NOTE TO USERS

Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DE ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Rimon Zarif ELIAS

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Ph.D.(Computer Science)

GRADE - DEGREE

School of Information, Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Towards Obstacle Reconstruction through Wide Baseline Set of Images

R. Laganière

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

É. Dubois

A. Mitiche

D. Nussbaum

E. Petriu

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Towards Obstacle Reconstruction Through Wide Baseline Set of Images

By

Rimon Elias

*A Thesis Submitted to the Faculty of Graduate and Postdoctoral Studies of the University of Ottawa in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science**

* The Ph.D. program in Computer Science is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

Your file Votre référence
ISBN: 0-494-01699-X
Our file Notre référence
ISBN: 0-494-01699-X

Abstract

In this thesis, we handle the problem of extracting 3D information from multiple images of a robotic work site in the context of teleoperation. A human operator determines the virtual path of a robotic vehicle and our mission is to provide him with the sequence of images that should be seen by the teleoperated robot moving along this path.

The environment, in which the robotic vehicle moves, has a planar ground surface. In addition, a set of wide baseline images are available for the work site. This implies that a small number of points may be visible in more than two views. Moreover, camera parameters are known approximately. According to the sensor error margins, the parameters read lie within some range. Obstacles of different shapes are present in such an environment. In order to generate the sequence, this ground plane as well as the obstacles must be represented. The perspective image of the ground plane can be obtained through a homography matrix. This is done through the virtual camera parameters and the overhead view of the work site. In order to represent obstacles, we suggest different methods; these are volumetric and planar.

Our algorithm to represent obstacles starts with detecting junctions. This is done through a new fast junction detection operator we propose. This operator provides the location of the junction as well as the orientations of the edges surrounding it. Junctions belonging to the obstacles are identified against those belonging to the ground plane through calculating the inter-image homography matrices.

Fundamental matrices relating images can be estimated roughly through the available camera parameters. Strips surrounding epipolar lines are used as a search range for detecting possible matches. We introduce a novel homographic correlation method to be applied among candidates by reconstructing the planes of junctions in space. Two versions of homographic correlation are proposed; these are SAD and VNC. Both versions achieve matching results that outperform non-homographic correlation. The match set is then turned into a set of 3D points through triangulation. At this point, we propose a hierarchical structure to cluster points in space. This results in bounding boxes containing obstacles. A more accurate volumetric representation for the obstacle can be achieved through a voxelization approach.

Another representation is suggested. That is to represent obstacles as planar patches. This is done through mapping among original and synthesized images. Finally, steps of the different algorithms presented throughout the thesis are supported by examples to show the usefulness we claim of our approaches.

Keywords

obstacle modeling, 3D modeling, 3D reconstruction, JUDOCA, junction detection, stereo vision, binocular vision, wide baseline matching, sparse view matching, homographic correlation, point clustering, voxel occupancy, voxelization.

Contents

List of Figures

List of Tables

List of Symbols

Axes

X_p	Pixel (image) $x-$axis	19
X_w	World (space) $x-$axis	19
Y_p	Pixel (image) $y-$axis	19
Y_w	World (space) $y-$axis	19
Z_w	World (space) $z-$axis	19

Angles

ϑ	The rotation angle used with the similarity matrix	142
μ	The inclination angle of an edge of a junction detected by JUDOCA operator	91
$\rho_{(n)}$	A viewing direction at a point $\dot{\mathbf{m}}_n$	144
$\rho_{(k,k+1)}$	The direction of the motion between two points $\dot{\mathbf{m}}_k$ and $\dot{\mathbf{m}}_{k+1}$	143
φ	The rotation angle used to compensate for the swing effect	146

Points

$\dot{\mathbf{m}}$	A 2D vector representing a 2D point expressed in inhomogeneous coordinates	16
$\dot{\mathbf{m}}'$	A 2D vector representing a 2D point, corresponding to $\dot{\mathbf{m}}$, expressed in inhomogeneous coordinates	31
$\mathbf{m}$	A 3D vector representing a 2D point expressed in homogeneous coordinates, a 2D line, or a direction (where the last term = 0)	16
$\mathbf{m}'$	A 3D vector representing a 2D point, corresponding to $\dot{\mathbf{m}}$, expressed in homogeneous coordinates	16
$\hat{\dot{\mathbf{m}}}$	A 2D vector representing the estimated position of a 2D point, $\dot{\mathbf{m}}$, expressed in inhomogeneous coordinates	33
$\hat{\dot{\mathbf{m}}}'$	A 2D vector representing the estimated position of a 2D point, $\dot{\mathbf{m}}'$, expressed in inhomogeneous coordinates	33

Points (continued)

$\hat{\mathbf{m}}$	A 3D vector representing the estimated position of a 2D point, $\mathbf{m}$, expressed in homogeneous coordinates	33
$\hat{\mathbf{m}}'$	A 3D vector representing the estimated position of a 2D point, $\mathbf{m}'$, expressed in homogeneous coordinates	33
$\mathbf{m}_l$	A 3D vector representing the intersection point between an epipolar line and the left image edge	89
$\mathbf{m}_r$	A 3D vector representing the intersection point between an epipolar line and the right image edge	89
$\dot{\mathbf{M}}$	A 3D vector representing a 3D point expressed in inhomogeneous coordinates	16
$\hat{\dot{\mathbf{M}}}$	A 3D vector representing the estimated position of a 3D point, $\dot{\mathbf{M}}$, expressed in inhomogeneous coordinates	31
$\dot{\mathbf{M}}_\infty$	A 3D vector representing the intersection of the optical ray passing through the point $\dot{\mathbf{M}}$ and its image $\dot{\mathbf{m}}$ with the plane at infinity	31
$\dot{\mathbf{M}}'_\infty$	A 3D vector representing the intersection of the optical ray passing through the point $\dot{\mathbf{M}}$ and its image $\dot{\mathbf{m}}'$ with the plane at infinity	31
$\tilde{\mathbf{M}}$	A 3D vector representing a 3D point on the planar ground surface expressed in homogeneous coordinates	67
$\mathbf{M}$	A 4D vector representing a 3D point expressed in homogeneous coordinates or a direction (where the last term = 0)	16
$\mathbf{M}_{new}$	A 4D vector representing a 3D point $\dot{\mathbf{M}}$ under 3D transformation	22

Lines

$\mathtt{L}$	A 2×4 matrix representing a line in 3D space	17
$\mathbf{l}$	A 3D vector representing the epipolar line on the first image plane in a binocular system	23
$\mathbf{l}'$	A 3D vector representing the epipolar line on the second image plane in a binocular system	23
$\mathbf{l}_\infty$	A 3D vector representing the line at infinity	28
$\mathbf{l}'_a$	A 3D vector representing an edge of the strip used as a search range for matching	89
$\mathbf{l}'_b$	A 3D vector representing the other edge of the strip used as a search range for matching	89
$\mathbf{l}_l$	A 3D vector representing the left image edge	88
$\mathbf{l}_r$	A 3D vector representing the right image edge	88

Planes

Π	A 4D vector representing a plane	17
Π'	A 4D vector representing the second image plane in a binocular system	23
Π_g	A 4D vector representing the equation of the ground plane	67
Π_{new}	A 4D vector representing the plane equation expressed in a new coordinate system	67
Π_i	A 4D vector representing a plane formed through the optical center and line $\mathbf{l}_i$	91
Π'_i	A 4D vector representing a plane formed through the optical center and a line $\mathbf{l}'_i$	91
$\mathbf{N}$	A 3D vector representing the normal to the plane	93
$\mathbf{N}_{new}$	A 3D vector representing the normal to the plane expressed in the new coordinate system	67
$\mathbf{N}_i$	A 3D vector representing the normal to the plane formed through the optical center and a line $\mathbf{l}_i$	92
$\mathbf{N}'_i$	A 3D vector representing the normal to the plane formed through the optical center and a line $\mathbf{l}'_i$	92
d	A scalar quantity, the fourth term in a 4D plane vector, representing the distance from the origin to the plane	29
d_{new}	A scalar quantity, the fourth term in a 4D plane vector, representing the distance from the origin to the plane expressed in the new system coordinates	67
d_i	A scalar quantity, the fourth term in a 4D plane vector, representing the distance from the origin to the plane formed through the optical center and a line $\mathbf{l}_i$	92
d'_i	A scalar quantity, the fourth term in a 4D plane vector, representing the distance from the origin to the plane formed through the optical center and a line $\mathbf{l}'_i$	92

Images

$\mathcal{B}$	A binary image created by imposing a threshold on the gradient image	40
$\mathcal{B}^+$	A binary image containing the points of $\mathcal{B}$ that are local maxima in the direction of the gradient	40
$\mathcal{I}$	An image	40
$\mathcal{I}(x, y)$	The intensity at the pixel $[x, y]^T$	63
$\mathcal{I}_j$	An image number j	130
w	The width of an image	89

Clusters

$\mathcal{C}_{(i,j)}$	A cluster, $\mathcal{C}_j$, consists of a number points at level $\mathcal{L}_{(i)}$	118
$\mathcal{L}_{(i)}$	A level consists of a number of clusters of points	117

Camera Intrinsic Parameters

α_u	$= fk_u$, see below	20
α_v	$= fk_v$, see below	20
γ	The skew element	20
θ	The angle between retinal axes	20
f	The focal length of the camera	20
k_u, k_v	The horizontal and vertical scale factors. Their inverses determine the size of the pixel in world coordinate units	20
r	The aspect ratio $= \frac{\alpha_u}{\alpha_v} = \frac{f\ k_u}{f\ k_v}$	20
$[u_0, v_0]^T$	The principal point of the camera	20

Camera Extrinsic Parameters

t_x	Translation along the $x-$axis	55
t_y	Translation along the $y-$axis	55
t_z	Translation along the $z-$axis	55
ω	The first rotation angle in the ω, ϕ, κ system	57
ϕ	The first rotation angle in the ω, ϕ, κ system	57
κ	The third rotation angle in the ω, ϕ, κ system	57
ρ	The first rotation angle in the ρ, τ, ψ system	59
τ	The second rotation angle in the ρ, τ, ψ system	59
ψ	The third rotation angle in the ρ, τ, ψ system	59

Optical Centers

$\dot{\mathbf{C}}$	A 3D vector representing the optical center of the camera in inhomogeneous coordinates	21
$\dot{\mathbf{C}}'$	A 3D vector representing the second optical center in a binocular system expressed in inhomogeneous coordinates	23
$\mathbf{C}$	A 4D vector representing the optical center of the camera in expressed homogeneous coordinates	21
$\mathbf{C}'$	A 4D vector representing the optical center of the second camera expressed in homogeneous coordinates in a binocular system	32

Epipoles

Calibration Matrices

Rotation Matrices

Translation Vectors

$\mathbf{T}$	A 3D vector representing the translation of the camera	62
$\mathbf{T}'$	A 3D vector representing the translation of the second camera in a binocular system	67
$\mathbf{T}'_{new}$	A 3D vector representing the translation of the second camera in a binocular system expressed in the new coordinate system	68
$\mathbf{t}$	A 2D vector representing the translation	28
$\mathbf{t}_{lr}$	A 2D vector representing the translation that deviates from the point detected by the homography matrix along the left-right direction	69
$\mathbf{t}_{rl}$	A 2D vector representing the translation that deviates from the point detected by the homography matrix along the right-left direction	73

Projection Matrices

$\mathtt{P}$	The 3×4 perspective projection matrix	18
$\mathtt{P_a}$	$= \mathtt{AP_c}$, a 3×4 matrix, representing the simple case of the perspective projection matrix where $f \neq 1$	??
$\mathtt{P_c}$	The 3×4 matrix, $[\mathtt{I}\vert\mathtt{0}]$, representing the simple case of the perspective projection matrix	18
$\mathtt{P}'$	The 3×4 perspective projection matrix of the second camera in a binocular system	68
$\mathtt{P}_{new}$	The 3×4 perspective projection matrix under transformation	23
$\mathtt{P}'_{new}$	The 3×4 perspective projection matrix of the second camera in a binocular system expressed in the new coordinate system	68
$\tilde{\mathtt{P}}$	A 3×3 matrix composed of the first 3 columns of the perspective projection matrix	21
$\tilde{\mathtt{P}}'$	A 3×3 matrix composed of the first 3 columns of the perspective projection matrix of the second camera in a binocular system	32
$\mathtt{P}^+$	The pseudo-inverse of $\mathtt{P}$	24
$\mathbf{P}_i$	A 4D vector representing the $i-$th row of the perspective projection matrix	21
$\dot{\mathbf{P}}$	A 3D vector representing the $4-$th column of the perspective projection matrix	21
$\dot{\mathbf{P}}'$	A 3D vector representing the $4-$th column of the perspective projection matrix of the second camera in a binocular system	25

Transformations

$\mathtt{B}$	The 3×3 affine transformation matrix	28
$\mathtt{C}$	The 2×2 scaling matrix	142
$\mathtt{E}$	The 3×3 essential matrix	26
$\mathtt{F}$	The 3×3 fundamental matrix	24
$\mathtt{H}$	The 3×3 homography matrix	27
$\mathtt{H}_z$	The 3×3 planar perspective matrix	55
$\mathtt{H}_g$	The 3×3 matrix representing the homography of the planar ground surface between two views	67
$\mathtt{H}_{jk}$	The 3×3 matrix representing the homography of the planar ground surface between two images j and k	130
$\mathtt{S}$	The 3×3 similarity matrix	28
$\mathtt{T}$	A 4×4 non-singular matrix used for 3D transformation	22
$\mathtt{V}$	The 3×3 pure projective transformation matrix	28

Chapter 1

Introduction

The ultimate goal of Computer Vision is to simulate the vision system in human beings. Humans can visually recognize the surrounding environment using a sequence of calibrated stereo two-dimensional images captured by their eyes. In this case,

1. the parameters of the camera; i.e., the eye, are known;
2. color images are transferred to the brain;
3. the sequence is of stereo type;
4. the baseline; i.e, the distance between the two eyes, is small;
5. certain cells in the visual pathways of the brain often have an approximately Gaussian response [FPWW94] that can be used to reduce the level of noise.

As a result of the above-mentioned facts, humans can understand the surrounding environment in a three-dimensional fashion. This information is valuable for making decisions; e.g., avoiding obstacles as they move.

As a matter of fact, the problem of 3D reconstruction of objects is the center of interest for many researchers in different fields, in addition to Computer Vision, including Computer Graphics, Computer Aided Design (CAD), Virtual Reality and Computer Animation [BS91, CC92, MBL+91, PS91].

In this thesis, we propose to study the problem of building a 3D representation of an environment in the context of a telerobotic application where mobile robots, equipped with cameras, have to perform some tasks in a distant work area.

1.1 Teleoperation

Telerobotics may be the answer for machinery operating in harsh environments; e.g., mining, waste disposal, forestry, or oil platform maintenance environment. However, conventional teleoperation techniques require direct operator-equipment transactions and create three major disadvantages [AAH+]:

1. The complex machine-environment interactions produce continual operator-equipment dialogs that require the operator's constant attention.
2. Collisions or control errors may result from the direct nature of the transactions;
3. Transmission delays to and from the operator's site affect the efficiency, relevance and safety of the specifications.

In order to avoid the above-mentioned disadvantages, an alternative solution, called SMART,[1] has been proposed by a number of Canadian universities. It is based on the concept of augmented reality, which enables the dissociation of key aspects of task specification and task execution. This endeavor is centered on four main schemes [AAH+]:

1. Interactive 3D sensing: This field is concerned with limiting the amount and frequency of data transmitted (to the human operator from the work site) to a small representative subset of all necessary information. Minimizing the amount of data involves reducing the number of images and viewpoints transmitted to the operator. In order to compensate, the system should provide the teleoperator with 3D interpretation of the scene. Moreover, the remaining portion of the work site should be synthesized upon request by the human operator.
2. Enhanced sensori-motor interaction: Mediating between task specification and task execution through the embedding of virtual replica of the equipment in the scene as perceived by the operator.
3. Intelligent mediation system: The system should be intelligent enough to monitor sequential autonomous tasks and decide whether human operator assistance is required. This allows the operator to introduce new commands, and modifies objective and control using direct teleoperation.
4. Intelligent control: This refers to the capability of a control system to execute subtasks previously defined by a human operator.

Our work in this thesis falls under the first scheme. It aims at extracting 3D information from the environment and providing the operator with a sequence of synthesized images that

[1] SMART stands for Sensori-Motor Augmented Reality for Telerobotics.

represent the remaining part of the site. In the next sections, we will closely define our segment of the problem. we will also state the assumptions we have considered.

1.2 Definition of the Problem

In the SMART project, the mining environment was targeted to illustrate the concepts developed. In such an environment, teleoperated machinery operates over a relatively flat work surface. Furthermore, rocks of different sizes and shapes may also exist, constituting obstacles that must be avoided when moving the equipment. It is therefore not an easy task for a robotic vehicle to navigate in these conditions. In today's teleoperation technology, this requires the human operator's uninterrupted attention. Also, a great deal of data is transmitted to and from the operator (e.g., commands and images), which slows down the operation. This suggests that we should increase the level of automation by developing intelligent control systems, which implies less supervision on the part of the human operator.

Using the concept of 3D sensing, only a small number of natural images of widely separated views of the work site are taken as a first step, using cameras mounted on the navigating robotic vehicle. This is combined with *approximate* camera parameters; i.e., position and orientation, at every viewpoint. Camera parameters are assumed to be obtained through the use of sensors such as GPS or inertial devices [BDU+99].[2] However, the accuracy of these measurements will be limited by the error tolerance of the devices. Any information inferred from these measurements must take these uncertainties into account.

In order to provide the operator with information about the remaining portion of the work site, 3D information must be extracted from the available images to generate a virtual sequence of images that can be seen by the robotic vehicle along a defined path. In other words, the system should have the capability to generate views of extended portions of terrain, including obstacles. This helps make decisions, e.g., avoiding obstacles or changing paths.

From the definition of the problem, we can assume that:

1. the ground on which the obstacles reside is flat. This allows us to establish a relationship among the projections of the ground in different images. Hence, the obstacles can be detected and the matching process can be facilitated;
2. several views of the work site are available. These views are widely separated (i.e., the differences among successive camera poses are large) so that the amount of data transmitted to the human operator is reduced as well; and

[2]The examples presented in this thesis use another source of information: PhotoModeler Pro software copyrighted by Eos Systems Inc. This software provides the three parameters for the position of the camera and other three parameters for the rotation angles for every image in a given set of images.

3. the parameters of the camera are known for each view. This includes three parameters that determine its position and three others representing its orientation about all three axes. Based on the sensor specifications, the values of these parameters are within a certain range.

From the above-mentioned assumptions, our goal is to obtain a representation of the site showing the work surface, the obstacles lying over it through a virtual sequence along a predetermined path; i.e, each frame of this sequence represents the scene as observed by a camera moving along that path.

To reach this goal, different – but related – research areas contribute to our investigation. These are:

- Projective Geometry (e.g., homography estimation in Chapter 2); and
- Computer Vision (e.g., feature correspondence in Chapter 5)

that are the main fields of our research. In addition,

- Image Processing (e.g., feature detection in Chapter 3); and
- Computer Graphics (e.g., mapping in Chapter 7)

are important complementary areas.

The last point to emphasize is that this thesis does not consider the telerobotics, teleoperations or telecommunications aspects of the project. It assumes that images have been correctly captured and transmitted and that our task is to obtain three-dimensional information from them.

1.3 Related Work

Many papers discuss projects related to SMART. For example, [THF+03] presents two systems for acquiring accurate volumetric maps of underground mines. [DK01] describes a real time localization and map-building system. (Localization is the problem of determining a robot's pose from sensor data, where the term "pose" refers to the robot x-y-coordinates in the environment along with its heading direction [TBB+00].) Mapping is defined here as the problem of estimating the occupancy of all locations in the environment [BCF+99].) Some other works discuss non-vision aspects of projects similar to SMART. For example, [RC01, DeS97] discuss the kinematics and control of vehicles used in mining operations. However, in this section we will concentrate our discussion on works related to the vision-based aspects of the SMART project.

Pears and Bojian presented a system in which they integrate multiple visual cues in order to help a mobile robot navigate in an indoor work site [PL02, Pea02]. (A visual cue is a source of information in an image or sequence of images, such as edges, corners, colors, texture or motion [Pea].) Their idea was to detect and segment the ground plane from the rest of the scene using monocular uncalibrated camera [PL01]. Corners are first detected using the Plessy-Harris operator. Those corners are tracked in n frames and grouped to form coplanar regions using a proposed method they called *H-based tracker*. Ground plane regions are then detected and region-growing algorithm using color classification is applied to achieve ground plane segmentation. An improvement to quadtree-based split-and-merge segmentation has been developed [LP02a]. In [LP02b], they presented some of their achievements. One of them was the computation of the plane homography (see Section 2.3.5) when the camera motion was restricted to pure translation; i.e., elation. In this case, there are only four degrees of freedom and hence only two pairs of corresponding points are used to compute the homography. Also in [LP02b], they presented a new method for applying planar homographies to measure the height of some feature above the ground. Using this information, a decision can be made as to whether the point belongs to the ground plane; i.e., if height reaches zero, and hence the robot may drive over this point, or if the point belongs to obstacles and must be avoided or driven under if they are overhanging and of sufficient height.

In [CJZ03], Cobzas *et al.* presented a cylindrical image-depth model that can be automatically acquired and then used to support both robot localization and predictive display. They presented a system that provides real time feedback images that replace delayed real images. Their system consists of three main modules. In the first module, a panoramic image is captured by rotating a camera about a vertical axis passing through its optical center. In addition, a laser range-finder is used to provide depth values. Data are then registered with respect to a common reference frame. This is followed by plane fitting and line detection. The second module is the online localization system that matches line features in the model with those of the current view. This allows the robot location to be acquired. (Refer to [CZ00a] for the localization problem as well.) As a result, the predicted view can be rendered which represents the third module. Note that no full 3D reconstruction is achieved in this system.

RHINO, an autonomous interactive tour-guide robot software, is presented in [BCF+99]. This robot was built to entertain people in public places; e.g., museums. The software, which consists of 20 modules, addresses many challenges including localization, mapping, collision avoidance, planning, and Web-based telepresence. For example, the collision avoidance module which they called μDWA, addresses the problems of the inability to handle invisible obstacles and consider dynamics. It uses a hand-supplied map for the environment to avoid collision with obstacles that cannot be sensed. Collision avoidance and reaction to people are related in this system. A number of actions are taken by the robot in order to avoid

collision ranging from slowing down to blowing the horn when the obstacle is not on the map (e.g., a person), to finding a local detour and finally finding a global detour. Minerva is a second generation interactive tour-guide robot developed by the same team of researchers and presented in [TBB+00].

In the following section, we will present the elements of the system that we developed to solve our problem of extracting 3D information and representing the work site.

1.4 Overview of the Proposed Scene Modeling Approach

As mentioned above, our goal is to obtain a representation of the site showing the surface, and the obstacles and their positions such that virtual exploration of the site becomes possible. In other words, our task is to perform scene modeling. In our case, the modelization requires detecting, locating and representing the obstacles on the ground plane. We can summarize our scene modeling approach into two main streams:

1. representing the ground plane; and
2. representing obstacles.

These objectives may be achieved through the procedure illustrated in Figure 1.1 on Page 8, which can be summarized as follows. Using the camera parameters, a rough estimation for the representation of the planar work surface is obtained. Through the available error margins of the sensors used, we may identify junction features that putatively belong to obstacles; i.e., those that are not lying on the ground plane.

Based on the epipolar geometry derived from the available parameters, these junctions must be matched. The matching process leads to 3D point reconstruction resulting in a set of 3D points. These points are then grouped in clusters based on some proximity criteria. The points in each cluster are used to define a bounding box inside which an obstacle must be present.

Finally, a representation for each obstacle is to be obtained. In addition to the bounding boxes, two alternative models are proposed: the first approach involves a full 3D reconstruction for each obstacle based on a voxel occupancy scheme defined across the bounding box; the other approach approximates the obstacle as planar patches where silhouette images or the projections of the bounding boxes are used as opacity maps.

Using the above-mentioned ideas, we can split the scene modeling problem into:

1. representing the ground plane;
2. extracting feature points;
3. detecting feature points on obstacles;
4. matching feature points;
5. estimating the three-dimensional positions for feature points;
6. clustering points that leads to bounding boxes; and
7. rendering obstacles which includes:
   (a) three-dimensional rendering through voxel occupancy; and
   (b) planar rendering through opacity mapping.

In the following sections, we analyze each of the above-mentioned subproblems.

1.4.1 Ground Plane Representation

As mentioned, the ground level of the work site is assumed to be planar. If the top view of the scene is obtained as an image, the relationship between this image and the projection of the planar ground in the synthesized view sought will be a 3×3 homography matrix. This matrix can be calculated using the virtual camera parameters. This is the "Ground Plane Representation" module represented in Figure 1.1. The inputs to this module are the overhead view image, the camera parameters and the virtual camera parameters. The output is used to represent the ground plane where obstacles are represented as bounding boxes, 3D structures or planar patches. The output is also used as a primary step to establish relationships among the projections of the ground plane in the different images.

1.4.2 Feature Detection

Features are used to refine the homography, epipolar geometry and to detect and locate the obstacles. Different types of features, such as corners, can be used at this step. However, we are looking for a feature type that can provide a richer information. In our case, we want to detect not only the position of the corner but also its orientation; i.e., the inclination angles of the edges forming that corner, in order to facilitate the matching process. In other words, we want to detect junctions of different shapes. This is the module called "Junction Detection" in Figure 1.1. The input to this module is composed of the available images. The output is used as an input to the next module; i.e., "Junction Selection."

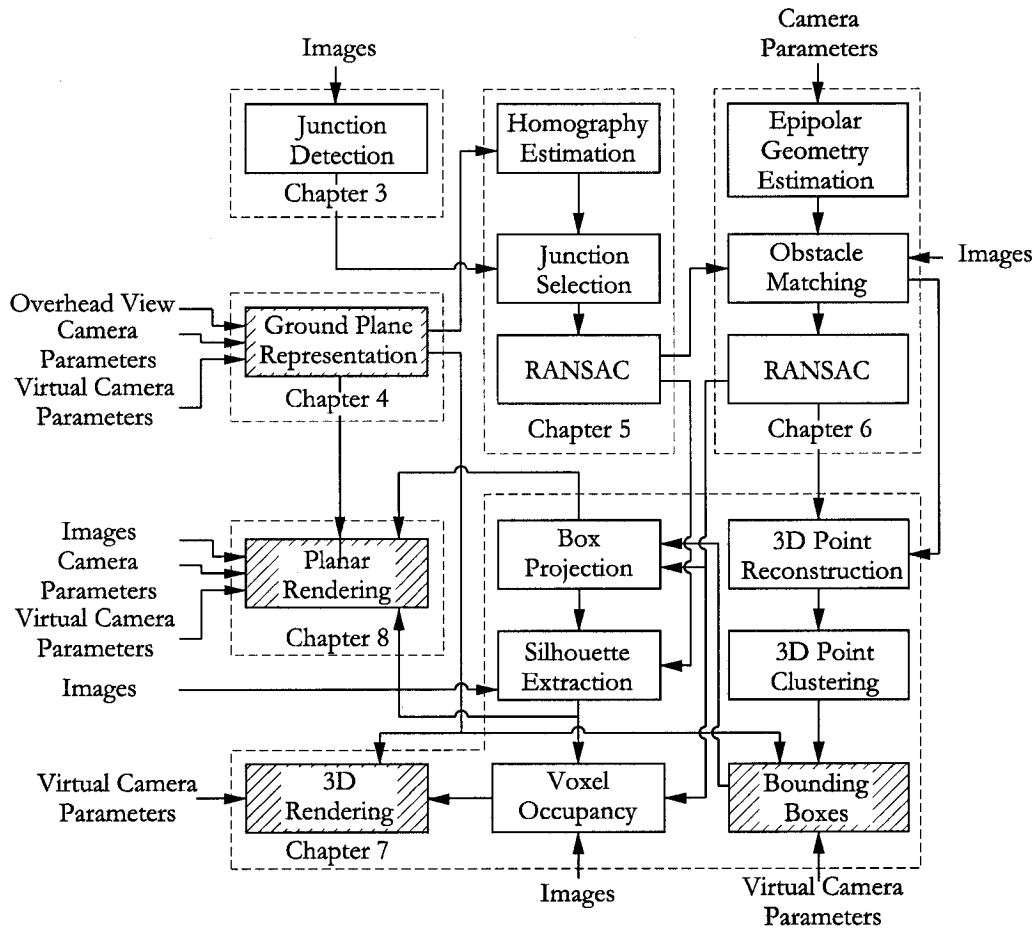


Figure 1.1: Scene modeling system. The hatched modules represent elements of the output. The dashed lines represent the limits of each chapter.

