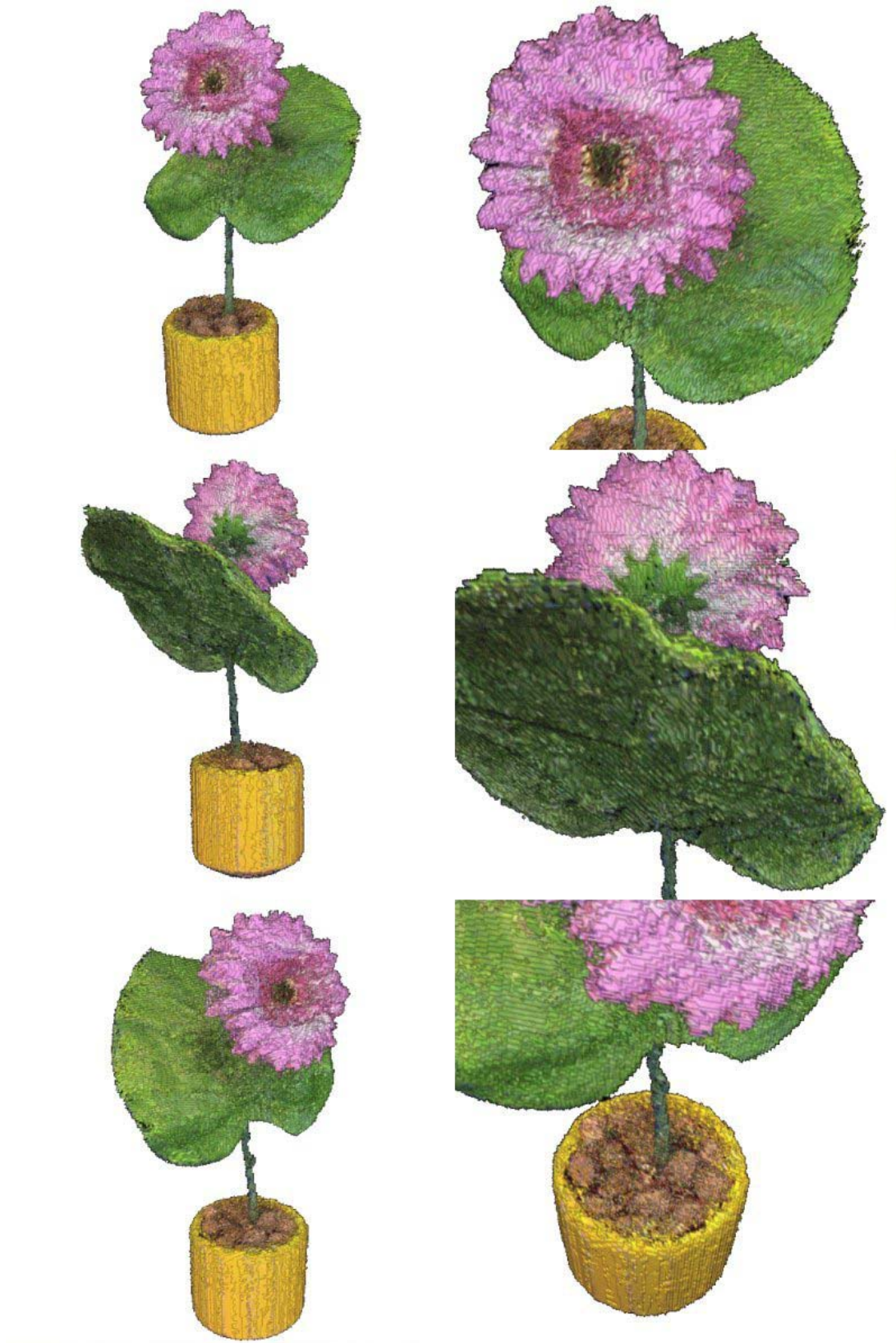


Figure 4-21: Reconstruction of a flower by voxel coloring using 16 images

4.3 Model Refinement by Image Matching

One of the traditional methods to reconstruct 3D surfaces from images is area-based image matching. The results of image matching are point clouds which can easily be represented by integer voxel coordinates. However the reconstruction would not be dense. Hence, for denser reconstructions a hole filling method might be used. Image matching methods can also be applied to voxel-based object reconstruction methods. In Chapter 4.1, a visual hull reconstruction from image silhouettes is examined. Both, silhouette methods and image matching methods have advantages and disadvantages. For example image matching fails on homogenous and occluded areas. However, the visual hull cannot recover concave areas even when large numbers of images are used because this method uses the ray of sights from the projection center through the image silhouettes that are tangential to the surfaces of the object. On the other hand, the visual hull contains the true shape of the object. Volumetric intersection method improves the speed of the reconstruction while image matching is helpful to recover the concave areas, thus these two methods can be used to recover the true shape of the object. In the following chapter first the classical image matching methods will be explained.

4.3.1 Image Matching

Human eye captures light from two different positions in the left and right eye. The displacement of the 3D feature in the two retinal images is called disparity. Disparity is computed by matching corresponding points in two images. One of the most fundamental problems in photogrammetry is to find conjugate points automatically in two or more images. The goal of image matching is to compute 3D location of the matched points in object space.

There are two image matching techniques: area-based and feature based matching. In area based matching, gray levels of image patches are compared using correlation or least squared techniques. In feature based matching edges or other features are compared to determine conjugate features. The similarity like shape, strength of edges is measured by a cost function. We use correlation in order to find conjugate points in two images because the objects might have natural textures without sharp edges.

Before color image matching is discussed, basic single channel image matching is introduced in the following chapters.

4.3.1.1 Correlation

Correlation techniques have been investigated in photogrammetry since fifties. The main idea is to measure similarity between two image regions. Various names can be found in literature; however we will refer to the templates as master and slave, describing a very clear relationship. We define a position in the master image, create a master template around it, and search for this template in the second, the slave image. While the master template remains fixed and constant, the slave template will be moved within a limited search area, until a maximum similarity is found. Figure 4-22 depicts these relations:

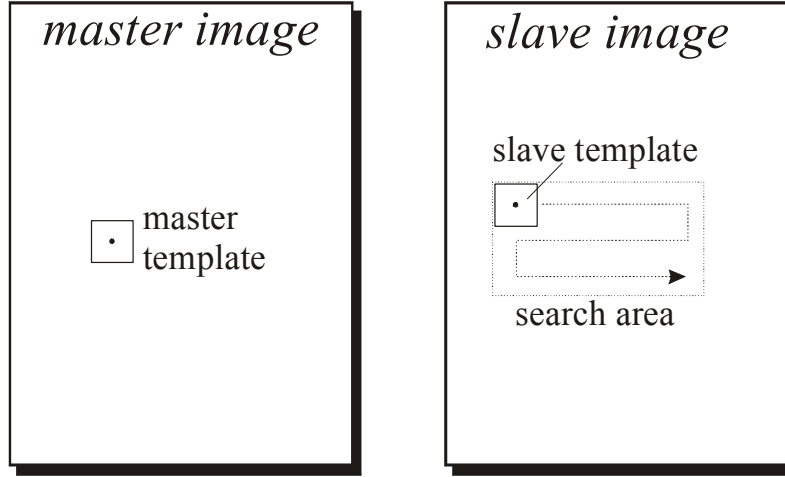


Figure 4-22: Area-based template matching

Correlation is based upon texture information, so in homogeneous regions a correlation coefficient will give a false result, since it always returns a high correlation factor. Consequently, it is not wise to perform cross correlation in homogeneous regions. Moreover, due to occlusions a similar region might not exist and correspondences are hard to find when base line increases.

4.3.1.2 Normalized Cross Correlation

The similarity is computed by a correlation factor. The result of the normalized cross correlation ranges from -1 to +1. The value 1 is obtained if the master and slave template are identical. If there is no correlation between image patches then the cross correlation factor is 0. And if the two templates are identically inverse, the correlation is -1. So with this strictly defined range of the result we have an absolute measure of similarity, i.e. we are not only able to tell that template 'A' is more similar to the master template than 'B', but we can also tell how similar it is.

Differences in brightness and contrast of the image pairs are taken into consideration by observing the differences of each pixel to the templates mean value. The following computer optimized equation can be used to correlate the corresponding pixels. It is optimized in a way that, mean value and variance can be computed in one loop only.

$$r = \frac{\sigma_{g_1 g_2}}{\sqrt{\sigma_{g_1}^2 \cdot \sigma_{g_2}^2}} = \frac{\sum (g_1 - \bar{g}_1) \cdot (g_2 - \bar{g}_2)}{\sqrt{\sum (g_1 - \bar{g}_1)^2 \cdot \sum (g_2 - \bar{g}_2)^2}} = \frac{\sum g_1 \cdot g_2 - n \cdot \bar{g}_1 \cdot \bar{g}_2}{\sqrt{(\sum g_1^2 - n \cdot \bar{g}_1^2) \cdot (\sum g_2^2 - n \cdot \bar{g}_2^2)}} \quad (4.3)$$

where:

r : correlation coefficient

g_1, g_2 : densities (gray values) in master and slave template

\bar{g}_1, \bar{g}_2 : Arithmetic means of the densities of the master and slave template

n : number of samples

The normalized cross correlation is the most common similarity calculation in photogrammetry. Some other approaches will be introduced and evaluated in chapter 4.3.

4.3.1.3 Rank Correlation

The idea of rank correlation is to further improve the validity of the correlation result. In the normalized cross correlation as described earlier, the input observations are of unknown range. Depending of the images contrast, they can range from anything up to 255. Rank correlation does not use the image's gray values directly; instead it uses their rank in a sorted list. Finally, the observations are part of a very specific set of numbers, consisting of $0 \dots n$. With this clearly defined set of numbers, which always contain the same numbers in the master and slave template, it is hoped to better apply statistical tests to the result, in order to check the similarities significance. The calculation of the rank correlation is identical to normalized cross correlation, only with different input values, however, the algorithm requires a sorted list, and a sorting algorithm is always time consuming. So the rank correlation is another option, but it will not be focused any further.

4.3.1.4 Difference Correlation

If we are only interested in the most similar position, and not the quality of similarity, we may consider the pure differences in the template pixels. Hence, the equation may look like this:

$$r_D = \frac{\sqrt{\sum (g_1 - g_2)^2}}{n} \quad (4.4)$$

or,

$$r_D = \frac{1}{n} \cdot \sum Abs(g_1 - g_2) \quad (4.5)$$

In opposition to cross correlation we are here looking for the smallest result as highest similarity. The range of the result is however unknown and it strongly depends on both input templates. So we can tell the highest similarity, but we can not assess this value any further. For example if one image is globally darker and lower in contrast than its partner, the results will be quite high. Nevertheless, the smallest value still refers to the best match. If the similarity needs to be evaluated, one could find the best match - or a number of best matches - with this difference operator, and evaluate those with the cross correlation, applying a sensible threshold.

However, since we know the value range of the input data, we can normalize the result to a range of 0 (different) to 1 (equal), in the following way:

$$r'_D = 1 - \frac{r_D}{(g_{\max} - g_{\min})} \quad (4.6)$$

This approach has the clear advantage of performance speed, obvious when compared to the equation for cross correlation.

4.3.1.5 Least Squares Matching (LSM)

The previous algorithms can determine a best match at the accuracy of one pixel. If we are interested in a better accuracy, we will somehow have to consider subpixel information. Furthermore, any of the previous approaches compares two templates, directly grabbed from the input images. We want to use area based matching as a means to find corresponding points in two images for three dimensional reconstruction. Here we have two contradicting requirements. For the spatial reconstruction it is better to have lines intersecting with a large angle, i.e. images that converge with a large angle. For the image matching, it is better to have most similar images, i.e. images taken from very close positions. It will not be discussed in this chapter how this contradiction can be solved, but it is important to stress out that the images are different; therefore, the search templates are different. And they will be distorted further, due to perspective projections. The estimation of these distortions and therefore improved results is the goal of least squares matching.

Least squares matching was introduced in eighties by [ACKERMANN, 1984] and [GRUEN, 1985] and least-squares matching in object space was first mentioned by [WROBEL, 1987] and [EBNER, 1987]. Just as cross correlation, LSM is based on the similarity of gray values. The idea of least squares matching is to minimize the gray level differences between the master and slave template. This is done by the adjustment process where the position and the shape of the slave template are determined. The position and the shape of the slave template are changed until the gray level differences reach a minimum so that all pixels in the window become conjugate with the master template.

Due to surface slope, position and depth differences of the cameras, the conjugate images might have geometrical distortions. Besides, illumination and reflection might cause radiometric differences in the images. Moreover, noise and distortions of sensor might affect geometric and radiometric correspondence of the conjugate images. Therefore cross correlation alone can not solve these problems and it is not enough for the reconstruction of 3D objects. However, least squares matching is suitable because it is the most accurate image matching technique. LSM is mostly used after image correlation as an improvement, because the accuracy of LSM is dependant on the previous approximation. The approximations should be known within the accuracy of a few pixels.

The relation between the master and the slave template can be approximated by a transformation, for example the affine transformation. If there were no radiometric errors and the transformation parameters were known exactly, the gray levels of template and search images can be transformed identically.

$$g_1(i, j) = g_2[T_G(i, j)] = g_2(x, y) \quad (4.7)$$

where:

$g_1(i, j)$: original master template

$g_2(x, y)$: transformed slave template

T_G : geometric transformation

However, due to radiometric errors and because we do not know transformation parameters exactly there will be gray levels differences between conjugate images. The idea of LSM is to estimate transformation parameters from gray level differences by a least squares adjustment.

A radiometric transformation of the matching window is described in order to compensate the differences between brightness and contrast.

$$T_R[g(i, j)] = r_0 + r_I \cdot g(i, j) \quad (4.8)$$

where:

r_0 : radiometric brightness shift

r_I : radiometric contrast stretching

This transformation will adjust the matching window radiometrically to the template. Radiometric transformation can be introduced during the least squares process. However it is generally adjusted in a pre-process.

Affine transformation models with six parameters are considered to sufficiently approximate projective distortions. It will be applied as follows:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t_0 \\ t_3 \end{pmatrix} + \begin{pmatrix} t_1 & t_2 \\ t_4 & t_5 \end{pmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} \quad (4.9)$$

The affine transformation describes two translations, two scaling, one rotation and a skewing. With the geometric transformation we can write the following generic observation equation:

$$v_g(i, j) = g_2(T(i, j)) - g_1(i, j) = g_2(x, y) - g_1(i, j) \quad (4.10)$$

where:

T : geometric affine transformation

$v_g(i, j)$: residuals of gray level differences between master and transformed slave template

In the following, $g(x, y)$ will refer to the transformed gray values $g_2(i, j)$ of the slave template, with rational coordinates. The linearization of $g(x, y)$ with respect to the transformation parameters:

$$g(x, y) \approx g^0(x, y) + \frac{\partial g(x, y)}{\partial T_x} \left[\frac{\partial T_x}{\partial t_0} \Delta t_0 + \frac{\partial T_x}{\partial t_1} \Delta t_1 + \frac{\partial T_x}{\partial t_2} \Delta t_2 \right] + \frac{\partial g(x, y)}{\partial T_y} \left[\frac{\partial T_y}{\partial t_0} \Delta t_3 + \frac{\partial T_y}{\partial t_1} \Delta t_4 + \frac{\partial T_y}{\partial t_2} \Delta t_5 \right] \quad (4.11)$$

and the partial derivatives:

$$\begin{aligned}
 \frac{\partial g(x, y)}{\partial T_x} &= g_x & \frac{\partial g(x, y)}{\partial T_y} &= g_y \\
 \frac{\partial T_x}{\partial t_0} &= 1 & \frac{\partial T_x}{\partial t_3} &= 1 \\
 \frac{\partial T_x}{\partial t_1} &= 1 & \frac{\partial T_y}{\partial t_4} &= x \\
 \frac{\partial T_x}{\partial t_2} &= y & \frac{\partial T_x}{\partial t_5} &= y \\
 g^0(x, y) &= g_2(i, j) & l &= g_l(i, j) - g_2(i, j)
 \end{aligned} \tag{4.12}$$

with:

g_x : gradient in x direction

g_y : gradient in y direction

pixel	Δt_0	Δt_1	Δt_2	Δt_3	Δt_4	Δt_5	const
1, 1	g_{x1}	$g_{x1} \cdot x_1$	$g_{x1} \cdot y_1$	g_{y1}	$g_{y1} \cdot x_1$	$g_{y1} \cdot y_1$	$g_l(1, l) - g_2(1, l)$
2, 1	g_{x2}	$g_{x1} \cdot x_1$	$g_{x2} \cdot y_1$	g_{y1}	$g_{y1} \cdot x_2$	$g_{y1} \cdot y_1$	$g_l(2, l) - g_2(2, l)$
...							
n, m	g_{xn}	$g_{xn} \cdot x_n$	$g_{xn} \cdot y_m$	g_{ym}	$g_{ym} \cdot x_n$	$g_{ym} \cdot y_m$	$g_l(n, m) - g_2(n, m)$

The observation equations, the solution and the estimated variance of unit weight are given as follows:

$$v = A\hat{t} + l \quad : \text{observation equations} \tag{4.13}$$

$$\hat{t} = (A^T P A)^{-1} A^T P l \quad : \text{solution vector} \tag{4.14}$$

$$\hat{\sigma}^2 = \frac{v^T \cdot P \cdot v}{n \cdot m - 6} \quad : \text{variance factor} \tag{4.15}$$

Since least squares matching is a non linear problem, the solution is found iteratively. The first iteration begins with the approximate location of the slave template (R_s, C_s). The coefficients of the design matrix are computed with the coefficients of the table and the transformation parameters Δt_1 .

Figure 4-23 finally describes a possible effect of least squares matching on the slave template, resulting in a sub-pixel correspondence.

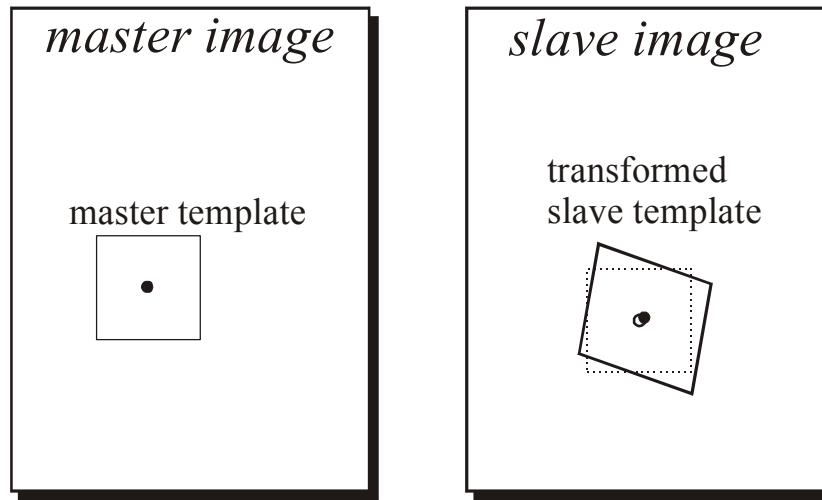


Figure 4-23: Transformed slave template after least squares matching

4.3.1.6 Epipolar Geometry

The search for a best match within a limited search area is still a time consuming process, since the slave template changes with every new position. Here we can use epipolar constraints to further limit the search area. Instead of considering the entire image, or a rectangular window region as a search space, we can reduce it by using epipolar geometry. Epipolar lines are efficient constraint for conjugate entities. If a matching entity in one image is chosen, the search region in the second image can be narrowed down to a broad line since the image orientation is known. Although an object point corresponds to exactly one image point, the reversion is not valid. Instead, every object point along a line of sight may be projected into the same pixel. Only by the use of a second image, we are able to derive a unique point in space. But we can also use this line of sight to limit the positions in the second image, where the object point may appear. This situation is illustrated in Figure 4-24 and it is known as the epipolar line.