1.4.3 Junction Selection

Many texture features may exist on the ground plane along with those on obstacles. Since our goal is to match features on obstacles only in order to localize them, there is a need to extract junctions that are on the obstacles, or in other words, exclude junctions that are not part of the obstacles. That can be achieved through the matrix mentioned in Section 1.4.1. This step is represented by three modules in Figure 1.1. The input to the first module, "Homography Estimation," is the homography matrix calculated in Section 1.4.1 and obtained through the original approximate camera parameters. The output of this module along with the complete set of junctions detected previously in Section 1.4.2 are the inputs of the second, "Junction Selection," in which junctions are selected and passed to the next module, named "RANSAC" where homography matrices may be refined using the match set that belong to the ground plane. Hence, more ground features can be detected as a result. Consequently, other features that putatively belong to the obstacles are passed to the next step; i.e., the matching process. Also, the output of the "RANSAC" module is used as an input to the "Silhouette Extraction" module.

1.4.4 Wide Baseline Matching

Wide baseline or sparse view matching step is represented by three modules in Figure 1.1 called "Epipolar Geometry Estimation," "Obstacle Matching" and "RANSAC."

The input to the first module is composed of the approximate camera parameters. Through this information, we can calculate the epipolar geometry where a corresponding point should lie on a specific line. Due to the inaccuracy of camera parameters, a point may not lie exactly on the line but in its vicinity. We therefore consider a strip whose shape and width can be determined according to the error margins of the camera parameters. Using this strip, the search for possible matches can be restricted to fewer candidates. The output of this module is used by the "Obstacle Matching" module.

The inputs to the second module, "Obstacle Matching," are three types of information. These are: the strips obtained through the epipolar geometry; the features that survived the previous step (Section 1.4.3); and the original images. The strips are used to restrict the search area when looking for matching candidates. The second piece of information is used to construct a planar patch in space that can be formed through the edges of the candidate junction. Using the third input; i.e., the images, and through correlation techniques, we determine the best candidate for a given feature. The last stage in the matching process is disambiguating the match set. This is done by calculating the epipolar geometry that agrees with most of the matches. This is followed by rejecting matches that do not agree with

the new epipolar geometry. The output of this module is the improved match set. This is represented by the module "RANSAC."

1.4.5 Three-Dimensional Point Reconstruction

Given that obtaining 3D information for the scene is of primary concern, in this step we try to convert our information into 3D form. This step is represented by only one module called "3D Point Reconstruction" whose input is the match set obtained in Section 1.4.4. Every match pair contributes to two rays that should intersect in space at the location of the corresponding 3D point. However, because of the assumption that the camera parameters are known approximately, the two rays are most likely to be skew. In this case, we calculate the midpoint of the perpendicular to both rays. This produces an output set of 3D points.

1.4.6 Three-Dimensional Point Clustering

After calculating the 3D locations of the feature points in the previous step, we group those points into clusters, where each cluster belongs to one obstacle. This point clustering is performed in order to determine a bounding box for every obstacle. This step is represented by one module called "3D Point Clustering," whose input is the set of 3D points and whose output is a few sets of 3D points where the points of each set should belong to one obstacle.

1.4.7 Obstacle Representation

For some applications, a simple bounding box representing the obstacle might be sufficient. This is represented by the "Bounding Boxes" module. The inputs to this module are three types of information. These are: the sets of points obtained in the previous step (Section 1.4.6); the matrix that represents the ground as obtained in Section 1.4.1; and virtual camera parameters used to calculate a projection matrix. However, in order to obtain a more accurate representation of the terrain and to improve the quality of rendering, two alternative approaches will be considered: reconstructing the obstacles in 3D space and representing them as planar patches.

1.4.7.1 Obstacles as 3D Models

The 3D reconstruction of the obstacle is represented by four modules. These modules are "Box Projection," "Silhouette Extraction," "Voxel Occupancy" and "3D Rendering."

The inputs to the "Box Projection" module are the dimensions and locations of the bounding boxes as well as the updated projection matrices obtained through the enhanced match set. The output of this step is used to extract silhouettes or as opacity maps for planar rendering. The next module is named "Silhouette Extraction" where silhouettes are extracted

using the boxes projected in the previous module as well as refined homography matrices of Section 1.4.3 and the original images. The output of this module is used towards voxelization. Also, it is used as opacity maps for planar rendering. The inputs to the next module, "Voxel Occupancy," are: the silhouette images obtained previously; the set of original images; and the updated projection matrices estimated through the enhanced match set. Those three pieces of information are to determine the occupied voxels through the "Voxel Occupancy" module. The output is the set of occupied voxels, which becomes an input to the "3D Rendering" module in addition to the matrix that represents the ground as obtained in Section 1.4.1; and virtual camera parameters used to calculate a projection matrix.

This reconstruction of the obstacle is performed through a voxelization technique. The idea is to split the bounding boxes obtained previously into small cubes. Starting from the location of a given cube in space, we can project it onto the images. If the information at the projected points is similar in as many images as possible, it is considered a part of the obstacle; otherwise, it should be discarded. The outputs of the "Voxel Occupancy" module are the location and color of every occupied voxel. The 3D representation of an obstacle in "3D Rendering" module is therefore the union of the accepted voxels.

1.4.7.2 Obstacles as Planar Patches

When an approximate representation of the work site is sufficient for the operator to work, a simpler way to represent the obstacles can be considered. In this case, the obstacles may be represented as planar patches. The inputs to the "Planar Rendering" module are: the original images; the associated parameters; the matrix that represents the ground as obtained in Section 1.4.1; virtual camera parameters used to calculate a projection matrix; and the projections of the bounding boxes, or optionally, silhouette images that may be used as opacity maps to add realism to the output. This representation is achieved by determining the boundaries of the planar patch using the projection of the bounding boxes. Mapping is then performed between the appropriate original image and the patch in the synthesized image.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 explores many aspects of Projective Geometry that are used extensively in today's Computer Vision research. The last part of that chapter concentrates on three-dimensional reconstruction. The use of projective geometry in this area is discussed. Different approaches are surveyed.

Chapter 3 discusses a new approach, called JUDOCA, which we introduce to detect junctions of different shapes. We use junctions detected by our method to facilitate the matching process discussed in Chapter 5.

Chapter 4 derives the homography matrix used to establish the perspective view of the planar ground surface where the obstacles reside. This matrix is also used as a primary step to detect features on obstacles and exclude those on the ground. Detecting features on obstacles is an important step in order to facilitate the matching process (discussed in Chapter 5).

Chapter 5 addresses the problem of wide baseline matching. Using the junctions survived the process in the previous chapter, we introduce an algorithm to achieve point correspondences given the constraint of wide baseline. The idea is based on the fact that every two edges that form a junction are in fact forming a plane in space. Relationships can be established among different projections of the same plane as discussed in Chapter 2. Consequently, through these relationships of the planes at junction locations, correlation techniques may be applied; and hence, we can achieve good matching results.

Chapter 6 discusses three main steps in our system. These are: 3D point reconstruction; 3D point clustering; and voxelization. The discussion starts with reconstructing 3D points in space as done through their projections onto images. This is followed by clustering points in space through a hierarchical structure we propose. The idea we introduce is based on the concept of irregular pyramids. The result is a bounding box for each obstacle, which can be used as a final representation for the obstacle. Alternatively, it can be used as a primary step towards 3D modeling through voxelization. The last part of the chapter concentrates on a voxel occupancy technique that determines the 3D shape of the obstacles and renders the resulting shapes.

Chapter 7 investigates an approach to represent the obstacles approximately as planar patches. This is introduced to speed up the representation process if the time factor is critical and high spatial accuracy is not required.

Our work in this thesis deals with many parameters and thresholds. In order to clarify matters, Appendix A summarizes the parameters and thresholds used and defines their role in different stages of computation. Appendix B lists the details of each term of the planar perspective matrix discussed in Chapter 4 as well as the projection and the calibration rotation matrices and the translation vector. Appendix C lists the derivations of some formulas used throughout the thesis. Finally, Appendix D handles three-dimensional modeling from a Computer Graphics viewpoint.

1.6 Main Contributions

Two demos of our work were presented at two IRIS/Precarn[3] conferences [EL00, EL01b] held in Montréal and Ottawa in 2000 and 2001, respectively. Those demos showed sequences of images created along a path determined by a user on an overhead view of the work site. Two different representations of obstacles were presented: volumetric and planar. Besides, during the course of this thesis, we have presented many other contributions that may be summarized throughout the rest of this section.

Traditional corner detectors involve finding the location of a corner point. Other information, such as the orientation of that corner and the shape characterizing it, may get neglected. In [EL02], we proposed a new data structure called *cone data structure* to detect junctions where corner locations and orientations were detected. Its algorithm works on the intensity values of the image that was time consuming. In order to speed up the process, we proposed another algorithm to detect junctions that we called *JUDOCA* [LE, LE04]. This operator works on binary edge maps. Avoiding working with the intensity values helped save time. The effectiveness and the ability of this operator to work were demonstrated on both synthetic and real-world images.

Wide baseline matching is always a tough problem that regular correlation techniques usually fail to solve. The main reason for this is the existence of perspective deformation, which usually presents between pairs of wide baseline images. In [Eli04], we presented a novel approach to match feature points among wide baseline pairs of images. We deployed a measure of invariance to overcome this difficulty by applying homographic transformation that can estimate this deformation for planar patches formed by the junctions detected by the above JUDOCA detector. We have shown through examples and analysis that our approach is effective compared with standard correlation techniques. In [HEL03, HEL04], we used our wide baseline approach to localize and detect obstacles.

In Image Analysis, pyramid levels (regular or irregular) usually process an image along different stages. In this thesis, the concept of irregular pyramids has been deployed in a new way. In [Eli03a], we proposed to use the concept of irregular pyramids on a set of 3D points, rather than an image, to perform point clustering. We have shown that this idea may be used to localize and detect obstacles [HEL03, HEL04] and obtain bounding boxes containing them for use as a step towards obstacle modeling or even for use as a representation by itself.

[3]IRIS, "Institute for Robotics and Intelligent Systems," managed by Precarn Incorporated, is a federally funded Network of Centres of Excellence that brings together top Canadian researchers to combine efforts and expertise on projects that focus on the essential elements of an intelligent system - the ability to perceive, reason and act [Inc03b].

In [EL04], we used those bounding boxes to extract silhouette images for obstacles and deployed the updated versions of the fundamental matrices to obtain more accurate projection matrices and hence achieve volumetric representation of obstacles through voxel occupancy scheme.

Geometry plays a key role throughout this thesis particularly Projective Geometry. Understanding geometric relations among different entities is essential for achieving good recognition algorithms. We discussed the main concepts of this important branch of geometry in [EL01a, Eli03b].

The orientation of a virtual camera can be represented through different types of angles resulting in different rotation systems. In [EL03], we derived the homography matrix that relates an overhead view of the site to a virtual camera viewpoint. This derivation was accomplished in detail through two different rotation systems.

Finally, we intend to present another journal paper summarizing our work in this thesis.

1.7 Summary

This chapter started with an introduction to Computer Vision and teleoperation. It provided a brief overview of the SMART project, which involves four main schemes. We mentioned that our work in this thesis was related to the first scheme of the project. We defined the problem and stated the assumptions that were applied to this thesis and to the SMART project as well. Some related works were discussed. We provided an overview of the proposed scene modeling system. This was followed by a discussion of the organization of the thesis and the contents and topics of each chapter. Finally, we briefly stated our contributions through this work.

Chapter 2

Projective Geometry for Three-Dimensional Computer Vision

Homogeneous coordinates simplify the camera properties formulation. This is one of the reasons why Projective Geometry is an attractive framework in Computer Vision [Fau96, MZ92]. This chapter explores many aspects of Projective Geometry that have been used in Computer Vision. It starts with declaring the notation used. Then, it discusses the camera model. It, then, moves to investigate a very important aspect in three-dimensional reconstruction, namely binocular vision. Under this title, we explore many topics including the fundamental and the essential matrices and the plane homography. The last part of this chapter addresses the three-dimensional reconstruction problem. It provides a general overview for different poses investigated to reconstruct an object in space. Then it moves to explore the role of Projective Geometry in this matter.

Many concepts explained in this chapter are used throughout the rest of the thesis. They have been divided into the following main sections:

- The projective model discussed in Section 2.2 is used in Chapter 4 to compute the homography matrix of the ground;
- Binocular vision concept discussed in Section 2.3 is used in Chapters 4 and 5 to compute the fundamental matrix; and
- 3D reconstruction discussed in Section 2.4 is used in Chapter 5 to reconstruct lines in space and Chapter 6 to reconstruct points in space.

Our discussion in this chapter is available in [EL01a, Eli03b]. In order to get started, the next section introduces the notation that will be used throughout this thesis.

2.1 Notation

This section is a collection of the Projective Geometry notations used in this thesis. These notations draw on materials from [ZLA98, ZDFL94, LV94, aODF89].

- The scalar quantities are represented by italic letters (e.g., s).
- The vectors are represented by:
  - uppercase bold letters when they refer to 3D space (e.g., $\mathbf{M}$) or,
  - lowercase bold letters when they refer to 2D space (e.g., $\mathbf{m}$).
- The matrices are represented by uppercase fixed size letters (e.g., $\mathtt{A}$).
- When several views are included (several coordinate systems), these are identified using prime notation (e.g., $\mathbf{m}, \mathbf{m}'$).
- When transformation in space is carried out, it will be identified by the subscript $_{new}$ to indicate it (e.g., $\mathbf{M}_{new}$ to indicate transformed $\mathbf{M}$).
- The estimated position of the points are indicated by the hat notation (e.g., $\hat{\mathbf{m}}$).
- For any vector $\mathbf{T}=[t_1, t_2, t_3]^T$, there is an antisymmetric matrix $\mathtt{T}_\times$ defined as:

$$\mathtt{T}_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{2.1}$$

  such that

$$\mathtt{T}_\times \mathbf{x} = \mathbf{T} \times \mathbf{x} \tag{2.2}$$

  where $\times$ denotes the cross product.

2.1.1 Points

In order to differentiate between the Cartesian and projective or homogeneous coordinates, the following notation is used. A point, $\dot{\mathbf{M}} = [x, y, z]^T$, in 3D space is represented by its homogeneous coordinates, $\mathbf{M} = [x, y, z, 1]^T$. Similarly, its image on the retinal (or image) plane, $\dot{\mathbf{m}} = [x, y]^T$, is represented by its homogeneous coordinates, $\mathbf{m} = [x, y, 1]^T$.

In general, the last entry in homogeneous coordinates can be different from 1; i.e., we may have $\mathbf{M} = [x, y, z, t]^T$. In this case, the relation between $\mathbf{M}$ and $\dot{\mathbf{M}}$ is:

$$\dot{\mathbf{M}} = \left[\frac{x}{t}, \frac{y}{t}, \frac{z}{t}\right]^T \tag{2.3}$$

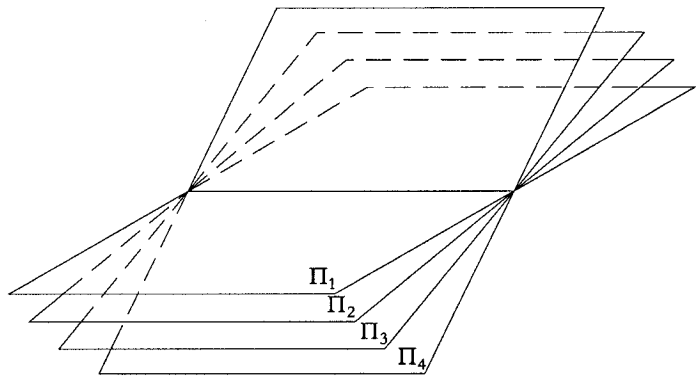


Figure 2.1: Pencil of planes.

2.1.2 Lines

Similar to the point notation, a line in 2D space is represented by a 3D vector. If a point $\dot{\mathbf{m}}$ is on a line $\mathbf{l}$ (where $\mathbf{l} = [a, b, c]^T$) then $\mathbf{l}^T\mathbf{m}$=0. Consequently, there is no difference between the representation of a line and the representation of a point. This fact is known as the *duality principle*. The equation of a line $\mathbf{l}$ going through two points $\mathbf{m_1}$ and $\mathbf{m_2}$ is given by $\mathbf{l} = \mathbf{m_1} \times \mathbf{m_2}$. Similarly, the intersection between two lines $\mathbf{l_1}$ and $\mathbf{l_2}$ is a point $\mathbf{m} = \mathbf{l_1} \times \mathbf{l_2}$.

A number of methods have been proposed to represent lines in 3D space. One possible representation is that a line is the axis of a pencil of planes (see Figure 2.1 on Page 17). This axis can be represented by the intersection of any two planes from the pencil. A similar representation is that a line is a pencil of collinear points and is defined by any of two of these points. To represent that mathematically, we consider a line as the span of two vectors. Thus, if a line is the intersection of two planes Π_1 and Π_2 (see Section 2.1.3), it can be represented as:

$$\mathrm{L} = \begin{bmatrix} \Pi_1^T \\ \Pi_2^T \end{bmatrix} \tag{2.4}$$

2.1.3 Planes

Also, a plane is represented by a 4D vector. If a point $\dot{\mathbf{M}}$ is on a plane Π (where $\Pi = [a, b, c, d]^T$) then $\Pi^T\mathbf{M}$=0.

2.1.4 Directions

A direction is a point on the plane at infinity. Points at infinity are called *ideal points* [Fis97b]. In homogeneous coordinates, such points have their last entry equal to zero (e.g., $\mathbf{d} = [x, y, 0]^T$, $\mathbf{D} = [x, y, z, 0]^T$).

2.2 The Projective Model

The main property of the camera model is that the relationship between the world coordinates and the pixel coordinates is linear projective. This relationship remains linear regardless of the choice of both coordinates. Thus, $\mathbf{M}$ and $\mathbf{m}$ are related by:

$$s\mathbf{m} = \mathtt{P}\mathbf{M} \tag{2.5}$$

where

- s, a scale factor, is called the *depth* [HA98];
- $\mathtt{P}$ is a 3×4 matrix, called the *perspective projection matrix* (also known as the *camera matrix* [RCF95]).

Because s is an arbitrary scale which could take any value, we will simply denote Equation (2.5) as:

$$\mathbf{m} \cong \mathtt{P}\mathbf{M} \tag{2.6}$$

If the origin of the world coordinate system is located at the optical center of the camera (see Figure 2.2 on Page 19) whose focal length is 1, then $\mathtt{P}$ is simply:

$$\mathtt{P} = \mathtt{P}_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

In the general case, the origin of the world coordinate system is located arbitrarily in space (see Figure 2.3 on Page 19) and image points are denoted by a different coordinate system (pixel or camera coordinates). Changing coordinates in space is equivalent to multiplying matrix $\mathtt{P}$ to the right by a 4×4 matrix while changing coordinates in the retinal plane is equivalent to multiplying matrix $\mathtt{P}$ to the left by a 3×3 matrix. Thus, the perspective projection matrix, $\mathtt{P}$, could be decomposed into 3 matrices:

$$\mathtt{P} = \mathtt{A}\mathtt{P}_c\mathtt{D} \tag{2.8}$$

where

- $\mathtt{A}$ is called the *camera calibration matrix* which maps the normalized image coordinates to the retinal image coordinates. This matrix has a number of entries called the *intrinsic parameters* (of the camera). A camera is calibrated if $\mathtt{A}$ is known. The camera calibration matrix, $\mathtt{A}$, is defined as:

$$\mathtt{A} = \begin{bmatrix} fk_u & fk_u cot\theta & u_0 \\ 0 & \frac{fk_v}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

  The intrinsic parameters are:

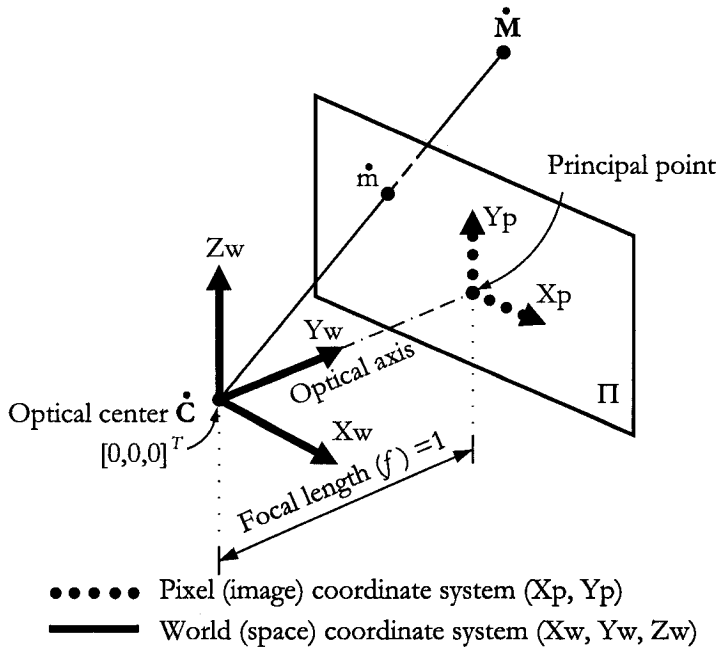


Figure 2.2: Special case: The origin of the coordinate system coincides on the optical center of the camera whose focal length is equal to 1.

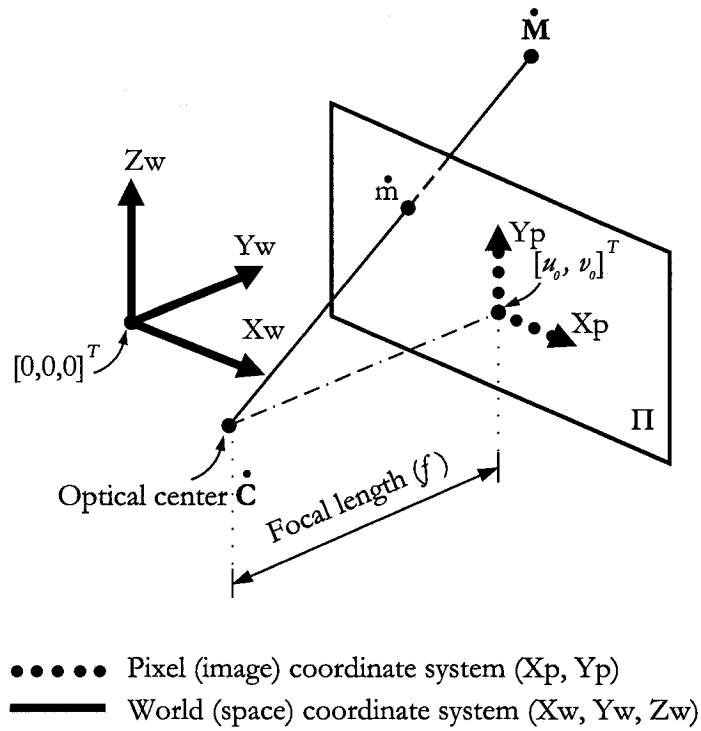


Figure 2.3: General case: The origin of the coordinate system is arbitrarily located in space.

1. f, the focal length of the camera;
2. k_u and k_v, the horizontal and vertical scale factors. Their inverses determine the size of the pixel in world coordinate units;
3. u_0 and v_0, the coordinates of the intersection between the optical axis; i.e., the line passing through the optical center and perpendicular to the retinal plane, and the retinal plane (the *principal point* of the camera);
4. θ, the angle between the retinal axes. This can take care of the fact that the pixel grid may not be orthogonal; i.e., in the general case, θ may not be equal to 90°. When $\theta = 90°$, the matrix A is defined as:

$$\mathtt{A} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Because we cannot separate the focal length of the camera from the scale factors, we have five intrinsic parameters $\alpha_u = fk_u$, $\alpha_v = fk_v$, u_0, v_0 and θ which contribute to the *skew* element γ ($\gamma = fk_u cot\theta$). So, assuming that $sin\theta \cong 1$, the matrix A could be rewritten as:

$$\mathtt{A} = \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Note that the ratio $r = \frac{\alpha_u}{\alpha_v}$ is the *aspect ratio* (i.e., the ratio of the width to the height). If f as well as the format height and width are expressed in millimeters, α_u and α_v can be expressed in pixels as:

$$\begin{aligned} \alpha_u &= f\frac{IW}{FW} \\ \alpha_v &= f\frac{IH}{FH} \end{aligned} \tag{2.12}$$

where IW, IH are the image width and height expressed in pixels, FW, FH and f are the format width and height and focal length expressed in millimeters.

- D is a 4 × 4 matrix defined as:

$$\mathtt{D} = \begin{bmatrix} \mathtt{R} & \mathbf{T} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.13}$$

where

1. R is a 3 × 3 rotation matrix (which has 3 parameters). It can defined as:

$$\mathtt{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.14}$$

2. $\mathbf{T}$ is a 3D translation vector (which has 3 parameters). It can be defined as:

$$\mathbf{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.15}$$

The matrix D describes the change of the pose of the camera (the world coordinate system) and has 6 independent parameters (3 parameters for R and 3 others for $\mathbf{T}$). These parameters are called the *extrinsic parameters.*

A geometric interpretation for the perspective projection matrix, P, can be obtained by decomposing the matrix P as follows [Fau96]:

$$\mathtt{P} = \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \mathbf{P}_3^T \end{bmatrix} \tag{2.16}$$

where $\mathbf{P}_i | i \in \{1, 2, 3\}$ are 4D vectors. Each vector represents a projective plane with point equation:

$$\mathbf{P}_i^T \mathbf{M} = 0 \tag{2.17}$$

Another decomposition of the perspective projection matrix, P, makes explicit the relationship between the matrix P and the optical center of the camera $\dot{\mathbf{C}}$ [LF96]:

$$\mathtt{P} = [\tilde{\mathtt{P}} | \dot{\mathbf{P}}] \tag{2.18}$$

where

- $\tilde{\mathtt{P}}$ is a 3×3 matrix of rank 3;
- $\dot{\mathbf{P}}$ is a 3D vector.

Assume that the optical center, $\dot{\mathbf{C}}$, is not at infinity and its homogeneous coordinates is the 4D vector, $\mathbf{C} = [\dot{\mathbf{C}}^T, 1]^T$. Then, the optical center, $\dot{\mathbf{C}}$, satisfies the equation:

$$\mathtt{P}\mathbf{C} = \mathbf{0} \tag{2.19}$$

which means that:

$$\dot{\mathbf{C}} = -\tilde{\mathtt{P}}^{-1} \dot{\mathbf{P}} \tag{2.20}$$

Also, the matrix P can be decomposed into [HA98]:

$$\mathtt{P} = \begin{bmatrix} rf & \gamma f & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} [\mathtt{R} | -\mathtt{R}\mathbf{T}] \tag{2.21}$$

where

- r is the aspect ratio;
- f is the focal length;
- γ is the skew;
- $[u_0, v_0]^T$ is the principal point;
- $\mathtt{R}$ is the rotation matrix; and
- $\mathbf{T}$ is the translation vector.

The camera is called $non-skew$ if $\gamma = 0$, $aspect-free$ if $r = 1$ and *with Euclidean image plane* when both $\gamma = 0$ and $r = 1$. The camera calibration matrices

$$\begin{bmatrix} rf & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} f & \gamma & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

are called *non-skew*, *aspect-free* and *Euclidean* calibration matrices respectively.

If we consider a 3D transformation of the world coordinates, so that:

$$\mathbf{M}_{new} = \mathtt{T}\mathbf{M} \tag{2.23}$$

where $\mathtt{T}$ is a 4×4 non-singular matrix, then image measurements will not be affected and we can write:

$$\mathbf{m} = \mathtt{P}\mathbf{M} = \mathtt{P}\mathtt{T}^{-1}\mathbf{M}_{new} = \mathtt{P}_{new}\mathbf{M}_{new} \tag{2.24}$$

This means that $\mathtt{P}_{new} = \mathtt{P}\mathtt{T}^{-1}$ under the transformation $\mathtt{T}$ [BZM97]. When $\mathtt{P}$ has the form:

$$\mathtt{P} = [\mathtt{A}\mathtt{R}| - \mathtt{A}\mathtt{R}\dot{\mathbf{C}}] \tag{2.25}$$

where

- $\dot{\mathbf{C}}$ is the optical center; and
- $\mathtt{A}\mathtt{R}$ (see Equation (2.21)) is a 3×3 nonsingular matrix, which requires the optical center not to lie on the plane at infinity [BZM97].

then,

$$\mathtt{T}^{-1} = \begin{bmatrix} [\mathtt{A}\mathtt{R}]^{-1} & \dot{\mathbf{C}} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.26}$$

Notice that $\dot{\mathbf{C}}$ is the translation as well.