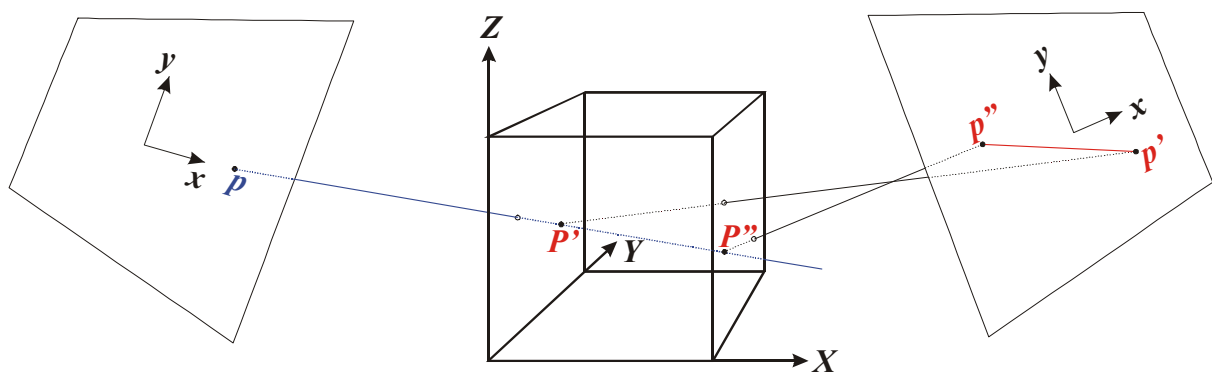


Figure 4-24: Epipolar geometry

We can use this line to limit the search area for image matching, since it is a very time consuming process.

4.3.2 Color Image Matching

Area-based image matching using single channel is a well-known technique in photogrammetry. However one of the most important information to identify objects is color. Color image matching is not new, especially in remote sensing some authors used image matching algorithms with 3 or more channels. Hahn and Brenner have investigated quality differences between area-based image matching of multi-channel images and images with one channel. [HAHN, 1995]. Also refer to Chapter 4.3.3.1 for polychromatic block matching [KOSCHAN-I, 1993].

When color images are available, we should make use of color information to further improve our matching results. As shown in Chapter 4.3.2.2, color can be crucial and when ignored can lead to false results. Now, the question is how the color information should be considered. Normally, the color is directly available as an RGB triple, whether from a 24 bit true color image or an 8 bit image with color palette. This is the basic color information at hand. If desired, this can be transformed into other color spaces, like CMYK or IHS (see chapter 4.1.2).

In the following sections a few color image matching approaches are introduced and evaluated. These color image matching algorithms are used in Chapter 5.

4.3.2.1 Color Theory

When we see color on the monitor, we see a combination of red, green and blue pixels of different intensity. So color can be expressed by an additive mixing of red, green and blue (RGB). This means to add the single channels together. The opposite, which we will encounter with printers, is the subtractive model, consisting of yellow, magenta and cyan. Here, we can think of the single channels, being removed from the basic white (paper). The mixture of two basic colors of one model gives a basic color of the other:

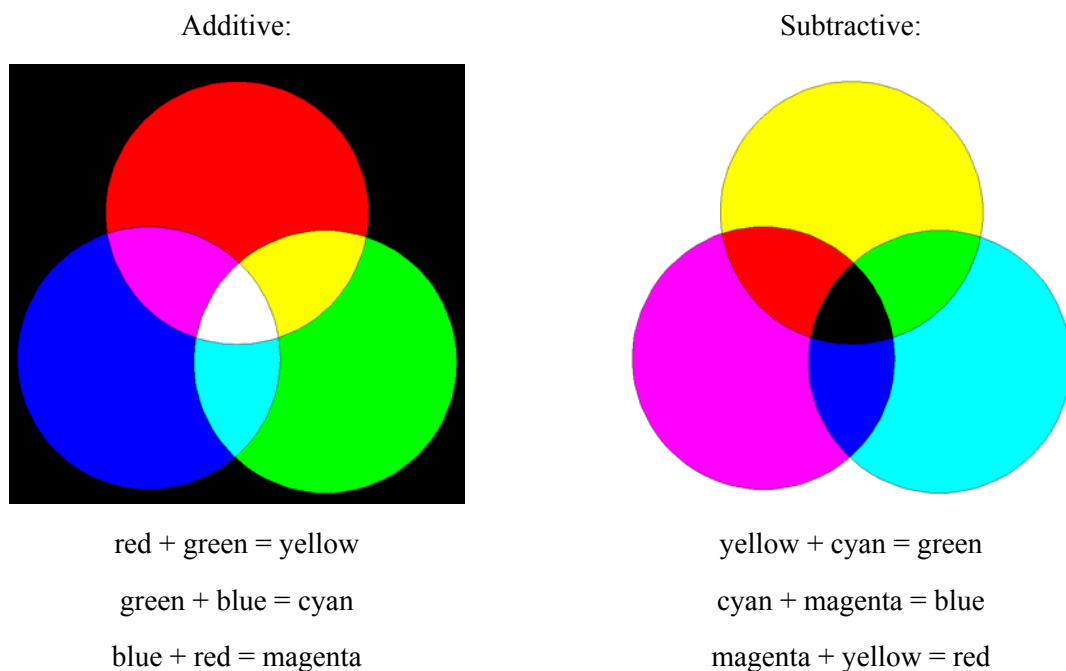


Figure 4-25: Basic color theory

Another important model is the HSL-model. This triple contains information about the color value (Hue), its intensity (Lightness), and the degree of saturation. The color value is defined as a circular

sequence, starting with red, passing yellow, green, cyan, blue, magenta and ending again with red. The value is defined as the circular angle 0° to 359° ($0^\circ = 360^\circ = \text{red}$). The intensity defines the lightness of the color, where it is black with zero intensity. The color's purity is described by the saturation, i.e. the less saturated, the more the color fades to gray. In the cases of zero saturation or zero lightness, the color value is undefined – black has no color.

We can use the HSL-model to reveal or enhance some phenomenon, which are not visible in the normal RGB-color space. In particular we make use of the HSL-model to perform color segmentation (see section). Figure 4-26 shows how we can distinguish a single color, which is disturbed by lighting conditions, leading to differences in saturation and lightness. However, we can see that in the RGB- and even in the gray scaled image, the turntable on the bottom can clearly be separated from the background. In the hue image, there is no visible difference because the color value is the same, lightness and saturation are suppressed.

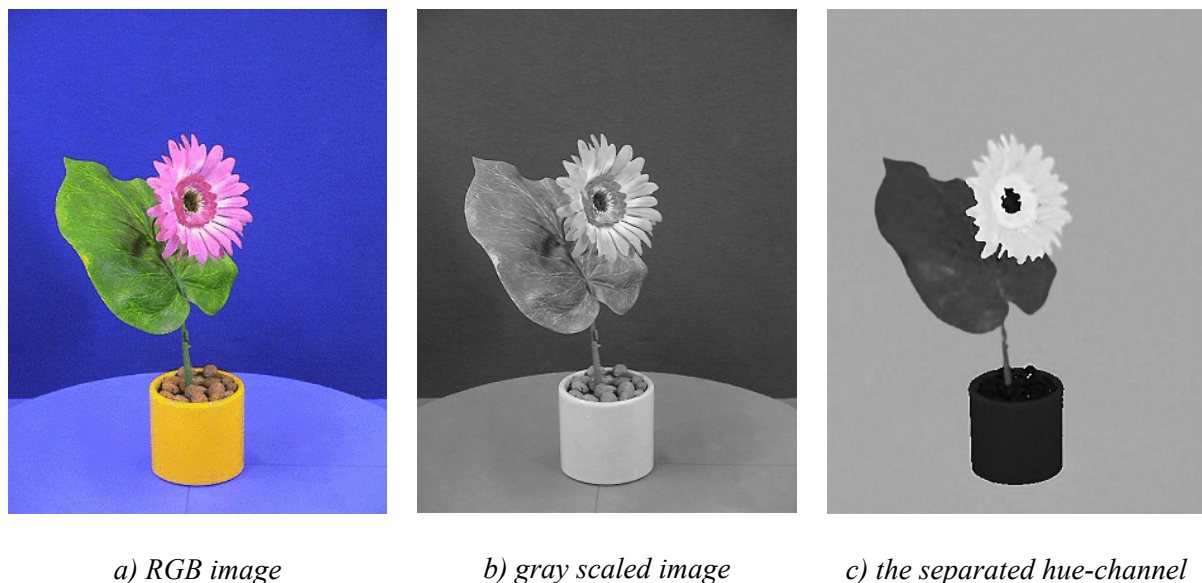


Figure 4-26: An image showing the properties of the hue-channel

4.3.2.2 Considering Color

We introduced some efficient approaches to search for image correspondences. But we did not take into consideration that we do have color information available. The previous equations only require simple gray values. However, it is obvious that an RGB-triplet contains more information than a single gray value (refer to Figure 4-26).

Independent of a specific algorithm, the color information can be considered in various ways. In general we might classify the possible approaches into two groups. First, we can throw all color channels into one equation and get one correlation factor as a result. Second, we calculate a correlation factor for each channel separately.

To the second approach, we can add additional information, like the variance in the single channels and apply an according weight to them. This would enhance or suppress the channels according to their texture. Figure 4-27 (a) shows a blue cross on a red background (it will not be visible in a black and white copy). The colors were intentionally carefully chosen, so that in the gray scaled image the cross would disappear. But the separated red and blue channels reveal the information very clearly. This example stresses out the how important the consideration of color might be.

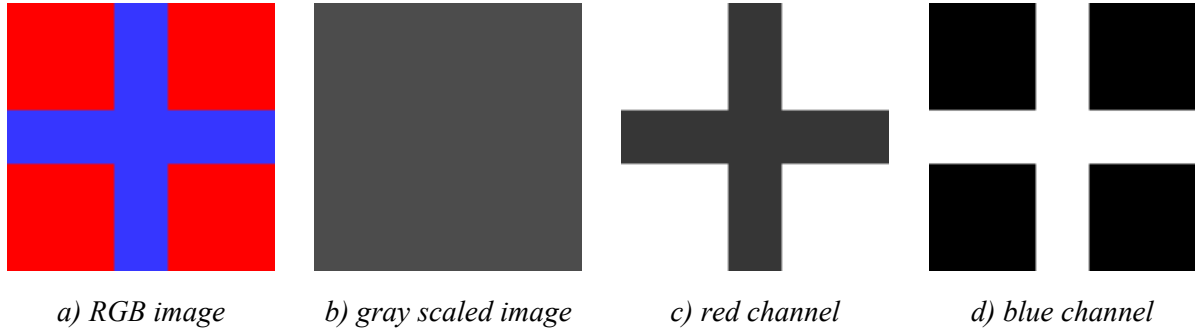


Figure 4-27: The importance of color

Furthermore, we can consider different color spaces, like RGB or HLS (see section). Which approach is the most sensible, will be described later. We will then go into detail and present experimental results.

4.3.2.3 Single Vector Correlation

As several tests have shown that the simplest and at the same time the most reliable solution is the correlation of all the input data in one vector. Assuming the normal case of having three channels of color (RGB, CMY, IHS), we will now have three times as many observations as in gray images:

$$n = \text{width} \cdot \text{height} \cdot 3$$

The idea is to put all these observation in one vector, resulting in one single correlation coefficient:

$$r = \frac{\sum_{ch} \sum_x \sum_y (g_1 - \bar{g}_1) \cdot (g_2 - \bar{g}_2)}{\sqrt{\sum_{ch} \sum_x \sum_y (g_1 - \bar{g}_1)^2 \cdot \sum_{ch} \sum_x \sum_y (g_2 - \bar{g}_2)^2}} \quad (4.16)$$

4.3.2.4 Separate Channel Mean Value

A small alteration of the above takes the channels separately into consideration, so that:

$$r^{red} = \frac{\sum_x \sum_y (g_1^{red} - \bar{g}_1^{red}) \cdot (g_2^{red} - \bar{g}_2^{red})}{\sqrt{\sum_x \sum_y (g_1^{red} - \bar{g}_1^{red})^2 \cdot \sum_x \sum_y (g_2^{red} - \bar{g}_2^{red})^2}}, \quad (4.17)$$

r^{green} and r^{blue} accordingly.

This way we will get three results, for each channel separately. The most obvious way is to derive the mean value.

$$r = \frac{r^{red} + r^{green} + r^{blue}}{3} \quad (4.18)$$

4.3.2.5 Weighted Channel Mean Value

As with the above, we will again have three separate correlation coefficients. But now we do not simply calculate the mean value. Instead, we would like to take differing texture properties within the separate color channels into consideration. That is why we build a weighted mean value, according to:

$$r = \frac{w_1 \cdot r_{red} + w_2 \cdot r_{green} + w_3 \cdot r_{blue}}{w_1 + w_2 + w_3} \quad (4.19)$$

This way, we would like to support channels with a stronger texture, i.e. whose variance is greater, at the same time suppressing homogeneous channels. So we will have to base this weight upon the variance in the single channels:

$$w_{ch} = \sqrt{\frac{\sum_x \sum_y (g_{xy}^{ch} - \bar{g})^2}{m \cdot n - 1}} \quad (4.20)$$

It is sensible to calculate this way upon the values of the master template, since this is the patch of reference.

4.3.2.6 Difference Correlation

For this method, we can apply the same approaches, as for the normalized cross correlation. Hence, we can calculate one value by summing up all the differences over the three channels (as will be done in the future), or we can derive three separate values and calculate a weighted and a non-weighted mean value.

As mentioned, we will only consider the single vector variant for this approach:

$$r_D = 1 - \frac{\sum_{ch} \sum_x \sum_y Abs(g_1 - g_2)}{3 \cdot width \cdot height \cdot (g_{\max} - g_{\min})} \quad (4.21)$$

4.3.2.7 RGB-Correlation

Another approach is to regard the RGB triple itself as an observation vector and calculate the correlation factor with a second one. This way the equality of color can be expressed.

$$r = \frac{R_1 \cdot R_2 + G_1 \cdot G_2 + B_1 \cdot B_2 - \frac{(R_1 + G_1 + B_1) \cdot (R_2 + G_2 + B_2)}{3}}{\sqrt{[(R_1^2 + G_1^2 + B_1^2) - \frac{(R_1 + G_1 + B_1)^2}{3}] \cdot [(R_2^2 + G_2^2 + B_2^2) - \frac{(R_2 + G_2 + B_2)^2}{3}]}} \quad (4.22)$$

The summation of these factors over a surrounding area about the center pixel will give the similarity measure of an image patch.

4.3.2.8 Evaluation

The introduced approaches have been tested on three generated images (see Figure 4-28). The basic image shows a target (in this case a simple cross) in its basic colors (red, green, blue, cyan, magenta, yellow, white and black) on a slightly lighter background varying over the same basic colors. To the basic image 10% noise were added, and a third was created, by adding 75% noise.

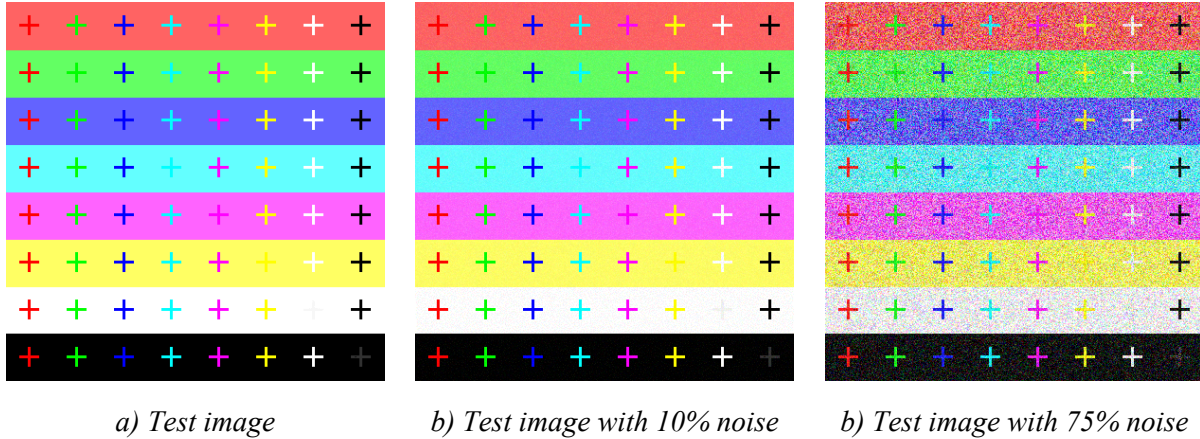


Figure 4-28: Test image to evaluate color image matching

The testing was done by taking each of the 64 targets as a master template and performing a matching over the complete image itself, so that there should be at least one point of absolute similarity.

In case of the perfect non-noise image however, we observed cases where there were no similarities at all or several points with 100% similarity. This is due to the absolute homogeneity in single channels, so that in the correlation equation the denominator becomes zero and delivers a result that cannot be interpreted. For example the separate channel approach, with the red cross on a blue background, delivers 100% similarities for the red cross on green, blue and cyan backgrounds. In most cases this methods delivers ambiguous results.

The RGB-correlation does not recognize targets on a lighter background of the same color, but works fine otherwise. So it should not be applied on regions which do not vary in their color value.

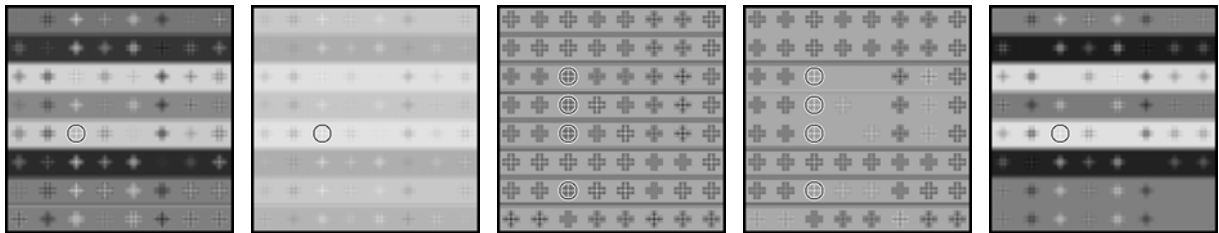
Only the single vector correlation and the difference correlation delivered reliable results. The only exception was that the single vector correlation could not distinguish the very light gray cross on white

background from the black cross on white background. Since by definition this method ignores differences in brightness and contrast, this is a logical consequence.

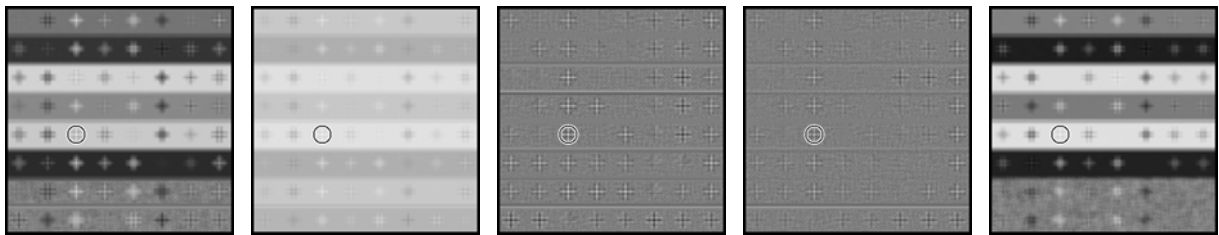
However, in the presence of noise, all the approaches deliver reliable results. In these artificial images if there was more noise, they would even become better.

Summarizing, the most reliable results were delivered by the single vector and the difference correlation, since they worked best in all three test images. They are also the simplest equations so that faster performance can be expected, especially the difference correlation.

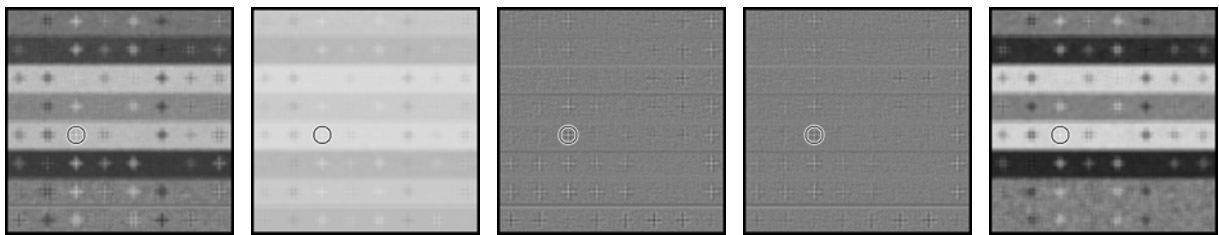
Finally, Figure 4-29 shows the visualized results of the different matching algorithms. Note the ambiguity in the separate channel approaches in the upper row.



Results on a noise-free image (from left to right: single vector, difference, weighted RGB, non-weighted RGB, RGB correlation). Best matches are marked.



Results on an image with 10% noise



Results on an image with 75% noise

Figure 4-29: Visualizations of the color matching algorithms

4.3.3 A Visual Hull Refinement Algorithm Using Block-Matching

4.3.3.1 Polychromatic Block Matching

As mentioned previously, the visual hull contains more voxels than the true shape of the object should contain due to concave areas on the object's surface. If the surface contains enough texture, the missing object concavities can be detected. The modeling error of the visual hull can be minimized using a block matching approach of Koschan [KOSCHAN-I, 1993]. Although block matching is not the most reliable technique, it is less time consuming than standard image matching. In Chapter 4.3, color image matching is described as a more reliable technique to refine the object's approximate model.

The mathematical background of block matching algorithm is similar to standard image matching. The main idea of Block Matching is a similarity check between two equal sized blocks in two images (area-based stereo). The main difference from standard image matching is; instead of shifting the master patch pixel by pixel in the master image, it is shifted block by block. The mean square error MSE between the pixel values inside the blocks defines a measure for the similarity. However, the slave patch is searched by shifting it pixel-wise inside a search area that is defined by the epipolar geometry.

$$D = \min_{|\Delta| \leq d_{MAX}} \{MSE_{COLOR}(x, y, \Delta)\} \quad (4.23)$$

The block disparities (parallaxes) D are median filtered to avoid outliers and a rough depth map is generated. Then a dense depth map is generated using a pixel selection technique. In this technique for each pixel in question the best match is searched along the epipolar line shifting the block according to its neighbor disparities.



Figure 4-30: Computed stereo depth maps of the flower (left, middle) and the combination of three neighbored views (right)

4.3.3.2 Hierarchical Matching Using Image Pyramids

The enhanced algorithm [KOSCHAN-II, 1997] is more robust and shows better results. The hierarchical approach can be implemented very efficiently in parallel to achieve high speed execution. Furthermore, the combination of three views improves the quality of the matching results (see Figure 4-30). The correspondence analysis of a reference image with the neighbored images eliminates most of the artifacts.

4.3.3.3 Integration of Volume Data and Depth Maps

The combination of the voxel data and the depth maps is realized in the voted-based carving stage. During the voting-based voxel carving stage the voxels have been assigned votes representing how many images they have been projected into background pixels. Each voxel get additional votes according to the information provided by the depth maps. The results (see Figure 4-31) show the importance of precise calibration data and additional constraints are necessary to reduce the number of outliers.



Figure 4-31: The enhanced concave head of the flower using block matching (right)

Chapter 5 : Proposed Volumetric Reconstruction Algorithm

5.1 Utilities for Volumetric Processing

5.1.1 Line Tracing

For many tasks it is necessary to process each voxel along a specific line. Visibility information can be recovered this way, as will be described in the next chapter. In texture mapping, we would make use of a line tracing, and if we were to simulate real world conditions, like lighting, shadow casting and atmospheric effects, the appropriate technique is ray-tracing, hence voxel-line tracing.

As mentioned in chapter 2.2.4, there are different degrees of neighborhood, which influence the resulting line. In the following, we will introduce two algorithms to construct a 6-connected and an 18-connected line (also refer to Figure 2-3).

When we are constructing a line with a start and an end voxel, we want to visit each voxel in between exactly once. While the voxel is defined by integer coordinates, there might be occasions, where rational coordinates are advantageous. But still, for the sake of performance, each voxel should only be visited once.

Figure 5-1 shows the difference between a 6-connected and an 18-connected line (for simplicity in 2D).

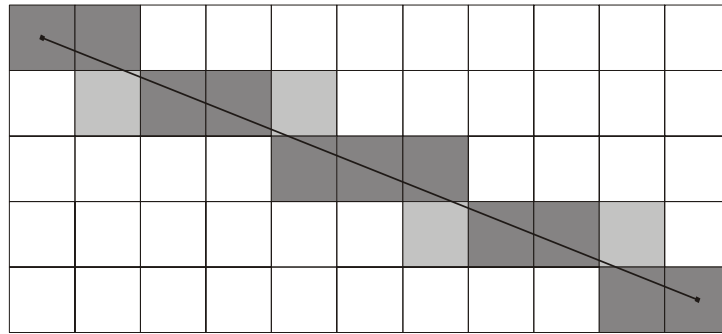


Figure 5-1: 18-connected (dark pixels) and 6-connected line (light+dark pixels) in 2D

5.1.1.1 18-Connected Line

The construction of the 18-connected line is quite simple, as it is shown in the following pseudo-code. Although it is a simpler implementation than the 6-connected line, it produces rational coordinates, which can easily be rounded to the closest voxel (nearest neighbor).

The basic idea is to find that direction of $(\Delta X, \Delta Y, \Delta Z)$, in which the line has the greatest length. According to this direction, we will normalize the steps for each direction, so that:

$$\Delta x = \frac{\Delta X}{\Delta_{\max}}$$

$$\Delta y = \frac{\Delta Y}{\Delta_{\max}}$$

$$\Delta z = \frac{\Delta Z}{\Delta_{\max}}$$

For the leading direction, the stepsize will become one, which will ensure the visiting of each voxel. The other stepsizes are rational numbers between zero (including) and one (including). The following pseudo-code should explain the functionality. It is furthermore depicted in Figure 5-2.

```
function linetracing_18connected (P0=start, P1=end)
{
    let dx = P1.x - P0.x
    let dy = P1.y - P0.y
    let dz = P1.z - P0.z

    let maxlength = MAX[ABS(dx), ABS(dy), ABS(dz)]

    let sx = dx / maxlength
    let sy = dy / maxlength
    let sz = dz / maxlength

    let P = P0

    for (0 ... maxlength)
    {
        call operator (P)

        P.x = P.x + sx
        P.y = P.y + sy
        P.z = P.z + sz
    }
}
```

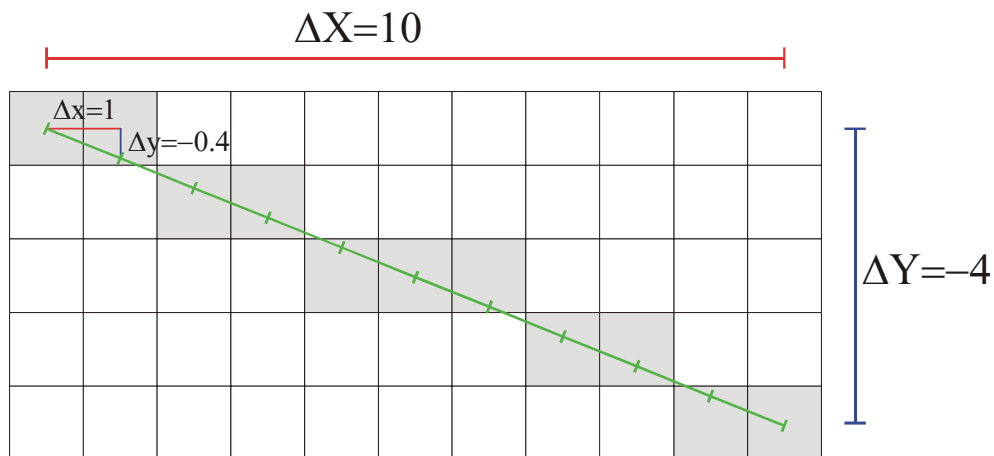


Figure 5-2: The basic idea of an 18-connected line tracing

5.1.1.2 6-Connected Line

The construction of a 6-connected line is a little bit more complicated than of an 18-connected line. We use a stepsize in each direction in this algorithm as well. But in this case, the stepsize describes the length of a line section in which it remains inside the same voxel of the particular direction. In Figure 5-3 this is shown as Δx and Δy . A counter variable for each direction will be available, and for each step the smallest counter is increased by its corresponding stepsize.

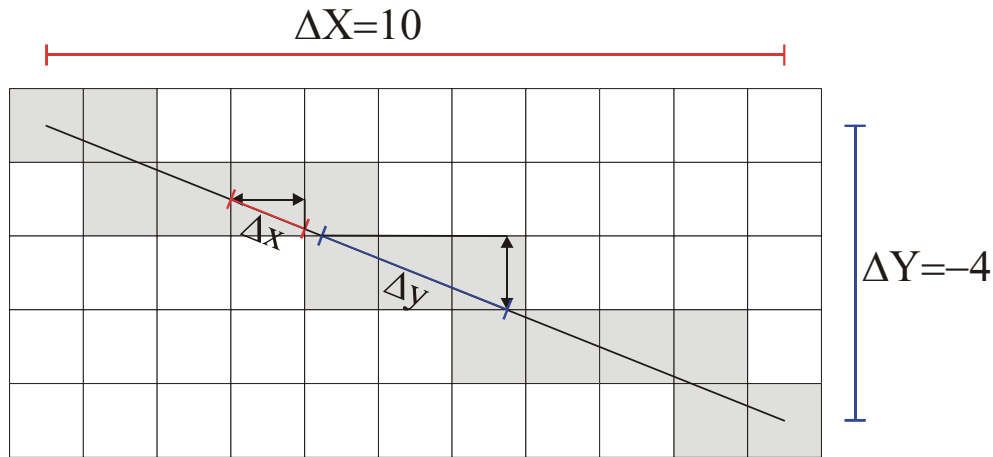


Figure 5-3: 6-connected line tracing

In the example in Figure 5-3, the ray has to travel 1.077 units in order to traverse a voxel in x-direction, but 2.692 units in y-direction. That means that every 2.499 voxels in average we need to change the direction. The pseudo-code should clarify this algorithm. It is introduced in detail by [AMANATIDES, 1987].

```
linetracing_6connected (P0=start, P1=end)
{
    vector    v;
    vector    Dp, dp, l, p;
    double    len, dlen;

    let Dx = P1.x - P0.x
    let Dy = P1.y - P0.y
    let Dz = P1.z - P0.z

    let px = SIGN(Dx)
    let py = SIGN(Dy)
    let pz = SIGN(Dz)

    let length = SQRT(Dx2 + Dy2 + Dz2);

    /* division by zero possible */
    let dx = length / Dx
    let dy = length / Dy
    let dz = length / Dz

    /* Initialize: */
    let lx = dx / 2
    let ly = dy / 2
    let lz = dz / 2

    P = P0;
```

```
    loop
    {
        call operator (P)

        // Break condition:
        if (lx, ly, lz > length)
            break;

        if (lx = MIN[lx, ly, lz])
        {
            P.x = P.x + px;
            lx = lx + dx;
        }

        else if (ly = MIN[lx, ly, lz])
        {
            P.y = P.y + py;
            ly = ly + dy;
        }

        else if (lz = MIN[lx, ly, lz])
        {
            P.z = P.z + pz;
            lz = lz + dz;
        }
    }
}
```

5.1.1.3 Implementation of Line Tracing

Since we have a means of tracing a line voxel by voxel, we can now use this algorithm in several tasks such as in checking if this voxel is within the defined voxel space after all. Hence we have to define an operator which will work on the current voxel. This operator should be able to do arbitrary tasks and return a value to indicate success or failure, continue or break, respectively.

Very briefly, our line tracing algorithm would look as follows:

```
do initial work
for (p=P.start ... P.end)
{
    call operator (p)
    if (break signaled)
        break loop
    else
        increase p
}
do cleanup work (success / failure)
```

The choice of C++ as programming language simplifies the implementation of such operator constructs. As an example, we want to find out that voxel, which is first encountered, when we trace a pixel back into voxel space. The operator will check if an object voxel is encountered, or if the defined voxel space has been left. In both cases the operator should send a break signal, so that the shell algorithm will stop. In the following chapters, some other operators will be introduced, when appropriate.

Now, without going too much into detail about object oriented programming (OOP), we would like to give an idea, how it is accomplished.

A crucial element of OOP is the use of classes (objects) and the consequences that go along with it. For instance, a derived class inherits all data members and methods from its parent classes. In general one can say that a derived class is a specialization of its parent. Since each object knows, from where it is derived (if at all), it can very simply make use of methods, defined in the parent class.

In the following Figure 5-4, we use a simple example to clarify the idea of derivation. The basic class of a family of geometric objects is a point. From this class we derived three more classes, each of which inherits the point coordinates: a circle, a square and a line. Again, from the circle and the square, we derived an ellipse and a rectangle. Each class is a specialization of its parents. So each of these geometric entities has a position, and each child specializes its shape by an additional parameter. The advantage of inheritance is that we do not have to implement a method for each object. While each object would be responsible to draw itself, (apparently a square differs from a circle, although they can be described by the same parameters) we will find some operations, which all object have in common. For instance, if we want to move an entity, it is sufficient to move the base coordinate defined in the point-class. Hence a move-method only needs to be defined in the very base class and will be inherited by all its children.

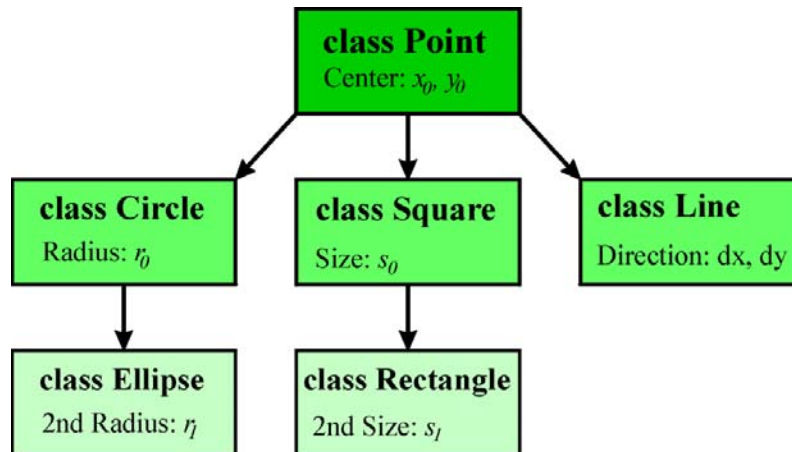


Figure 5-4: The use of classes and inheritance

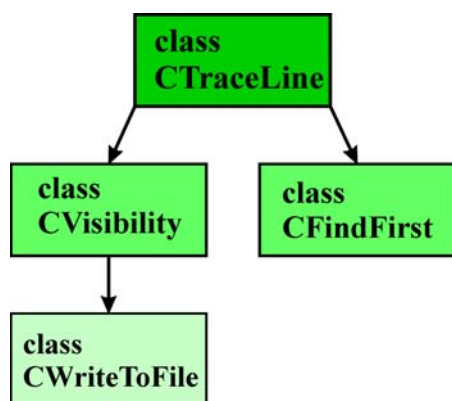
We have seen that a child can easily use the methods of its parents. Sometimes it may be useful to go up the family tree. For example, if we create a mixed list of geometrical entities, it would consist of a number of point-derived objects. There might be occasions, where we want to call a method of an object, but we would rather want the method of the youngest child to be executed first. In terms of C++ this is realized by using *virtual* methods.

Coming back to the line tracing, we have implemented a base class to operate on a voxel. In this base class we defined the three main virtual methods to do:

- An initialization
- A per-voxel operation (returning success or failure)
- A clean-up, depending on success or failure

Since this base class is only serving as an inheritance template, its methods are doing nothing. They need to be defined, in order to make use of virtual method calls.

In specific, our base class is described in Figure 5-5, along with some derived operators.



```

// operator base class for the C++ method:
class CTraceLine
{
public:
    CTraceLine();
    virtual ~CTraceLine();

// methods:
    virtual void    Init(vector& p0, vector& p1);
    virtual void    CleanUp(bool) {};
    virtual bool    operator ()(vector& v);

// data members:
    vector m_ptVoxel;
    vector m_ptStart, m_ptEnd;
};
  
```

Figure 5-5: The operator-class family

Since the programmer can derive his own class, he has full control of its behavior. In Figure 5-5 we have given three example implementations for possible operators. As mentioned above, the base class only serves as a template. As an example, the visibility class might recover visibility information of a

voxel (see also chapter 5.1.4) and the *CWriteToFile*-class uses the method of its parent, but additionally writes the results into a file.

The pseudo code of the actual line tracing algorithm has been introduced in the beginning of this chapter. In particular, its function prototype looks like this:

```
bool trace_line(vector p0, vector p1, CTraceLine& op);
```

The algorithm traces a line according to the two given vectors, where for each voxel the operator ‘op’ is called. Since we are free to derive our own operator, this way of performing a line tracing is highly flexible. Moreover, C++ allows storing arbitrary information in an object. So we cannot only tell, that the line tracing failed (as provided by the boolean return value), but we can also implement means to tell us why and where.

Some operators will be introduced in the following chapters, when appropriate.

5.1.2 Line Segment Limiting

Tracing a line is a process which consumes time, directly related to the number of elements. Hence, if we can narrow the line down to half length for instance, we would save half of the processing time. In the previous chapter we wanted to find the first object voxel, corresponding to a given image point. The line is therefore defined by the pixel and the images projection center. To be more accurate, it begins with the image projection center, faces directly away from the image point and is of unknown length. It would definitely be a waste of time to process the line like this.

It would be right to look for a sensible geometric limitation with two simple preconditions:

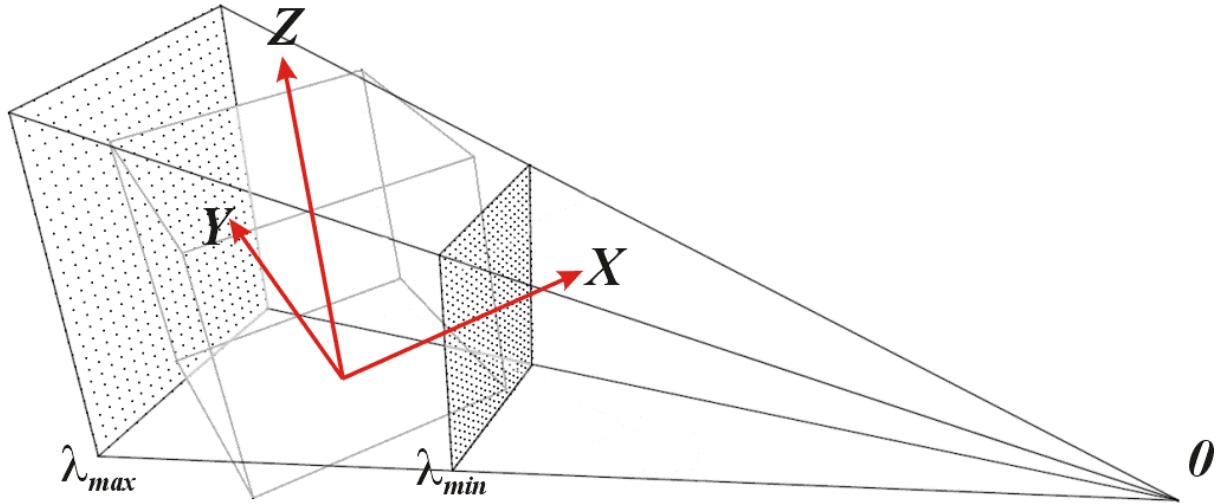
- Each voxel must be covered by the line.
- Not too much empty space should be swept by the line.

With the help of equation (3.4) from chapter 3.1.1, several approaches can be derived. Here is the equation again, as a reminder:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R^{-1} \cdot \lambda \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \\ -c \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}$$

5.1.2.1 Global Minimum/Maximum Limitation

For the first approach, we will choose two values for λ , to define a start- and an end-point for the line. It is based on the assumption, that a λ -factor according to the farthest and the nearest corner of the voxel cube will completely enclose the whole cube. For this definition, all we have to do is to calculate the distance to each of the eight corners, and determine the minimum and maximum of these values. Those two λ -factors will be globally valid for all pixels of one image. Figure 5-6 shows a visualization of this limitation. It might not be the best limitation, but its clear advantage is that it only needs to be derived once per image.