Generally, in order to transform to a new coordinate system through a rotation matrix R and a translation vector $\mathbf{T}$, a 4×4 transformation matrix, T, is used. That is:

$$\mathtt{T}^{-1} = \begin{bmatrix} \mathtt{R}^{-1} & -\mathtt{R}^{-1}\mathbf{T} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.27}$$

For example, to transform the projection matrix, P, to a new coordinate system through R and $\mathbf{T}$, the matrix P should be multiplied to the right by the transformation matrix $\mathtt{T}^{-1}$.

$$\mathtt{P}_{new} = \mathtt{P}\mathtt{T}^{-1} \tag{2.28}$$

2.3 Binocular Vision

2.3.1 The Epipolar Geometry

The epipolar geometry is the basic constraint arising in stereo vision. Let us consider using two cameras to view a point $\dot{\mathbf{M}}$ in space. We will get two images for this point, one on each image plane. This is shown in Figure 2.4 on Page 24. Given a point $\dot{\mathbf{M}}$ in space and two cameras with optical centers $\dot{\mathbf{C}}$ and $\dot{\mathbf{C}}'$, $\dot{\mathbf{m}}$ is the image of $\dot{\mathbf{M}}$ formed through $\dot{\mathbf{C}}$ on the image plane Π and $\dot{\mathbf{m}}'$ is the image of the same point on Π'. So, given an image point $\dot{\mathbf{m}}$, its matching point $\dot{\mathbf{m}}'$ belongs to the line segment, $\mathbf{l}'$, formed by intersecting the plane $\dot{\mathbf{M}}\dot{\mathbf{C}}\dot{\mathbf{C}}'$ with the image plane Π'. This line is called the *epipolar line* associated with $\dot{\mathbf{m}}$, while the epipolar line $\mathbf{l}$ associated with $\dot{\mathbf{m}}'$ is the line formed by intersecting the plane $\dot{\mathbf{M}}\dot{\mathbf{C}}\dot{\mathbf{C}}'$ with Π. These two lines are called *conjugate epipolar lines.* Any point on the first epipolar line has its match on the second and vice versa. Finally, the points $\dot{\mathbf{e}}$ and $\dot{\mathbf{e}}'$ are the intersections of the line $\dot{\mathbf{C}}\dot{\mathbf{C}}'$ with Π and Π' respectively and they are on the epipolar lines $\mathbf{l}$ and $\mathbf{l}'$ and called the *epipoles.*

By definition, the epipole $\mathbf{e}$ is the projection of the optical center $\mathbf{C}'$ and they are related by the relation:

$$\mathbf{e} = \mathtt{P}\mathbf{C}' \tag{2.29}$$

Substituting the value of $\mathbf{C}'$ using Equation (2.20), we get:

$$\mathbf{e} = \mathtt{P}\begin{bmatrix} -\mathtt{P}'^{-1}\dot{\mathbf{P}}' \\ 1 \end{bmatrix} \tag{2.30}$$

Finally, given a pair of stereo images, they are called *weakly calibrated* when the epipolar geometry between the images is known [LF94].

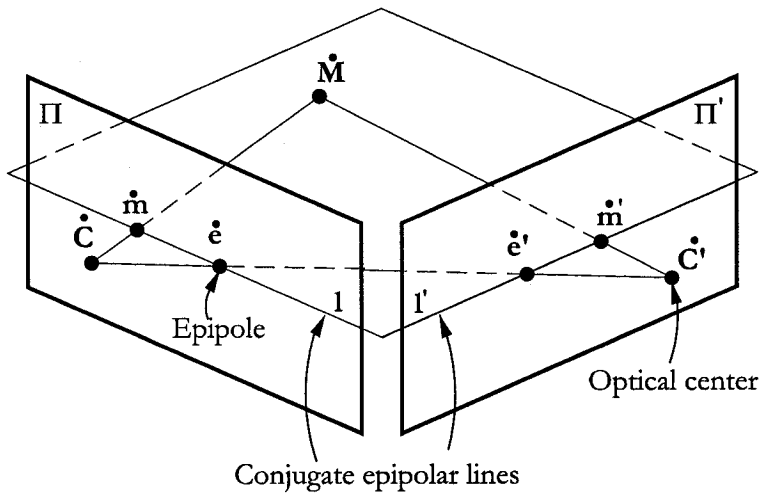


Figure 2.4: Binocular vision: Using two cameras to view a point in 3D space; $\dot{\mathbf{M}}$ is a point in the space, $\dot{\mathbf{C}}$ and $\dot{\mathbf{C}}'$ are the optical centers, $\dot{\mathbf{m}}$ and $\dot{\mathbf{m}}'$ are the images of $\dot{\mathbf{M}}$ on the image planes Π and Π'.

2.3.2 The Fundamental Matrix

The relationship between the pixel coordinates of a point; e.g., $\dot{\mathbf{m}}$, and its corresponding epipolar line, $\mathbf{l}'$, is linear projective because the relations between $\dot{\mathbf{m}}$ and $< \dot{\mathbf{C}}, \dot{\mathbf{m}} >$, and $< \dot{\mathbf{C}}, \dot{\mathbf{m}} >$ and its projection $\mathbf{l}'$ are both linear projective. The 3×3 matrix which describes this correspondence is called the *fundamental matrix* $\mathtt{F}$ [LF96].

If we partition the perspective projection matrix $\mathtt{P}$ as:

$$\mathtt{P} = [\mathtt{AR}| - \mathtt{AR}\dot{\mathbf{C}}] \tag{2.31}$$

where $\dot{\mathbf{C}}$ is the first optical center and the 3×3 $\mathtt{AR}$ matrix is not singular, then the fundamental matrix can be expressed as:

$$\begin{aligned} \mathtt{F} &= [\mathtt{A}'\mathtt{R}']^{-T}[\dot{\mathbf{C}} - \dot{\mathbf{C}}']_{\times}[\mathtt{AR}]^{-1} \\ &= [[\mathtt{A}'\mathtt{R}'][\dot{\mathbf{C}} - \dot{\mathbf{C}}']]_{\times}[\mathtt{A}'\mathtt{R}'][\mathtt{AR}]^{-1} \\ &= [\mathtt{A}'\mathtt{R}']^{-T}[\mathtt{AR}]^{T}[[\mathtt{AR}][\dot{\mathbf{C}} - \dot{\mathbf{C}}']]_{\times} \end{aligned} \tag{2.32}$$

In [HZ00], Hartley and Zisserman pointed out that:

$$\mathtt{P}\mathtt{P}^{+} = \mathtt{I} \tag{2.33}$$

where $\mathtt{P}^{+}$ is the pseudo-inverse of $\mathtt{P}$. In other words, $\mathtt{P}^{+}$ can be written as:

$$\mathtt{P}^{+} = \begin{bmatrix} [\mathtt{AR}]^{-1} \\ \mathbf{0}_3^T \end{bmatrix} \tag{2.34}$$

then the fundamental matrix can be expressed as:

$$\mathtt{F} = [\mathbf{e}']_{\times}\mathtt{P}'\mathtt{P}^{+} \tag{2.35}$$

Using the notation of the fundamental matrix, we can express the epipolar constraint. For a point $\dot{\mathbf{m}}$, the projective representation of its epipolar line is represented as:

$$\mathbf{l}' = \mathtt{F}\mathbf{m} \tag{2.36}$$

Since $\dot{\mathbf{m}}' \in \mathbf{l}'$, it follows that:

$$\mathbf{m}'^{T}\mathtt{F}\mathbf{m} = 0 \tag{2.37}$$

Reversing the role of both images, the fundamental matrix will change to its transpose, $\mathtt{F}^{T}$. By transposing Equation (2.37), we get:

$$\mathbf{m}^{T}\mathtt{F}^{T}\mathbf{m}' = 0 \tag{2.38}$$

The epipolar line of the epipole $\dot{\mathbf{e}}$ is $\mathtt{F}\mathbf{e}$. As shown in Figure 2.4 on Page 24, $\dot{\mathbf{e}}'$ is the image of the optical ray $< \dot{\mathbf{C}}, \dot{\mathbf{e}} >$ on the plane Π'. It follows that this line is reduced to a point $\mathbf{e}'$, or the second epipole. This implies that $\mathtt{F}\mathbf{e} = \mathbf{0}$. The same is applied for $\mathbf{e}'$, which means:

$$\mathtt{F}\mathbf{e} = \mathtt{F}^{T}\mathbf{e}' = \mathbf{0} \tag{2.39}$$

As a result, the rank of $\mathtt{F}$ is less than or equal to 2. Generally, the rank is equal to 2 [LF96]. Furthermore, the fundamental matrix is defined up to a scale factor, and depends upon 7 parameters [LF94].

2.3.3 The Relation Between the Fundamental and Projection Matrices

The perspective projection matrix can be written as $\mathtt{P}=[\tilde{\mathtt{P}}|\dot{\mathbf{P}}]$ where $\tilde{\mathtt{P}}=\mathtt{AR}$ is a 3×3 matrix. Then, the projections of a point $\mathbf{M} = [\dot{\mathbf{M}}^{T}, m]^{T}$, according to Equation (2.5), are:

$$s\mathbf{m} = \tilde{\mathtt{P}}\dot{\mathbf{M}} + m\dot{\mathbf{P}} \tag{2.40}$$

and

$$s'\mathbf{m}' = \tilde{\mathtt{P}}'\dot{\mathbf{M}} + m\dot{\mathbf{P}}' \tag{2.41}$$

where m may be equal to zero [RFC95]. Eliminating $\dot{\mathbf{M}}$ from these constraints results in:

$$s\mathbf{e} = \dot{\mathbf{P}} - \tilde{\mathtt{P}}\tilde{\mathtt{P}}'^{-1}\dot{\mathbf{P}}' \tag{2.42}$$

and

$$s'\mathtt{F} = [\mathbf{e}']_{\times}\tilde{\mathtt{P}}'\tilde{\mathtt{P}}^{-1} \tag{2.43}$$

This represents the epipolar constraints between the two cameras.

We can set $\mathtt{P} = [\mathtt{I}|\mathbf{0}]$ where $\mathtt{I}$ is the 3×3 identity matrix. This parametrization and Equation (2.42) and Equation (2.43) result in:

$$s\mathbf{e}' = \dot{\mathbf{P}}' \tag{2.44}$$

and

$$s'\mathtt{F} = [\mathbf{e}']_\times \tilde{\mathtt{P}}' \tag{2.45}$$

Both $\tilde{\mathtt{P}}'$ and $\dot{\mathbf{P}}'$ are constrained so that:

$$\mathtt{P}' = [[\mathbf{e}']_\times \mathtt{F}|\mathbf{e}'] \begin{bmatrix} \mathtt{I} & \mathbf{0}_3 \\ \mathbf{A}_3^T & a \end{bmatrix} \tag{2.46}$$

where $\mathbf{A}$ is an arbitrary vector and a is a non-zero scalar. Thus, given the matrix $\mathtt{F}$, we can find solutions for $\mathtt{P}$ and $\mathtt{P}'$ in a projective frame.

2.3.4 The Essential Matrix

When the intrinsic parameters of both views are known (*strong-calibration* case), the fundamental matrix becomes the *essential matrix* $\mathtt{E}$ [LH81]. This matrix depends upon the 5 parameters describing the relative displacement between the views [LF94]. Specifically, the first three parameters are of the 3D rotation and the other two parameters define the direction of the translation [LF96].

If $\dot{\mathbf{m}}$ and $\dot{\mathbf{m}}'$ are images of a 3D point $\dot{\mathbf{M}}$ on both images and ($\mathtt{R}$, $\mathbf{T}$) is the motion from the first camera to the second, under the pinhole model, we have the following two equations [Zha98]:

$$s\mathbf{m} = \mathtt{A}_1[\mathtt{I}|\mathbf{0}]\mathbf{M} \tag{2.47}$$

and

$$s'\mathbf{m}' = \mathtt{A}'[\mathtt{R}|\mathbf{T}]\mathbf{M} \tag{2.48}$$

where $\mathtt{A}$ and $\mathtt{A}'$ are the intrinsic , or calibration matrices of both cameras. (Refer to Equations (2.9) through (2.11).) Eliminating s, s' and $\mathbf{M}$, we get:

$$\mathbf{m}^T = \mathtt{F}\mathbf{m}' = 0 \text{ with } \mathtt{F} = \mathtt{A}'^{-T}\mathtt{T}_\times\mathtt{R}\mathtt{A}^{-1} \tag{2.49}$$

where $\mathtt{T}_\times$ is an antisymmetric matrix defined by $\mathbf{T}$ as:

$$\mathtt{T}_\times\mathbf{x} = \mathbf{T} \times \mathbf{x} \tag{2.50}$$

Note that $\times$ denotes the cross product (refer to Section 2.1). The essential matrix $\mathtt{E}$ can be written as:

$$\mathtt{E} = \mathtt{T}_\times\mathtt{R} \tag{2.51}$$

Now, we can write Equation (2.49) as:

$$\mathbf{m}'^T\mathtt{E}\mathbf{m} = 0 \tag{2.52}$$

Hence, the magnitude of $\mathbf{T}$ cannot be recovered from two images, we set $\|\mathbf{T}\| = 1$ [Zha98]. The matrices $\mathtt{E}$ and $\mathtt{F}$ are related by:

$$\mathtt{F} = \mathtt{A}'^{-T}\mathtt{E}\mathtt{A}^{-1} \tag{2.53}$$

and

$$\mathtt{E} = \mathtt{A}'^{T}\mathtt{F}\mathtt{A} \tag{2.54}$$

The essential matrix, $\mathtt{E}$, Is characterized by two constraints found by Huang and Faugeras [aODF89]. These constraints are the nullity of the determinant and the equality of the two non-zero singular values.

2.3.5 Plane Homography

This important concept is known as *plane projectivity* or *plane collineation* $\mathtt{H}$ [LO98]. It relates two views of a plane in 3D space. More specifically, $\mathtt{H}$ is a nonsingular 3×3 matrix needed to relate two views of a plane. Thus, if $\dot{\mathbf{m}}$ is a point in the first view and $\dot{\mathbf{m}}'$ is the corresponding point in the second view, then:

$$\mathbf{m} \cong \mathtt{H}\mathbf{m}' \tag{2.55}$$

Note that $\mathtt{H}$ as well as $\mathtt{F}$ can be estimated up to scale factors. They are related by:

$$\mathtt{F}^{T}\mathtt{H} + \mathtt{H}^{T}\mathtt{F} = \mathbf{0} \tag{2.56}$$

where $\mathtt{F}^{T}\mathtt{H}$ is a skew-symmetric matrix. Equation (2.56) can be satisfied if and only if [LF96]:

$$\mathtt{F} = [\mathbf{e}']_{\times}\mathtt{H} \tag{2.57}$$

where $[\mathbf{e}']_{\times}$ is an antisymmetric matrix formed using the second epipole coordinates. As a consequence of Equation (2.56), we get:

$$\mathtt{H}\mathbf{e} = \mathbf{e}' \tag{2.58}$$

which represents the homography of a plane in general position. Also,

$$\mathtt{H}\mathbf{e} = \mathbf{0} \tag{2.59}$$

that represents the case where the plane contains the optical center $\dot{\mathbf{C}}'$, which leads to non-invertible correspondence.

Also, matrix $\mathtt{H}^{T}$ maps epipolar lines to corresponding epipolar lines. Thus, we have (up to a scale factor):

$$\mathtt{H}^{T}\mathbf{l}' = \mathtt{H}^{T}\mathtt{F}\mathbf{m} = \mathtt{F}^{T}\mathtt{H}\mathbf{m} = \mathtt{F}^{T}\mathbf{m}' = \mathbf{l} \tag{2.60}$$

Matrix H can be decomposed into a concatenation of three matrices S, B and V representing *similarity*, *affine* and *pure projective* transformations respectively [LZ98]:

$$\mathtt{H} = \mathtt{SBV} \tag{2.61}$$

where

$$\mathtt{S} = \begin{bmatrix} s\mathtt{R} & \mathtt{t} \\ \mathbf{0}_2^T & 1 \end{bmatrix} \tag{2.62}$$

where

- R is a 2×2 matrix representing the 2D rotation;
- t is the 2D translation vector; and
- s is an isotropic scaling;

$$\mathtt{B} = \begin{bmatrix} \frac{1}{\beta} & -\frac{\alpha}{\beta} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.63}$$

where

- α and β specify the image of the circular points [SK79, LZ98]. α and β determine two degrees of freedom for this matrix.

$$\mathtt{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \tag{2.64}$$

where

- $\mathbf{l}_\infty = [l_1, l_2, l_3]^T$ is the vanishing line of the plane (see Figure 2.5 on Page 29). The vector $\mathbf{l}_\infty$ has two degrees of freedom. Obviously, this line may pass through an infinite number of planes.

2.3.5.1 Homography Given the Plane

Suppose that the world origin is at the first camera, then the projection matrices are:

$$\mathtt{P} = \mathtt{A}[\mathtt{I}|\mathbf{0}] \tag{2.65}$$

and

$$\mathtt{P}' = \mathtt{A}'[\mathtt{R}'|\mathbf{T}'] \tag{2.66}$$

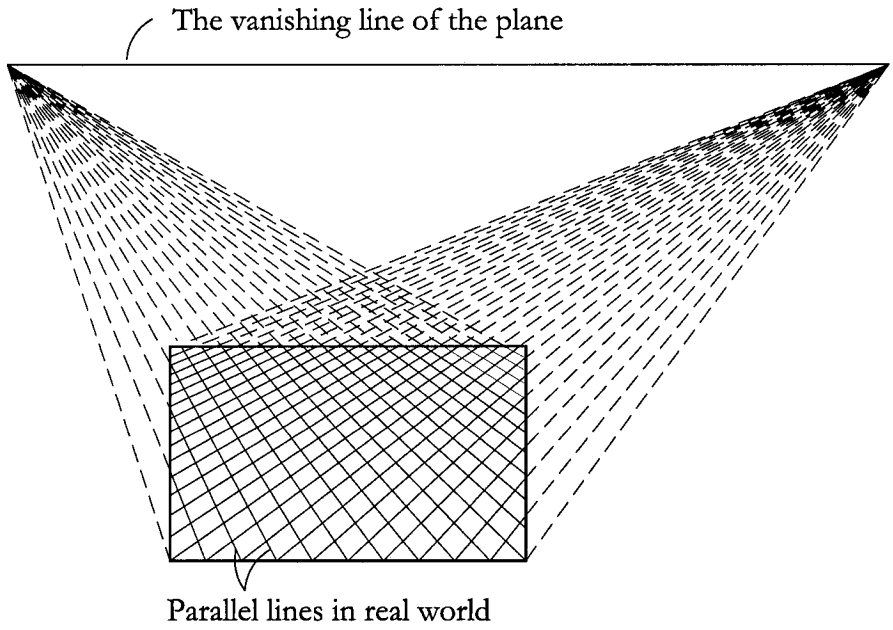


Figure 2.5: All parallel lines intersect at one vanishing line.

and suppose that a 3D plane has the equation $\Pi = [\mathbf{N}^T, d]^T$, so for any point $\dot{\mathbf{M}}$ on that plane, we have:

$$\mathbf{N}^T\dot{\mathbf{M}} + d = 0 \tag{2.67}$$

then the homography $\mathtt{H}$ is calculated as [HZ00]:

$$\mathtt{H} = \mathtt{A}'\left[\mathtt{R}' - \frac{\mathbf{T}'\mathbf{N}^T}{d}\right]\mathtt{A}^{-1} \tag{2.68}$$

2.4 Three-Dimensional Reconstruction

2.4.1 Varieties of the Problem

Techniques are developed to determine three dimensional object reconstruction. These techniques use groups of images taken by cameras. Variations of the problem include 3D surface and scene reconstruction. Also, they include:

- 3D reconstruction from uncalibrated monocular image sequence [FCZ98, PKVG98];
- 3D reconstruction from calibrated monocular image sequence [NH95];
- 3D reconstruction from stereo images. This includes pairs of images taken at the same time by two cameras or at two different instants by one camera provided that the scene is static;

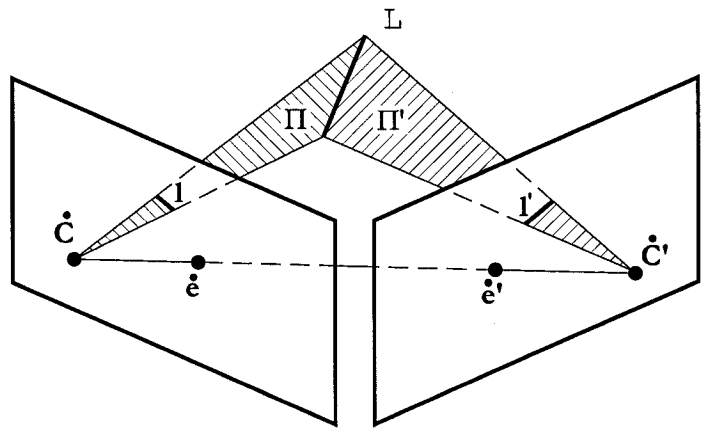


Figure 2.6: A 3D line as the intersection of two planes formed by the its projections in both images and the optical centers.

Usually, The solution is divided into two steps [Zha95]. These steps are:

1. extracting and matching features between corresponding images; and
2. determining structure from corresponding features.

However, the success of the whole process can be affected dramatically by the reliability of the first step.

Chapters 3, 4 and 5 discuss methods we that implemented to achieve the goal of the first step. The rest of this chapter is devoted to discuss a major part of the second step where Projective Geometry plays a key role in determining structures.

2.4.2 Three-Dimensional Line Reconstruction

Assume that a 3D line $\mathtt{L}$ is projected into lines in two views as $\mathtt{l}$ and $\mathtt{l}'$ (see Figure 2.6 on Page 30). Each of $\mathtt{l}$ and $\mathtt{l}'$ with the corresponding optical center form a 3D plane $\Pi = \mathtt{Pl}$ and $\Pi' = \mathtt{P}'\mathtt{l}'$. Considering the notation mentioned in Section 2.1.2, the 3D line can be represented as:

$$\mathtt{L} = \begin{bmatrix} \mathtt{l}^T\mathtt{P} \\ \mathtt{l}'^T\mathtt{P}' \end{bmatrix} \tag{2.69}$$

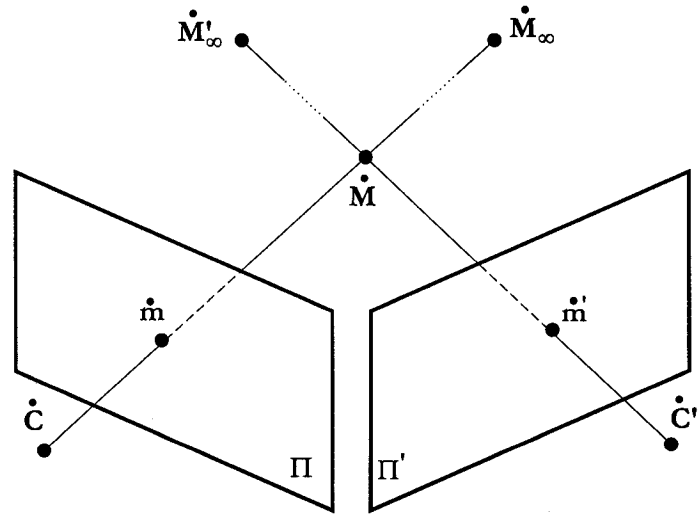


Figure 2.7: $\dot{M}$ is the intersection of $< \dot{C}, \dot{m} >$ and $< \dot{C}', \dot{m}' >$.

2.4.3 Three-Dimensional Point Reconstruction

2.4.3.1 Distance Minimization

Using Equation (2.47), we can estimate the 3D coordinates of a point in space $\dot{M}$ if we know the motion (R', T') and the matching pair $(\dot{m}, \dot{m}')$ as follows [Zha98]:

$$\hat{\dot{M}} = \arg\min_{\dot{M}}(\|\mathbf{m} - A[I|0]M\|^2 + \|\mathbf{m}' - A'[R'|T']M\|^2) \tag{2.70}$$

Note that in this equation we are minimizing the distance between the back-projection of the 3D reconstruction and the image point. Also, this method requires an initial estimate for M.

2.4.3.2 Intersecting Camera Rays

A 3D point is the intersection of two rays passing through the optical centers $\dot{C}$ and $\dot{C}'$, and corresponding image points $(\dot{m}, \dot{m}')$ (see Figure 2.7 on Page 31). The three points $\dot{m}$, $\dot{m}'$ and $\dot{M}$ form a triangle. Thus, this is also referred to as *triangulation.*[1] The directions of these rays are $[\dot{M}_\infty^T, 0]^T$ and $[\dot{M}_\infty'^T, 0]^T$ (refer to Section 2.1.4) where M_∞ and M'_∞ are the intersections of both rays with the ideal plane Π_∞. Then, M can be determined by [RCF95]:

$$M = \alpha_1 \begin{bmatrix} \dot{C} \\ 1 \end{bmatrix} + \beta_1 \begin{bmatrix} \dot{M}_\infty \\ 0 \end{bmatrix} = \alpha_2 \begin{bmatrix} \dot{C}' \\ 1 \end{bmatrix} + \beta_2 \begin{bmatrix} \dot{M}'_\infty \\ 0 \end{bmatrix} \tag{2.71}$$

[1] This is not to be confused with Delaunay triangulation that may be used to generate the mesh required for 3D modeling.

Note that $\mathbf{m} = \mathtt{P}[\dot{\mathbf{M}}_\infty^T, 0]^T$ and $\mathbf{m}' = \mathtt{P}'[\dot{\mathbf{M}}_\infty'^T, 0]^T$. If $\mathtt{P} = [\tilde{\mathtt{P}}|\dot{\mathbf{P}}]$ and $\mathtt{P}' = [\tilde{\mathtt{P}}'|\dot{\mathbf{P}}']$, where $\tilde{\mathtt{P}}$ and $\tilde{\mathtt{P}}'$ are both 3×3 invertible matrices, then $\mathbf{m} = \tilde{\mathtt{P}}\dot{\mathbf{M}}_\infty$ and $\mathbf{m}' = \tilde{\mathtt{P}}'\dot{\mathbf{M}}_\infty'$. As a consequence, we get:

$$\mathbf{M}_\infty = \begin{bmatrix} \dot{\mathbf{M}}_\infty \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{\mathtt{P}}^{-1}\mathbf{m} \\ 0 \end{bmatrix} \tag{2.72}$$

and

$$\mathbf{M}_\infty' = \begin{bmatrix} \dot{\mathbf{M}}_\infty' \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{\mathtt{P}}'^{-1}\mathbf{m}' \\ 0 \end{bmatrix} \tag{2.73}$$

Also, because $\mathtt{P}\mathbf{C} = 0$ and $\mathtt{P}'\mathbf{C}' = 0$ then by utilizing Equation (2.20), we get:

$$\dot{\mathbf{C}} = -\tilde{\mathtt{P}}^{-1}\dot{\mathbf{P}} \tag{2.74}$$

and

$$\dot{\mathbf{C}}' = \tilde{\mathtt{P}}'^{-1}\dot{\mathbf{P}}' \tag{2.75}$$

Using Equation (2.71) through Equation (2.75) leads to:

$$\begin{bmatrix} \dot{\mathbf{M}}_\infty' \\ 0 \end{bmatrix} = \frac{\alpha_1}{\beta_2}\mathbf{C} + \frac{\beta_1}{\beta_2}\begin{bmatrix} \dot{\mathbf{M}}_\infty \\ 0 \end{bmatrix} - \frac{\alpha_2}{\beta_2}\mathbf{C}' \tag{2.76}$$

which represents four equations and has only three unknowns $\frac{\alpha_1}{\beta_2}$, $\frac{\beta_1}{\beta_2}$ and $\frac{\alpha_2}{\beta_2}$. Then $\dot{\mathbf{M}}$ is obtained by back substitutions into Equation (2.71).

2.4.3.3 Triangulation with Nonintersecting Rays

In the case of noise, the rays are most likely to be skew. Thus, the 3D position can be estimated as the midpoint of the perpendicular to both rays (see Figure 2.8 on Page 33). The midpoint, $\dot{\mathbf{M}}$, can be obtained from the two optical centers and the directions of the rays, $\mathbf{M}_\infty$ and $\mathbf{M}_\infty'$ [BZM97]. This is done by calculating:

$$\hat{\dot{\mathbf{M}}} = \left([\mathtt{I} - \dot{\mathbf{M}}_\infty\dot{\mathbf{M}}_\infty^T] + [\mathtt{I} - \dot{\mathbf{M}}_\infty'\dot{\mathbf{M}}_\infty'^T]\right)^{-1}\left(\dot{\mathbf{C}} + \dot{\mathbf{C}}' - [\dot{\mathbf{C}}^T\dot{\mathbf{M}}_\infty]\dot{\mathbf{M}}_\infty - [\dot{\mathbf{C}}'^T\dot{\mathbf{M}}_\infty']\dot{\mathbf{M}}_\infty'\right) \tag{2.77}$$

where $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}_\infty'$ are normalized to unit magnitude. The values of $\dot{\mathbf{M}}_\infty$, $\dot{\mathbf{M}}_\infty'$, $\dot{\mathbf{C}}$ $\dot{\mathbf{C}}'$ are calculated using Equation (2.72) through Equation (2.75).