Figure 5-6: Min/Max λ limitation of a line

5.1.2.2 Spherical Limitation

A second limitation can be accomplished, by wrapping a sphere around the cube and calculating the intersection of each ray with the sphere. If there is no or only one intersection point (the tangential), the line can be skipped in the first place. In case of two intersection points, these will define the limitation of the line. The disadvantage is obviously that a limitation needs to be calculated for each image point separately, although it is probably a better approximation. This approach is described in Figure 5-7.

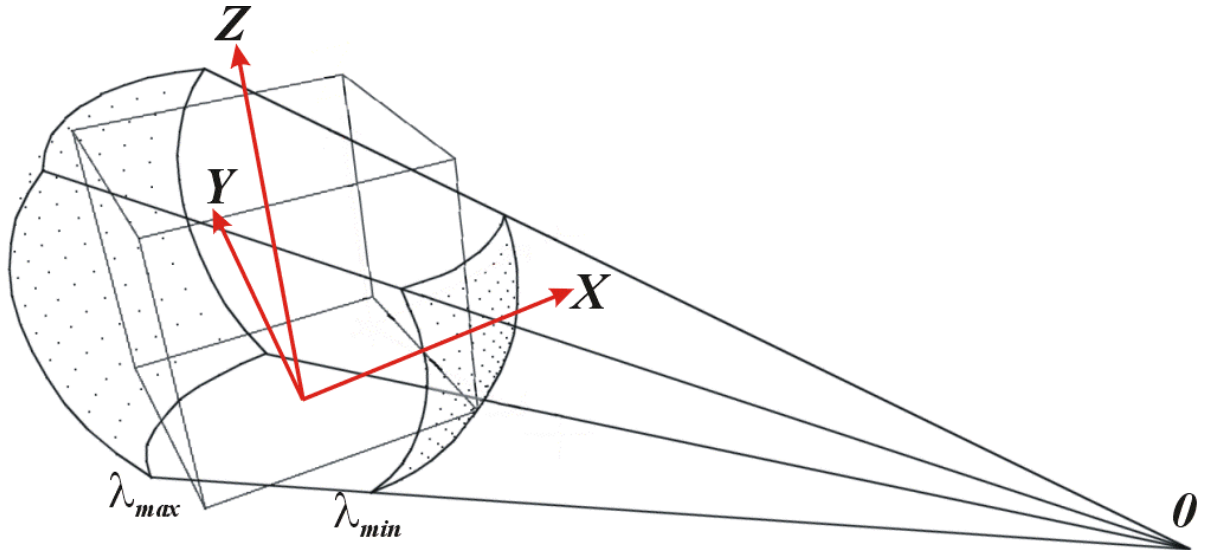


Figure 5-7: Sphere-intersection limitation

The intersection of a line and a sphere is calculated as follows:

Vector equation of a line:
$$P = P_0 + u (P_1 - P_0) \quad (5.1)$$

Equation of a sphere:
$$(x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = r^2 \quad (5.2)$$

Substituting the line into the sphere yields: $a \cdot u^2 + b \cdot u + c = 0$ (5.3)

with: $a = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$ (5.4)

$$b = 2[(x_2 - x_1)(x_1 - x_3) + (y_2 - y_1)(y_1 - y_3) + (z_2 - z_1)(z_1 - z_3)] \quad (5.5)$$

$$c = x_3^2 + y_3^2 + z_3^2 + x_1^2 + y_1^2 + z_1^2 - 2[x_3 x_1 + y_3 y_1 + z_3 z_1] - r^2 \quad (5.6)$$

Solving $a \cdot u^2 + b \cdot u + c = 0$ for u yields:

$$u = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2a} \quad (5.7)$$

The behavior of the square root determines the number of solutions. If the term $b^2 - 4 \cdot a \cdot c$ is negative, there is no solution at all and hence, no intersection. If the term is equal to zero, there is but one solution, which is a tangential line to the sphere. A positive value yields two solutions, the two intersection points we are actually looking for.

5.1.2.3 Cubical Limitation

The most accurate limitation is to determine the intersection of the line with the defined voxel space itself. But that means we have to check the line for intersecting all six faces of the voxel cube. Also in this approach, the subsequent line processing can be ignored if there is no intersection at all. Figure 5-8 describes this approach, where the limitation is individual for each image point, as in the spherical approach.

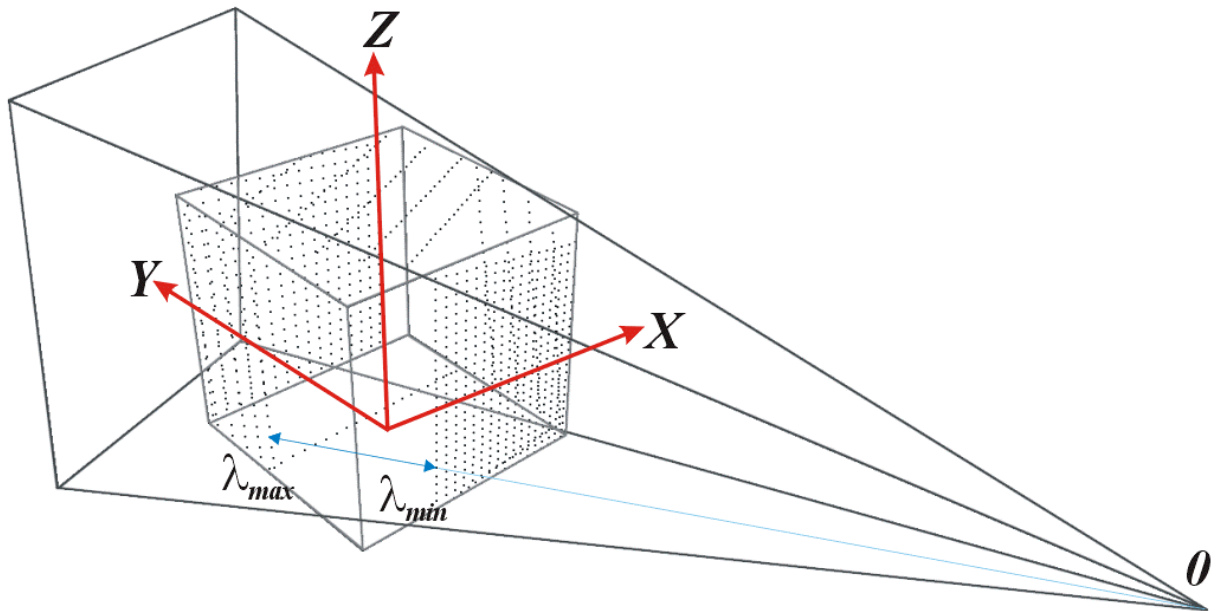


Figure 5-8: Cubical intersection limitation

In order to find the intersections of the line with the cube sides, we first need to find the intersection of the line with the corresponding plane. This is always the case, as long as the plane and the line are not parallel. But now we need to check if the point of intersection lies within the rectangular region of the cube side. Only if this condition is met, we have found one of two possible intersection points.

5.1.3 Line Snapping

In some cases, we have to work with rational coordinates, for the sake of accuracy. But eventually, the line tracing algorithm can only deliver integer coordinates of the voxel center. For instance, if we want to trace the line of a reference image into the cube, and for each voxel we want to determine the projection in another image. This would cause a tremendous inaccuracy if we used integer coordinates, always depending on the voxel size, hence the cube resolution.

In most cases however, the definition of the line, the start- and end point are well known, just the realization of the tracing algorithm may lose the rational part. Here we can use a simple trick to snap the integer point onto the true line. Figure 5-9 (a) shows how the algorithm is projecting the points onto the line.

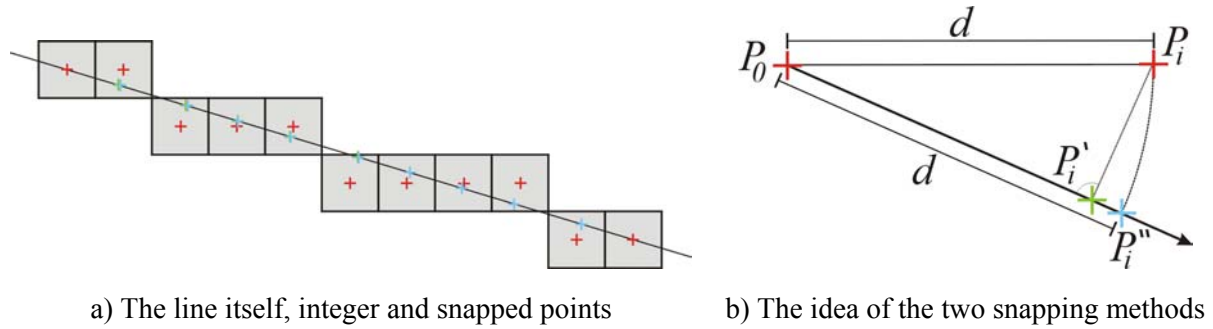


Figure 5-9: Snapping of integer points to the true line

There are two approaches to snap a point to a line, as seen in Figure 5-9(b). In the first approach the point is projected perpendicular into the closest point P' along the line. The second approach simply rotates the point P'' into the line. In the following two chapters, the mathematical background will be given.

5.1.3.1 Perpendicular Projection

The perpendicular projection of a point to a line is looking for the shortest distance between that point and the line. We will name the point to be projected with P_i and the projection on the line P' .

A vector line is given by:

$$P = P_0 + u \cdot (P_1 - P_0) \quad (5.8)$$

The vector describing the shortest distance is perpendicular to the original line, so that:

$$(P_i - P') \cdot (P_1 - P_0) = 0 \quad (5.9)$$

and substituting P' with the equation for the line:

$$(P_i - P_0 - u \cdot (P_1 - P_0)) \cdot (P_1 - P_0) = 0 \quad (5.10)$$

We have to solve this equation for u , in order to find the projected point P' .

$$u = \frac{(P_i - P_0) \cdot (P_1 - P_0)}{(P_1 - P_0)^2} = \frac{(x_i - x_0)(x_1 - x_0) + (y_i - y_0)(y_1 - y_0) + (z_i - z_0)(z_1 - z_0)}{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} \quad (5.11)$$

$$P_i' = P_0 + u \cdot (P_1 - P_0) = P_0 + \frac{(P_i - P_0) \cdot (P_1 - P_0)}{(P_1 - P_0)^2} \cdot (P_1 - P_0) \quad (5.12)$$

5.1.3.2 Point Rotation

The second approach is to rotate the integer point into the point P'' on the line, so that $|P_0 P_i| = |P_0 P''|$. The length of the line is:

$$d_{line} = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} \quad (5.13)$$

and the distance of the integer point to the beginning of the line is given by

$$d = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \quad (5.14)$$

We have to construct a new point on the line with the same distance as the integer point:

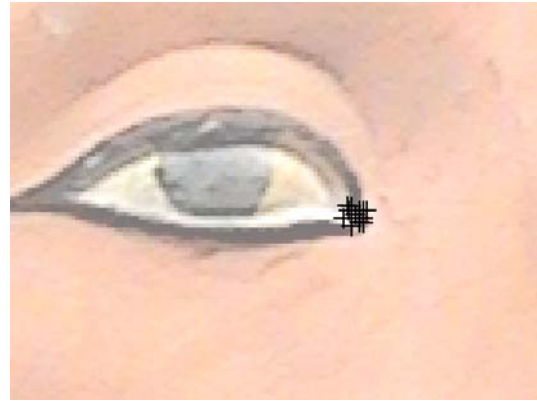
$$P_i'' = P_0 + \frac{d}{d_{line}} \cdot (P_1 - P_0) = P_0 + (P_1 - P_0) \cdot \sqrt{\frac{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2}{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}} \quad (5.15)$$

As can be seen from Figure 5-9 (b) the resulting points differ especially in the beginning of the line, when the enclosing angle is larger. It becomes less significant, when the line advances further away from its start point. Since there is no real difference, the choice can be made upon performance. When we compare the two algorithms, we can see that they are almost the same, except for one square root, by which the second approach is more computationally expensive.

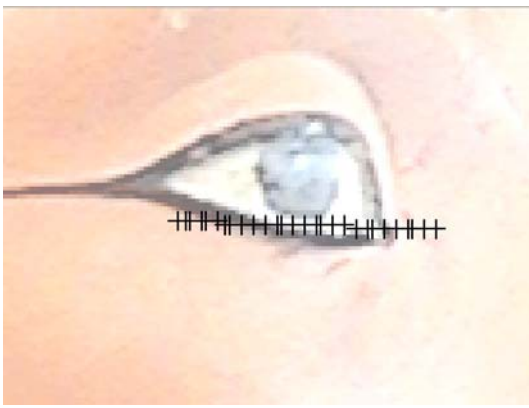
In Figure 5-10, we are describing the consequences of processing voxels, which have inaccuracies due to double-integer conversions. In Figure 5-10 (a), we defined a reference point, from which we are tracing a line into the voxel cube. The next image shows each erroneous voxel projected back into the reference image. A kind of flickering about the center is clearly visible. The two images in the second row are showing the same line from another viewpoint. The right one shows the projected erroneous voxels, while in the left image the snapped voxels were projected. Only the corrected voxels on the left are forming a straight line, namely the epipolar line, in opposition to the noisy line on the right.



a) The reference point



b) The integer-line projected back into the reference image



c) The snapped voxels projected into a second image



d) The rounded integer voxels projected into the second image

Figure 5-10: Projecting corrected and uncorrected voxels

5.1.4 Recovering Visibility Information

It is often crucial to find out, when a voxel is visible and when it is occluded. There are mainly two reasons, why a voxel is not visible to a specific image. First, it can be positioned on that side of the object, which is facing away from the camera. The second reason is: it is occluded by another voxel in between. When high realistic rendering is required, we might take atmospheric effects into consideration such as fog. Then a voxel can just disappear due to a great distance to the camera. Although with a few modifications the following proposal might cover this problem as well, we will not consider it in detail.

When we compare visibility checking with polygonal models, we will find a clear separation between the two mentioned cases of invisibility. The first case can very easily be determined by calculating the normal vector of the according polygon. In many cases, those models are made of triangles. Hence the calculation of the normal vector is simply the cross product of two of its sides. The point in question can be excluded, as soon as the normal vector is facing away from the viewing direction that is if the normal vector and the viewing direction enclose an angle greater than 90 degrees.

All other elements can still be occluded by triangles in between. In order to check this, each and every triangle has to be processed in one way or the other.

One way is to sort the triangles according to the distance to the camera and draw them from back to front, so that further triangles will be overdrawn by the closer ones.

The second approach makes use of a distance buffer, the so called z-buffer. It corresponds in size to the output screen resolution. Every pixel that is drawn into the screen buffer will have its distance written into the z-buffer. If a pixel has a greater distance than the one stored in the z-buffer, it will be skipped. Closer voxels will replace the distance value in the z-buffer, as well as the color value in the screen buffer.

In volumetric models, we can not simply take over those techniques. The surface normal vector is a criterion which can be applied, although it is much more expensive to compute and less accurate at the same time. The use of a z-buffer is a sensible solution, which can easily be converted to volumetric purposes. However, instead of using z-buffer, in this chapter another method is introduced which is more general, fast, easy to implement and can detect all kinds of invisibility in one algorithm.

The proposed algorithm is based on the line tracing which is explained previously. The basic idea is to define the line according to the voxel in question and the image projection center. We will trace this line towards the image and we will stop as soon as an opaque voxel is encountered. If only transparent voxels are encountered until the line exits the defined voxel space, we can assume that the voxel is visible.

The distinction between different kinds of visibility works as follows: if the first voxel of the line is already opaque, then we are most likely dealing with a voxel on the backside of an object. If we first encounter some transparent voxels, but then again an opaque voxel, we are looking at a voxel facing the camera, but occluded by others. Figure 5-11 describes the idea.

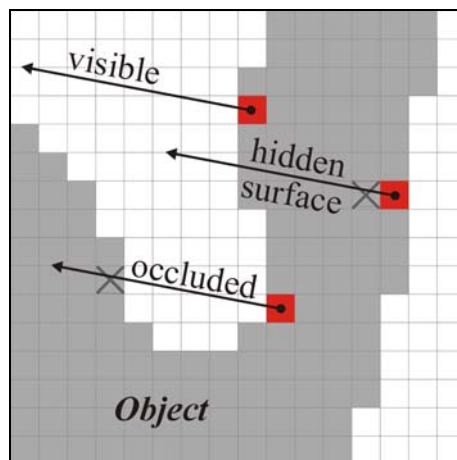


Figure 5-11: Recovering visibility information

Referring to Chapter 5.1.1.3, this is a method which makes use of the basic line tracing algorithm with an accordingly designed operator.

The pseudo-code for the initialization and per-voxel processing is as follows:

```
void Init(vector& p0, vector& p1)
{
    init base class with (p0, p1)

    // initialize all values with true
    // until proven otherwise:
    set visible = true
    set occluded = true;
    set hidden = true;
}

bool operator(vector v)
{
    set current_voxel = v

    if (v is outside)
        return false // stop algorithm

    if (v is object-voxel)
    {
        set visible = false
        set occluded = NOT hidden
        return false // stop algorithm
    }
    else
        set hidden = false

    return true // continue algorithm
}
```

The introduced algorithm covers all cases of invisibility. It works very fast for hidden voxels (those on the backside) and it can tell us, why a voxel is invisible and at which position this has been discovered. However, it is advisable to ignore on or two voxels at the very beginning of a line before evaluating the operator, since a line might contain a neighboring surface voxel, although the true line just passes by. As mentioned, the initial voxels to be ignored have to be surface voxels. As soon as an interior voxel is encountered, the algorithm can stop anyway. This concession to tolerance can be implemented into the operator itself, so that we don't have to alter the end points of the line, nor the line tracing algorithm. See Figure 5-12 for clarification.

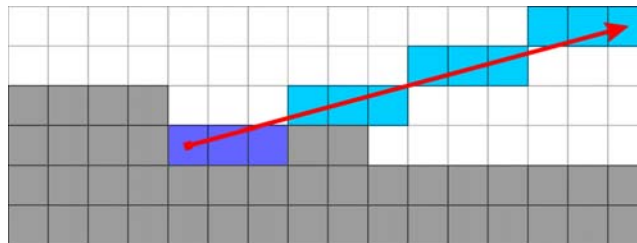


Figure 5-12: Tolerance for recovering visibility information

When we are supplied with alpha values (transparency values), we can enhance the operator so that it can take atmospheric effects into consideration. Consequently, when tracing a line the individual alpha values are combined and when a certain threshold has been reached, the voxel can be considered invisible. In this thesis however, this will not be applied. But it is worth mentioning.

Finally, we will demonstrate how the consideration of different voxels will affect a rendered image in Figure 5-13.

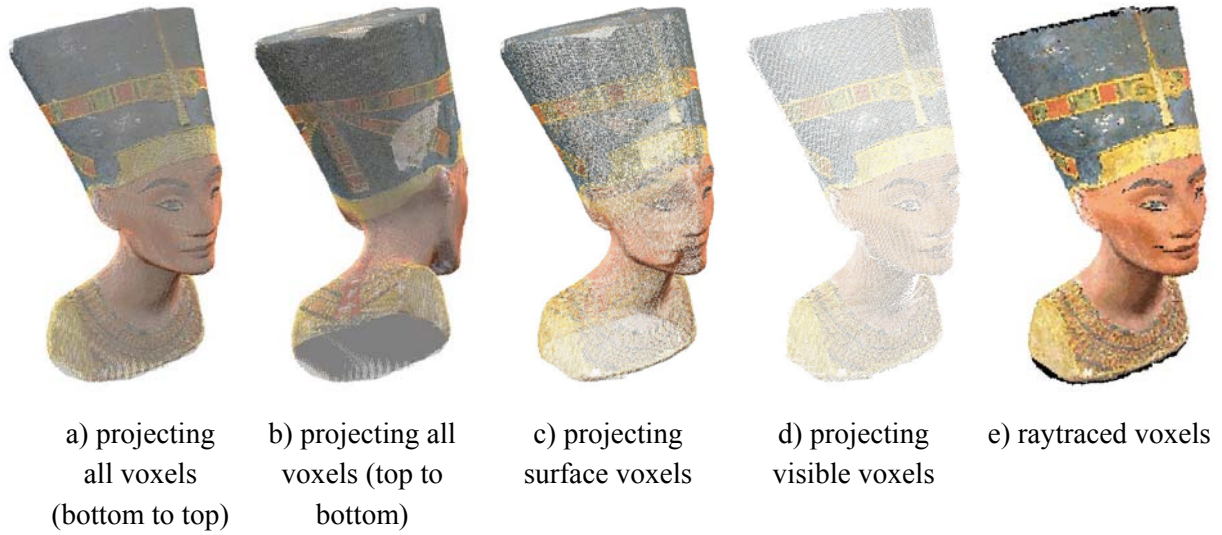


Figure 5-13: Different results for voxel rendering

On the very left, all voxels (including interior voxels) have been projected into a new image. A medium gray value is given to the interior voxels. If inside voxels were white, there would hardly be any difference to Figure 5-13(c). However, in Figure 5-13(b), we can see inside voxels through the surface. The second image shows exactly the same algorithm, except for the sweeping direction in which the voxels were projected (top to bottom).