2.4.4 Noise and Inaccuracy Correction

Different approaches [BZM97] are possible to deal with noise which is most likely to occur. Methods developed may include the following:

1. Orthogonal projection on image plane. The steps are the following:
   - Compute the epipolar line $\mathbf{l}'$ for a given image point $\mathbf{m}$.
   - Compute the orthogonal projection of $\mathbf{m}'$ onto $\mathbf{l}'$.

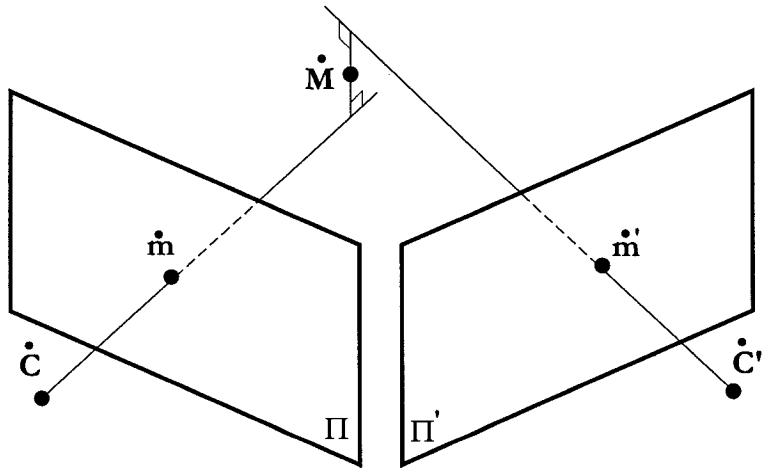


Figure 2.8: A point in space can be considered as the midpoint of the perpendicular to both rays.

- Use the first point $\mathbf{m}$ and the orthogonal projection of its corresponding one to get two rays guaranteed to intersect in space since they are on the same epipolar line.
- It is assumed that all the errors are in the second image which may be a wrong assumption.

2. Distance minimization on image plane:

   Generally, two corresponding points, $\dot{\mathbf{m}}$ and $\dot{\mathbf{m}}'$ do not satisfy the epipolar constraint due to the existence of noise or the inaccuracy in camera parameters. Thus, it is required to estimate the points $\hat{\dot{\mathbf{m}}}$ and $\hat{\dot{\mathbf{m}}}'$ that:

   - minimize the function:

   $$d(\mathbf{m}, \hat{\mathbf{m}})^2 + d(\mathbf{m}', \hat{\mathbf{m}}')^2 \tag{2.78}$$

   where $d()$ represents the Euclidean distance on the image plane; and

   - agree with the epipolar constraint:

   $$\hat{\mathbf{m}}'^T \mathbf{F} \hat{\mathbf{m}} = 0 \tag{2.79}$$

This method ensures that the two optical rays passing through $\hat{\mathbf{m}}$ and $\hat{\mathbf{m}}'$ will intersect in space. Thus, correction is a good starting point to detect the 3D position of the point. In other words, finding $\hat{\mathbf{m}}$ and $\hat{\mathbf{m}}'$ leads to $\dot{\mathbf{M}}$ detection by triangulation (Section 2.4.3.2) since the corresponding back-projection rays should intersect precisely in space.

In [HS95, HS94], Hartley *et al* reformulated the problem differently. They requested to minimize the function:

$$d(\mathbf{m}, \mathbf{l})^2 + d(\mathbf{m}', \mathbf{l}')^2 \tag{2.80}$$

where $\mathbf{l}$ and $\mathbf{l}'$ range over all choices of corresponding epipolar lines. The steps of this method, which is referred to as *optimal correction*, are the following:

(a) Parametrize the pencil of the epipolar lines in the first image by a parameter t.

(b) Compute the conjugate epipolar line, $\mathbf{l}'$, of $\mathbf{l}$ using the fundamental matrix $\mathtt{F}$.

(c) Express Equation (2.80) as a function of t.

(d) Take the derivative of the last step.

(e) Maxima and minima happen when the derivative equals to 0.

(f) The last equality (where the derivative $= 0$) is a polynomial of degree 6 that may have up to 6 real roots. Solving such a polynomial makes this method computationally expensive.

This method ensures that the two optical rays passing through $\hat{\mathbf{m}}$ and $\hat{\mathbf{m}}'$ will intersect in space.

3. Distance minimization in space:
An estimation for the 3D point can be achieved using the method described in Section 2.4.3.3.

2.5 Summary

In the preceding chapter, we explored many fundamental aspects of Projective Geometry that are used frequently in Computer Vision research. We started by declaring the notation used throughout the thesis; e.g., points, lines, planes and directions. This was followed by explaining the projective model and the role of the perspective projection matrix, $\mathtt{P}$, that determines the relationship between the world coordinate system and the image coordinate system. This matrix was shown in its simplest and general forms. Binocular vision concept was discussed. This was followed by a discussion about the fundamental matrix, $\mathtt{F}$, and its relation with the projection matrix. Then, we moved to investigate the essential matrix, $\mathtt{E}$. This was followed by exploring the concept of plane homography. The last main part of this chapter handled some important three-dimensional reconstruction concepts. It discussed different approaches for 3D line and point reconstructions. Finally, this chapter ended with a discussion on noise and inaccuracy correction.

Materials drawn from this chapter will be used in different phases of our modeling system. For example, the projective model, discussed in Section 2.2 will be used to estimate the planar perspective matrix of Chapter 4. The inter-image homography, discussed in Section 2.3.5, will be estimated in Chapter 4 as well. The fundamental matrix, discussed in Section 2.3.2, will be used in Chapter 5 to estimate the epipolar geometry. Also, 3D line reconstruction, discussed

in Section 2.4.2, will be used in Chapter 5 as well. Point reconstruction, discussed in Section 2.4.3, will be used in Chapter 6. The similarity matrix of Section 2.3.5 will be utilized in Chapter 7. The topics covered in this chapter have been published in [EL01a, Eli03b].

Chapter 3

Junction Detection

In Section 2.4.1, we mentioned that a solution to the three-dimensional reconstruction problem can be split into two distinctive steps. Those steps are:

1. extracting and matching features between corresponding images; e.g., wide baseline matching; and
2. determining structure from corresponding features.

Distinctive features of an image carrying meaningful information are often called *interest points*. A 3D reconstruction algorithm as well as many algorithms in Image Analysis and Pattern Recognition rely on the extraction of interest points; i.e., the first part of Step 1 mentioned above. The choice of interest points plays a key role in many of those algorithms. In addition, the richness of information an interest point operator can provide, the speed of getting it as well as the accuracy of such an information can affect the whole process significantly. In a tough application like wide baseline matching, which we address in Chapter 5 (the second part of Step 1), gathering as much rich, fast and accurate information as possible among views becomes a priority.

This chapter presents a new junction detection algorithm that can be used as an interest point detector. This operator assumes that the branches of the detected junctions can be well approximated as straight lines in the vicinity of the point of convergence. These junction radial lines can be found from the identification of what we have called *circumferential anchor* (CA) points. This operator detects not only junction locations but their orientations and strengths as well resulting in rich information that helps make the wide baseline matching process more viewpoint invariant and hence more reliable. Furthermore, binary maps rather than their original images are used which results in speeding up the process. We claim that this operator can provide rich, fast and accurate information about the localization and characterization of the interest point. Tests made show that orientation angles gathered through this operator

are of very good accuracy. This is the most important feature concerning this operator that distinguishes it from an ordinary corner detector.

The rest of the chapter is organized as follows. The next section presents a quick review of some interest point operators. Section 3.2 discusses the problem of junction detection. Section 3.3 describes the proposed JUDOCA algorithm while some experimental results are presented in Section 3.4.

3.1 Interest Point Detection

In the past, several works have proposed various operators that can detect interest points. Most of the time, these operators look for prominent features that correspond to high-curvature boundaries in the image. For this reason, the detected features are often referred to as *corners* even if, in reality, these operators may be sensitive to a wider variety of intensity patterns. This leads us to a possible way of defining interest points. This is to describe them as image points where two or more edges meet. Interest points that comply with this definition could be called corners or *junctions*. The number of converging edges and their respective orientations constitute the signature characterizing the junction. In fact, the existence of such a characterization constitutes probably the way to differentiate corner detectors and junction detectors. Indeed, most interest point detectors output the strength of their detected feature. Once a threshold is applied to these values, the subsequent processing stages are left with a binary image carrying only information about the feature location. In contrast, a junction detector would extract the relevant attributes of the detected feature, thus subsuming a more informative model of the point of interest and of its vicinity.

Methods for characterizing junctions have been proposed in the past. However, their computational complexity generally do not allow them to be used for junction detection and localization. Instead, they would be applied to selected interest points having been previously detected by some efficient corner detector. This is the case of the method proposed in [WY98], which is based on the successive application of rotated wedge averaging filters around a given key point. A 1D derivative filter is applied with respect to the polar angle to the filter responses and the junction is then characterized by the maxima of the resulting function. The approach described in [PGH98] detects junctions using a piecewise constant function that partitions a circular template into wedge-shaped regions. The detection and characterization is achieved, in this case, through energy minimization in order to find the best approximating junction model. Although the method proposed leads to an effective solution, the required computational effort remains important. The corner detector in [Roh92] is based on a similar approach where junctions are detected by minimizing the sum of squared

differences between a parametric model and the observed intensity values. The work presented in [RT01] is another approach based on the use of a model. However, this time, the model does not represent a neighborhood by segments of constant intensity but by a color distribution. This makes the detector particularly suitable for detection at a large scale in color images.

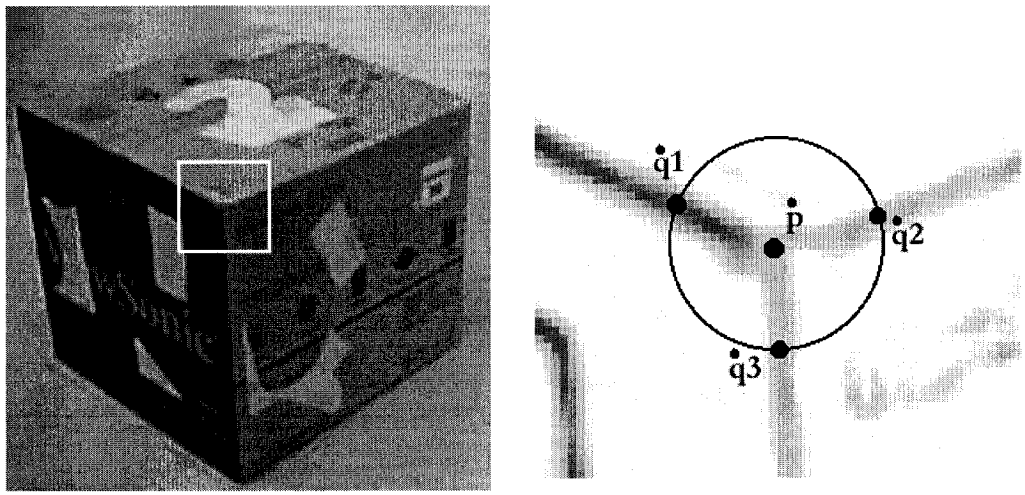
Another category of approaches proceeds to the detection of junction by looking at the edge image. For example, in [Bey91], a junction detector is proposed that makes use of the local properties of the gradient magnitude near junction points to reconnect broken edges. In [FD00], junctions are detected by measuring the rate of change of the orientation vector along lines within a given neighborhood. The detector in [BP01] uses a two-pass Hough transform to identify points of intersection in an edge map. The corner detector presented in [MS98] also includes the detection of T-junctions but this is done mainly to eliminate false corners. For a review of different corner detectors, one may refer to [SB95].

3.2 Junction Detection

Junctions in images occur when several uniform regions join at one prominent point; i.e., the point of junction, where the boundaries of the adjacent regions meet. When a junction is produced by only two regions, this definition assumes a high curvature boundary where the point of junction corresponds to the point of maximal curvature. In this case, the angle between the two edges forming the junction deviates from 180°.

Assuming a symmetrical transition between the adjacent regions, then locations of junctions will correspond to points where ridges in the gradient domain of the image intersect. This suggests that junction detection can be accomplished in the gradient domain. Moreover, in order to facilitate the detection of junctions, a further restriction can be introduced; the class of junctions to be detected will be restricted to those where the joining edges can roughly be approximated by straight lines in the vicinity of the location of the junction. The detector proposed in this chapter is built around these ideas. It defines the vicinity of a putative point of junction as a circular neighborhood of radius λ and centered at the point of junction. The radial lines that form the junction must therefore lie on some rays of this disc, as illustrated in Figure 3.1 on Page 39. In addition, the following two properties should hold for such junctions:

1. The value of the gradient of the points of intersection between the circle that delimits the junction's neighborhood and each of the junction radial lines must correspond to a local maximum in the direction of the gradient (a ridge point).



(a) (b)

Figure 3.1: (a) One corner of a box that produces a Y-junction. (b) The junction at $\dot{\mathbf{p}}$ with three circumferential anchors $\dot{\mathbf{q}}_1, \dot{\mathbf{q}}_2$ and $\dot{\mathbf{q}}_3$ (superimposed on the gradient image).

2. The value of the gradient along the ray on which each junction branch lies must always be greater than a predetermined value (not necessarily on a ridge though, in order to permit some minor deviations from the straight edge model).

Note that constraints on the orientation of the gradient on each junction radial line could have been added. However, it can be shown that the above two properties somewhat subsume that the gradient orientation will be consistent with the orientation of each junction branches.

Consequently, there will be a junction at a given image point if at least two rays obeying properties 1 and 2 can be emitted from this point. We must add that, if only two such rays are found, then the angle between the two rays must not be within 180° range that is determined according to a predefined value. The strength of such a junction should be proportional to the sum of the gradient magnitudes along each accepted ray.

Properties 1 and 2 form the basic principles through which junction detection and characterization can be performed. An operator that makes use of these properties is presented in the next section. The underlying algorithm is based on the identification of local directional maxima of the intensity gradient values located on the circumference of a circular neighborhood. When devising this algorithm, one concern was to obtain an operator that can perform junction detection in a fast efficient manner. For this reason, it operates on binary edge maps. As it will be seen, this simplification significantly reduces the computational cost of the method while still detecting junctions that adhere to properties 1 and 2. The proposed

algorithm has been named JUDOCA for JUnction Detection Operator from Circumferential Anchor points.

3.3 The JUDOCA Algorithm

In order to detect junctions, the proposed operator relies on the intensity gradient. This can be conveniently estimated using Gaussian derivative filters. The size of the Gaussian filter, which is specified by a variance parameter, σ, also offers the advantage of allowing edge detection at different scales; i.e., only more prominent edges being detected for larger value of σ. Note, however, that if good localization is required, smaller σ value should be used.

The algorithm proceeds as follows [LE, LE04]:

1. Apply vertical and horizontal Gaussian derivative filters on image $\mathcal{I}$.
2. Compute the gradient magnitude and create two binary images from it.
   (a) The first image, $\mathcal{B}$, is created by imposing a threshold, $t_{\mathcal{B}}$, on the gradient image.
   (b) The second image, $\mathcal{B}^+$, contains the points of $\mathcal{B}$ that are local maxima in the direction of the gradient.
3. For each point $\dot{\mathbf{p}}$ in $\mathcal{B}$, consider a circle of radius λ centered at this point and obtain the list of candidate points $\dot{\mathbf{q}}_i$ in $\mathcal{B}^+$ that lie on the circumference of this circle. These so-called circumferential anchor (CA) points are the extremities of potential radial lines for the putative junction (see Figure 3.1).
4. For each CA point, $\dot{\mathbf{q}}_i$ in the list, consider the points located at a distance less than one pixel to the segment that joins the current CA point to the central point $\dot{\mathbf{p}}$. Two sets are used; one that belongs to $\mathcal{B}$ and the other belongs to $\mathcal{B}^+$. The first set is used to determine if the corresponding putative junction radial line will be accepted and the other to determine what the strength of this junction line will be. That is:
   (a) To be accepted as a junction edge, a continuous path of $\mathcal{B}$ points joining the CA point and the central point $\dot{\mathbf{p}}$ must exist. If not, then reject this radial line and repeat the scanning operation with the next CA point.
   (b) The strength, $S_J(< \dot{\mathbf{p}}, \dot{\mathbf{q}}_i >)$, of this junction radial line is considered as the percentage of the points allocated in $\mathcal{B}^+$ along the line segment $< \dot{\mathbf{p}}, \dot{\mathbf{q}}_i >$. Instead, it may be defined as the sum of the squared distances from the $< \dot{\mathbf{p}}, \dot{\mathbf{q}}_i >$ line segment to the $\mathcal{B}^+$ points in the currently considered set. This strength is normalized by the length of this segment.

5. If the strength of this junction radial line is smaller than a predetermined threshold, s_J, then reject this radial line. Otherwise, $< \dot{\mathbf{p}}, \dot{\mathbf{q}}_i >$ becomes one of the branch of the putative junction at $\dot{\mathbf{p}}$.

6. If the number of branches found at $\dot{\mathbf{p}}$ is less than 2, then there is no junction at this location. Otherwise, record the orientation of the accepted junction radial lines and set the strength, $S_J(\dot{\mathbf{p}})$, of this junction as being the minimum across all strengths $S_j(< \dot{\mathbf{p}}, \dot{\mathbf{q}}_i >)$.

In the particular case of 2-edge junction, an extra step must be undertaken in order to ensure that this junction is not, in fact, a simple line. This can be verified by looking at the angle between the two radial lines. If this one is close to 180°, then the junction should be rejected. In practice, a threshold is set on the maximal and minimal acceptable angle for 2-edge junctions.

In Step 3, the image points located on a circular circumference must be considered. This can be done by pre-computing a discretized circular contour of appropriate radius. Identifying the circular anchor at each location can then be done in a very efficient manner.

Additionally, the strength of the junction can be used to perform a non-maxima suppression as a post-processing phase to eliminate clusters of junction that could arise, especially if a permissive threshold is used in Step 2(a). Note that different criteria for the junction strength could have been adopted. However, the one given in Step 6 is based on an assertion stating that a junction is as weak as its weakest branch. This criterion also implies that a weak N-edge junctions can be transformed into a stronger $(N-1)$-edge junction.

3.4 Experimental Results

Figure 3.2 on Page 42 illustrates an experiment where the detector has been tested on a synthetic image to which Gaussian noise has been added. By increasing the value of σ, it has been possible to considerably reduce the impact of noise on the junction detection operation. In Figure 3.3 on Page 42, the noise level has been further increased. This time, it can be seen that it is by the reduction of the edge detection threshold t_B, it has become possible to detect the junction of the low contrasted square of our test image.

The accuracy of the junction orientation is better with larger value of λ. In order to demonstrate this fact, we took different pictures of a triangular shape (28 in total, some of them being shown in Figure 3.4 on Page 43). Knowing that, no matter what the point of view is, the sum of the three angles should result in 180°. For each of these images, we computed the sum of the junction angles as detected by JUDOCA. We repeated this experiment for

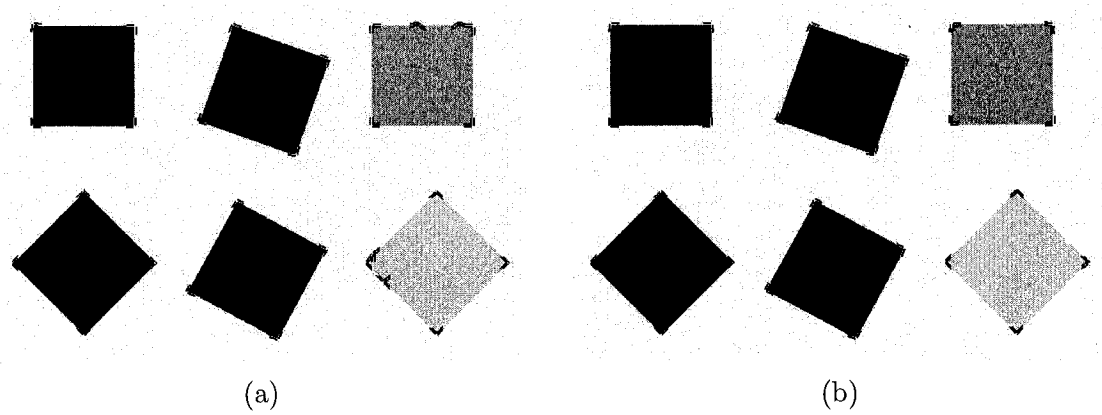


Figure 3.2: Junction detection on a synthetic test image with Gaussian noise of variance=400. (a) Detected junctions with $\sigma = 1.0$. (b) Detected junctions with $\sigma = 1.5$.

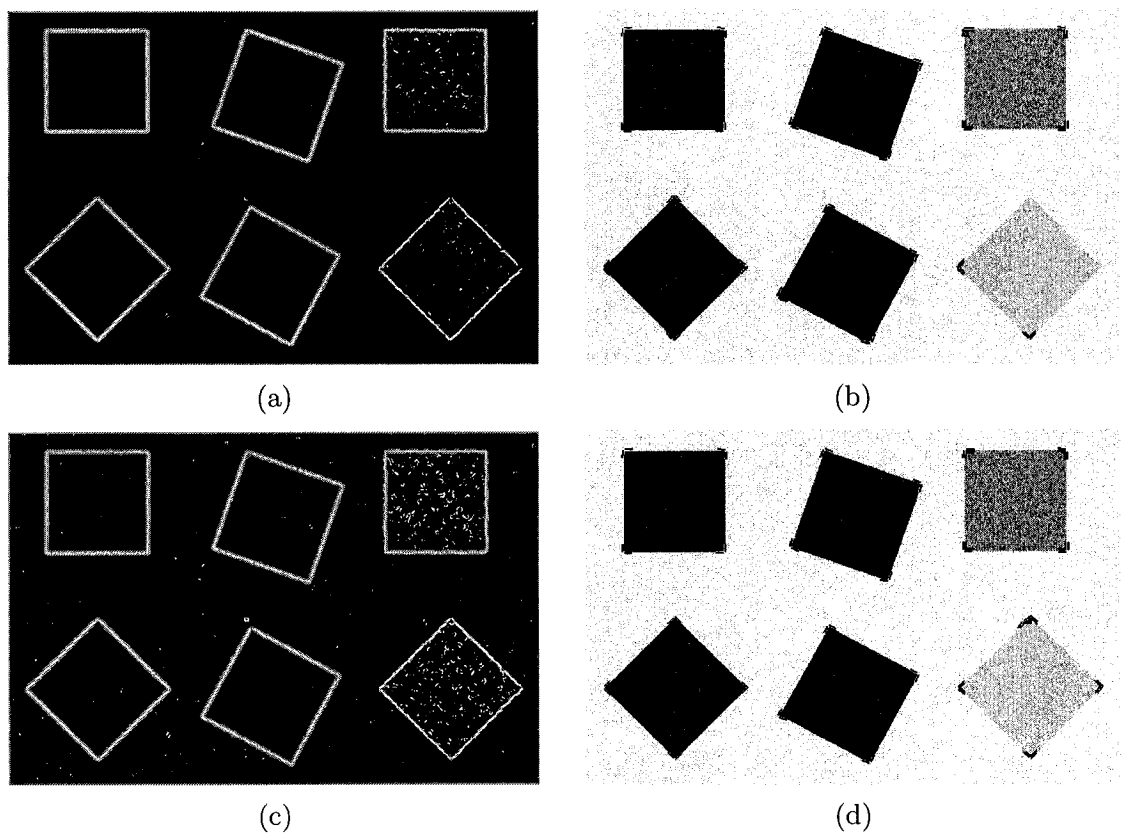


Figure 3.3: Junction detection on a synthetic test image with Gaussian noise of variance=1000. (a) Edge image with $\sigma = 1.5$ and $t_{\mathcal{B}} = 10$. (b) Detected junctions with $\mathcal{B}^+$ and $\mathcal{B}$ computed from Image (a). (c) Edge image with $\sigma = 1.5$ and $t_{\mathcal{B}} = 8$. (d) Detected junctions with $\mathcal{B}^+$ and $\mathcal{B}$ computed from Image (c).

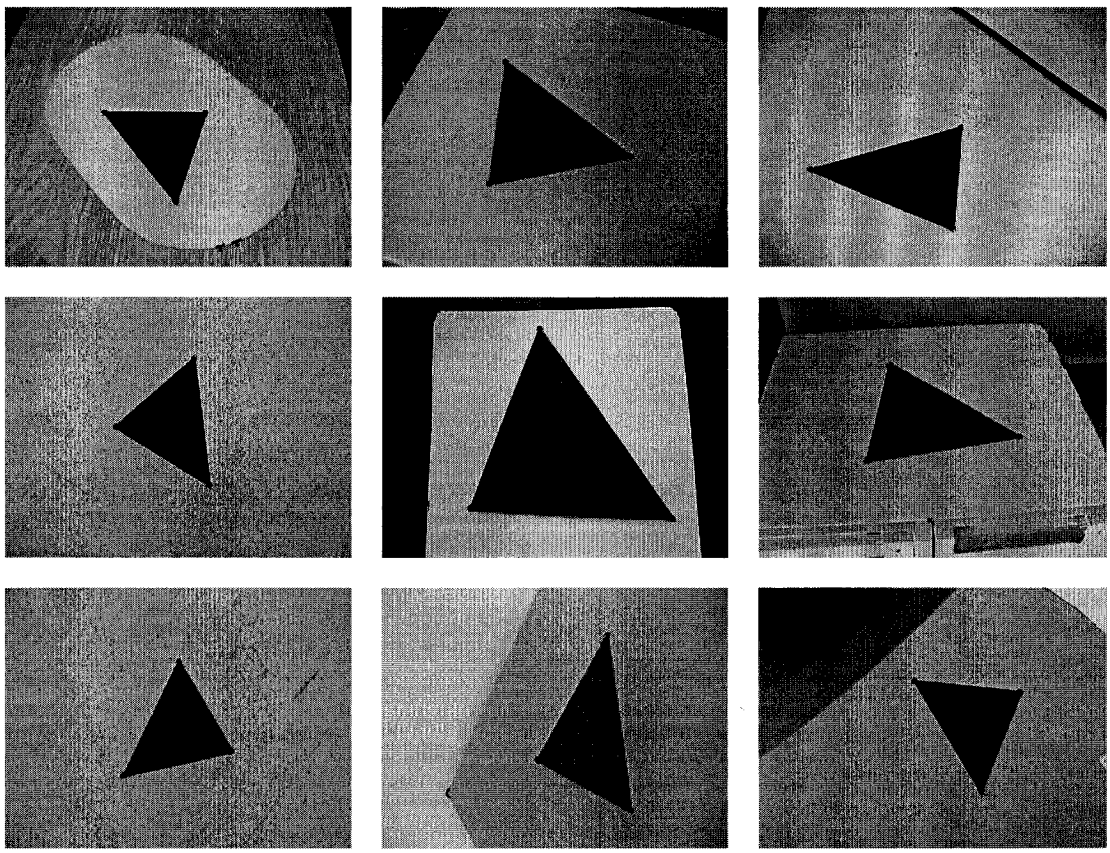


Figure 3.4: Different images of a triangle used to test junction orientation accuracy.

different values of λ. The results are shown in the graph of Figure 3.5 on Page 44. As expected, the mean value of this sum approaches 180° as λ increases while the variance decreases with increasing λ. This shows that not only do longer radial lines increase the accuracy of the angles but also make these measurements more reliable. Also, note that, this experiment shows that the operator tends to slightly overestimate the value of the angles. This bias can be explained by the smoothing effect of the Gaussian filter that blunts the tip of sharp corners.

Another experiment to support our claim about the accuracy provided by the operator is the following. Consider the "chess board" in Figure 3.6(a) on Page 45. Suppose that a 3D histogram is established such that one axis spans the horizontal angle values from +315° (or −45°) to +45° and from 135° to +225° (or −135°) and the second axis spans the values from 45° to 135° and from 225° (or −135°) to 315° (or −45°) while the last axis counts the histogram values. In that sense, there is no difference between the angle δ and the angle $\delta \pm 180°$. Both angles will contribute to the same entry in the histogram. Applying this to

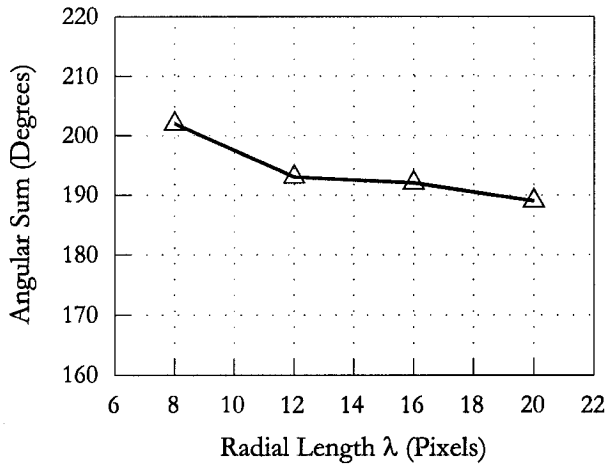


Figure 3.5: Average angular sum of the junctions detected on 28 different images of a triangle as computed for different values of λ.

the image in Figure 3.6(a), the histogram achieved is shown in Figure 3.6(b). The angle step used in this histogram is 5°. The peak of the histogram happens around angles $-5°$ and $+90°$, which is in the expected range.

Figures 3.7, 3.8 and 3.10 on Pages 46, 47 and 49 show the results obtained when applying the JUDOCA operator to more complex real-world images. In Figure 3.7, we categorized junctions by their number of linear ridges forming them. Figures 3.7(b) shows 2-edge junctions only while Figure 3.7(c) shows 3-edge junctions. Notice that many junctions were detected in the plant regions in Figure 3.7. The reason is the existence of high frequency regions that result in short but many ridges intersecting in the gradient domain constituting junctions.

As noted, the accuracy of the detector is better with larger value of λ. In some cases, using large value for λ may result in rejecting correct junctions. For example, consider some cases shown in Figure 3.8(a). The face of the box is reproduced in Figure 3.9(a) on Page 48. The gray edges represent the binary image $\mathcal{B}$ while the black edges represent the binary image $\mathcal{B}^+$. Note that no junctions were detected at the centers of the circles shown. The radii of these circles represent the value of λ used. It is obvious from this image that the CA points were detected on other ridges other than the right ridges. The reason is that the correct ridges were too short comparing to the value of λ, which results abandoning those junctions. Notice that the left circle passes beside the correct point but does not pass through it. We repeated the process again using the same parameters after modifying λ from 13 to 12 and this particular junction was successfully detected.

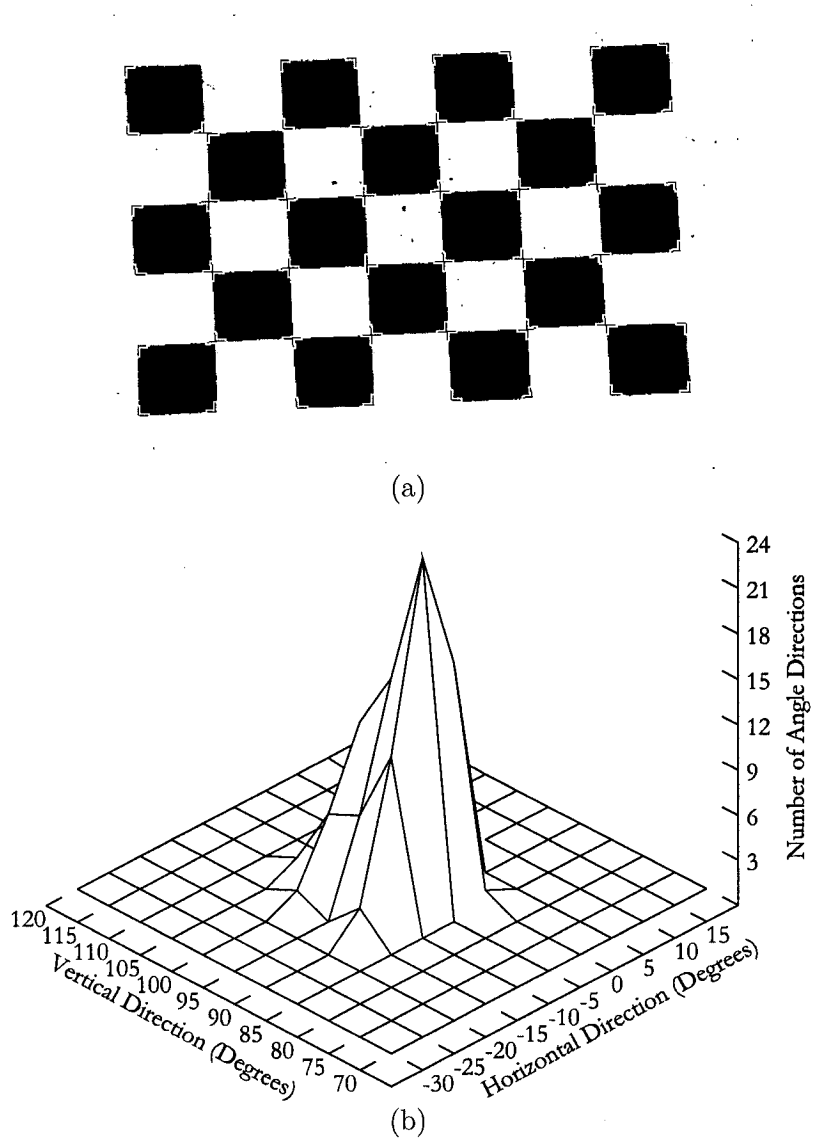


Figure 3.6: (a) Accuracy example: Chess board. (b) The histogram of the angle directions. The parameters used are $\sigma = 1.0$, $t_B = 10$, $\lambda = 15$ and $s_J = 0.5$.

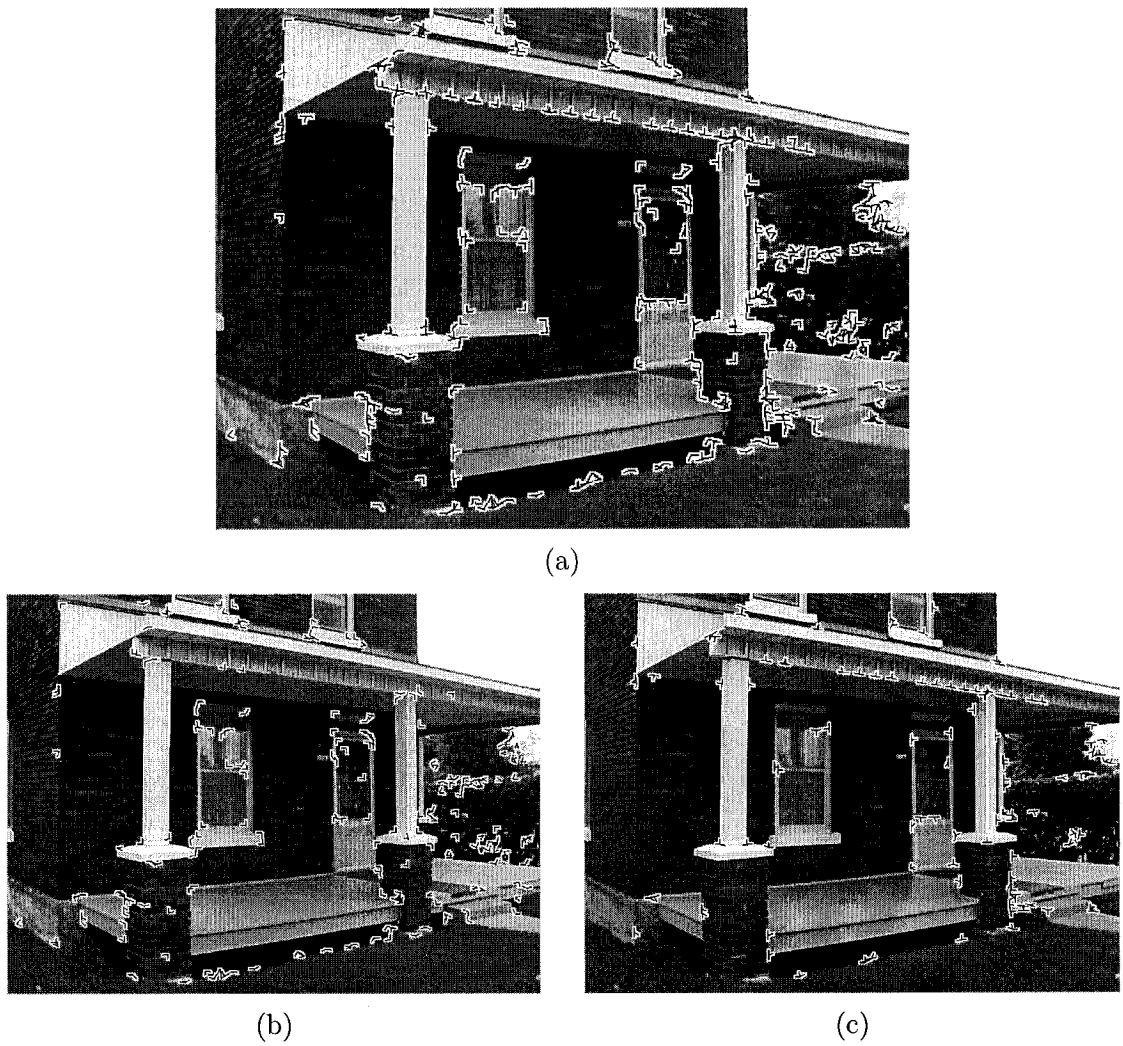
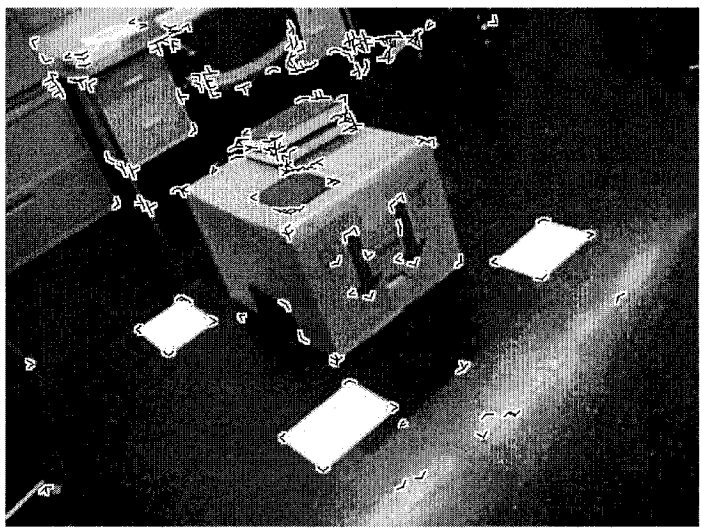
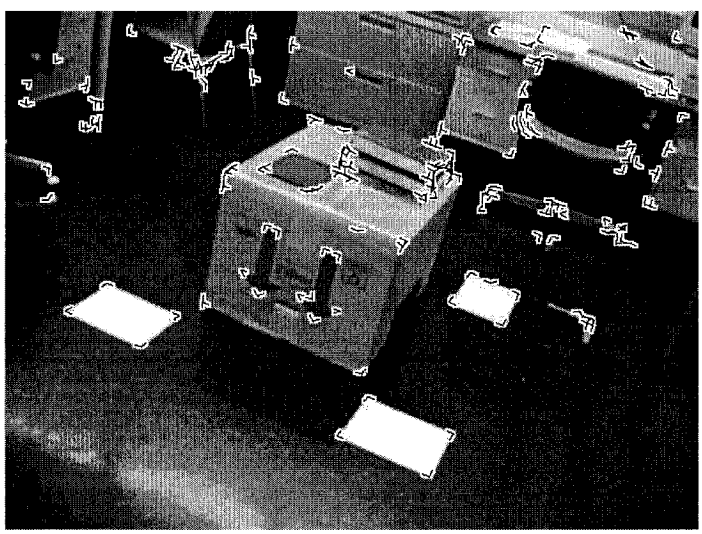


Figure 3.7: (a) Junction detection using JUDOCA with $\sigma = 1.5$, $t_B = 6$, $\lambda = 10$ and $s_J = 0.5$. (b) 2-edge junctions only. (c) 3-edge junctions only.

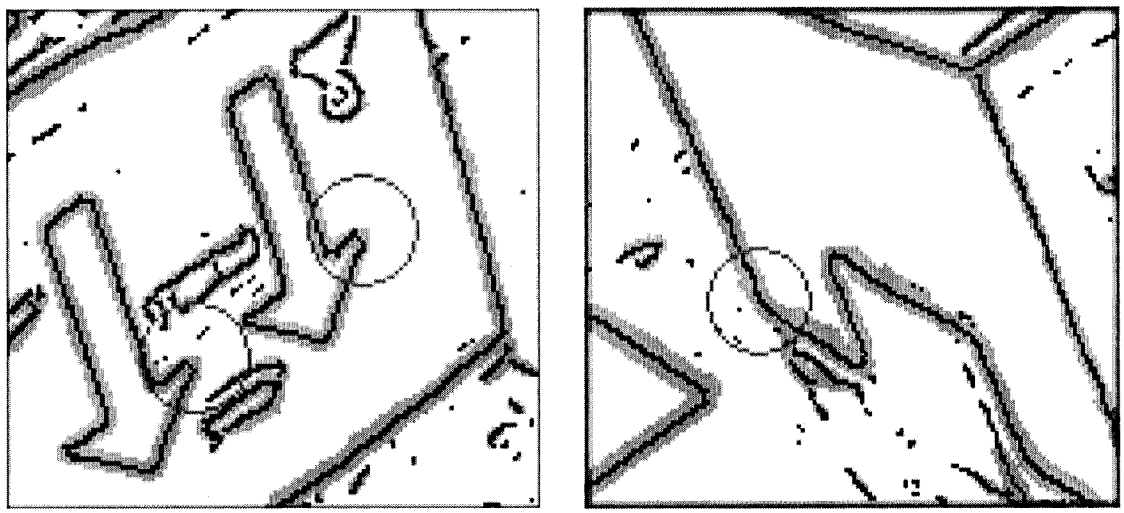


(a)



(b)

Figure 3.8: A pair of wide baseline images. JUDOCA parameters used are $\sigma = 1.2$, $t_{\mathcal{B}} = 5$, $\lambda = 13$ and $s_J = 0.45$.

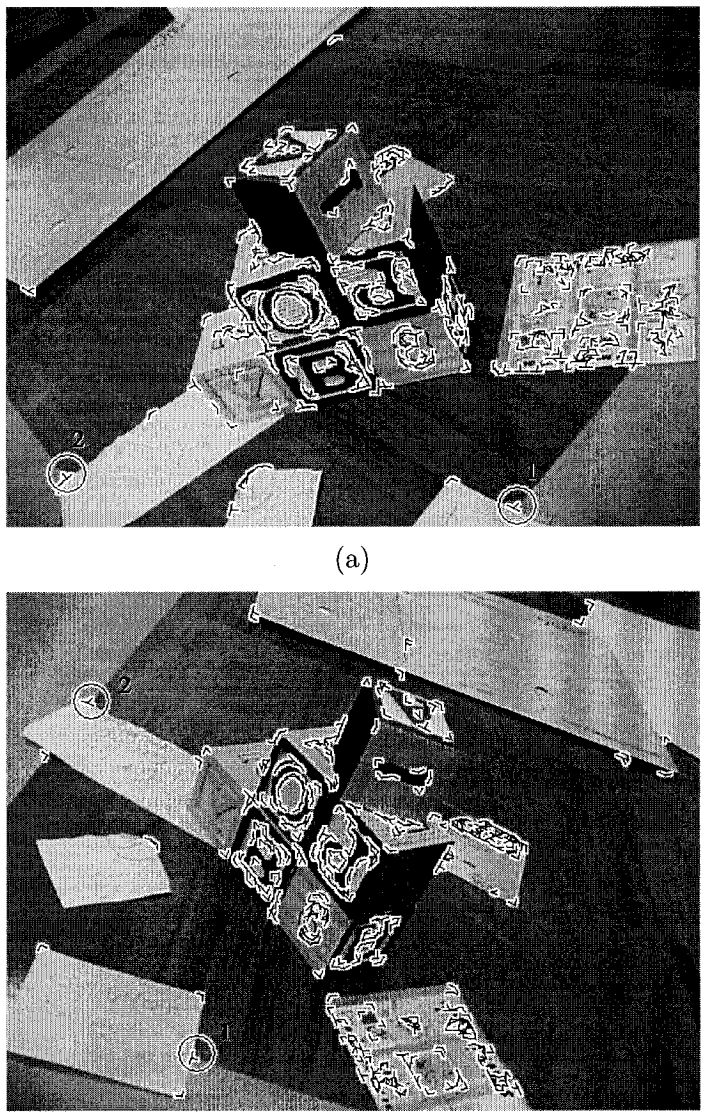


(a) (b)

Figure 3.9: Limitations in detecting junctions.

Also, stated above is that if a 2-edge junction is detected, the angle between the branches should be tested in order to check if it is not a simple line. We have to determine maximal and minimal acceptable angles. Consider the junction at the center of the circle of Figure 3.9(b). This junction was not detected because the angle between edges was not in the acceptable range.

In Figure 3.10, we show another test for the accuracy of our detector. This is done through checking the sum of angles where straight lines are present in junctions. Straight lines remain straight under projective transformation. Thus, the sum of the angles contributing to a straight line should be 180°. To test that, consider the junctions in circles "1" and "2." In Figure 3.10(a), the angles for junction "1" are 29.05°, 203.96° and 315.00° while in Figure 3.10(b), the angles are 16.70°, 106.70°, 281.31°. Thus, the sum of angles representing a straight line for junction "1" is 185.09° or 174.91° for Figure 3.10(a) and 185.39° or 174.61° for Figure 3.10(b). Now consider the junction "2" where the angles detected are 143.13°, 233.13°, 323.13° for Figure 3.10(a) and 23.96°, 196.70°, 315.00° for Figure 3.10(b). This provides us with a sum of exactly 180° for Figure 3.10(a) and 187.26° or 172.74° for Figure 3.10(b).



(a)

(b)

Figure 3.10: A pair of wide baseline images. The parameters used to detect junctions are $\sigma = 1.2$, $t_{\mathcal{B}} = 8$, $\lambda = 10$ and $s_J = 0.5$.

3.5 Parameters and Thresholds

In the JUDOCA algorithm, there are four parameters and thresholds that control the detection of junctions. The first two, σ and $t_{\mathcal{B}}$, are concerned with the creation of the two binary edge maps, $\mathcal{B}$ and $\mathcal{B}^+$. Their values influence the sensitivity of the detection and determine the scale at which the junction detection is performed. For example, comparing Figure 3.11(a) with Figure 3.11(b) on Page 51, it appears that several textural details can be filtered out, if wished, by a proper choice of σ. Parameter σ plays also an important role regarding the noise robustness of the operator as illustrated in Figure 3.2. In low contrasted situations like that of Figure 3.2, the reduction of the edge detection threshold $t_{\mathcal{B}}$ plays a significant role in detecting edges and hence detecting junctions.

The length, λ, of the radial lines that are scanned during the detection process, has an impact on the number of junctions found. Normally longer lines are more desirable since they better support the defined junction model. By choosing smaller length, the detector will act as a "regular" corner detector sensitive to prominent high curvature points. This is illustrated in Figure 3.11(c) where a smaller radial length has been used. We found that values between 8 and 20 usually give good results. Note also that setting λ at a too high value can significantly increase the computational cost of the detection.

The threshold s_J defines the degree of adherence to the straight edge assumption. Indeed, because of noise, shading and contrast variation, the edge contours in $\mathcal{B}^+$ will not be perfectly straight (this effect is clear in Figures 3.3(a) and (c)). Consequently, a level of tolerance must be defined in the acceptance of a junction. If one wants to perform a more selective junction detection, the value of this threshold can be made higher, at the price of missing some good junctions. Generally, s_J varies between 50% and 90%. This percentage roughly represents the minimal acceptable portion of the radial line having a $\mathcal{B}^+$ point close to it. Obviously, the strength of an accepted junction will always be higher than this value. Figure 3.11(d) shows a result obtained when using a stricter value for s_J. Finally, Section A.2 summarizes the role of each of the above-mentioned parameters and thresholds.

3.6 Summary

Junctions in images are important distinctive features. This chapter presented a new junction detection operator that can be used as an interest point detector. This operator was introduced so that a novel and effective wide baseline matching algorithm applied in the context of calibrated scenes can be devised. Our operator defined junctions as points where linear ridges in the gradient domain intersect. The radial lines composing the junction were therefore identified by searching, in a circular neighborhood, for directional maxima of the

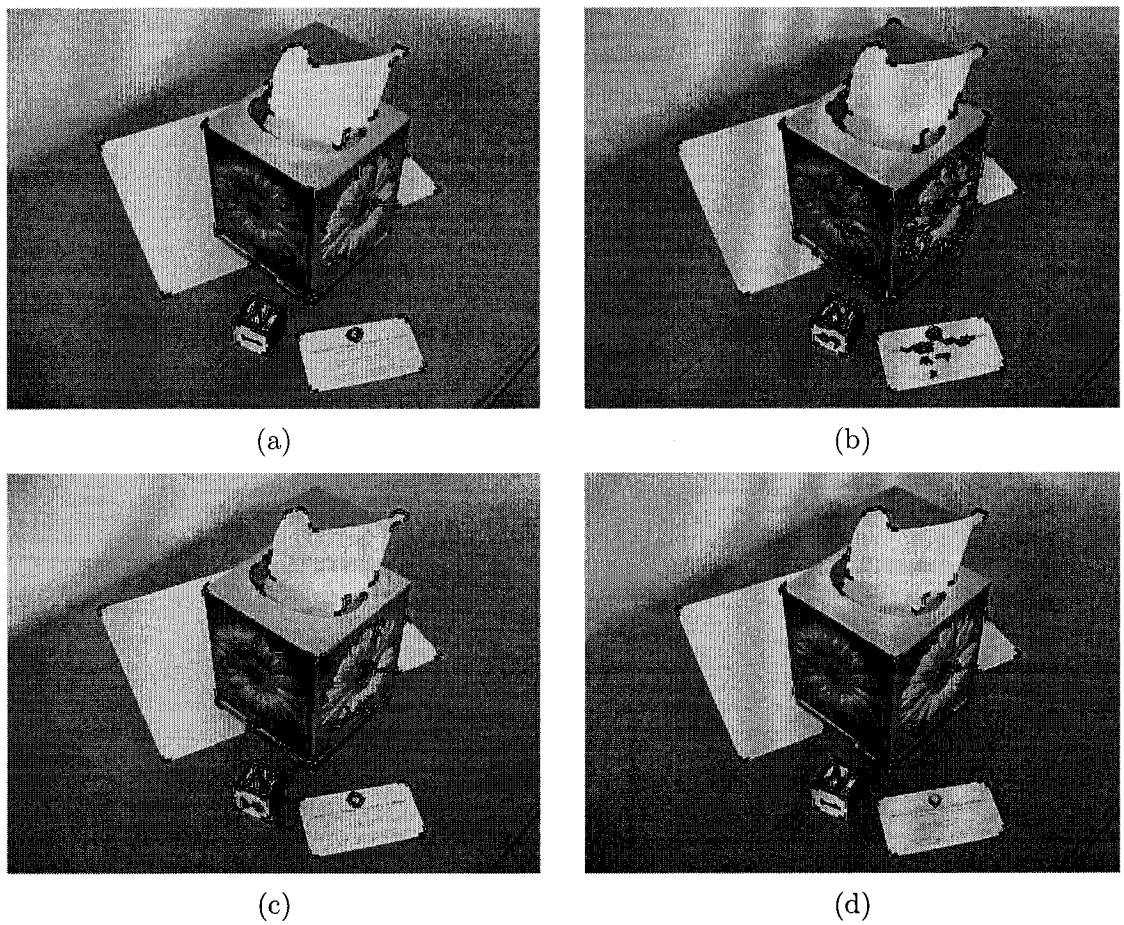


Figure 3.11: Junction detection on a simple image using JUDOCA with; (a) $\sigma = 1.8$, $t_{\mathcal{B}} = 7$, $\lambda = 10$ and $s_J = 0.6$. (b) $\sigma = 1.0$, $t_{\mathcal{B}} = 7$, $\lambda = 10$ and $s_J = 0.6$. (c) $\sigma = 1.8$, $t_{\mathcal{B}} = 7$, $\lambda = 7$ and $s_J = 0.6$. (d) $\sigma = 1.8$, $t_{\mathcal{B}} = 7$, $\lambda = 10$ and $s_J = 0.7$.

intensity gradient. The proposed algorithm operated on two binary edge maps. Thus, the computational time of the detection process was considerably reduced. This operator characterized the identified junctions by the number of converging branches and their respective orientations. A measure was also proposed for the strength of a junction. Through its efficiency, this operator can provide rich and reliable information to subsequent steps, such as matching.

The output obtained by our detector will be used in the following chapters. In Chapter 4, we will use the junction positions provided by JUDOCA in an algorithm to exclude feature points that belong to the ground plane. In other words, it will be used as a corner detector. However, the big impact of this operator will appear in Chapter 5 where the orientations of edges forming a junction will be utilized; consequently, this will allow us to propose a novel wide baseline matching algorithm that outperforms the ordinary techniques. The materials of this chapter are the topics of [LE, LE04]. An implementation of the JUDOCA algorithm has be made available to interested researchers through our Web site.[1]

[1] http://www.site.uottawa.ca/school/research/viva/projects/judoca/

Chapter 4

The Planar Perspective Matrix

In this chapter, we will discuss the derivation of what we have called the planar perspective matrix. This 3×3 matrix will be used to achieve two goals. The first one is to generate perspective views for the planar ground surface. As we will show in Chapter 7, this matrix will be integrated with a path representing different robot locations that form a set of viewpoints for which a sequence of images is requested. This sequence should represent the environment as seen by a robot moving along this path. As a second goal, this matrix will be used as a step to identify feature points that do not belong to the work surface as they deviate from the homography mapping. Hence, matching process, discussed in Chapter 5, may be carried out on those features only. Consequently, this chapter includes two main parts; i.e., the derivation of the planar perspective matrix and detecting obstacle features.

The first part of this chapter is represented by Sections 4.1 through 4.6. In this part and in order to understand how this matrix is obtained, we begin by analyzing the geometric relationships that affect the appearance of a perspective view. Then, we go through the details of each possible camera movement (i.e., rotation, translation) and integrate all of them together to get the planar perspective matrix. An example is provided where the planar perspective matrix is utilized to generate a perspective view for a pattern. A detailed description of each term constituting this matrix can be found in Appendix B.

The other part is represented by Sections 4.7 through 4.10. This part deals with distinguishing feature points (detected by JUDOCA operator) that belong to obstacles from those belong to the ground plane. This will be done through a matching technique where feature points that do not obey to the homographic relation will be excluded. We will describe the steps of the technique we use. At the end of the chapter, we will show examples where we use our method on pairs of images.

4.1 Overhead View Transformation

The overhead or top view of a plane can be obtained, from images showing that plane, by applying the appropriate transformation. Many authors have used overhead views in different applications such as augmented reality and video surveillance. Bradshaw *et al.* [BRM97] used known ground plane features, such as lines, to transform planar motion into an overhead view and then recover the trajectory of moving objects on the plane. Intille and Bobick [IB95] used overhead view to track football players on the coordinate system of the field rather than the image coordinates. The overhead view is calculated using known measurements of the lines on the football field. Lee, Romano and Stein [LRS00] used multiple cameras to obtain an overhead view without depending on known landmarks or manual registration. Instead, they detected objects moving simultaneously with overlapping views. The object centroids are used as potential point correspondences to recover the homographies between camera pairs. Laganière [Lag00] used an overhead view to build a bird's eye view of a work site. The overhead view is calculated assuming known intrinsic camera parameters and known orientation with respect to the observed plane. In the sections to follow, we will show how to generate a perspective view from the overhead view of a ground plane. Appendix D discusses an approach where Computer Graphics concepts are utilized to generate the whole environment including the overhead or top view.