In the next image (c) only surface voxels are considered, therefore voxels of the other side have a chance to be projected through the gaps of the front side. According to the introduced algorithm, only visible voxels are seen in Figure 5-13(d). Although it is the least dense image, it does not contain false information. The last image shows a result that has been accomplished by tracing each pixel back into the cube, until hitting the first object voxel. It corresponds to indirect pixel resampling, where we are rather performing an inverse transformation to look for the source pixel. This gives naturally the best result since it does not depend on the images but only depends on the cubes resolution.

5.1.5 Normal Vector on Voxel Surfaces

As introduced in Chapter 2.2.5, it might be interesting to learn about the viewing direction of a specific voxel. We will now describe a way to derive the tangent surface to the surroundings of a voxel.

The definition of a three dimensional plane is given as follows:

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \quad (5.16)$$

Expanding the term yields:

$$n_x \cdot x - n_x \cdot p_x + n_y \cdot y - n_y \cdot p_y + n_z \cdot z - n_z \cdot p_z = 0 \quad (5.17)$$

Renaming and rearranging:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \quad (5.18)$$

Now, we have to solve for the unknowns a , b , c and d . However, the translation and therefore the exact position of the plane is given by the parameter d . But we are only interested in the direction, i.e. the orientation of the plane, which is only given by the parameters a , b and c .

Now we can write the equation system with matrices:

$$A \cdot X = 0, \quad (5.19)$$

with the unknown vector:

$$X = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \quad (5.20)$$

and the design matrix:

$$A = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \end{bmatrix} \quad (5.21)$$

Before we can solve the mostly over conditioned system, we have to clarify, where the coordinates in the design matrix come from.

When we want to learn about the tangential surface of a very specific voxel, we have to take a look at its surroundings. Hence, we define a radius r of voxels, in which we specify surface voxels. Their center coordinates will be introduced to the design matrix. Furthermore, we will consider them as local coordinates, i.e. the origin voxel is assigned (0, 0, 0) and the surrounding voxels will only vary in integer values between $-r$ and $+r$. This assures a well balanced design matrix.

Figure 5-14 shows a part of a model, with its surface voxels, the area specified by the radius and the surface normal vector. For a better understanding, Figure 5-15 shows a larger area, with several surface normal vectors.

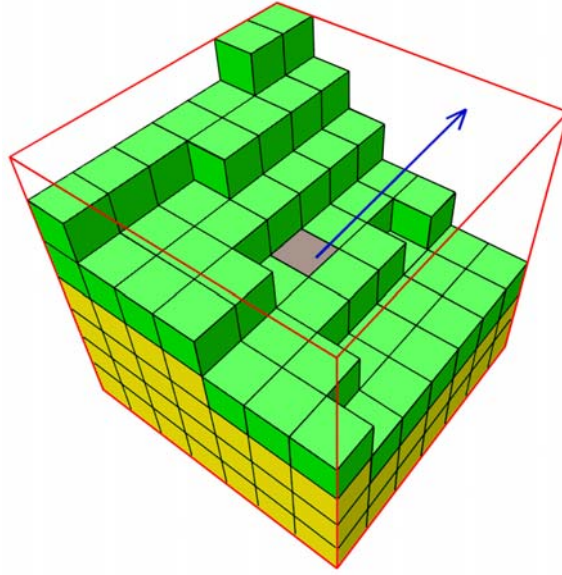


Figure 5-14: Surface voxels (green) within the radius (red) around the voxel of interest (dark grey) and their surface normal vector (blue).

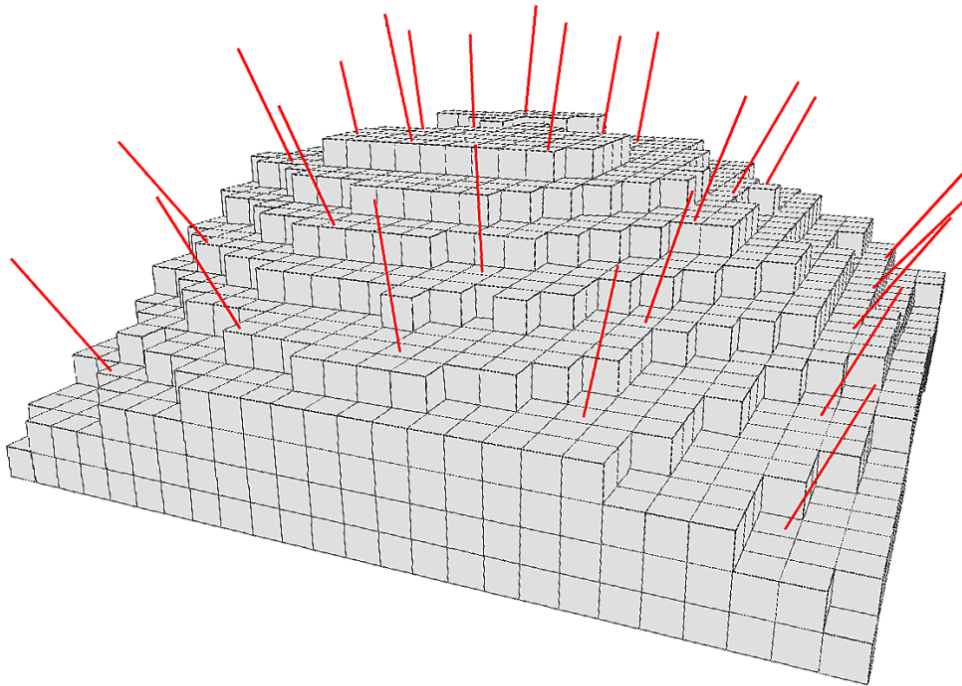


Figure 5-15: Several surface normal vectors seen on a larger surface fragment

Within the defined radius, we will derive all first order surface voxels. On one hand, this will speed up the search; on the other hand it delivers enough surface points for the solution of the tangential surface. Obviously, the set of points will not exactly lie on one plane, so the solution is that of an adjustment. Hence, it will only be a best fit surface.

Coming back to mathematics, we have to solve $A \cdot X = 0$, where we are not interested in the obvious solution $X = 0$. We will use the singular value decomposition (SVD) to compute the best solution, with $X \neq 0$. We will not go into detail about the SVD; instead we refer to literature about this subject [HARTLEY, 2000]. Nevertheless, the SVD will be used to split the design matrix A into three new matrices U , D and V^T , such that $A = U \cdot D \cdot V^T$, where U and V are orthogonal matrices and D is a

diagonal matrix with non-negative entries. The SVD can be performed for non-square matrices. This is especially the case when we have an over-estimated adjustment problem. Then we will have an \mathbf{A} matrix with clearly more rows than columns, i.e. more observations than unknowns. From adequate literature we can learn that the solution to an equation $\mathbf{AX} = \mathbf{0}$ corresponds to the column of the \mathbf{V} matrix, which corresponds to the smallest value in the \mathbf{D} matrix. The solution of the above equation can be written down as the following steps:

- Perform the SVD for the \mathbf{A} matrix.
- Let i be the index of the smallest value in the \mathbf{D} matrix.
- The solution vector \mathbf{X} corresponds to the i .th column in the \mathbf{V} matrix.

The recently derived unknown vector \mathbf{X} directly contains the elements of the desired normal vector, so that $\mathbf{X} = \mathbf{n}$. Furthermore, since we are only interested in the direction of the normal vector, we do not have to worry about its length, as long as its not zero-length.

Now, let \mathbf{P} be the vector of projection, along which the voxel of interest is projected onto the image plane.

We can now compute the angle between the surface normal vector and the projection vector (Figure 5-16), according to the scalar product (or dot product):

$$\cos \alpha = \frac{\vec{n} \cdot \vec{P}}{|\vec{n}| |\vec{P}|} \quad (5.22)$$

The sign of the surface normal vector, calculated as above, is not always clear. So we do not really know whether it is facing inside or outside the model. Assuming that we have a solid model, not only a shell, we can easily solve this ambiguity. Since the only undetermined property is the vectors sign, we can check the voxels on either side. On one side (inside) there should be object voxels present, on the other side (outside) they should be background. This assumption is valid since we are considering surface voxels. This final check was applied in Figure 5-15 and it can be seen that all vectors are pointing outside.

However, without knowledge of the sign of the surface normal vector, the angle will range from 0° (looking directly towards or away from the camera) to 90° (facing away perpendicular). If we have knowledge of the vectors sign, we can distinguish angles between 0° and 180° (looking straight away from the camera).

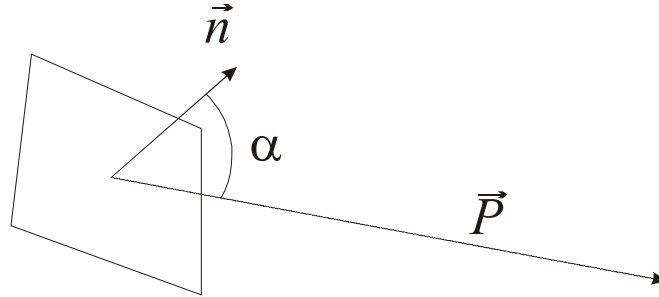


Figure 5-16: Angle α between the surface normal \mathbf{n} and the projection vector \mathbf{P}

Figure 5-17 is a visualization of the angle between the surface normal and the projection vector. The left picture is an original digital image, captured by camera, the middle and right picture show the angle according to the approximated volume intersection model where the lighter the pixel, the smaller is the angle. The right image is just a generalized version of the middle image. Angles are here separated into three classes: less than 30° (white), 30° - 60° (gray) and greater than 60° (dark gray).



Figure 5-17: Visualization of the angle between surface normal and projection vector

5.2 Model Refinement

As we have seen in the previous chapters, the model derived from volume intersection consists of the visual hull, which is a convex shape (see Figure 4-4). So if the object has some concave parts, as most objects do, we will have to find means to carve away those superfluous areas.

The very crucial condition of a true surface point is that if projected into two or more images in which it is visible, it will be projected into corresponding image points. In the following, we will always assume that we have accurately oriented images. In Figure 5-18, we can see that if we project a voxel of the (false) visual hull P_0' into two images it delivers the two image points p_0 and p_1' . But the images are taken from the real world, i.e. the true object. Therefore, the image point, or the texture at this point respectively, corresponds to the models texture at P_0 for the left image and P_1' for the right image, which is normally different. On the other hand, the texture at the image point p_1 in the right image corresponds to that of p_0 in the left image. They both are projections of the true surface point P_0 .

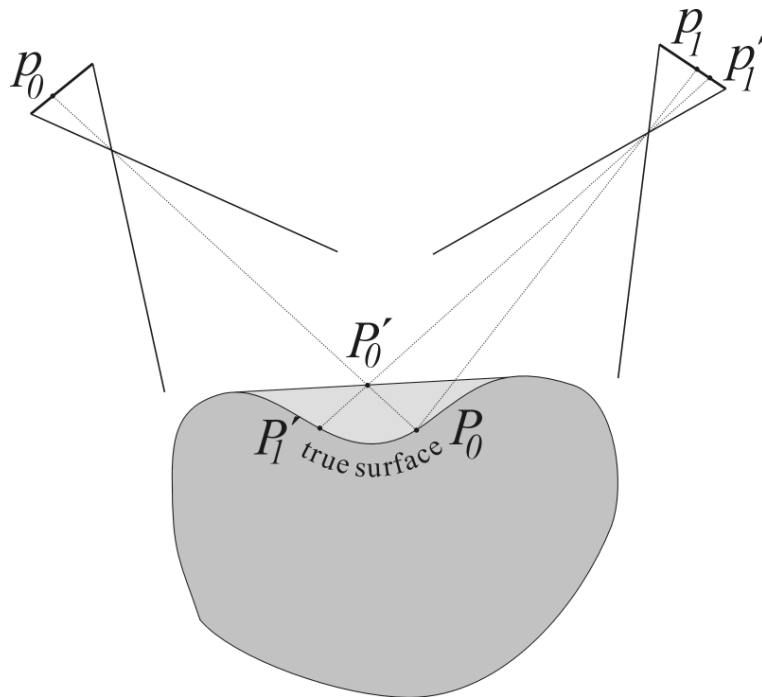


Figure 5-18: Projection error due to concavities

Hence, we have a means of telling, whether a point is a true surface point or not, by comparing their textures. Hence, if the object point is projected into different image points we assume that this voxel is false and therefore should be carved.

However, there are some obstacles. Many times we have mentioned that we are comparing image textures in small templates around the point of interest. If there is no texture, but only a region homogeneous in color and lightness, there is no way of telling which template in a search region is the best match. In other words, every homogenous master template is a very good match with any other homogenous slave template. Nevertheless, the human eye can still derive depth information from a homogeneous object, but there are many other influences which help us in our evaluation. Last but not least we simply recognize the object as a whole and just know its shape. However, with image matching we try to derive a distance upon two small image patches alone. So without further knowledge, we cannot make a decision for homogeneous areas.

Another drawback especially in close range photogrammetry is that the image templates may be highly distorted due to perspective geometry, or even rotated against each other. One solution is to apply the least-squares-matching in order to transform the slave template in a way that it best fits the master template. But still, this method requires good approximation values, at about one pixel, which is mostly acquired by cross correlation for example. In case of perspectively distorted image patches, there is hardly a way to find good matches with cross correlation.

In the following sections, knowledge will be introduced to the image matching algorithms in order to take perspective distortion into consideration.

Subsequently, two approaches are presented for carving the concave areas of the approximate model.

5.2.1 Knowledge-Based Patch Distortion

This chapter shows how we can introduce previous knowledge into the basic image matching algorithms. The more advanced algorithm, in particular the least-squares-matching will take an approximated patch and distort it in a way, that it best fits the master patch. Since the patch is very limited in size and shows only a small part of the original image, it is sufficient to model the distortion by an affine approach with six parameters. The correct description would require an eight parameter projective transformation, but many times it has been proven that an affine transformation is enough. The LSM algorithm looks for the one distortion, which delivers the least difference in the correction expression, hence least-squares-matching. This result does not necessarily have anything to do with the reality, it is just mathematically the best result.

It would be desirable to be able to introduce some knowledge for verification. First, we have to clarify what kind of knowledge we can offer. As mentioned above, we assume to have accurately oriented images. So this is our basic knowledge. Furthermore, since we are considering very specific voxels, we also have information about object space. Later it will become clear, that the latter information is not necessary.

Let us now describe how we can apply this knowledge to the image matching. The idea is to estimate, how the slave patch should be transformed in order to perspectively correspond to the master patch. Since we have knowledge about distances and image orientations, it is possible.

In particular, the algorithm projects the master patch into object space. This can be done because we know which distance we are dealing with. Furthermore, the patch is uniquely defined in the master image, therefore delivering four image points. These four points will be projected into object space, using the same distance as the voxel of interest. Those four object points are now simply projected into the second image(s), forming the transformed slave patch. For clarification, Figure 5-19 shows the same example situation from two viewpoints.

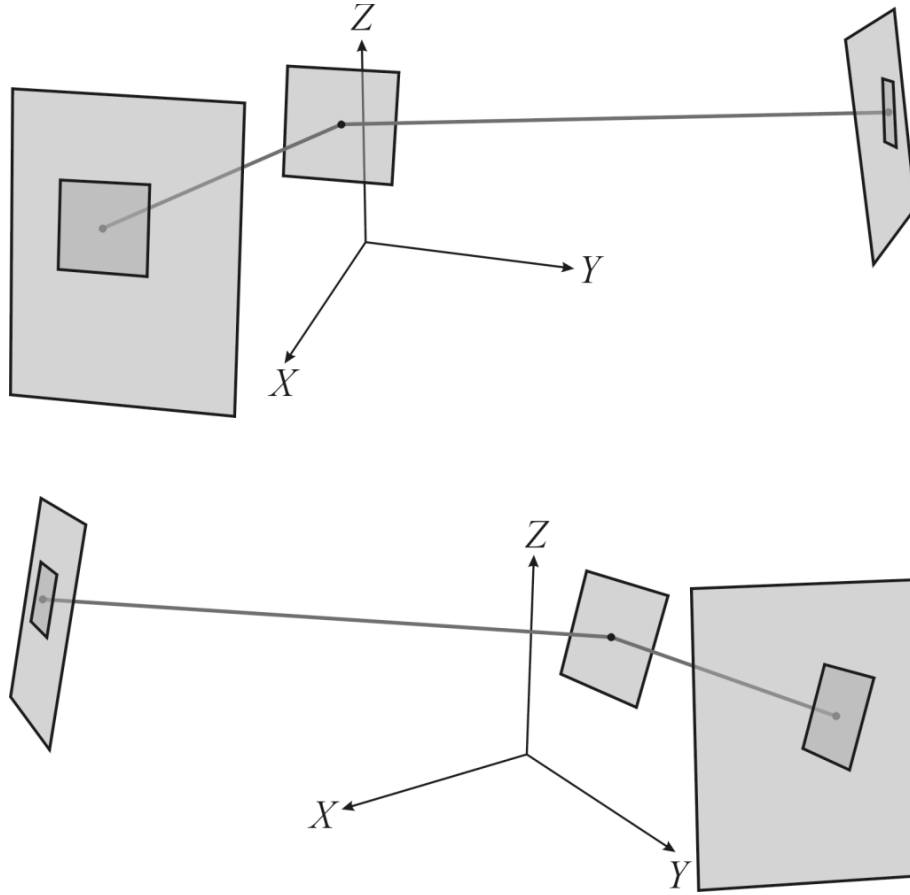


Figure 5-19: Master patch and distorted slave patch from two viewpoints

If we do not have knowledge about a particular point in space, we will have to estimate a position in the slave image, which we want to compare. However, the position of the master and the slave patch deliver an object point, when performing a forward section. So if we are looking for a best match, we can project the slave patch for each position in the slave image individually, so that it will be correctly transformed, according to the actual image point.

When we take a look at the mathematics behind, we first have to derive the distance. We will again make use of the λ -factor from equation (3.4) (see chapters 3.1.1 and 5.1.2):

$$\lambda = \frac{r_{11} \cdot (X - X_0) + r_{21} \cdot (Y - Y_0) + r_{31} \cdot (Z - Z_0)}{x_m - x_0} \quad (5.23)$$

Using the same λ -value, we can construct four new object points, corresponding to the four corners of the master patch:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R^{-1} \cdot \lambda \cdot \begin{pmatrix} x_i - x_0 \\ y_i - y_0 \\ -c \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}$$

with,

x_i, y_i : the four corners of the master patch

These four object points are now being projected into the second image, using the collinearity equations (chapter 3.1.1):

$$x_i^{slave} = x_0 - c \frac{r_{11}(X_i - X_0) + r_{21}(Y_i - Y_0) - r_{31}(Z_i - Z_0)}{r_{13}(X_i - X_0) + r_{23}(Y_i - Y_0) - r_{33}(Z_i - Z_0)}$$

$$y_i^{slave} = y_0 - c \frac{r_{12}(X_i - X_0) + r_{22}(Y_i - Y_0) - r_{32}(Z_i - Z_0)}{r_{13}(X_i - X_0) + r_{23}(Y_i - Y_0) - r_{33}(Z_i - Z_0)}$$

At this point, we already have the outline of the slave patch, but what we really want is a set of transformation parameters, so that we can grab the patch from the image accordingly. As mentioned, we will limit this transformation to a six parameter affine transformation for relatively small patches.

As the patch is stored in memory as a two dimensional array $g[0...width][0...height]$, we want the transformation to map these patch coordinates to the slave patch. Here we have two sets of corresponding 2D points:

The patches array coordinates (p^a_i):	The projected image coordinates (p^t_i):
$p^a_0 = (0 ; 0)$	$p^t_0 = (x_0 ; y_0)$
$p^a_1 = (width ; 0)$	$p^t_1 = (x_1 ; y_1)$
$p^a_2 = (width ; height)$	$p^t_2 = (x_2 ; y_2)$
$p^a_3 = (0 ; height)$	$p^t_3 = (x_3 ; y_3)$

We have four points in order to derive six affine transformation parameters. This is an over conditioned equation system, so the result is a best match, not necessarily the correct match. This is consistent with the fact, that in reality it is a projective transformation. But we will be satisfied with the affine result.

Now, here is how to find the transformation parameters. We have the design matrix A , the observation vector l and the unknown vector x , so that:

$$A \cdot x = l \quad (5.24)$$

Where:

$$A = \begin{pmatrix} x_0^a & y_0^a & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0^a & y_0^a & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}, l = \begin{pmatrix} x_0^t \\ y_0^t \\ \vdots \end{pmatrix} \quad (5.25)$$

and,

$$x = \begin{pmatrix} a1 \\ a2 \\ a3 \\ b1 \\ b2 \\ b3 \end{pmatrix} \quad (5.26)$$

for the affine transformation;

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a1 & a2 & a3 \\ b1 & b2 & b3 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (5.27)$$

The equation has to be solved for x . This can be done using the least-squares approach.

$$x = (A^T A)^{-1} A^T l, \quad (5.28)$$

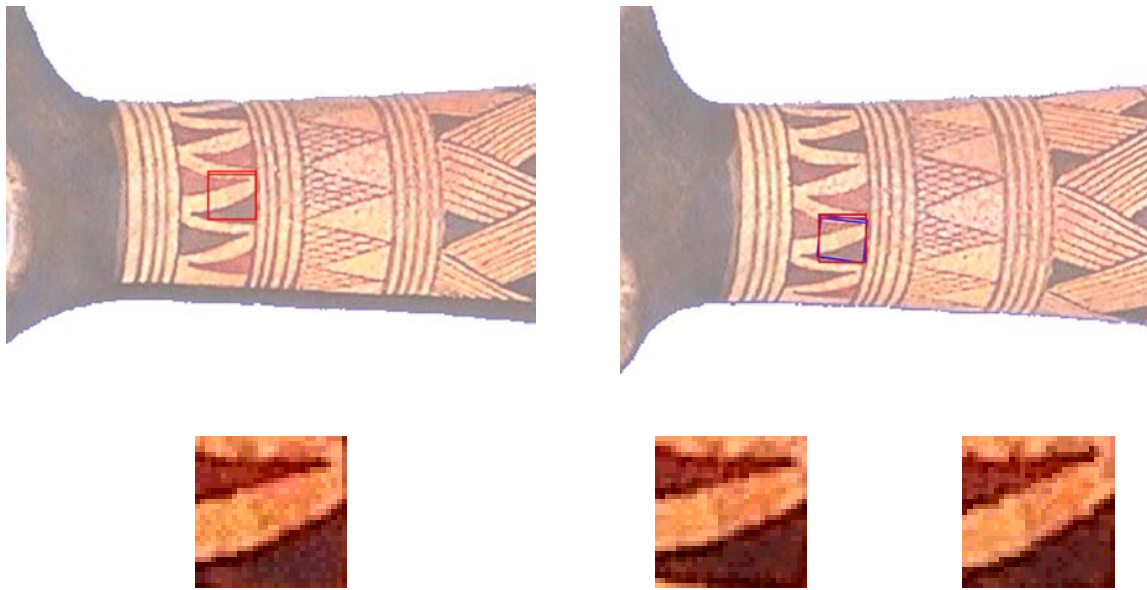
or the singular value decomposition, mentioned in chapter 5.1.5.

This transformation matrix can be used very conveniently to grab the template from the according image:

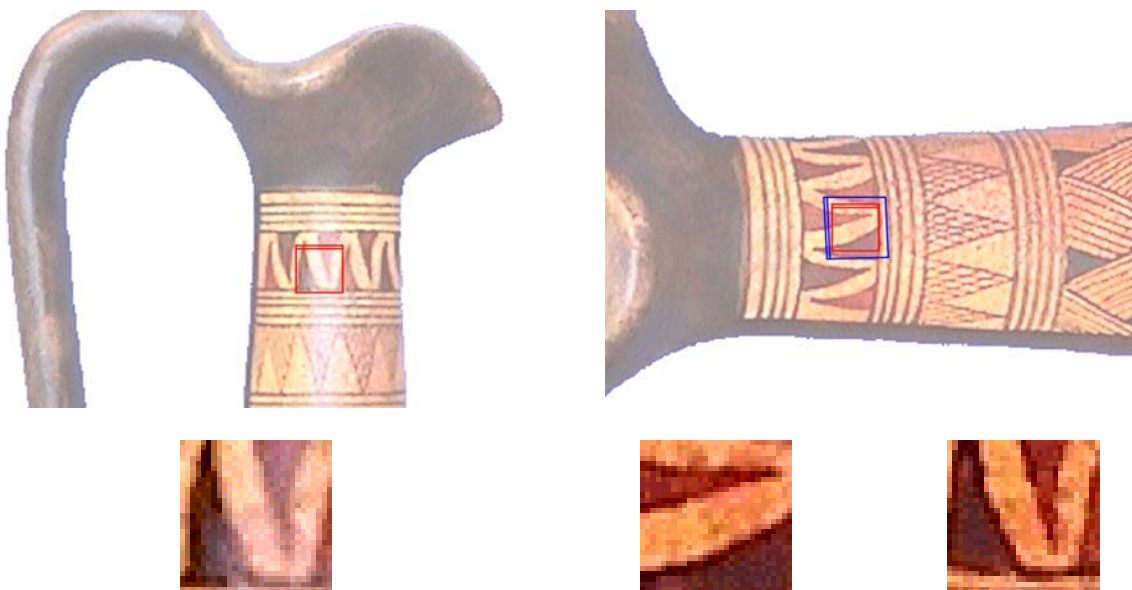
$$\begin{pmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{pmatrix} = T \cdot \begin{pmatrix} x_{template} \\ y_{template} \\ 1 \end{pmatrix}, \quad (5.29)$$

where the template coordinates are looping over the template array $[0...width][0...height]$. So this transformation corresponds to an indirect image resampling, so that we do not have to expect any gaps in the template.

Finally, the following figure will illustrate the effect of the knowledge based patch distortion. On the left side you will see a detail of the master image. Just below is the actual master patch. The right column shows the corresponding part of the slave image. Below on the left is the raw slave patch and on the right you can see the transformed patch. The patches are also illustrated as frames in the image details. The second example shows very impressive, that even rotated patches can be grabbed correctly, so that subsequent correlations will give reliable results.



a) Two neighboring images and their patches



b) Rotated images and their patches

Figure 5-20: The effect of knowledge-based patch distortion

5.2.1.1 Introducing Knowledge about the Approximate Object Shape

Additionally to the image orientation, we can offer the knowledge of the approximate shape of the object. This is especially the case, after performing volume intersection. The idea is to approximate the image patch on the surface of the object itself. This patch will then be projected into all candidates for the image matching. Hence, we need to construct the tangential surface at the voxel of interest. This solution is already offered in Chapter 5.1.5 where the computation of the surface normal vector is described. There we already have the surface in the form:

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \quad (5.30)$$

What we would like to do now, is to create a rectangular patch on that surface. It follows that we need a local 2D coordinate system, as seen in Figure 5-21.

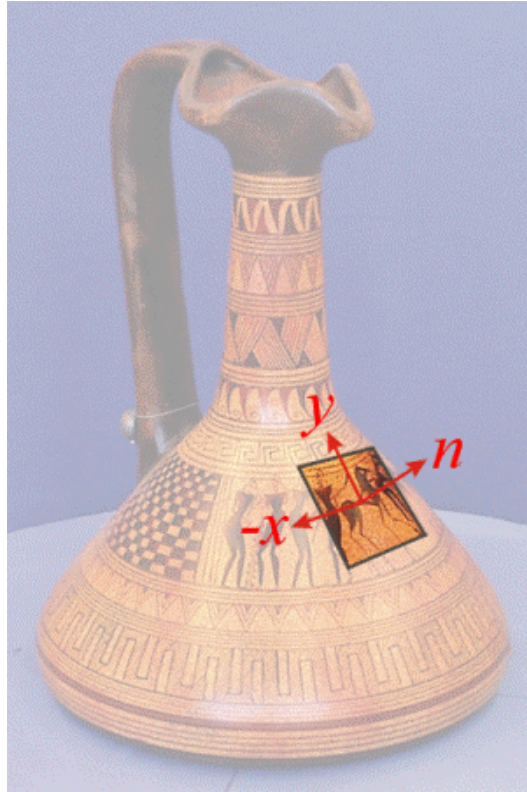


Figure 5-21: Creating a tangential surface patch

We will now create the vector x , with the following two conditions: it is perpendicular to n and parallel to the XY plane.

It follows that:

$$\vec{x} \parallel XY \Rightarrow \vec{x} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \quad (5.31)$$

and,

$$\vec{n} \cdot \vec{x} = 0 \Rightarrow x_n \cdot x_x + y_n \cdot y_x + z_n \cdot z_x = x_n \cdot x_x + y_n \cdot y_x = 0 \quad (5.32)$$

Choosing $x_x = n_y$ and $y_x = -n_x$ delivers the desired vector.

The construction of the vector y can be derived by taking the cross product of n and x .

Figure 5-22 shows the improvement to the patches when considering the object shape. We can clearly see a difference between the original slave patches while the knowledge based patches show less variety.

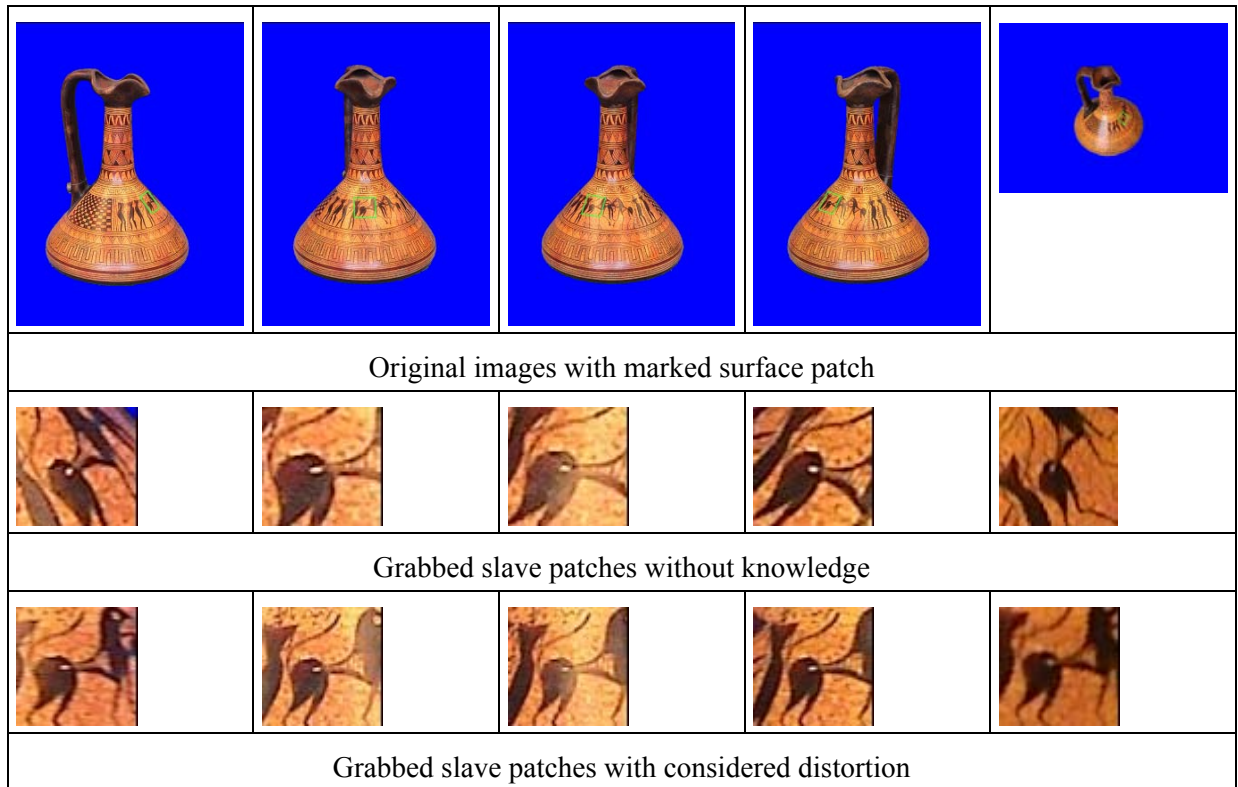


Figure 5-22: The improvement of taking the object shape into account

5.2.2 Shell Carving

In this chapter we will introduce an approach to carve away concave areas of a model. We will make use of some tools which we defined in Chapter 5.1.

As the name suggests, we want to carve away false voxels shell by shell. So we begin by creating the surface voxel list, which can be considered as a kind of outer shell. Now we will process each voxel in this list. For each voxel, we will check if it is part of the true surface, which was explained in the last chapter. The algorithm will stop, if in a shell no voxels are considered false, or if a maximum carving depth has been reached.

Very briefly, the algorithm can be separated into the following steps:

- Create surface voxel list.
- For each voxel:
 - Select three, at least two images with the best view.
 - Perform image matching in the selected images.
 - Carve the voxel, if considered different; Set a fixed-flag, if considered equal.

If we take a closer look at all these steps, we will have to do more detailed explanation. The creation of the surface voxel list (SVL) has already been explained. However, it might be worth mentioning, that in our case it is managed by a dynamic double-linked list. This means that each list item has a reference to its predecessor and its successor. Dynamic lists are predestined for inserting and deleting operations, and they are crucial, when the resulting size of the list is known only after some time, as in our case. As opposed, a fixed array needs to be allocated; therefore the size must be known in the very beginning.

Now, we want to check each voxel, if it is part of the true surface, hence if it is projected into corresponding image templates. Therefore, we need to find at least two images, in which this voxel is visible. From the set of available images, we can first exclude those, where the voxel is occluded. For this task, we have introduced an operator in chapter 5.1.4.

From the remaining set of images, we will have to choose two or three images, where we can assume the best visibility. This assumption is based on the surface normal vector. See chapter 5.2.4 for a detailed explanation. In order to make a comparison, we need at least two images. If only one image can be determined, an evaluation of this voxel is not possible and it is set as fixed in order not to be processed again.

If we have two images, we can perform an image matching to see whether the voxel projects into corresponding image points or not. When three images are available, we can check the similarity with another pair of templates to improve reliability. At this point, the knowledge based patch distortion can be applied for a more accurate template matching.

Once a voxel has been found true, a flag is set, stating that this voxel is true and should not be considered again. Voxels, which are projected to different image patches, will be carved. So shell by shell, we will normally have an increasing number of fixed voxels, and a decreasing number of carved voxels, until this number drops below a certain threshold or until a maximum carving depth has been reached.

The following pseudo code is once again summarizing the approach:

```

for (shell=1 ... maximum_depth)
{
    create_svl
    for each voxel from svl
    {
        if (voxel is fixed)
            continue loop

        select best visibility images
        if (less than two images selected)
        {
            set fixed for this voxel
            continue loop
        }

        if (master template is homogeneous)
        {
            set fixed for this voxel
            continue loop
        }

        if (different image1, image2)
        {
            carve voxel
            continue loop
        }

        if ((image3 available) AND (different image1, image3))
        {
            carve voxel
            continue loop
        }

        set fixed for this voxel
    }

    if (carved voxels in this shell < threshold)
        break loop
}

```

In this approach we have several variables, which will influence the result. First there is the size of the image patches. There is no thumb rule, which size is the best. It depends on the image resolution, the texture and the differences in the viewpoints.

Furthermore, we can manipulate all the thresholds which appear in the process. In the pseudo code it can be seen, that we skip homogeneous patches. We can influence the behavior by varying this parameter. Additionally, we can vary all the thresholds for the similarity checks. The lower we set those, the more tolerant the algorithm will be, and less voxels will be carved.

This approach is working rather fast, and is accelerating with each shell, due to the increasing number of fixed voxels. However, it is not the most reliable approach to fix a voxel as true surface, when a confidence threshold is exceeded. The voxel below can still deliver a better match, but it will never be

processed, because its predecessor was sufficiently convincing. It is comparable to the stock market. At one point, there has to be a decision whether to buy or sell, but the next day could have been much more profitable, we just did not consider it (because we cannot look into the future). In our carving algorithm, the information about the next voxel (the future) is there, but we do not make use of it.

The next chapter will show an alternate approach, which does take a set of voxels into consideration, to choose the best one at the end.

5.2.3 Shape Carving

As we have mentioned, the shell-carving approach is fast, but may not be too reliable. So the following approach takes over the idea of a milling machine. The machine carves away parts of the object in individual depths, so that after one pass, the object can theoretically be finished. As opposed, the previous approach would only take away one voxel in depth at a time.

In our case, the images take over the function of the milling machine. Image by image and pixel by pixel we will go forward and engrave into the object to a reasonable depth. The depth is, as in the last chapter, based upon image correlation.

We will need some further explanation. Thus, by defining a pixel in an arbitrary image, we will be able to find the corresponding object voxel. We have to define an operator for the line tracing, which will stop, as soon as an object voxel is encountered. This line can now be defined by the pixel, as

$$p = (x, y, -c)$$

and the image projection center

$$p_0 = (x_0, y_0, z_0)$$

We have to trace this line into the voxel cube and apply an operator to detect the first object voxel. This will also be the line which works as the engraving tool. It is in this direction, which the object will be carved.

Up to now, our machine is defined by the engraving point and direction. But we need information about the carving depth. In order to obtain this information, we need at least one other image. As in the previous chapter, we will select one or two more images which promise the best view, next to the master image.

Again, we will trace a line. It is the same line, which served to find the object voxel, but here, this very voxel serves as a start point. A previously defined maximum carving depths delivers the end voxel. When we define a much more complicated operator, we can trace this line and find the one voxel which projects into the most similar templates of the two slave images. The master template is constant throughout the whole tracing. Figure 5-23 illustrates how the depth is being derived.

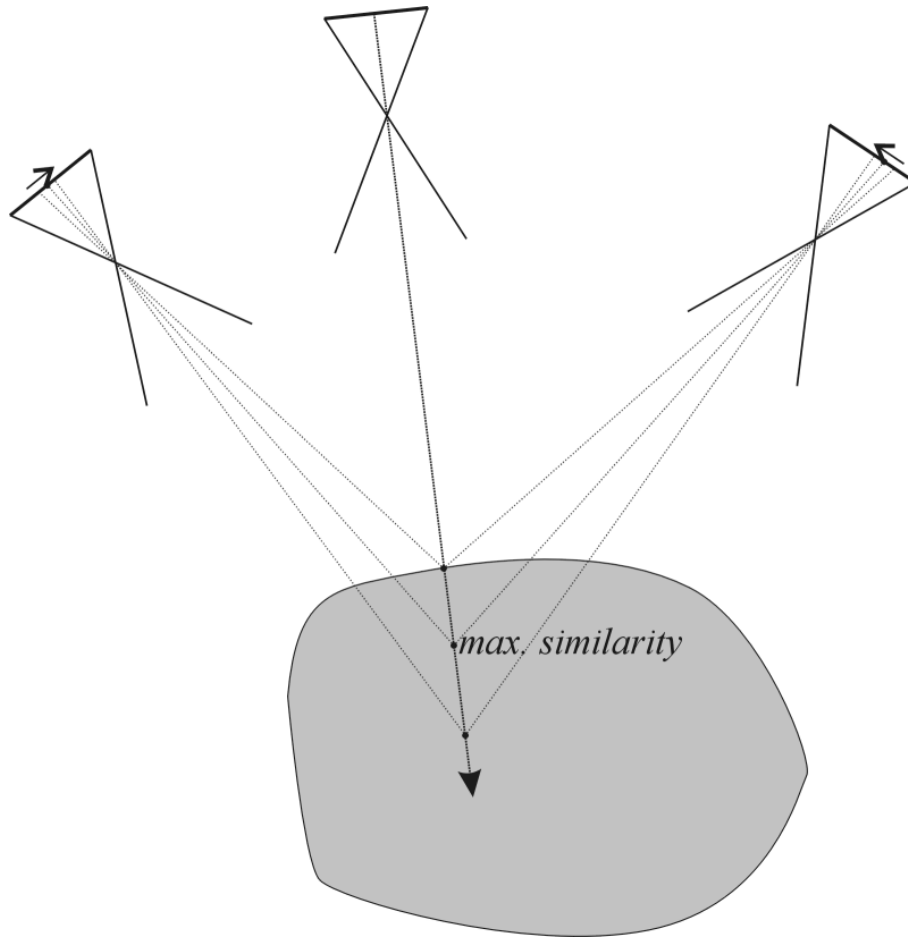


Figure 5-23: The 'milling machine' approach

During this operation, we can apply the knowledge based patch distortion to the template comparison. It will definitely slow down the algorithm, but it will also make it more reliable.