Note that, in the following sections, we assume that we are observing a world plane given in the form of an input image where one pixel in the input image is equal to one unit in space; and the synthesized perspective image is represented by the image coordinate system

4.2 Computing the Planar Perspective Matrix: A Special Case

As mentioned before in Equation (2.5), the relationship between a point $\dot{\mathbf{M}}=[X,Y,Z]^T$ in space and its projection, $\dot{\mathbf{m}}=[x,y]^T$, onto the image plane is represented by [EL01a]:

$$\begin{bmatrix} xs \\ ys \\ s \end{bmatrix} = \mathsf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.1}$$

As depicted in Figure 4.1 on Page 55, we suppose that the image of the horizontal plane, or the overhead view, is provided. Then, the world coordinate system can be placed conveniently such that the $Z-$coordinate of any point on that plane is always zero. In other words, we assume that the horizontal plane resides on the $XY-$plane. Moreover, we place the world origin at the lower left corner of the given overhead view. As a consequence, and substituting

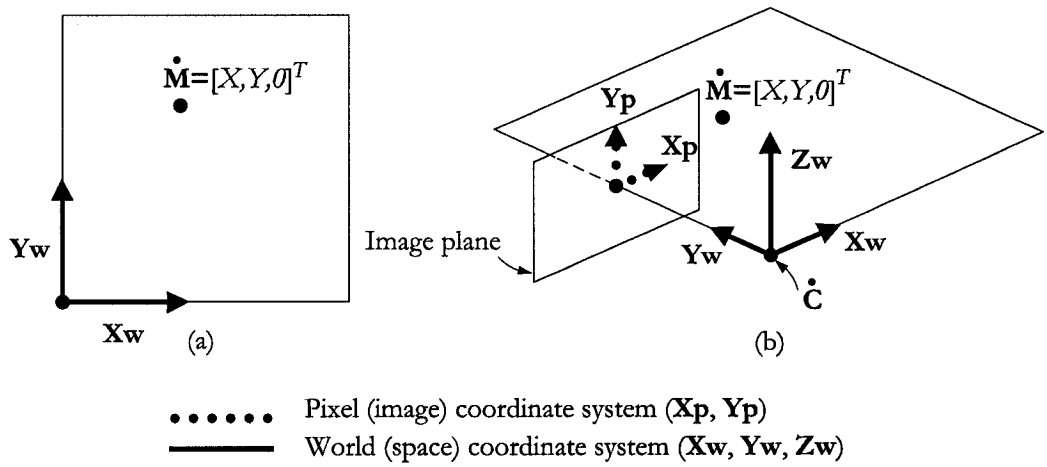


Figure 4.1: (a) The plane under consideration: The origin of world coordinate system origin is placed at the lower left corner of the overhead view. M is an arbitrary point on the plane; (b) 3D representation of (a). Note that $Z-$coordinate is always 0. The image plane showed is formed using rotation angles of 90°, 0° and 0° for ω, ϕ and κ (or 90°, 0° and 0° for the tilt, τ, pan, ρ, and swing ψ respectively).

the value of Z in Equation (4.1), we get:

$$\begin{bmatrix} xs \\ ys \\ s \end{bmatrix} = \mathtt{P} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{4.2}$$

No matter what the values in the third column of matrix P are, their products by the third term of the vector M will always be 0. As a result, the matrix P can be reduced to a 3 ×3 matrix while M is reduced to a 3D vector. Rewriting Equation (4.2), we get:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathtt{H_z} \begin{bmatrix} \frac{X}{s} \\ \frac{Y}{s} \\ \frac{1}{s} \end{bmatrix} \tag{4.3}$$

where

- x and y are the image coordinates;
- $\mathtt{H_z}$ is a 3×3 homography matrix called the *planar perspective matrix*;
- X and Y are the input world coordinates; and
- s is a scaling factor.

Finally, we can rewrite Equation (4.3) as:

$$\mathbf{m} = \mathtt{H_z}\mathbf{m}' \tag{4.4}$$

where

- $\mathbf{m}$ is a point in the synthesized image;
- $\mathbf{m}'$ is a point in the input image representing the overhead view.

Or, equivalently:

$$\mathbf{m}' = \mathtt{H_z}^{-1}\mathbf{m} \tag{4.5}$$

Suppose that the optical center of the camera is placed slightly above the observed plane where the translation of the camera with respect to the world origin is $[0, 0, t_z]^T$ and its angles of rotation are all equal to zeros. Moreover, assume that the focal length of the camera is equal to 1. In this case, the matrix $\mathtt{H_z}$ can be expressed as:

$$\mathtt{H_z} = \mathtt{H_{zc}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -t_z \end{bmatrix} \tag{4.6}$$

Note that the translation of the camera is reflected in the third column of the above matrix (as it will be explained in the next section). This means that the value of t_z must *not* be equal to 0. Otherwise, we will be trying to observe a plane passing through the optical center of the camera, which is invalid mathematically.

This matrix is used as a homography matrix that relates the top view and its perspective projection in 3D space. In the next section, we will derive the general form of the planar perspective matrix $\mathtt{H_z}$. A complete detailed version of this matrix can be found in Section B.3.

4.3 Computing the Planar Perspective Matrix: The General Case

Unfortunately, most of the time, we are not so lucky to have the matrix $\mathtt{H_z}$ as simple as mentioned above. In the general case, the camera is located arbitrarily in space and has a focal length which is different from the unit value. Consequently, the matrix $\mathtt{H_z}$ has to be adjusted to satisfy the general requirements.

In order to achieve this adjustment, the matrix $\mathtt{P_c}$ of Equation (2.7) has to be multiplied to the left by the camera calibration matrix, $\mathtt{A}$, to adjust the coordinates on the image plane. Furthermore, in the case where the camera is placed and oriented freely in space, the matrix $\mathtt{P_c}$ of Equation (2.7) is multiplied to the right by the rotation translation matrix, $\mathtt{D}$, to change the coordinates in space. In other words, we utilize Equation (2.8). That is,

$$\mathtt{P} = \mathtt{AP_cD} \tag{4.7}$$

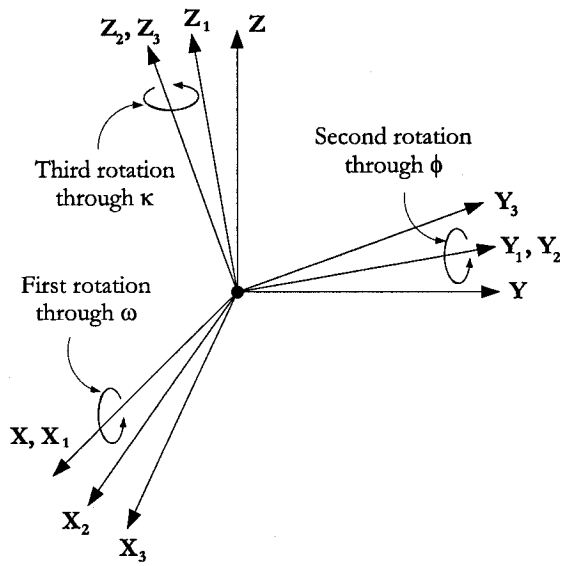


Figure 4.2: The overall rotation is performed as three sequential rotations through ω, ϕ and κ.

Then, by eliminating the third column, the planar perspective matrix, $\mathtt{H_Z}$ can be expressed as:

$$\mathtt{H_Z} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \tag{4.8}$$

Involved in these terms are the rotation and translation of the camera. In the following, we will show how to compute the rotation matrix through different rotation systems and the final translation vector as well. Those components constitute the elements of the matrix D of Equation (4.7) [EL03].

4.3.1 Different Rotations

As depicted in Figure 4.2 on Page 57, the overall rotation from XYZ system to $X_3Y_3Z_3$ system is performed in three steps:

1. Rotation through an angle ω about the $X-$axis to convert coordinates from the XYZ system to the $X_1Y_1Z_1$ system. This is through a rotation matrix, $\mathtt{R}_\omega$.

2. Rotation through an angle ϕ about the Y_1-axis to convert coordinates from the $X_1Y_1Z_1$ system to the $X_2Y_2Z_2$ system. This is through a rotation matrix, $\mathtt{R}_\phi$.

3. Rotation through an angle κ about the Z_2-axis to convert coordinates from the $X_2Y_2Z_2$ system to the $X_3Y_3Z_3$ system. This is through a rotation matrix, $\mathtt{R}_\kappa$.

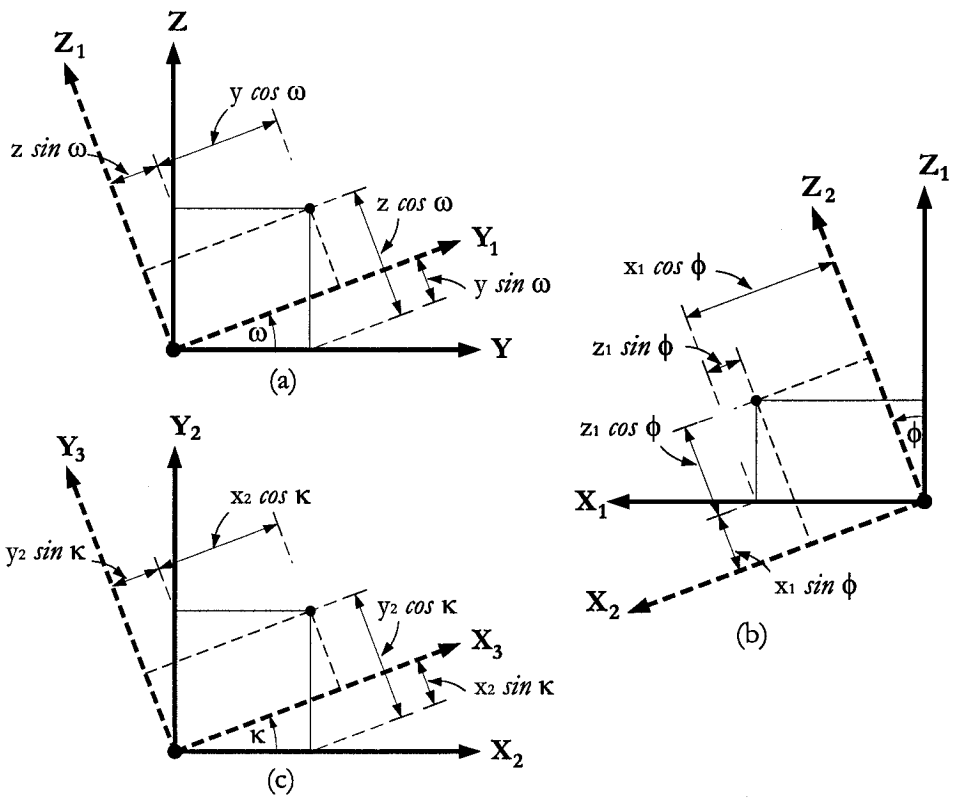


Figure 4.3: (a) Rotation through ω about the $X-$axis. (b) Rotation through ϕ about the Y_1-axis. (c) Rotation through κ about the Z_2-axis.

The details of this sequence of rotations are as follows:

- Rotation through the angle ω about the $X-$axis is shown in Figure 4.3(a) on Page 58. The new coordinates are given by:

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathrm{R}_\omega \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\ \omega & sin\ \omega \\ 0 & -sin\ \omega & cos\ \omega \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.9}$$

- Rotation through the angle ϕ about the Y_1-axis is shown in Figure 4.3(b). The new coordinates are given by:

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathrm{R}_\phi \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} cos\ \phi & 0 & -sin\ \phi \\ 0 & 1 & 0 \\ sin\ \phi & 0 & cos\ \phi \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \tag{4.10}$$

- Rotation through the angle κ about the Z_2-axis is shown in Figure 4.3(c). The new

coordinates are given by:

$$\begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \end{bmatrix} = \mathtt{R}_\kappa \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} cos\ \kappa & sin\ \kappa & 0 \\ -sin\ \kappa & cos\ \kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{4.11}$$

The final result of the rotations is calculated by concatenating the three rotation matrices. That is:

$$\begin{aligned} \mathtt{R} &= \mathtt{R}_\kappa\ \mathtt{R}_\phi\ \mathtt{R}_\omega \\ &= \begin{bmatrix} cos\ \phi\ cos\ \kappa & sin\ \omega\ sin\ \phi\ cos\ \kappa & -cos\ \omega\ sin\ \phi\ cos\ \kappa \\ & +cos\ \omega\ sin\ \kappa & +sin\ \omega\ sin\ \kappa \\ -cos\ \phi\ sin\ \kappa & -sin\ \omega\ sin\ \phi\ sin\ \kappa & cos\ \omega\ sin\ \phi\ sin\ \kappa \\ & +cos\ \omega\ cos\ \kappa & +sin\ \omega\ cos\ \kappa \\ sin\ \phi & -sin\ \omega\ cos\ \phi & cos\ \omega\ cos\ \phi \end{bmatrix} \end{aligned} \tag{4.12}$$

Another way to calculate the rotation matrix – in case of a camera observing a work surface – is by using pan, ρ, tilt, τ, and swing, ψ, angles instead of ω, ϕ and κ. Figure 4.4 on Page 60 shows these angles [WD00]. The optical axis – along with the vertical line passing through the optical center and perpendicular to the ground – form a plane called the *principal plane.* The line of intersection between the principal plane and the image plane is called *the principal line.* The angle – in the principal plane – between the optical axis and the vertical line is called the tilt, τ. The angle – in the ground plane – between the direction of the Y_w-axis and intersection between the principal plane and the ground plane is called the pan, ρ. The angle – in the image plane – between the direction of Y_3-axis and the principal line is called the swing, ψ.

As in the previous case, the overall rotation from XYZ system to $X_3Y_3Z_3$ system (depicted in Figure 4.5 on Page 61) is performed in three separate two-dimensional rotations:

1. Rotation through a pan angle, ρ, about the $Z-$axis to convert coordinates from the XYZ system to the $X_1Y_1Z_1$ system. This is through a rotation matrix, $\mathtt{R}_\rho$.

2. Rotation through a tilt angle, τ, about the X_1-axis to convert coordinates from the $X_1Y_1Z_1$ system to the $X_2Y_2Z_2$ system. This is through a rotation matrix, $\mathtt{R}_\tau$.

3. Rotation through a swing angle, ψ, about the Z_2-axis to convert coordinates from the $X_2Y_2Z_2$ system to the $X_3Y_3Z_3$ system. This is through a rotation matrix, $\mathtt{R}_\psi$.

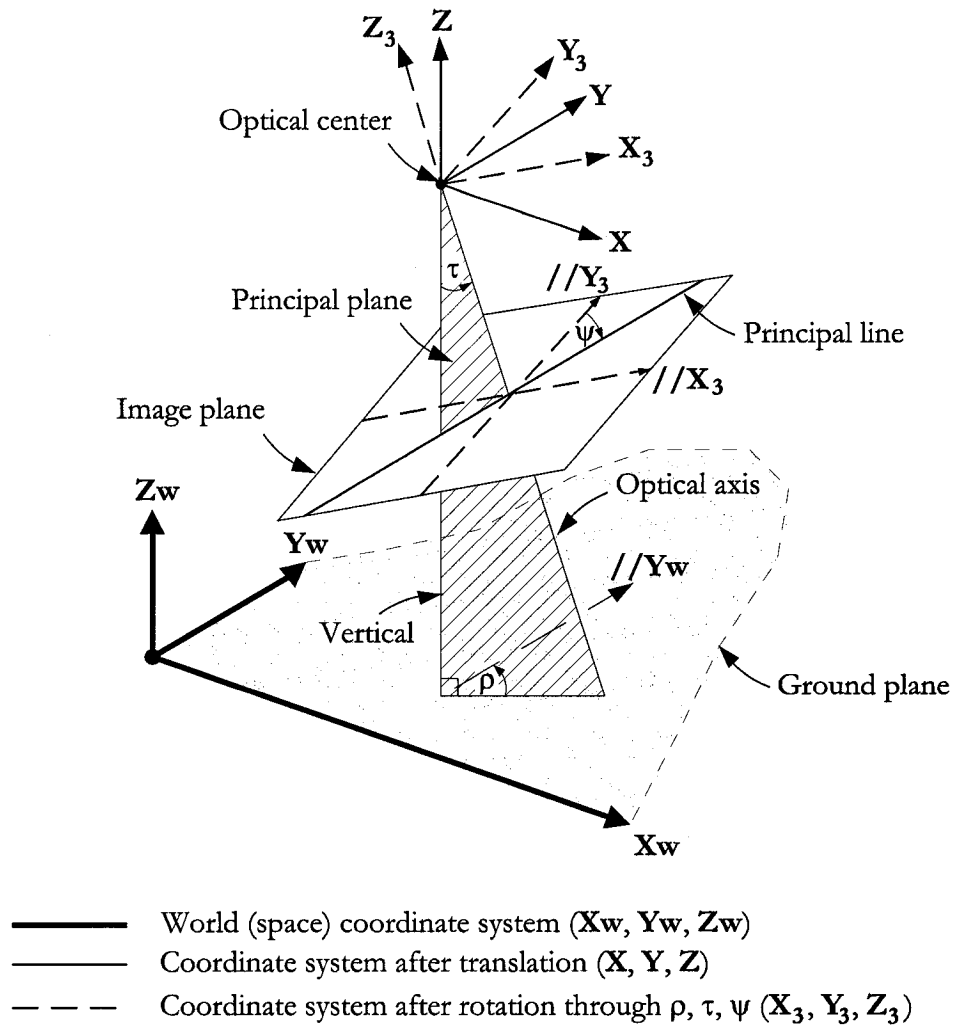


Figure 4.4: Pan, ρ, tilt, τ, and swing, ψ, rotation angles. In [WD00], the pan angle is called *azimuth*.

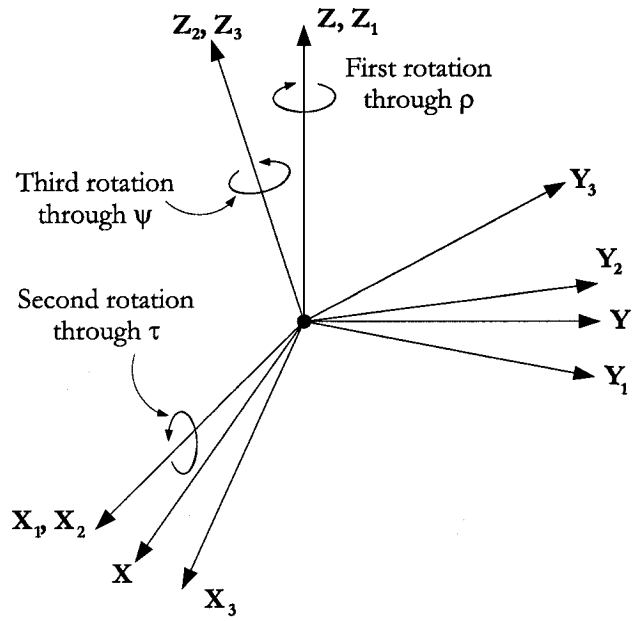


Figure 4.5: The overall rotation is performed as three sequential rotations through ρ, τ and ψ.

The final result of the rotations is calculated by concatenating the three rotation matrices. That is:

$$\begin{aligned} \mathrm{R} &= \mathrm{R}_{\psi}\ \mathrm{R}_{\tau}\ \mathrm{R}_{\rho} \\ &= \begin{bmatrix} cos\ \psi\ cos\ \rho & -cos\ \psi\ sin\ \rho & sin\ \psi\ sin\ \tau \\ +sin\ \psi\ cos\ \tau\ sin\ \rho & +sin\ \psi\ cos\ \tau\ cos\ \rho & \\ -sin\ \psi\ cos\ \rho & sin\ \psi\ sin\ \rho & cos\ \psi\ sin\ \tau \\ +cos\ \psi\ cos\ \tau\ sin\ \rho & +cos\ \psi\ cos\ \tau\ cos\ \rho & \\ -sin\ \tau\ sin\ \rho & -sin\ \tau\ cos\ \rho & cos\ \tau \end{bmatrix} \end{aligned} \tag{4.13}$$

Note that, both Equations (4.12) and (4.13) should result in the same rotation matrix. Thus, using those equations, we can calculate the values of the tilt, τ, the pan, ρ and the swing, ψ angles given ω, ϕ and κ [WD00].

$$\tau = cos^{-1}(r_{33}) = cos^{-1}(cos\ \omega\ cos\ \phi) \tag{4.14}$$

$$\rho = tan^{-1}\left(\frac{-r_{31}}{-r_{32}}\right) = tan^{-1}\left(\frac{-sin\ \phi}{sin\ \omega\ cos\ \phi}\right) \tag{4.15}$$

$$\psi = tan^{-1}\left(\frac{r_{13}}{r_{23}}\right) = tan^{-1}\left(\frac{-cos\ \omega\ sin\ \phi\ cos\ \kappa + \ sin\ \omega\ sin\ \kappa}{cos\ \omega\ sin\ \phi\ sin\ \kappa + \ sin\ \omega\ cos\ \kappa}\right) \tag{4.16}$$

The same idea can be used to calculate ω, ϕ and κ given τ, ρ and ψ.

$$\omega = tan^{-1}\left(\frac{-r_{32}}{r_{33}}\right) = tan^{-1}\left(\frac{sin\ \tau\ cos\ \rho}{cos\ \tau}\right) \tag{4.17}$$

$$\phi = sin^{-1}(r_{31}) = sin^{-1}(-sin\ \tau\ sin\ \rho) \tag{4.18}$$

$$\kappa = tan^{-1}\left(\frac{-r_{21}}{r_{11}}\right) = tan^{-1}\left(\frac{sin\ \psi\ cos\ \rho - cos\ \psi\ cos\ \tau\ sin\ \rho}{cos\ \psi\ cos\ \rho + sin\ \psi\ cos\ \tau\ sin\ \rho}\right) \tag{4.19}$$

4.3.2 New Camera Location

The new translation vector $\mathbf{T}$ should be calculated with respect to the original position of the camera. This includes re-rotating the camera through ω, ϕ and κ (or ρ, τ and ψ) before applying translation. Thus, the translation vector $\mathbf{T}$ is given by:

$$\mathbf{T} = -\mathtt{R}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.20}$$

The details of each term of this vector are listed in Section B.2. Finally, the planar perspective matrix can be estimated. The details of each term of this matrix are listed in Section B.3.

Once the planar perspective matrix, $\mathtt{H_z}$, has been computed, it can be used to:

1. display the perspective image of the ground plane; and
2. determine feature points that belong to the ground and those that belong to the obstacles. This result will be used in the matching procedure presented in Chapter 5.

In order to display the perspective image of the ground plane, the intersection between that plane and the plane at infinity; i.e., the line at infinity, must be detected. Moreover, interpolation is to be used to calculate the intensity of pixels of the result image. The next two sections explain these steps.

4.4 Equation of the Line at Infinity

The line at infinity is the vanishing line of the plane represented by $\mathtt{H_z}$. Only the points below this line must be considered when generating a perspective view of the plane. According to Equations (2.61) through (2.64), the last row of $\mathtt{H_z}$ represents the equation of the line at infinity $\mathbf{l}_\infty = [l_1, l_2, l_3]^T$ where $l_i = h_{3i}, i \in \{1, 2, 3\}$. In other words, the equation of the vanishing line given the homography matrix $\mathtt{H_z}$ is:

$$h_{31}x + h_{32}y + h_{33} = 0 \tag{4.21}$$

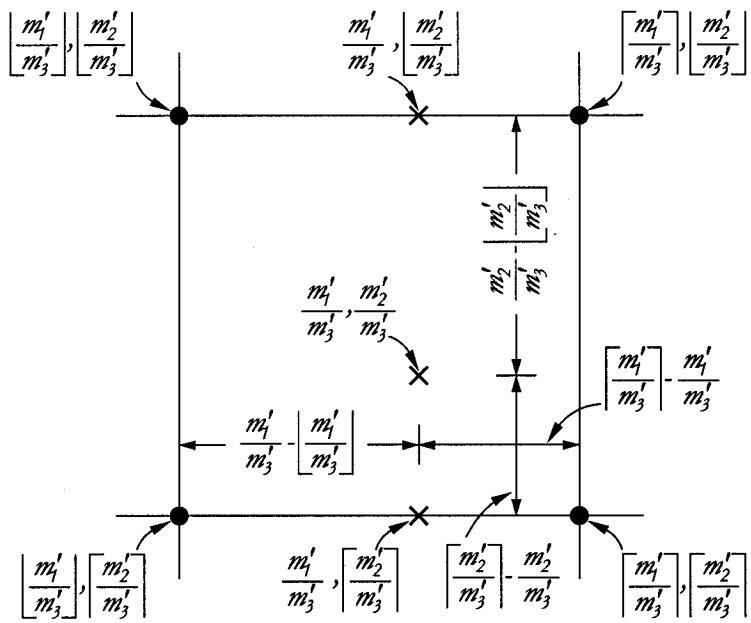


Figure 4.6: Interpolation: 1. interpolate along rows 2. interpolate along columns.

4.5 Determining Intensity by Interpolation

The intensity of any point $\dot{\mathbf{m}} = [x, y]^T$ in the final synthesized image can be calculated by first determining the location in the input image as:

$$\mathbf{m}' = \begin{bmatrix} m'_1 \\ m'_2 \\ m'_3 \end{bmatrix} = \mathtt{H_z}^{-1}\mathbf{m} \tag{4.22}$$

and then by interpolation, the intensity of $\dot{\mathbf{m}}$ is calculated as $\mathcal{I}\left(\frac{m'_1}{m'_3}, \frac{m'_2}{m'_3}\right)$. This can be done using bilinear interpolation that consists of successively interpolating intensity values along the two directions (see Figure 4.6 on Page 63) [Jai89]:

1. Interpolate along rows:

$$\begin{aligned} \mathcal{I}\left(\tfrac{m'_1}{m'_3}, \lfloor\tfrac{m'_2}{m'_3}\rfloor\right) &= \mathcal{I}\left(\lfloor\tfrac{m'_1}{m'_3}\rfloor, \lfloor\tfrac{m'_2}{m'_3}\rfloor\right)\left(\lceil\tfrac{m'_1}{m'_3}\rceil - \tfrac{m'_1}{m'_3}\right) \\ &+\mathcal{I}\left(\lceil\tfrac{m'_1}{m'_3}\rceil, \lfloor\tfrac{m'_2}{m'_3}\rfloor\right)\left(\tfrac{m'_1}{m'_3} - \lfloor\tfrac{m'_1}{m'_3}\rfloor\right) \end{aligned} \tag{4.23}$$

and

$$\begin{aligned} \mathcal{I}\left(\tfrac{m'_1}{m'_3}, \lceil\tfrac{m'_2}{m'_3}\rceil\right) &= \mathcal{I}\left(\lfloor\tfrac{m'_1}{m'_3}\rfloor, \lceil\tfrac{m'_2}{m'_3}\rceil\right)\left(\lceil\tfrac{m'_1}{m'_3}\rceil - \tfrac{m'_1}{m'_3}\right) \\ &+\mathcal{I}\left(\lceil\tfrac{m'_1}{m'_3}\rceil, \lceil\tfrac{m'_2}{m'_3}\rceil\right)\left(\tfrac{m'_1}{m'_3} - \lfloor\tfrac{m'_1}{m'_3}\rfloor\right) \end{aligned} \tag{4.24}$$

2. Interpolate along columns:

$$\begin{aligned}\mathcal{I}\left(\frac{m'_1}{m'_3}, \frac{m'_2}{m'_3}\right) &= \mathcal{I}\left(\frac{m'_1}{m'_3}, \lfloor\frac{m'_2}{m'_3}\rfloor\right)\left(\lceil\frac{m'_2}{m'_3}\rceil - \frac{m'_2}{m'_3}\right) \\ &+\mathcal{I}\left(\frac{m'_1}{m'_3}, \lceil\frac{m'_2}{m'_3}\rceil\right)\left(\frac{m'_2}{m'_3} - \lfloor\frac{m'_2}{m'_3}\rfloor\right)\end{aligned} \tag{4.25}$$

4.6 Experimental Results

Consider the image shown in Figure 4.7(a) on Page 65. If a camera is placed at the white dot and oriented as indicated by the arrow. The parameters of the camera are the following. The location of the camera is $[250, 250, 100]^T$ in pixels where the origin is at the lower left corner and the size of the image is 756 $\times$ 512 pixels. The orientation of the camera is set through the angles 70°, 90° and 0° for the tilt, τ, the pan, ρ and the swing, ψ (0°, $-70°$ and $-90°$ for ω, ϕ and κ respectively). The focal length along both x and y directions is 200 pixels. It is requested to generate the perspective view of that image as seen by a camera given the above parameters.

Figure 4.7(b) on Page 65 shows the perspective image generated that should be seen by that camera using the planar perspective matrix. Applying the planar perspective matrix, we get:

$$\mathtt{H_z} = \begin{bmatrix} 0.00 & -212.83 & -16837 \\ 106.42 & -249.02 & -3839.5 \\ 0.00 & -1.0 & 27.67 \end{bmatrix} \tag{4.26}$$

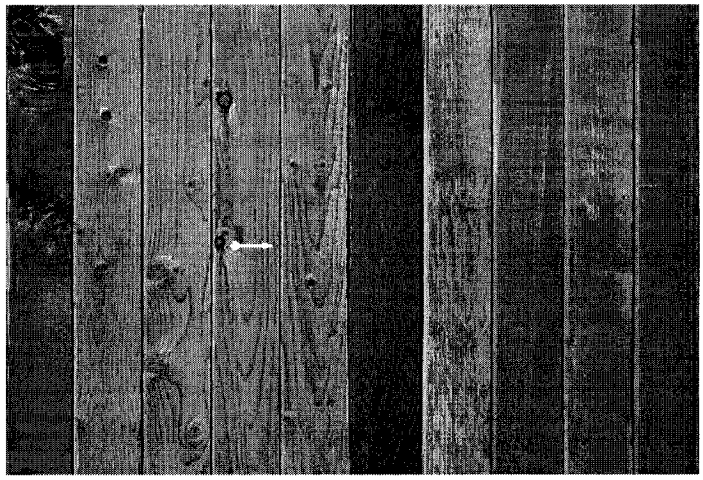
Note that, according to Section 4.4, the equation of the line at infinity is:

$$y = 27.67 \tag{4.27}$$

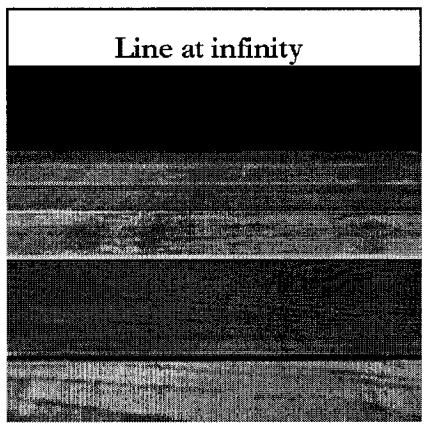
Only points below this line should be considered (according to the sign of the last equation).