After this step, we know the position of the most similar patch. We are not really interested in the image's patch coordinates, but only in the corresponding voxel itself. Once again, we will make use of the line tracing algorithm. As we now will have a line, strictly defined by the first encountered object voxel and the voxel of maximum similarity. We will run the line tracing with an operator, which sets all encountered voxels to a predefined value, in this case background.

For each image pixel, we have made use of the line tracing algorithm three times, with the following operators:

- Find the first object voxel.
- Find the voxel of maximum similarity.
- Set all voxels along the line to background.

While the basic algorithm remains unchanged, we only have to define three operators for these tasks. This shows again, how powerful and flexible this kind of programming is.

The engraving process is repeated for each pixel in each image. Again, we can set a fixed-flag for the voxels, to prevent ambiguous evaluation.

The pseudo code for this method:

```

for each image
{
    for each pixel
    {
        let v0 = trace line (find first object voxel)
        if (not found)
            continue loop

        select two images with best visibility
        if (none selected)
            continue loop

        let v1 = trace line (find maximum similarity voxel)
        if (similarity passes threshold)
            trace line [v0-v1] (set voxels to background)
    }
}

```

This code looks simpler than that of the shell carving, but a lot of work is hidden inside the operators, especially the maximum similarity operator. So it might be worth investigating this one a little bit further.

As we can see in Chapter 5.1.1.3 the operator has three functions which are called during the line tracing: an initialization method before the tracing starts, a cleanup method after the tracing breaks and the per-voxel operator.

Furthermore, since the individual operator is a C++ class, we can store as much information as necessary into the class. This information is valid and available through the whole tracing procedure and even after that. In this case we need to store at least the following information:

- Reference to the two slave images
- Two slave templates
- One master template
- Two maximum similarity factors and their positions

In the initialization method we can check the master for possible homogeneous areas. In these cases, we can already break the algorithm with a failure, before it even started. Furthermore, we need to initialize the maximum value flags with starting parameters.

The cleanup method is not really necessary, but the major work is done by the per-voxel operator. The master template is constant for the whole line, but the two slave templates change in position and, if knowledge is applied, in shape. Therefore, during each call of the operator, the current voxel is projected into the two slave images and the according templates are grabbed, with or without distortion. The correlation factors are calculated and the maximum value with its position is updated. After the tracing finishes, all class members remain valid and can be read from anywhere necessary. This is where the above pseudo code continues, and eventually the third line tracing starts.

It might be helpful to display some more pseudo code and the definition of the maximum similarity operator.

The definition of the operator-class:

```
class CMaxSimilarity : public CTraceCubeLine
{
public:
    CMaxSimilarity();
    virtual ~CMaxSimilarity();

    virtual bool    Init(const vector& start, const vector& end);
    virtual bool    CleanUp(bool);

    virtual bool operator () (const vector& v);

    CBuffer*    m_pSlave1;
    CBuffer*    m_pSlave2;

    CTemplate m_Master, m_Slave1, m_Slave2;
    CMatrix    T1, T2;

    short      m_nMaxDepth, m_nDepth;
    short      m_nCount[4];
    double     m_fMaxSimilarity[4];
};
```

Pseudo code for the initialization method:

```
bool CMaxSimilarity::Init(const vector& start, const vector& end)
{
    if (master template is empty OR homogeneous)
        return false    // don't even start

    m_fMaxSimilarity[0] = -1.0;
    m_fMaxSimilarity[1] = -1.0;
    m_fMaxSimilarity[2] = -1.0;

    m_nCount[0] = 0;
    m_nCount[1] = 0;
    m_nCount[2] = 0;

    m_nDepth = m_nMaxDepth;

    return true    // ok, start the algorithm
}
```

Pseudo code for the per-voxel operator:

```
bool CMaxSimilarity::operator ()(const vector& v)
{
    Project voxel into first slave image.
    Grab the slave1 template. (with/without knowledge distortion)
    let c = correlation(master, slave1)
    update (m_nCount[0], m_nMaxSimilarity[0]) with c

    if (slave2 available)
    {
        Project voxel into second slave image.
        Grab the slave2 template. (with/without knowledge distortion)
        let c = correlation(master, slave2)
        update (m_nCount[1], m_nMaxSimilarity[1]) with c

        // cross check:
        let c = correlation(slave1, slave2)
        update (m_nCount[2], m_nMaxSimilarity[2]) with c
    }

    // consider maximum 'Depth' voxels, then stop.
    m_nDepth--
    if (m_nDepth == 0)
    {
        m_nDepth = m_nMaxDepth;
        return false    // break the algorithm
    }

    return true    // continue next voxel
}
```

5.2.4 Voting Based Shell Carving

The previously shown approaches will carve the voxels according to one or two evaluated image patches. Hence, the result is likely to be noisy, i.e. falsely carved voxels.

There are four possible decisions and consequences:

Decision	True state	Result
Carve voxel	Non-surface voxel	Correct
Leave voxel	Surface voxel	Correct
Carve voxel	Surface voxel	Incorrect
Leave voxel	Non-surface voxel	Incorrect

Table 5-1: Different error types for carving

We can see that we have a fifty percent chance of making the wrong decision. It is therefore advisable, to make this decision upon reliable information, or on as much information as possible.

In this approach, we will evaluate every sensible combination of image matching, before deciding, whether to carve the voxel or not. For this purpose, a second cube is introduced, which will store the votes of the single decisions.

We will show in detail, how these votes are counted. As in the shell carving approach, we will evaluate every surface voxel, shell by shell. Now for every voxel of the list, we are going to run two more loops.

The first will run through all images, and take a master patch from those, in which the voxel is visible. Now, for each master patch, we are running a second loop through all following images, where we are grabbing a slave patch, wherever visible. Every combination of patches will now be available. We introduce two counters. The first one keeps track of all comparisons we made. The second counts the number of comparisons which were successful.

The comparison can be made with any of the previously introduced matching approaches, with or without knowledge based patch distortion. However, the two counter numbers tell us now the percentage of successful comparisons. Applying a threshold to this percentage will either carve or leave the voxel.

As usual, we have several ways to influence the behavior of the algorithm. First of all, there is the size of the patch that we use for comparison. The bigger the patch, the more smoothed will be the carving. Another crucial element is the decision, whether the image patches are considered similar. This decision alone consists again of different elements. For instance it depends on the algorithm, on whether to use patch distortion, and finally the threshold for a successful comparison.

In Figure 5-24, 5 possibilities out of 6 images will be tested considering only direct neighbors. The test results will give a percentage of success or failure. This percentage will serve as carving decision.

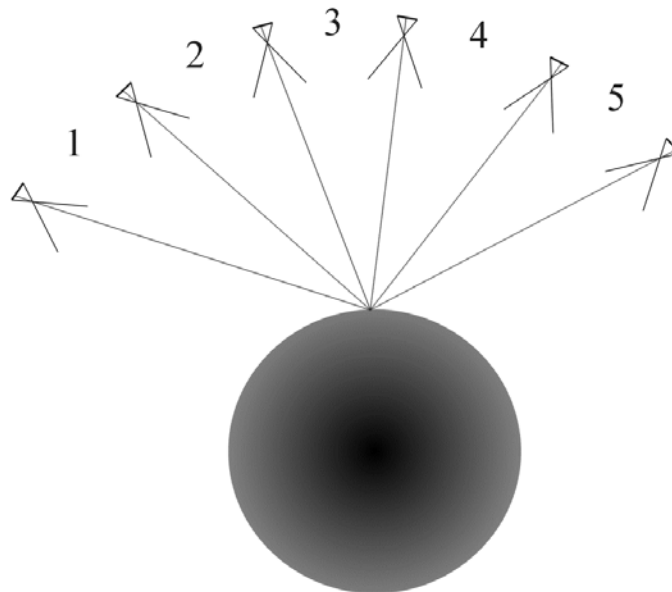


Figure 5-24: Voting-based carving

As a summary the pseudo code for this approach is given below:

```

for (depth=0...maxdepth)
{
    create surface voxel list
    for each voxel (surface voxel list)
    {
        set counter0 = 0
        set counter1 = 0
        for each image (n=0...maximage)
        {
            if (voxel is visible in image)
            {
                grab master patch
                for each image (m=n...maximage)
                {
                    if (voxel is visible in image)
                    {
                        increase counter0
                        grab slave patch

                        compare master and slave
                        if (successful)
                            increase counter1
                    }
                }
            }
        }

        set percentage = 100 * counter1 / counter0
        set vote-cube(voxel) to percentage
    }
}

```

This is a very rough version of the algorithm with many possibilities for optimization. For instance, we ask several times, if a voxel is visible in a specific image. Since this is a time consuming operation, this should be pre-calculated and stored in a list, so that the actual calculation is only done once per image. Secondly, when we are sure, that a voxel is definitely a surface voxel, we might set a flag, to prevent this voxel from being considered another time.

Nevertheless, we are storing the actual voting value inside a new cube. For visualization purposes, this value is scaled to fit the value range of 0...255, hence $\text{percentage} * 2.55$ is actually stored into the cube. When we now visualize the cube, we can clearly see how the single surface voxels were considered. The lighter the value, the more certainly it is a surface voxel, and the darker, the safer it is to carve.

The following figure illustrates the results of the voting.

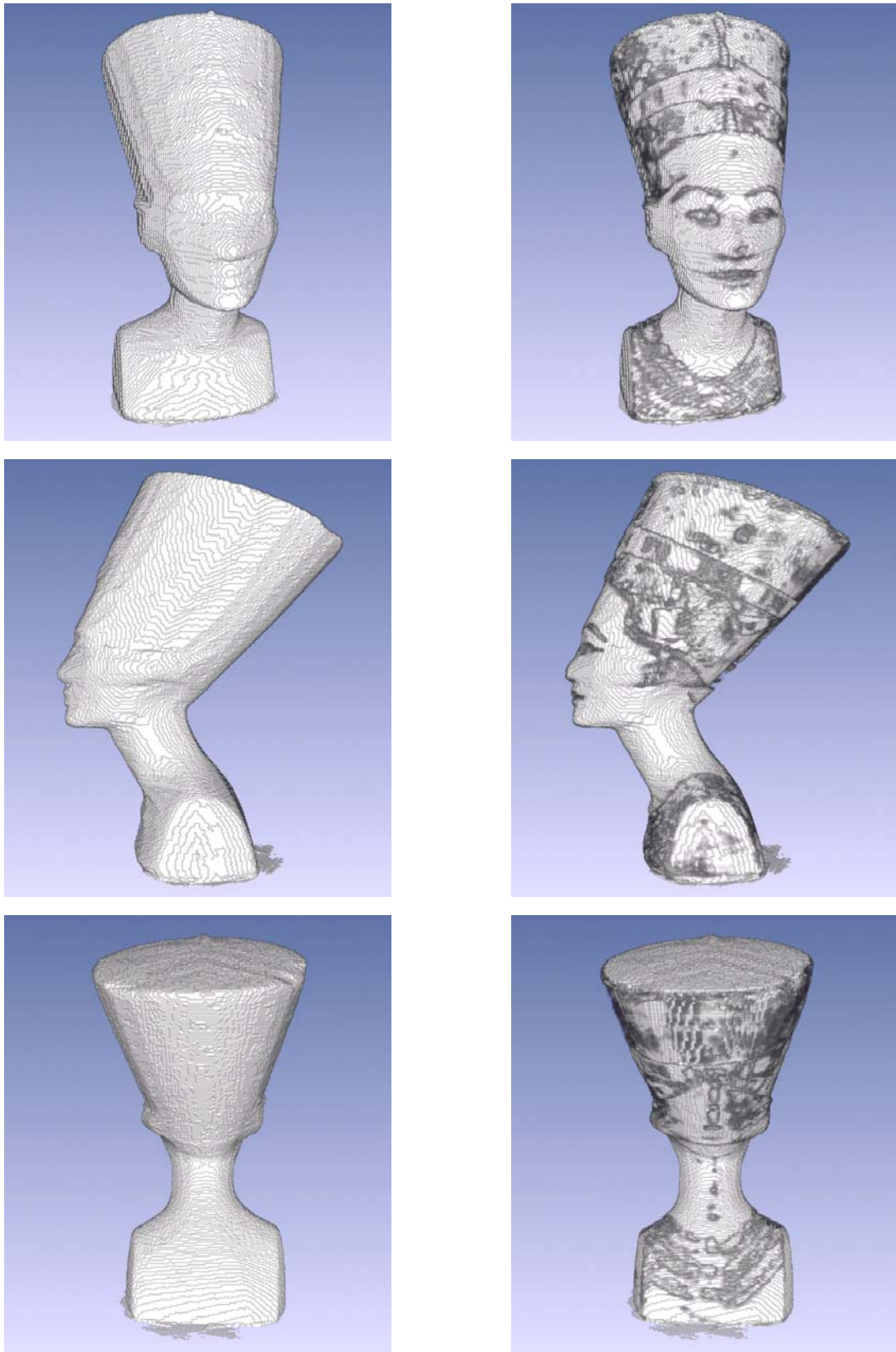


Figure 5-25: Different viewpoints of the Nefertiti cube. The result of shape from silhouette (left), refinement by voting based carving (right)

5.3 Texture Mapping

In Figure 5-26, you see an approximate, incomplete model acquired by volume intersection method. Even if the model represented the true shape, it would still not contain color. The comparison of textured model and non textured model is shown very clearly in Figure 5-26. As seen in the figure even if we have a rough model, the realism can be increased by applying texture mapping.



Figure 5-26: The approximate model without and with color information

The task is now to apply a color value, typically from a color table to every visible voxel, i.e. every surface voxel. The information we have is based on the available images from the object. From the preceding processing steps, we can assume that we have correct orientation data for the images. Furthermore, all the input images have been reduced 256 colors, so that all the images share the same color table.

As it is mentioned in previous chapters, only true surface voxels will be projected onto its correctly corresponding image pixel. Since we are here at the end of the processing chain, we are assuming that we are supplied with a correct model. So now we have to look for color information for every voxel. Since normally each voxel is visible in several images, we can choose, which image we take the color value from. Remembering chapter 5.1.5, we will use the surface normal vector to determine its enclosing angle with the current ray of sight:

$$\alpha = \arccos \frac{a \cdot b}{|a| \cdot |b|}$$

We can expect the best visibility, where this angle is smallest. However, the voxel should be visible in this image. The determination of the smallest angle is therefore not enough. The angle should only be chosen from the set of images, in which the voxel is visible in the first place. The pseudo-code is as follows:

```

create surface voxel list;
for each voxel in list
{
    set n = surface_normal_vector(voxel);
    set min_angle = 360;
    for each image
    {
        if (voxel is visible in image)
        {
            set angle = VECTOR_ANGLE(n, ray_of_sight);
            set min_angle = MIN(min_angle, angle);
        }
    }
    set img = image corresponding to min_angle;
    set color from projected voxel into img;
    set voxel_color(color);
}

```

If we take a closer look at the vase, we will see that the handle is occluding some voxels for some specific images. Hence these images, which have the best view to the occluded voxels, concerning the viewing angle, cannot see the voxels and therefore should not be concerned. In Figure 5-27 the example is depicted. We can see a slice from the vase and some camera positions. In three of those positions, the specified voxel is not visible, although these images would be best candidates according to the surface normal criterion. From the set of remaining images, the best candidate needs to be chosen.

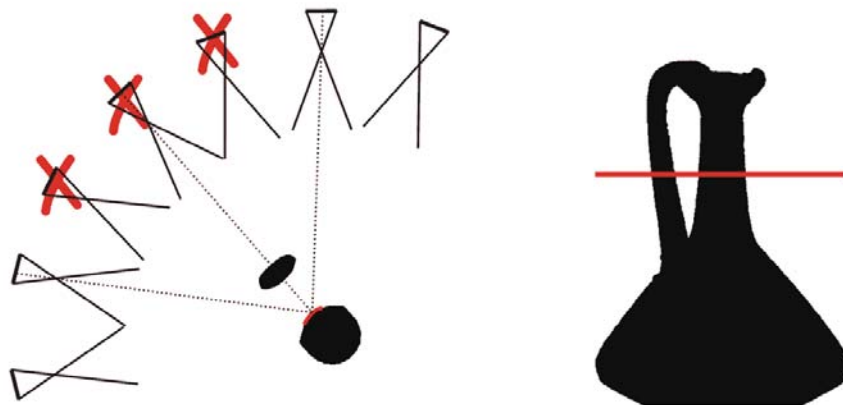
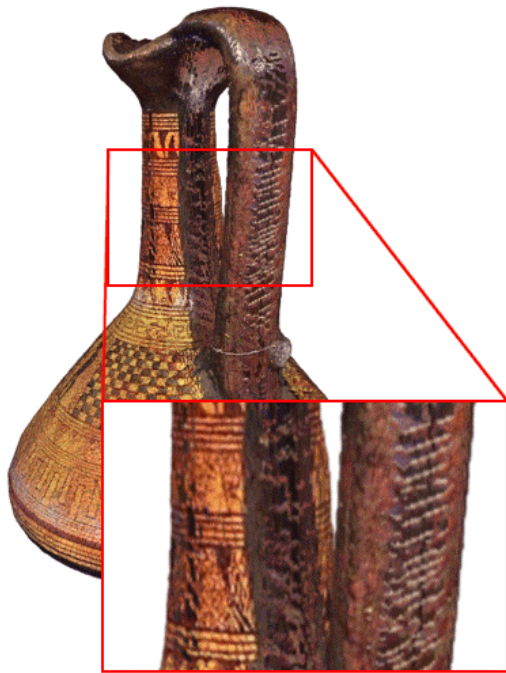


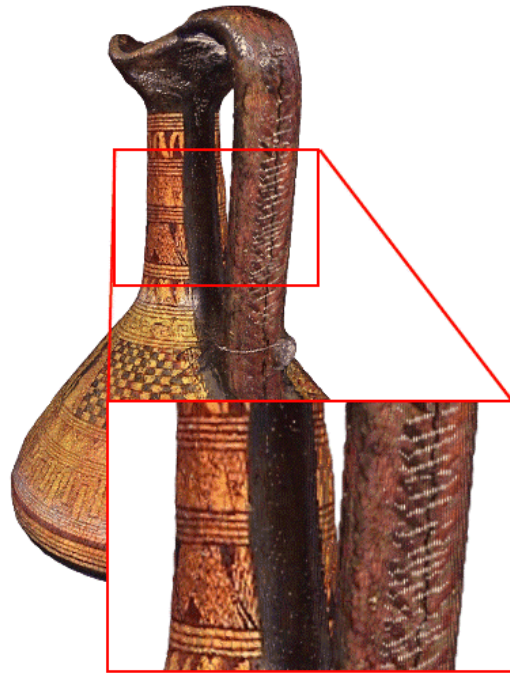
Figure 5-27: Considering occluded areas

To clarify this problem, we textured the above vase in the two described ways. It is not too obvious, but on the left image, where we did not consider visibility, the texture on the handle can also be seen underneath. Correctly textured, those voxels should contain a different texture, as shown in the right image, where they are correctly shown black.

This method of texture mapping shows very clearly, how we can achieve an impressive result with the use of a few simple tools. In this case we made use of the surface voxel list and the surface normal vector in combination with the line tracing algorithm. On the line tracing we applied the operator to recover visibility information, as introduced in chapter 5.1.4.