4.7 Obstacle Versus Ground Feature Detection

Many studies used planes in a variety of tasks. For example, in [VL01], Vincent and Laganière used RANSAC (Random Sample Consensus) [FB81, RW00] scheme to detect planar homographies in an image pair. The goal of [CRZ99] was to make measurements of world planes from their images and determine how inaccurate those measurement were. In [LO98], under the assumption of planar ground, Lourakis *et al.* presented an algorithm to detect obstacles. They used a set of corners matched between two views using cross correlation. Hence, the homography of the ground can be estimated and warping of the first image is performed with respect to the second. Then, subtraction is performed between the first and the warped image to detect obstacles.



(a)



(b)

Figure 4.7: (a) The top view. The position of the camera is indicated by the white circle. On a 756 × 512 image, the position is $[250, 250, 100]^T$ in pixels where the origin is at the lower left corner. The tilt angle, τ is 70°. The pan, ρ is 90°. The swing, ψ is 0°. (0°, −70° and −90° for ω, ϕ and κ respectively.) The focal length along both x and y directions is 200. (b) The perspective view generated using the planar perspective matrix.

In our case, we will exploit the assumption that the ground surface is planar to identify feature points (detected using JUDOCA operator discussed in Chapter 3) that belong to this plane to facilitate the matching problem discussed in Chapter 5. In order to reach that goal, we propose to use the planar perspective matrix discussed above to filter the obstacle feature points from the scene. In other words, we consider excluding features detected on the ground plane of the scene under consideration or we want to distinguish between two junctions sets; one that belongs to the ground and the other that belongs to obstacles. The two sets are complement to each other and their union represents the universal set; i.e., junctions detected by JUDOCA operator. We can summarize the steps of our algorithm to detect the features that belong to the ground plane as follows:

1. Roughly determine the homography of the ground plane between the two views using the planar perspective matrix or the equation of the plane.
2. Detect junctions in both images using JUDOCA operator (Chapter 3).
3. Apply the homography matrix achieved to every junction in the left image so that another corresponding point is obtained in the right image.
4. Use the error margins known for camera parameters to establish a window surrounding that point.
5. Use normalized correlation technique and compute the height of the reconstructed point in space to select the best candidate match included in this window.
6. Repeat the last three steps to the right-left direction in another phase.
7. Approve matches that are agreed upon in both phases. Those matches are likely to belong to the ground plane and should be excluded from the matching process discussed in Chapter 5.
8. Use the matches and RANSAC scheme to obtain a more accurate homography matrix.
9. Use the updated version of the homography matrix to exclude the rest of the ground features; i.e., those that are not part of matching pairs.

The following sections describe this procedure in more detail.

4.8 Obtaining the Inter-Image Planar Homography

4.8.1 Through the Planar Perspective Matrix

In case of a pair of images for which we have the camera parameters, the homography of the ground plane between the two views can be calculated through the planar perspective matrix discussed above. Suppose that a point $\dot{\mathbf{M}} = [X, Y, 0]^T$ is projected as points $\dot{\mathbf{m}} = [x, y]^T$ and

$\dot{\mathbf{m}}' = [x', y']^T$ on both images. The relation between $\dot{\mathbf{M}}$ and each of $\dot{\mathbf{m}}$ and $\dot{\mathbf{m}}'$ can be written as:

$$\mathbf{m} = \mathtt{H}_1\tilde{\mathbf{M}} \tag{4.28}$$

and

$$\mathbf{m}' = \mathtt{H}_2\tilde{\mathbf{M}} \tag{4.29}$$

where

- $\mathtt{H}_1$ and $\mathtt{H}_2$ are the planar perspective matrices associated with both views; and
- $\tilde{\mathbf{M}} = [X, Y, 1]^T$

We can rewrite Equation (4.28) as:

$$\tilde{\mathbf{M}} = \mathtt{H}_1^{-1}\mathbf{m} \tag{4.30}$$

Substituting the value of $\tilde{\mathbf{M}}$ in Equation (4.29), we get:

$$\mathbf{m}' = \mathtt{H}_2\mathtt{H}_1^{-1}\mathbf{m} \tag{4.31}$$

If

$$\mathtt{H}_g = \mathtt{H}_2\mathtt{H}_1^{-1} \tag{4.32}$$

then,

$$\mathbf{m}' = \mathtt{H}_g\mathbf{m} \tag{4.33}$$

where $\mathtt{H}_g$ is a 3×3 matrix describing the homography between the two plane images.

4.8.2 Through Ground Plane Equation

Another way to calculate the homography matrix, $\mathtt{H}_g$, is by using the ground plane equation. Suppose that the ground plane resides on the $XY-$plane of the world coordinate system. Then, the equation of this plane, Π_g, can be expressed as:

$$Z = 0 \tag{4.34}$$

or

$$\Pi_g = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.35}$$

If we placed the origin of the world coordinate system at the optical center of the first camera, then the equation of the ground plane can be re-expressed as:

$$\Pi_{new}^T = [\mathbf{N}_{new}^T \mid d_{new}]^T = [0\ 0\ 1\ 0] \begin{bmatrix} \mathtt{R}'^{-1} & -\mathtt{R}'^{-1}\mathbf{T}' \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{4.36}$$

where

- Π_{new} is the new equation of the plane;
- $\mathbf{N}_{new}$ is the normal to the plane;
- d_{new} is the distance to the plane from the origin;
- $\mathtt{R}'$ is the rotation of the second camera;
- $\mathbf{T}'$ is the translation of the second camera; and
- $\begin{bmatrix} \mathtt{R}'^{-1} & -\mathtt{R}'^{-1}\mathbf{T}' \\ \mathbf{0}_3^T & 1 \end{bmatrix}$ is the 4×4 matrix used to transform to the new coordinate system.

Then, in order to calculate the new rotation matrix and translation vector, the projection matrix, $\mathtt{P}'$, must be expressed in accordance to the new coordinate system. This is done by multiplying $\mathtt{P}'$ to the right by the 4×4 transformation matrix mentioned above. That is to transform $\mathtt{P}'$ to a new projection matrix according to the new coordinate system. The new projection matrix $\mathtt{P}'_{new} = \mathtt{A}'[\mathtt{R}'_{new}|\mathbf{T}'_{new}]$. Thus, in order to get the new rotation matrix and translation vector, $\mathtt{P}'_{new}$ should be multiplied to the left by $\mathtt{A}'^{-1}$. Thus:

$$[\mathtt{R}'_{new}|\mathbf{T}'_{new}] = \mathtt{A}'^{-1}\mathtt{P}' \begin{bmatrix} \mathtt{R}'^{-1} & -\mathtt{R}'^{-1}\mathbf{T}' \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{4.37}$$

where

- $\mathtt{R}'_{new}$ is the new rotation matrix of the second camera;
- $\mathbf{T}'_{new}$ is the new translation vector of the second camera;
- $\mathtt{A}'$ is the calibration matrix of the second camera; and
- $\mathtt{P}'$ is the projection matrix of the second camera as listed in Section B.4.

Finally, the homography is calculated as done in Equation (2.68):

$$\mathtt{H}_g = \mathtt{A}' \left[\mathtt{R}'_{new} - \frac{\mathbf{T}'_{new}\mathbf{N}_{new}^T}{d_{new}} \right] \mathtt{A}^{-1} \tag{4.38}$$

Both Equations (4.32) and (4.38) must result in the same homography matrix. Once this matrix is obtained, it becomes possible to reject feature points that are putatively on the ground which will leave those on obstacles. In the next section, we will deploy this matrix in order to facilitate the matching process in Chapter 5.

4.9 Ground Feature Detection

In order to identify the features on the ground, we start by applying the homography matrix, $\mathtt{H}_g$, to a given candidate, $\dot{\mathbf{m}} = [x, y]^T$, in the left image. This results in another point, $\dot{\mathbf{m}}' = [x', y']^T$, in the right image.

$$\mathbf{m}' = \mathtt{H}_g\mathbf{m} \tag{4.39}$$

Then, the candidate set is all junctions that are included in the region $[x'-\varepsilon_1, y'-\varepsilon_2]^T[x'+\varepsilon_3, y'+\varepsilon_4]^T$ where ε_1, ε_2, ε_3 and ε_4 are determined according to the error margins of the camera parameters; i.e., $\pm\Delta X$, $\pm\Delta Y$, $\pm\Delta Z$, $\pm\Delta\omega$, $\pm\Delta\phi$ and $\pm\Delta\kappa$. As depicted in Figure 4.8 on Page 71, each parameter error margin contributes to a window by itself. The final window corners are determined according to the minimum and the maximum coordinates among all windows. Thus, a candidate junction can be expressed as $\dot{\mathbf{m}}' + \mathbf{t}_{lr}$ where $\mathbf{t}_{lr} = [a, b]^T$, $a \in [x'-\varepsilon_1, x'+\varepsilon_3]$ and $b \in [y'-\varepsilon_2, y'+\varepsilon_4]$. A window of a predefined size surrounding each junction in the left image is established. The homography matrix calculated above and the the translation vector, $\mathbf{t}_{lr}$, are used to determine an area in the other image that corresponds to the previous window. Normalized correlation technique is applied between those areas. According to some threshold, junctions with low correlation values are rejected. This is to check:

$$\frac{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_l(x+m,y+n)-\overline{\mathcal{I}_l(x,y)}][\mathcal{I}_r(\dot{\mathbf{k}}^T+\mathbf{t}_{lr}^T)-\overline{\mathcal{I}_r(\dot{\mathbf{m}}'^T+\mathbf{t}_{lr}^T)}]}{\sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_l(x+m,y+n)-\overline{\mathcal{I}_l(x,y)}]^2}\sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_r(\dot{\mathbf{k}}^T+\mathbf{t}_{lr}^T)-\overline{\mathcal{I}_r(\dot{\mathbf{m}}'^T+\mathbf{t}_{lr}^T)}]^2}} \geq t_g \tag{4.40}$$

where

- $(N+1)^2$ is the area correlated in the left image;
- $\dot{\mathbf{m}} = [x, y]^T$ is the left point;
- $\mathbf{m}' = \mathtt{H}_g\mathbf{m} = \mathtt{H}_g[x, y, 1]^T$;
- $\dot{\mathbf{m}}' + \mathbf{t}_{lr}$ is the right point;
- $\overline{\mathcal{I}_l(x, y)}$ is the mean of the neighborhood surrounding $[x, y]^T$;
- $\mathbf{k} = \mathtt{H}_g[x+m, y+n, 1]^T$;
- $\overline{\mathcal{I}_r(\dot{\mathbf{m}}'^T + \mathbf{t}_{lr}^T)}$ is the mean of the neighborhood surrounding $\dot{\mathbf{m}}' + \mathbf{t}_{lr}$; and
- t_g is a threshold.

If (4.40) results in a true condition, then there is a good chance that $[x, y]^T$ and $\dot{\mathbf{m}}' + \mathbf{t}_{lr}$ are a match pair. The steps applied above to the left-right direction is repeated again for the opposite direction. Pairs that survived the above operations are searched for possible matches according to the height of the reconstructed 3D point in space. This is by calculating:

$$\hat{\dot{\mathbf{M}}} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} [\mathtt{I} - \dot{\mathbf{M}}_\infty\dot{\mathbf{M}}_\infty^T] + [\mathtt{I} - \dot{\mathbf{M}}'_\infty\dot{\mathbf{M}}'^T_\infty] \end{pmatrix}^{-1} \begin{pmatrix} \dot{\mathbf{C}} + \dot{\mathbf{C}}' - [\dot{\mathbf{C}}^T\dot{\mathbf{M}}_\infty]\dot{\mathbf{M}}_\infty - [\dot{\mathbf{C}}'^T\dot{\mathbf{M}}'_\infty]\dot{\mathbf{M}}'_\infty \end{pmatrix} \tag{4.41}$$

where

- $\dot{\mathbf{M}}$, $\dot{\mathbf{C}}$, $\dot{\mathbf{C}}'$, $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are the inhomogeneous representation of the point in space, the optical centers and the intersection of the rays with the plane at infinity respectively;
- $\mathtt{I}$ is a 3×3 identity matrix; and
- $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are normalized to unit magnitude.

Now, we check if:

$$-t_h \leq Z \leq t_h \tag{4.42}$$

where

- Z is the third term in $\dot{\mathbf{M}}$; and
- t_h is a threshold that represents a narrow range around zero.

If (4.42) results in a true condition, then the pair is a possible match. Those matches are likely to belong to the ground and should be excluded from further matching steps (discussed in Chapter 5). Notice that applying correlation using the homography matrix without triangulating the pair; i.e, without checking the height of the reconstructed point in space, might be enough procedure in case of real matches (as the height in this case will be around zero). However, recall that at this point, the homography matrix is not accurate and we are searching for a candidate among a few within some area. Correlation may lead to mismatch in the occluded areas. In this case, triangulation will probably lead to a height that exceeds $\pm t_h$.

At this point, we may end up with many feature points that do belong to the ground plane but, at the same time, are not part of matching pairs. In order to exclude those feature points, we use RANSAC scheme [FB81, RW00, VL01] with the matching pairs that belong to the ground plane. This step results in an updated version of the ground plane homography matrix, $\mathtt{H}_g$. The idea can be summarized as follows:

1. A homography matrix is computed using the relation:

$$\mathbf{m}' \times \mathtt{H}_g\mathbf{m} = \mathbf{0} \tag{4.43}$$

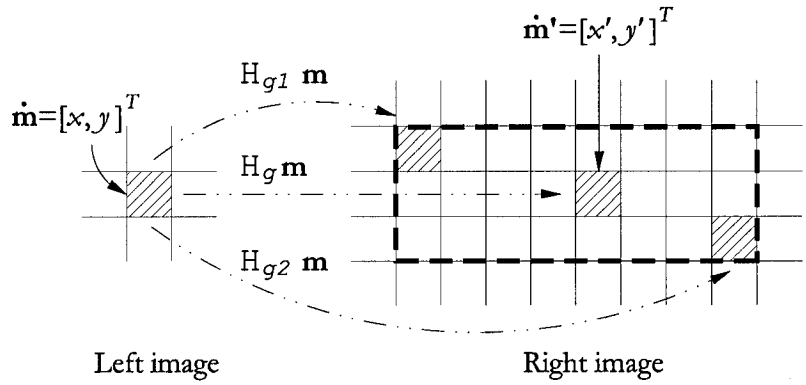
   Four correspondences which contribute to 8 linear equations are sufficient to determine $\mathtt{H}$. Thus, randomly select four putative correspondences from the above list.

2. Check if the other pairs in the list agree with the matrix computed above. That is to check:

$$d(\mathtt{H}_g\mathbf{m}, \mathbf{m}') < t_{\mathtt{H}} \tag{4.44}$$

   where

   - $d()$ represents the distance between the points $\mathtt{H}\mathbf{m}$ and $\mathbf{m}'$; and
   - $t_{\mathtt{H}}$ is a threshold,



Window created by an error margin of one parameter.

H_g The homography matrix of the ground where $\Delta\omega = 0$

$H_{g1,2}$ The homography matrix of the ground where $\Delta\omega = \pm d$ ($d \neq 0$)

Figure 4.8: A window is created by an error margin of one parameter. Two values of the error margin result in two different points. The window enclosing the minimum rectangular area that contains the two points is considered for this parameter. ω angle is used as an example.

If (4.44) results in a true condition, then the pair ($\dot{\mathbf{m}}$, $\dot{\mathbf{m}}'$) agrees with the matrix H_g.

3. Repeat the last two steps until a sufficient number of pairs are consistent with the computed homography [VL01].

Normalized correlation represented in (4.40) is re-applied again with the updated version of the homography matrix, $\mathbf{H}_g$, where $\mathbf{t} = [0, 0]^T$, to the other left and right junctions that do not represent ground matching pairs. For a given junction, if this test, (4.40), results in a true condition, then this junction belongs to the ground plane.

4.10 Experimental Results

The above technique is applied to a pair of images shown in Figure 4.9 on Page 72 where the locations of junctions detected by JUDOCA are displayed as dotted white squares. The error margins are ±10 mm, ±10 mm, ±10 mm, ±0.8°, ±0.8° and ±0.8° for ΔX, ΔY, ΔZ, $\Delta\omega$, $\Delta\phi$ and $\Delta\kappa$ respectively.

Figures 4.9(c) and (d) on Page 72 show two windows of size 11 × 11 pixels surrounding corresponding junctions that belong to the ground plane. Those junctions are enclosed in the circles shown in Figures 4.9(a) and (b). By using the technique explained above; i.e., through

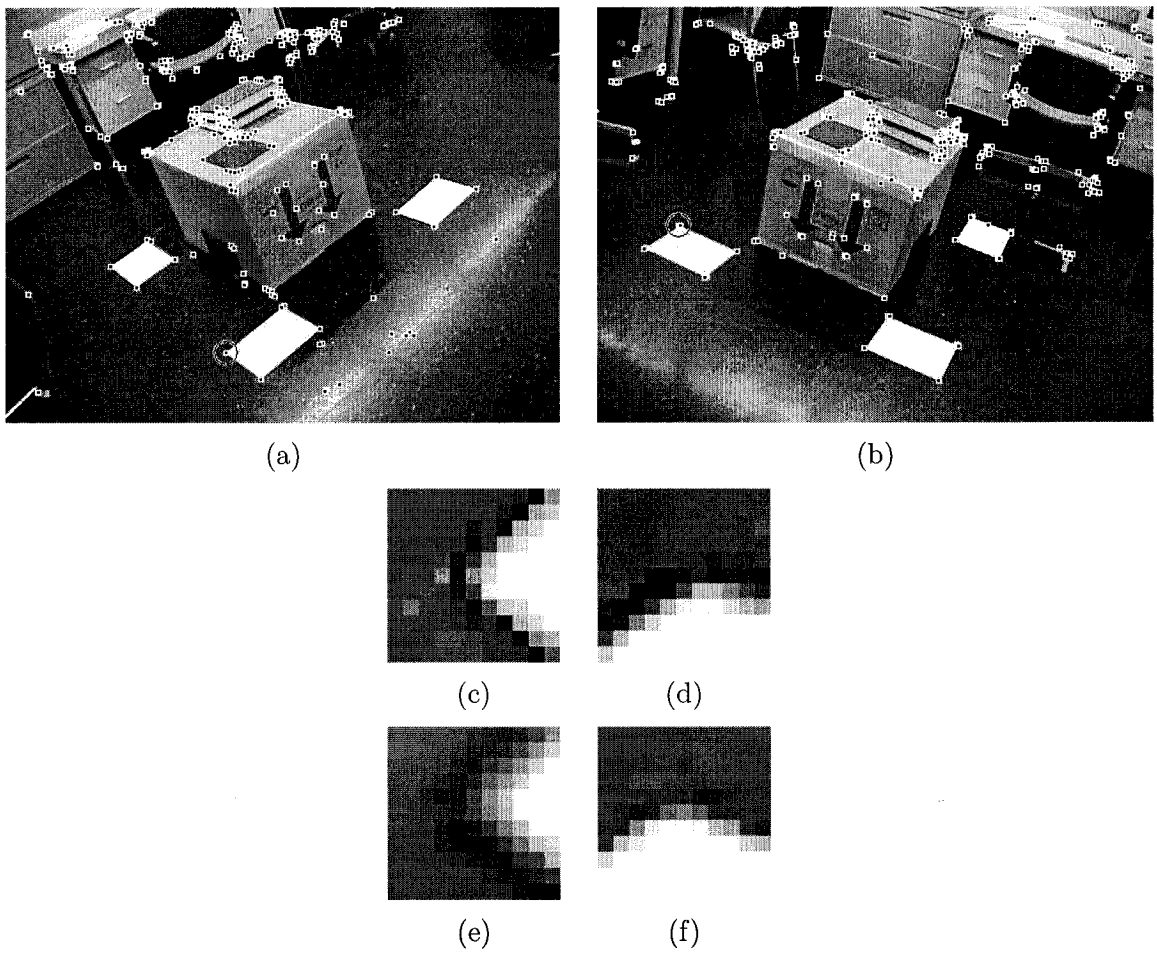


Figure 4.9: The points marked with the dotted white squares are the candidate points as detected by JUDOCA operator. The parameters used are $\sigma = 1.2$, $t_{\mathcal{B}} = 5$, $\lambda = 13$ and $s_J = 0.45$. (a) Left image. (b) Right image. (c) An 11 $\times$ 11 window in the left image. (d) An 11 $\times$ 11 window in the right image. (e) The area correlated to the window (c) after transforming using the homography matrix of the ground plane from the left image to the right image and applying the translation vector $\mathbf{t}_{lr}$l; i.e., the intensity is supplied from the right image. The correlation value is 0.85. (f) The area correlated to the window (d) after transforming using the homography matrix of the ground plane from the right image to the left image and applying the translation vector $\mathbf{t}_{rl}$; i.e., the intensity is supplied from the left image. The correlation value is 0.79. Images in the bottom rows are magnified 10 times.

$\mathtt{H}_g$, $\mathbf{t}_{lr}$ and $\mathbf{t}_{rl}$, we could get the windows transformed as shown in Figures 4.9(e) and (f) for both direction with correlation values of 0.85 and 0.79. In this case, the translation vectors, $\mathbf{t}_{lr} = [1, -1]^T$ and $\mathbf{t}_{rl} = [3, 0]^T$.

The results of detecting ground features are shown in Figures 4.10(a) and (b) on Page 74. The normalized correlation threshold used, t_g, is 0.7, the window size is 21 $\times$ 21 pixels and the threshold t_h is 20.0 mm. Through a threshold $t_{\mathtt{H}}$ =1.0, RANSAC is applied to the match set and enhanced homography matrix is obtained. Using this matrix and a threshold, t_g, of 0.8, the remaining feature points potentially belonging to the obstacles are obtained. These are shown in Figures 4.10(c) and (d) on Page 74. Among these features, some are on the floor, which correspond to the parts of the ground plane occluded in one image. Also, the features on the floor at the edges of the obstacles have not been identified as ground features because correlation failed due to the dissimilarity of the neighborhood. Finally, note that other ground features remain; these ones are false junctions corresponding to the reflection of light on the floor. These will be discarded by the subsequent steps of our system described in the following chapters.

Another pair of wide baseline images is shown in Figure 4.11 on Page 75 where the locations of junctions detected by JUDOCA operator appear as dotted white squares. As in the last example, the error margins are $\pm$10 mm, $\pm$10 mm, $\pm$10 mm, $\pm$0.8°, $\pm$0.8° and $\pm$0.8° for ΔX, ΔY, ΔZ, $\Delta\omega$, $\Delta\phi$ and $\Delta\kappa$ respectively.

In order to show the effects of the correlation window size as well as the correlation threshold, Table 4.1 on Page 75 lists the total number of matches accepted by the algorithm (i.e., considered ground features), the number of correct matches and the percentage of correct matches for a variety of window sizes and correlation thresholds. This table shows that the larger the size of the correlation window, the better the results. Also, larger correlation values tend to provide better results. The behavior of the percentage of the correct matches is shown in Figure 4.12 on Page 76.

Let us consider the case where the correlation window size is 21 $\times$ 21 and the correlation threshold is 0.55. Applying the above algorithm, Figures 4.13(a) and (b) on Page 78 show the features that represent match pairs and should be excluded from the matching process. The source of the few errors is due to the similarity in the vicinities of the mismatch pair. For example, refer to the mismatch circled in Figure 4.13(a). Using the homography matrix as well as the translation vector lead to correlation patches shown in Figures 4.13(e) through (h). Those patches are similar which, in turn, results in a high correlation value and consequently causes the features to be picked up as ground points. Figures 4.13(c) and (d) on Page 78 show the candidate features to be used as an input to the matching process. Using a window

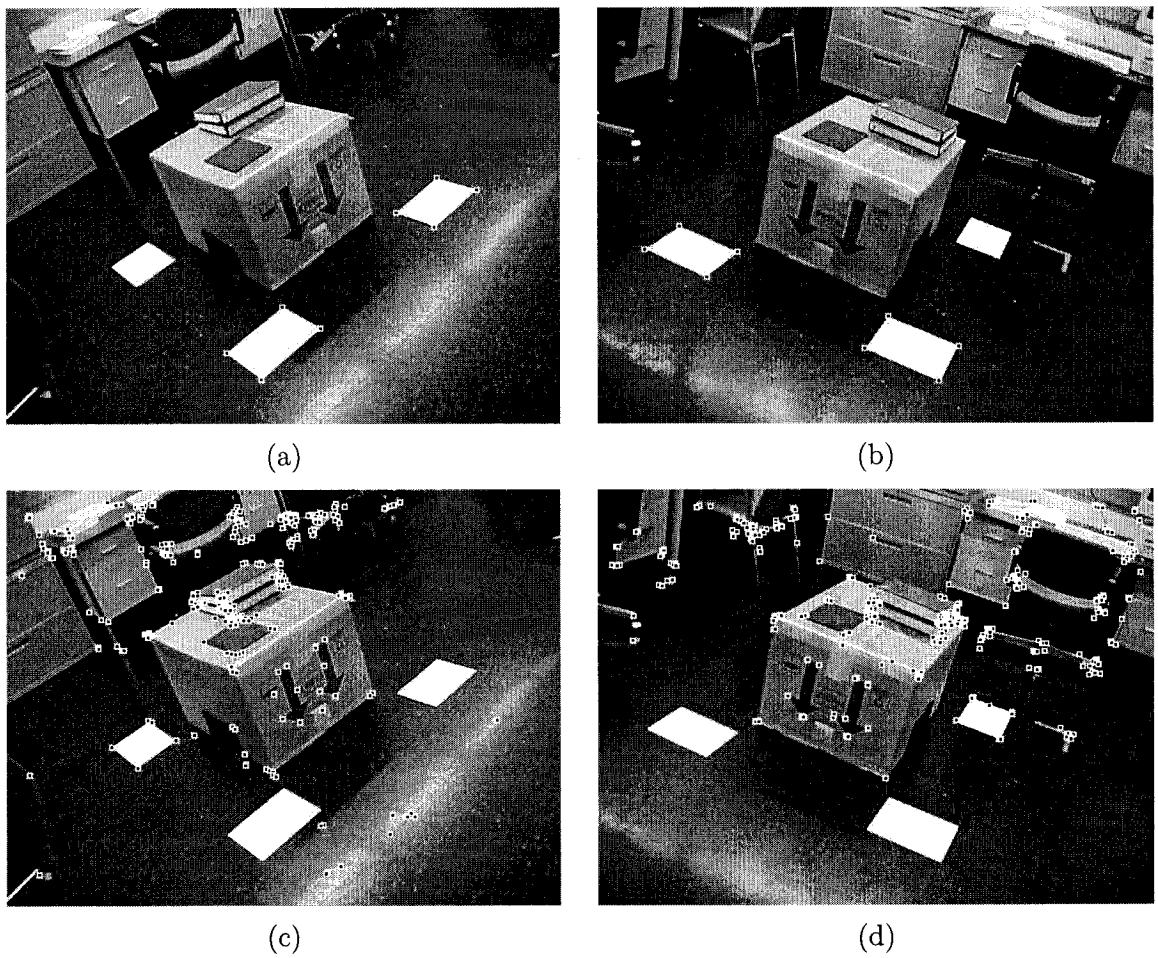
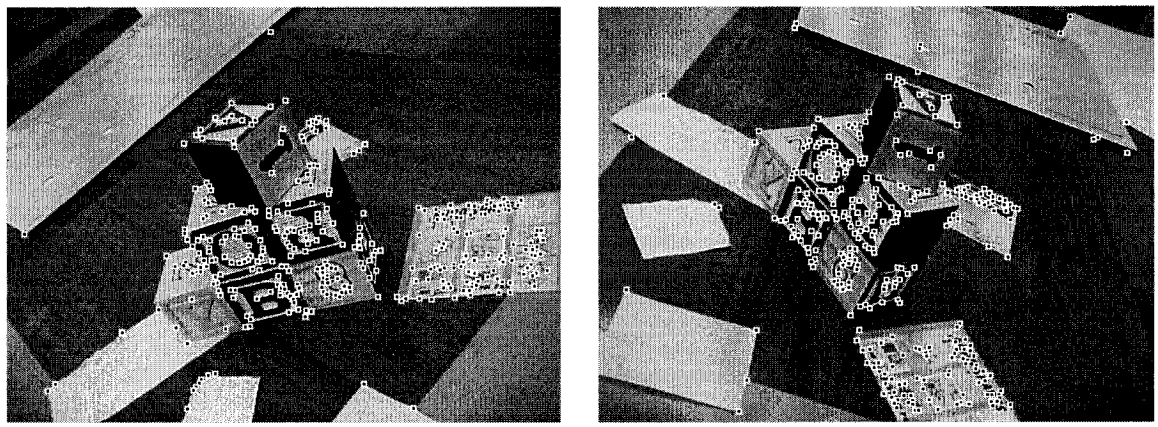


Figure 4.10: (a) and (b) Features detected through homography and the camera parameter error margins. The features are indicated with dotted white squares. Those features should be excluded from the obstacle matching process since they have been identified to lie on the ground. (c) and (d) Candidates for the obstacle matching process. The thresholds are as follows: $t_g = 0.7$ and 0.8, t_h=20.0 and t_{H}=1.0.