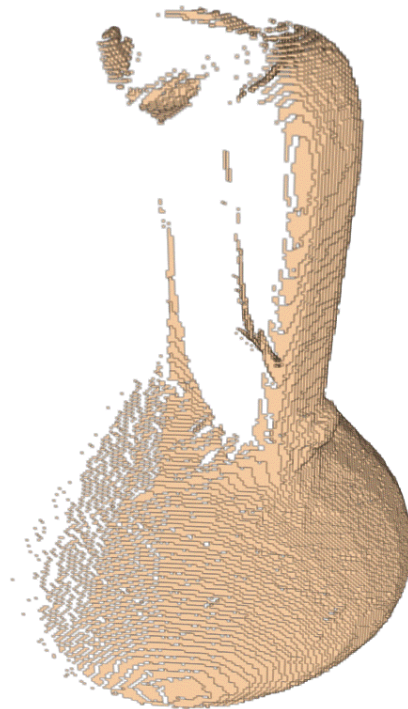
a) Projecting color regardless of visibility.



b) Projected visible color.



c) Reference image



d) Voxels visible in the reference image.

Figure 5-28: Projecting color without and with consideration of visibility

Chapter 6 : Contributions and Future Work

6.1 Contributions

For the problems stated in the Chapter 1.5, several algorithms have been proposed. In this thesis fast and robust voxel processing algorithms are used and the accuracy is improved by traditional photogrammetric methods. For example, the improvement of the image orientations by bundle block adjustment resulted in more reliable object reconstructions. Furthermore, since traditional photogrammetric orientation parameters are introduced to the images individually, we are no longer limited to a turntable setup, but instead we can chose our viewpoints freely.

The flexible definition of the voxel cube allows a good approximation of the bounding cube around the object minimizing areas of empty space and therefore allowing making better use of memory, resulting in a better voxel resolution.

Several algorithms for color image matching are proposed which give an advantage over matching the images by their grey values ignoring the information stored in the color channels.

The image matching by cross correlation is lacking the ability to compare highly distorted or even rotated image patches. The image matching, which is the means of comparing the image patches, was significantly improved by the knowledge-based patch distortion. Knowing the image orientation, a way to exploit this information is introduced in order to improve the robustness and reliability of cross correlation. Although time consuming, it is now insensitive against rotation and scaling. However, the found match can always be improved by least-squares-matching.

2D raster operations can be easily transferred to 3D space. By using 3D morphological filters, the noise artifacts can be removed from the voxel space.

In the reconstruction process, many tools have turned out to be very helpful. The surface voxel list is used in many occasions, for example in the computation of the surface normal vector. The surface normal vector, as the normal vector to the tangential surface, proved to be very helpful, when referring to an image from a voxel. This vector itself has been used for evaluating candidates for image matching.

Visibility recovery is very crucial to many tasks. Visibility information is computed by using voxel line tracing, which has turned out to be one of the most versatile and powerful tools, whose purpose serves beyond this thesis, since it has been programmed in the most flexible way. The finding of the first object voxel along the line of sight is yet another example for the simple use of the line tracing. A more complicated example is the calculation of the highest similarity of image patches, considering epipolar geometry. An operator is introduced that could solve this problem.

The approximate model from volume intersection is improved successfully by combining all these tools.

6.2 Future Work

The major limitation of creating the approximate model by shape from silhouette is the silhouette extraction which is performed by image segmentation. This requires a homogeneous background and therefore poses an obstacle to the camera setup. We always have to ensure to place the object in front of a homogeneous background, which is especially unlikely in outdoor projects under non perfect conditions. It is preferable to get rid of this limitation, by either finding an extraction method without homogeneous background or by getting rid of the approximation itself.

However, when worked under laboratory conditions, the results can be improved by optimizing the illumination. This would also allow us to support critical regions by other methods like shape from shading. Although the combination with the color image matching to get into the concavities gives good results for textured regions of the object, it is not reliable for non-textured regions.

In the future, we would like to check the current algorithms for further automation, for example the automatic measurement of tie points and the elimination of errors in the adjustment process. The scope of this thesis is to model 3D objects from its images, therefore automatic measurement of tie points were not investigated. The algorithms proposed here can be automated referring to the Ph.D. thesis of Rodehorst [RODEHORST, 2004].

Chapter 7 : Literature

- [ACKERMANN, 1984] Ackermann F.: Digital Image correlation: performance and potential application in photogrammetry, *The Photogrammetric Record*, 11(64), 1984, pp. 429-439.
- [ALIAGA, 2001] Aliaga D. G., Carlbom I.: Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs, *Proceedings of SIGGRAPH*, 2001, pp. 443-450.
- [AMANATIDES, 1987] Amanatides, J., Woo A.: A Fast Voxel Traversal Algorithm for Ray Tracing, *Proc. Eurographics '87*, pp. 1-10.
- [BOTTINO, 2001] Bottino A., Laurentini A.: Non Intrusive Silhouette Based Motion Capture, *Proc. Of 4th World Multiconference on Computer Graphics, Visualization and Computer Vision WSCG'2001*, pp. 5-9, 2001.
- [CHEN-I, 1993] Chen S. E., Williams L.: View interpolation for image synthesis, *Computer Graphics, SIGGRAPH'93*, pp. 279-288.
- [CHEN-II, 1995] Chen S. E.: Quick Time VR – an image based approach to virtual environment navigation, *Computer Graphics (SIGGRAPH'95)*, pp. 29-38.
- [CHIEN, 1986] Chien C. H., Aggarwal J. K.: Identification of 3D Objects from Multiple Silhouettes Using Quadrees/Octrees, *Computer Vision, Graphics, and Image Processing* 36, 1986, pp. 256-273.
- [COLLINS, 1996] Collins R. T.: A space-sweep approach to true multi-image matching, In *Proceedings Computer Vision and Pattern Recognition Conference*, 1996, pp. 358-363.
- [CULBERTSON, 1999] Culbertson W. B., Malzbender T., Slabaugh G.: Generalized Voxel Coloring, *International Workshop on Vision Algorithms*, Corfu, Greece, 1999, Springer Verlag Lecture Notes on Computer Science, ISBN 3-540-67973-1, pp. 100-115.
- [CURLESS-I, 1996] Curless B. and Levoy M.: A Volumetric Method for Building Complex Models from Range Images, *Proc. ACM SIGGRAPH 96*, 1996, pp. 303-312.
- [CURLESS-II, 1997] Curless B.: New methods for Surface Reconstruction from Range Images, *Ph.D. Thesis*, Stanford University, 1997.
- [DEBEVEC, 1996] Debevec P. E.: Modeling and Rendering Architecture from Photographs, *Ph.D. Thesis*, University of California at Berkeley, Computer Science Division, CA, 1996.
- [DE BONET, 1999] De Bonet J. S. and Viola P.: Responsibility Weighted 3D Volume Reconstruction, *Proceedings of the IEEE International Conference on Computer Vision*, 1999, Vol 1, pp. 415-425.

- [EBNER, 1987] Ebner H., Fritsch D., Gillessen W., Heipke C.: Integration von Bildzuordnung und Objectrekonstruktion innerhalb der Digitalen Photogrammetrie, Bildmessung und Luftbildwesen: Zeitschrift fuer Photogrammetrie und Fernerkundung, 1987, 55 (5):194-203.
- [DYER, 2001] Dyer C. R.: Volumetric Scene Reconstruction from Multiple Views, Foundations of Image Understanding, 2001, pp. 469-489.
- [EISERT, 1999] Eisert P., Steinbach E. and Girod B.: Multi Hypothesis, Volumetric Reconstruction of 3-D Objects from Multiple Camera Views, Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 1999, pp. 3509-3512.
- [FAUGERAS, 1998] Faugeras O., Keriven R.: Variational Principles, surface evolution, pde's, level set methods and the stereo problem, IEEE Transactions on Image Processing, 1998, vol. 7, no. 3, pp. 336-344.
- [GORTLER, 1996] Gortler S. J., Grzeszczuk R., Szeliski R., Cohen M. F.: The Lumigraph, SIGGRAPH'96, 1996, Computer Graphics Proceedings, Annual Conference Series, pp. 43-54.
- [GRUEN, 1985] Gruen A.: Adaptive least squares correlation: A powerful image matching technique, S Afr J of Photogrammetry, Remote Sensing and Cartography, Vol. 14, No. 3, 1985, pp. 175-187.
- [HAHN, 1995] Hahn M., Brenner C.: Area Based Matching of Colour Images, International Archives of Photogrammetry and Remote Sensing, vol. 30, part 5W1, Zurich, pp. 227-236.
- [HARTLEY, 2000] Hartley R., Zisserman A.: Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.
- [HORN, 1970] Horn B. K. P.: Shape From Shading: A New Method for Obtaining the Shape of a Smooth Opaque Object from One View, Ph.D. Thesis, Dept of EE, MIT, 1970.
- [HSBVIS] Sinram O.: <http://www.fpk.tu-berlin.de/~sinram/Cpp/>
- [JEBARA, 1999] Jebara T., Azarbajejani A., Pentland A.: 3D Structure from 2D Motion, IEEE Signal Processing Magazine, 1999, vol. 16, no. 3, pp. 66-84.
- [KAUFMAN-I, 1993] Kaufman A., Cohen D., Yagel R.: Volume Graphics, IEEE Computer, 1993, Vol. 26, No. 7, pp. 51-64.
- [KAUFMAN-II, 1996] Kaufman A.: Voxels as a Computational Representation of Geometry, SIGGRAPH Course Notes, 1996.
- [KOSCHAN-I, 1993] Koschan A.: Dense Stereo Correspondence Using Polychromatic Block Matching, In Proc. of the 5th Int. Conf. on Computer Analysis of Images and Patterns (CAIP'93),

- Chetverikov D., Kropatsch W. (eds.), Budapest, Hungary, 1993, pp. 538-542.
- [KOSCHAN-II, 1997] Koschan A., Rodehorst V.: Dense Depth Maps by Active Color Illumination and Image Pyramids, *Advances in Computer Vision*, Solina F., Kropatsch W.G., Klette R., Bajcsy R. (eds.), 1997, Springer, pp. 137-148.
- [KRAUS, 1997] Kraus K.: *Photogrammetry*, Dümmlers Verlag, 1997, Band 2. pp. 15-19, Band 1. pp. 349.
- [KUTULAKOS-I, 1998] Kutulakos K. N., Seitz S. M.: A Theory of Shape by Space Carving, *University of Rochester CS Technical Report 692*, 1998.
- [KUZU-I, 2001] Kuzu Y. and Rodehorst V.: Volumetric Modeling using Shape from Silhouette, *Fourth Turkish-German Joint Geodetic Days.*, 2001, pp. 469-476.
- [KUZU-II, 2002] Kuzu Y. and Sinram O.: Photorealistic Object Reconstruction using Voxel Coloring and Adjusted Image Orientations, *ACSM/ASPRS Annual Conference*, Washington DC, 2002, *Proceedings CD-ROM*, Proceed\00437.pdf.
- [KUZU-III, 2002] Kuzu Y.: Photorealistic object reconstruction using color image matching, *ISPRS Commission V Symposium 2002, Close-Range Vision Techniques*, Corfu, Greece, pp. 169-174, 2002.
- [KUZU-IV, 2003] Kuzu Y. and Sinram O.: Volumetric Reconstruction of Cultural Heritage Artifacts, *CIPA 2003, 19th International Symposium*, Antalya, Turkey.
- [LAURENTINI, 1995] Laurentini A.: How far 3D Shapes Can Be Understood from 2D Silhouettes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995, Vol.17, No.2.
- [LEVOY, 1996] Levoy M. and Hanrahan P.: Light Field Rendering, *Computer Graphics Proceedings, Annual Conference Series*, 1996, pp 31-42, *SIGGRAPH'96*.
- [LIPPMAN, 1980] Lippman A.: Movie-Maps: An Application of the Optical Videodisc to Computer Graphics, *Proceedings of SIGGRAPH 1980*, pp. 32-43.
- [LUHMANN, 2000] Luhmann T.: *Nachbereichsphotogrammetrie Grundlagen, Methoden und Anwendungen*, Herbert Wichmann Verlag, 2000, pp. 214-216, pp.463-464.
- [MAAS, 1992] Maas H. -G.: Robust Automatic Surface Reconstruction with Structured Light, *International Archives of Photogrammetry and Remote Sensing*, 1992, Vol. XXIX, Part B5, pp. 102-107.
- [MAAS, 1994] Maas H. -G., Stefanidis A., Gruen A.: From Pixels To Voxels: Tracking Volume Elements In Sequences Of 3-D Digital

- Images, International Archives Of Photogrammetry And Remote Sensing, 1994, Vol. 30, Part 3/2.
- [MARTIN, 1983] Martin W. N., Aggarwal J. K.: Volumetric Descriptions of Objects from Multiple Views, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983, Vol. PAMI-5, No.2.
- [MATHERON, 1975] Matheron G.: Random Sets and Integral Geometry, New York: Wiley, 1975.
- [MATUSIK, 2000] Matusik W., Buehler C., Raskar R., Gortler S. J., McMillan L.: Image-Based Visual Hulls, SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, 2000, pp. 369-374.
- [NAYAR, 1995] Nayar S., M. Watanabe, M. Noguchi: Real-time focus range sensor, Proceedings of IEEE International Conference on Computer Vision, 1995, pp.995-1001.
- [NIEM, 1994] Niem W.: Robust and Fast Modeling of 3D Natural Objects from Multiple Views, Proceedings of Image and Video Processing II, Vol. 2182, 1994, pp. 388-397.
- [OLIVERIA, 2002] Oliveira M. M.: Image-Based Modeling and Rendering Techniques: A Survey, RITA - Revista de Informática Teórica e Aplicada, October 2002, Volume IX, Number 2, pp. 37-66.
- [OSHER, 1988] Osher S. and Sethian J.: Fronts propagating with curvature dependent speed: Algorithms based hamilton-jakobi formulation, Journal of Computational Physics, 1988, vol. 79, pp. 12-49.
- [POTMESIL, 1987] Potmesil M.: Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images, Computer Vision, Graphics and Image Processing 40, 1987, pp.1-29.
- [PROCK, 1998] Prock A. C. and Dyer C.R.: Towards Real Time Voxel Coloring, Proc. 1998 Image Understanding Workshop, pp. 315-321.
- [RODEHORST, 2004] Rodehorst V.: Photogrammetrische 3D-Rekonstruktion im Nachbereich durch Auto-Kalibrierung mit projektiver Geometrie, Ph.D. Thesis, Berlin Technical University, 2004.
- [RUSINKIEWICZ-I, 2001] Rusinkiewicz S.: Sensing for Graphics, <http://www.cs.princeton.edu/courses/archieve/-fall01/cs597d>, 2001.
- [SEITZ-I, 1996] Seitz S. M., Dyer C. R.: View Morphing, Computer Graphics Proceedings SIGGRAPH'96, pp. 21-30.
- [SEITZ-II, 1997] Seitz S. M. and Dyer C. R.: Photorealistic Scene Reconstruction by Voxel Coloring, Proceeding of Computer Vision and Pattern Recognition Conference, 1997, pp. 1067-1073.

-
- [SERRA, 1982] Serra J.: Image Analysis and Mathematical Morphology, London: Academic Press, 1982.
- [SHADE, 1998] Shade J., Gortler S., He L. –W., Szeliski R.: Layered Depth Images, Computer Graphics Proceedings, SIGGRAPH'98, pp. 231-242.
- [SHUM, 1999] Shum H. –Y., He L. –W.: Rendering with concentric mosaics, Proceedings of SIGGRAPH'99, Los Angeles, pp. 299-306.
- [SLABAUGH-I, 2000] Slabaugh G., Culbertson W. B., Malzbender T. and Schafer R.: Improved Voxel Coloring via Volumetric Optimization, Technical Report 3, Center for Signal and Image Processing, Georgia Institute of Technology, 2000.
- [SLABAUGH-II, 2000] Slabaugh G., Malzbender T. and Culbertson W. B.: Volumetric Warping for Voxel Coloring on an Infinite Domain, Proceedings of the Workshop on 3D Structure from Multiple Images for Large Scale Environments (SMILE), 2000, pp. 41-50.
- [SLABAUGH-III, 2001] Slabaugh G., Culbertson W. B., Malzbender T. and Schafer R.: A survey of methods for volumetric scene reconstruction from photographs, In International Workshop on Volume Graphics, Stony Brook, New York, 2001.
- [SLABAUGH-IV, 2002] Slabaugh G., Schafer R. and Hans M.: Multi-Resolution Space Carving with Level Set Methods, in ICIR, 2002.
- [SRIVASTAVA, 1990] Srivastava S. K., Ahuja N.: Octree Generation from Object Silhouettes in Perspective Views, Computer Vision, Graphics and Image Processing 49, 1990, pp. 68-84.
- [STEINBACH, 2000] Steinbach E., Girod B., Eisert P., Betz A.: 3-D Object Reconstruction Using Spatially Extended Voxels and Multi-Hypothesis Voxel Coloring, Proc. ICPR'00, Barcelona, Spain, 2000.
- [STEVENS, 2002] Stevens R., Culbertson B., Malzbender T.: A Histogram Based Color Consistency Test for Voxel Coloring, Proceedings of ICPR, Quebec City, Quebec, Canada, 2002.
- [SZELISKI-I, 1990] Szeliski R.: Real-time octree generation from rotating objects, Technical Report 90/12, Digital Equipment Corporation, Cambridge Research Lab, December 1990.
- [SZELISKI-II, 1991] Szeliski R.: Shape from rotation, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91), 1991, pp. 625-630.
- [SZELISKI-III, 1994] Szeliski R. and Kang S. B.: Recovering 3D shape and motion from image streams using nonlinear least squares, Journal of Visual Communication and Image Representation, March 1994, 5(1):10-28.

- [SZELISKI-IV, 1998] Szeliski R. and Golland P.: Stereo Matching with transparency and Matting, *International Journal of Computer Vision*, 1998, pp. 517-523.
- [SZELISKI-V, 2001] Szeliski R.: Image and Video- Based Modeling and Rendering, The 4th International Workshop on Cooperative Distributed Vision, March 22-24, Kyoto, Japan, 2001.
- [TOMASI, 1992] Tomasi C., Kanade T.: Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 1992, 9(2):137-154.
- [VENDULA, 1998] Vedula S., Rander P., Saito H., Kanade T.: Modeling, Combining, and Rendering Dynamic Real-World Events from Image Sequences, *Proc. 4th Conference on Virtual Systems and Multimedia (VSMM98)*, November, 1998, pp. 326-332.
- [TVHP] The Visible Human Project, National Library of Medicine, http://www.nlm.nih.gov/research/visible/visible_human.html
- [WIEDEMANN, 2000] Wiedemann A., Hemmleb M., Albertz J.: Reconstruction of Historical Buildings Based on Images from the Meydenbauer archives, *IAPRS*, Vol. XXXIII, Amsterdam 2000, B5/2, pp. 887-893.
- [WROBEL, 1987] Wrobel B.: Digitale Bildzuordnung durch Facetten mit Hilfe von Objektraummodellen, *Bildmessung und Luftbildwesen*, 1987, Vol. 55, No 3, pp. 93-101.

Curriculum Vitae

Surname, Name: Kuzu, Yasemin

Date of Birth: 21. 01. 1971

Nationality: Turkish

Education:

- | | |
|-----------|--|
| 1999-2004 | Berlin Technical University, Faculty of Civil Engineering and Applied Geosciences, Division of Photogrammetry and Cartography, Ph.D. |
| 1993-1996 | Bogazici University, Kandilli Observatory and Earthquake Research Institute, Geodesy Department, M. Sc. Eng. |
| 1988-1992 | Yildiz Technical University, Faculty of Civil Engineering, Department of Geodesy and Photogrammetry, BS. |
| 1984-1987 | Erenkoy Kiz Lisesi, High School Degree |

Work Experience:

- | | |
|--------------|---|
| 1998-Present | Yildiz Technical University Division of Photogrammetry and Remote Sensing
Research Assistant |
| 1996-1998 | Baytur & Weidleplan J. V.
Chief engineer of Infrastructure Department. |
| 1992-1996 | Municipality, Surveying Engineer |