(a) (b)

Figure 4.11: The locations of junctions detected by JUDOCA operator. The parameters used to detect junctions are $\sigma = 1.2$, $t_B = 8$, $\lambda = 10$ and $s_J = 0.5$. (a) Left image. (b) Right image.

Window size	t_g	Total number	Number correct	% correct	Window size	t_g	Total number	Number correct	% correct
11 × 11	0.50	65	49	75.4	19 × 19	0.50	53	50	94.3
11 × 11	0.55	58	47	81.0	19 × 19	0.55	51	49	96.1
11 × 11	0.60	51	43	84.3	19 × 19	0.60	44	42	95.5
11 × 11	0.65	47	40	85.1	19 × 19	0.65	43	42	97.7
11 × 11	0.70	34	29	85.3	19 × 19	0.70	34	33	97.1
13 × 13	0.50	58	47	81.0	21 × 21	0.50	54	51	94.4
13 × 13	0.55	55	46	83.6	21 × 21	0.55	52	50	96.2
13 × 13	0.60	51	44	93.6	21 × 21	0.60	47	45	95.7
13 × 13	0.65	45	40	88.9	21 × 21	0.65	40	38	95.0
13 × 13	0.70	39	35	89.7	21 × 21	0.70	30	30	100.0
15 × 15	0.50	57	47	82.5	23 × 23	0.50	56	53	94.6
15 × 15	0.55	53	46	86.8	23 × 23	0.55	52	50	96.2
15 × 15	0.60	48	44	91.7	23 × 23	0.60	50	49	98.0
15 × 15	0.65	46	43	93.5	23 × 23	0.65	39	39	100.0
15 × 15	0.70	37	34	91.9	23 × 23	0.70	34	34	100.0
17 × 17	0.50	57	50	87.7	25 × 25	0.50	56	53	89.3
17 × 17	0.55	48	46	93.8	25 × 25	0.55	52	46	88.7
17 × 17	0.60	46	43	93.5	25 × 25	0.60	46	43	93.5
17 × 17	0.65	44	41	93.2	25 × 25	0.65	39	38	97.4
17 × 17	0.70	35	32	91.4	25 × 25	0.70	32	32	100.0

Table 4.1: The effect of the correlation window size and the correlation threshold, t_g, on the overall results of detecting ground features. The threshold, t_h, used is 20.0.

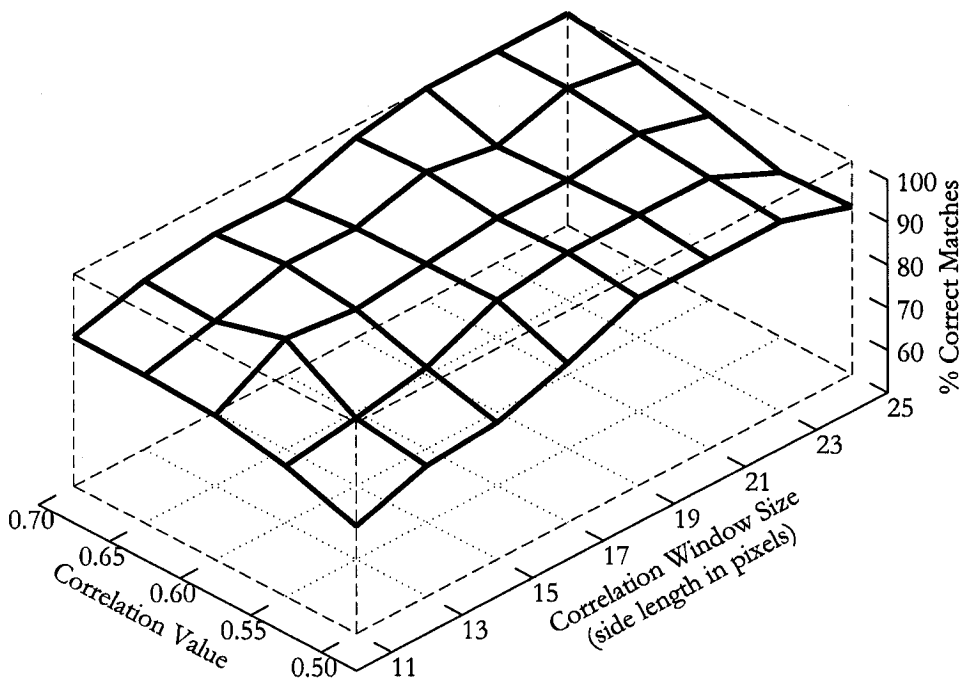


Figure 4.12: The behavior of the percentages of correct matches through different correlation window sizes and correlation thresholds.

size of 25 × 25 and a correlation threshold of,t_g, 0.70 results in avoiding the above errors as shown in Figure 4.14 on Page 79. Further processing of these results using the updated homography matrix, with thresholds, t_g, of 0.7, t_h, of 20.0 and t_{H}, of 6.0, lets us detect other ground features that are not part of match pairs. (The maximum distance recorded for t_{H} in this example was 3.2 pixels.) This is shown in Figure 4.15 on Page 80. Notice that ground features in occluded areas are still present in Figures 4.15(c) and (d).

Note that our goal at this step is not to detect all features on the ground plane but to exclude those possible matches on that plane. This means that ground features may still be present after this step with the majority of correspondences on the ground plane excluded. Although this objective may have been fulfilled prior to using the updated homography matrix since possible ground matches already have been detected, using this version of the matrix helps exclude more ground features. This, in turn, facilitates the coming matching process. In other words, this relieves some of the pressure off the next step.

One last point to demonstrate the usefulness of triangulation that triggers the threshold, t_h, can be mentioned where occlusion may happen. Consider the case shown in Figure 4.16 on Page 81 where correlation is applied to choose a possible match in the right image for the point marked in the left image. Abandoning the triangulation step and the threshold, t_h, causes Point "1" to be picked up as a potential match that belongs to the ground surface. Triangulation estimates the corresponding point in space to be 29.52 mm above the ground plane. We minimized this height in the vicinity of Point "1" by applying triangulation. We found that Point "2" produces a space point that is closer to the ground than that of Point "1" with a height of -0.1 mm. Hence, Point "2" could be a real match; however, it is occluded and could have never been picked up if triangulation was not applied causing Point "1" to be chosen instead, which results in a mismatch.

4.11 Parameters and Thresholds

There are several parameters and thresholds manipulated in this chapter. The parameters required to calculate the planar perspective matrix to display the perspective view of the ground plane are the intrinsic and extrinsic parameters of the virtual camera.

In order to estimate the inter-image homography, the planar perspective matrix may be used. In this case, to estimate the matrix, the parameters of the cameras detected by the sensors are to be used. The error margins of the extrinsic parameters are used at this stage to identify a search range for possible match candidates.

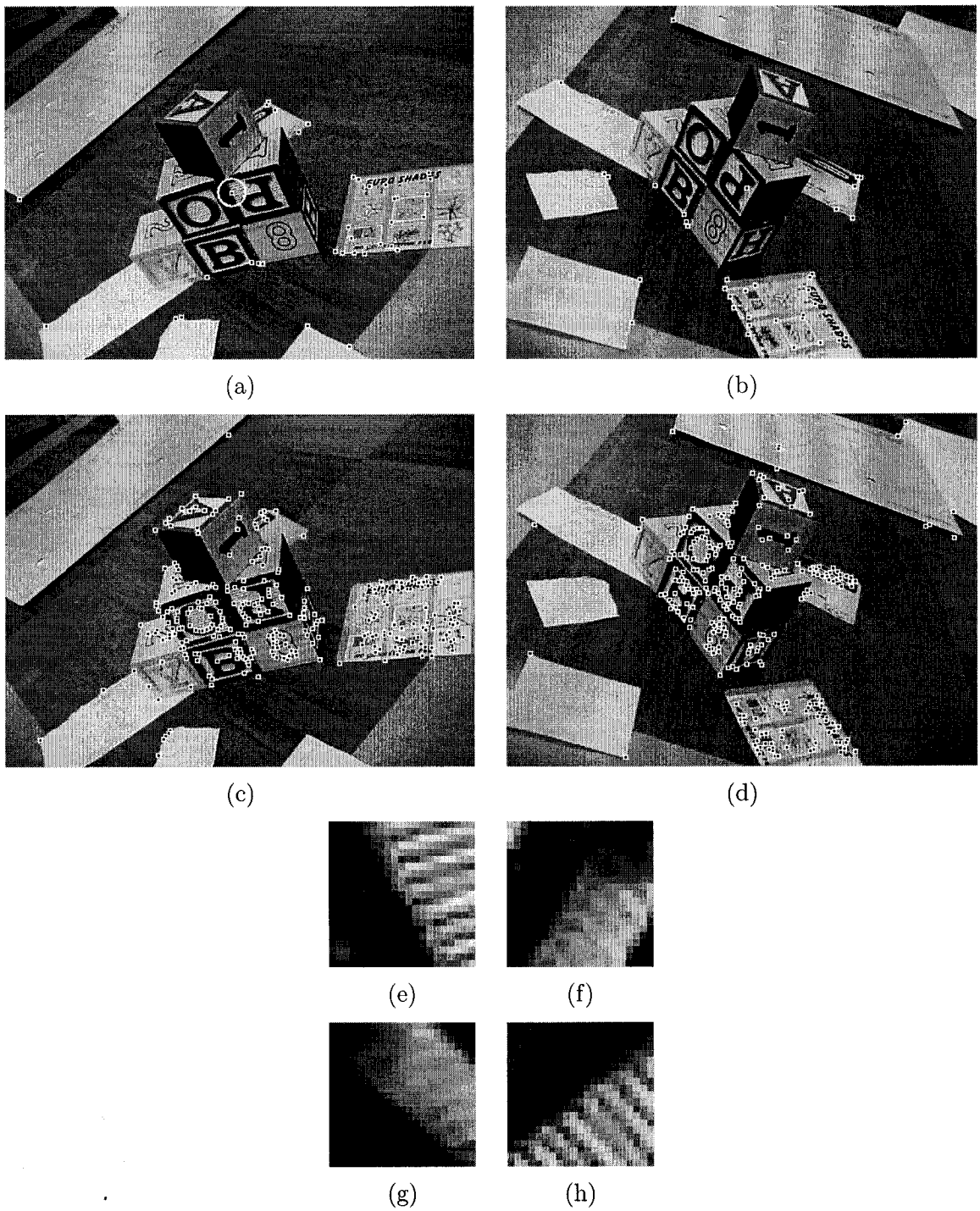


Figure 4.13: Using a window size of 21 × 21. (a) and (b) Ground features that represent match pairs. (c) and (d) The complement sets to (a) and (b). (e) A 21 × 21 window in the left image. (f) A 21 × 21 window in the right image. (g) The area correlated to the window (e) after transforming. (h) The area correlated to the window (f) after transforming. Images in the bottom rows are magnified 5 times.

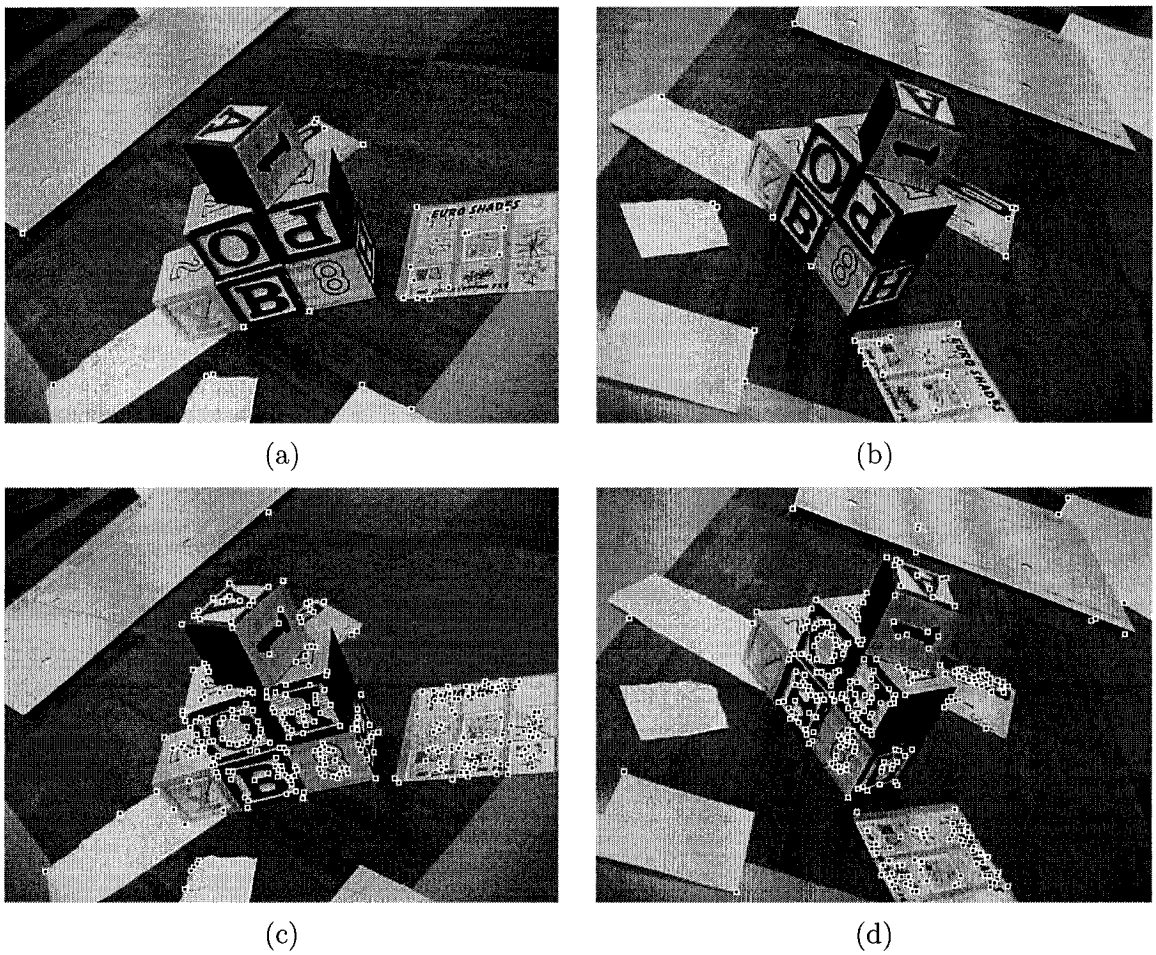


Figure 4.14: Using a window size of 25 × 25. Larger values for the correlation window size and the threshold avoid errors happened while detecting the ground features. (a) and (b) Ground match pairs that should be excluded from the matching process. (c) and (d) Candidates for the obstacle matching process.

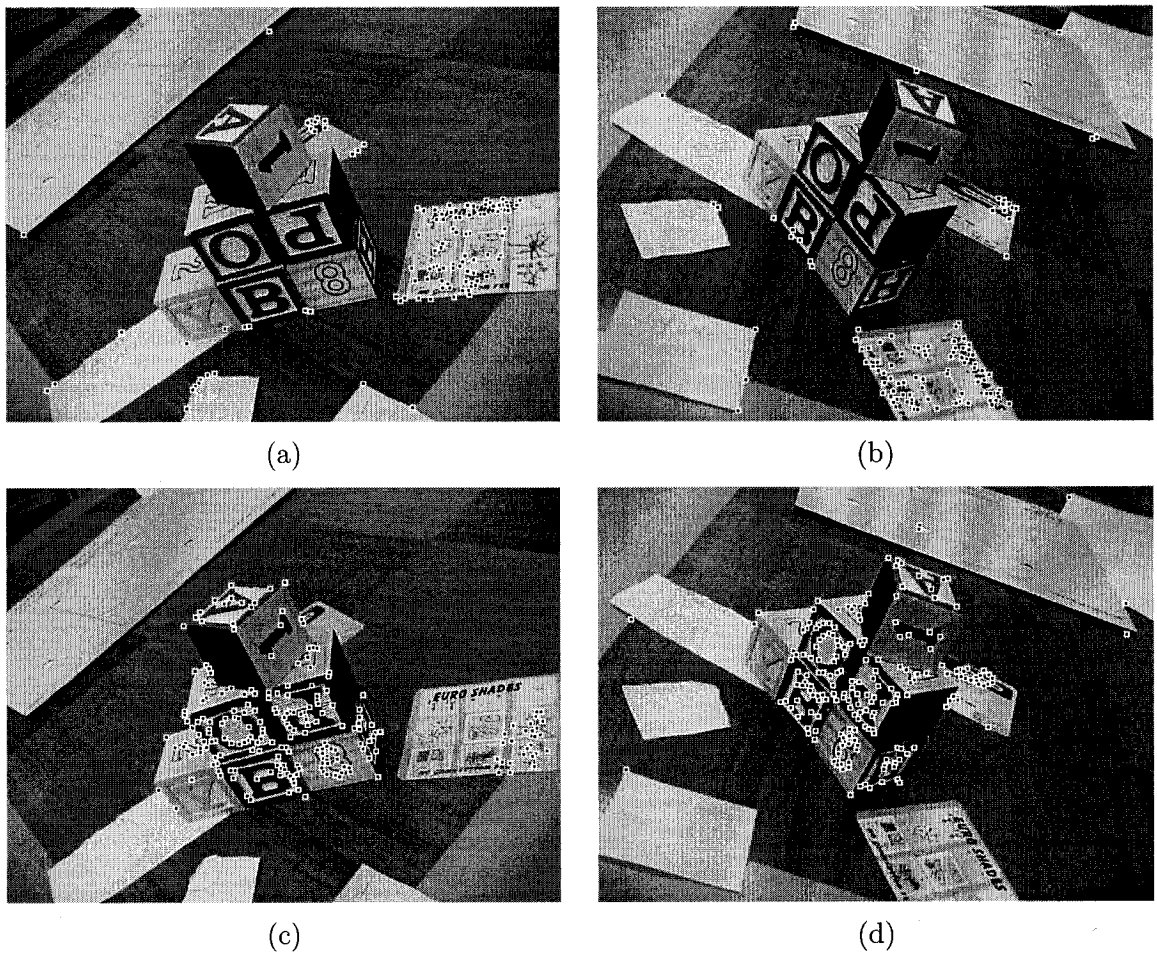
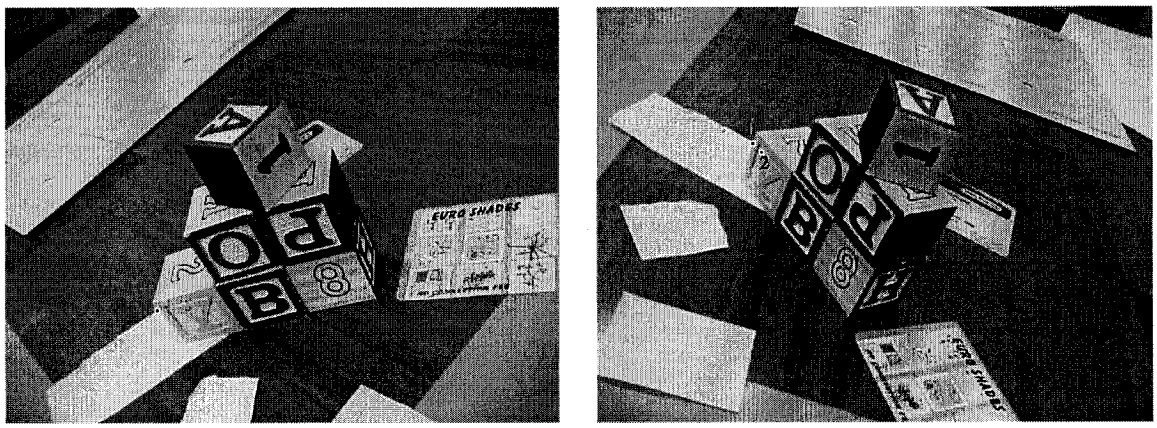


Figure 4.15: Using the updated homography matrix with a window size of 25 × 25 enhances the results. (a) and (b) Features that should be excluded from the matching process. (c) and (d) Candidates for the obstacle matching process. Notice that feature points in occluded areas still present in (c) and (d).



(a) (b)

Figure 4.16: The effect of the triangulation at the ground feature detection phase.

In addition to the above-mentioned parameters, there is another parameter that plays an important role in identifying match pairs on the ground plane. This is the size of the correlation window. We showed by experiments that the larger the window the better the results in terms of the percentage of correct matches. Another threshold that plays another important role is the threshold, t_g. This threshold represents a VNC correlation, which has a legitimate range [-1.0, 1.0] with -1.0 represents the worst case and 1.0 represents the perfect case. From our experiments, we can see that using a window size of ≥ 20 pixels with t_g set to around 0.7 gives very good results.

The last two thresholds mentioned in this chapter are t_h and t_{H}. The threshold t_h provides more restriction on the height of the point than that provided by the values of the error margins. In some mismatch cases, this threshold will be able to identify them. The last threshold t_{H} is used with RANSAC to represent the deviation from the potential location of the corresponding point. Finally, Sections A.1 and A.3 summarize the role of each of the above-mentioned parameters and thresholds.

4.12 Summary

The preceding chapter enclosed two related parts. The first part detailed the derivation of the planar perspective matrix. It began by a geometrical analysis of the relationships among the different coordinate systems under consideration utilizing materials from Chapter 2. Then, it went through computing the complete planar perspective matrix. This matrix will be used in Chapters 6 and 7 to generate the perspective view of the ground plane of the surrounding

environment.

In the other part of this chapter, we proposed an approach to identify the feature points, detected through the JUDOCA operator proposed in Chapter 3, on the ground surface of the scene under consideration. As a guide, we used a roughly calculated homography using the planar perspective matrix. (Alternatively, the equation of the ground plane may be used.) Through this homography and the error margins available for the camera parameters, a set of putative corresponding junctions in one image could be obtained for each junction in the other image. Applying normalized correlation technique along with calculating the height of the feature point in 3D space could determine the best matches. Those matches were likely to belong to the ground plane. At that point, we used the match set belonging to the ground plane with RANSAC scheme to update the homography of the ground. Hence, more feature points that belong to the ground but do not constitute match pairs could be detected. These features should be excluded from further processing carried out in Chapter 5.

Finally, we can summarize our contributions this chapter in the derivation of the planar perspective matrix through different rotation systems [EL03] and identifying a match set that belongs to the ground plane [Eli04]. The later contribution represents a special case of the homographic correlation we will present in the next chapter where we will introduce an approach to identify a match set on arbitrary planes.

Chapter 5

Wide Baseline Matching of Obstacle Features

The matching problem deals with finding point correspondences among different views of a scene. This is an important step towards 3D modeling. A reliable matching algorithm results in a good 3D estimation of object structures. In this chapter, we explore different formulations of the correspondence or matching problem and we present the algorithm that we developed to achieve correspondences in the context of obstacle localization. This algorithm uses, as its input, the junctions detected by JUDOCA method (described in Chapter 3) excluding those that were detected in Chapter 4 as part of the ground plane. The objective is to match feature points that belong to the obstacles of the scene in order to localize them. The result of this matching phase is therefore a match set that is located on obstacles' surfaces.

5.1 Point Correspondence Problem

The feature matching, or point correspondence problem has many formulations. Matching techniques developed can be divided into two main categories. These are:

1. area-based matching; and
2. feature-based matching.

In the first category, the techniques developed take advantage of the similarity in intensity between neighborhoods of corresponding points in both images. In the other category, images are preprocessed to detect features (e.g., corners, edges). The search is to be performed among these features. This category has two main advantages over the previous:

1. It is more flexible to surface discontinuities; e.g., the effect of occlusion might not be as severe as the first category; and

2. As the search is to be done on a small number of pixels, it is computationally less expensive.

A combination of the two categories is possible where the images are preprocessed for detecting features and correlation techniques are applied to the neighborhoods of those features. This is what we are doing in this chapter.

Also, one can split the matching techniques into two main streams according to the length of the camera baseline with respect to the viewed volume. These streams are:

1. short baseline matching; and
2. wide baseline, or sparse view matching.

Furthermore, we can split each of them into calibrated and uncalibrated cases according to whether or not the epipolar geometry is known.

5.1.1 Short Baseline Matching

This is the category for which the cameras are close to each other with respect to the viewed scene. In this case, small changes in intensity values between corresponding points in different views are expected. This results in similarity among the neighborhoods of the corresponding points, which facilitates the identification of those correspondences. In the calibrated case, the fundamental matrix is known and can therefore be used to guide the matching process. Thus, given a point in one image, the corresponding point in the second image lies on the corresponding epipolar line in the second image. The neighborhood of each candidate pair are then compared to determine the acceptable matches. A simple correlation measure can be obtained by computing the sum of the absolute pixel differences (SAD) inside a window. That is [LL96]:

$$C_1(x,y;i,j) = \sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} |\mathcal{I}_l(x+m,y+n) - \mathcal{I}_r(i+m,j+n)| \tag{5.1}$$

Then the best match would be the one that minimizes this similarity measure. Other functions can be used such as the variance normalized correlation (VNC) that is more robust to changes in lighting condition between views:

$$C_2(x,y;i,j) = \frac{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} \mathcal{I}_l(x+m,y+n)-\overline{\mathcal{I}_l(x,y)}][\mathcal{I}_r(i+m,j+n)-\overline{\mathcal{I}_r(i,j)}]}{\sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_l(x+m,y+n)-\overline{\mathcal{I}_l(x,y)}]^2}\sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_r(i+m,j+n)-\overline{\mathcal{I}_r(i,j)}]^2}} \tag{5.2}$$

where $\overline{\mathcal{I}_l(x,y)}$ and $\overline{\mathcal{I}_r(i,j)}$ are the mean values of the neighborhoods surrounding $[x,y]^T$ and $[i,j]^T$ respectively. Many researchers investigated this case; e.g., [Fal94, Fau96].

Variable block size and multiresolution techniques can be used [AN95, DTS94, Zie92]. Also, to deal with perspective deformations, linear transformations can be used [LAC93, SG92, RBYL93].

Since in the uncalibrated case, the camera parameters are not known, the search for matching points is done in a local neighborhood surrounding the original position resulting in a short disparity vector [DZLF94]. The RANSAC scheme (Random Sample Consensus) [FB81, RW00] is often used, in this context, to obtain an estimate for the fundamental matrix, F, from putative matches. Armed with this estimated fundamental matrix, guided matching can be applied as in the calibrated case [Fis97c].

5.1.2 Wide Baseline Matching

Here, the baseline between the cameras is significantly wide with respect to the viewed scene. This is a more difficult case than the previous one, since the neighborhoods of any corresponding two pixels are likely to be different because more important perspective deformations generally exist. In addition, occlusion becomes a significant problem in this case. This makes the images involved in the process substantially different.

Measures of invariance are needed to overcome this difficulty. Perspective deformation can be accurately estimated by a 3×3 homography matrix (Section 2.3.5) relating two homogeneous feature points if local planarity exists around those corresponding points. This matrix has 8 degrees of freedom (DOF) assuming that it is defined up to a scale factor. Usually, four corresponding pairs are needed to define a homography matrix. In [PZ98], Pritchett and Zisserman performed edge detection, linking and line fitting on both images. This is followed by line grouping with the constraint of parallelism and length equality to obtain parallelograms. Once the parallelograms have been grouped, they are matched pairwise between the images. This is done by computing the projective homography matrices from the four line correspondences and cross-correlating the regions. Another novel approach to compute the projective homography in order to achieve good matching algorithm is presented later in this chapter.

Because of that number of DOF's, it might be difficult to estimate the relation between every corresponding pair of planar patches. Thus, approximations to homographic transformation are often used [VLR00]. One such approximation is the affine homography that has 6 degrees of freedom and relating two inhomogeneous feature points. Examples of affine invariance measure techniques can be found in [TGDK99, TG01].

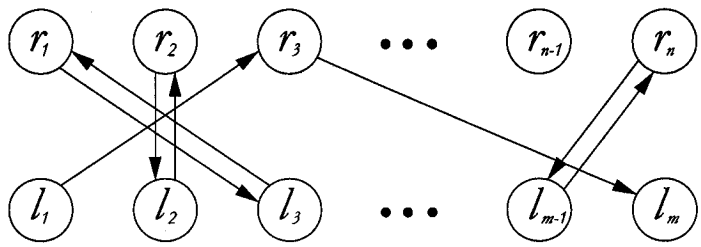


Figure 5.1: Stable marriage problem is related to point correspondence problem.

As in the case of short baseline, two categories exist; calibrated and uncalibrated wide baseline. In this chapter, we will consider the calibrated case. Indeed, in our case, the intrinsic camera parameters are known *approximately* and the pose difference between two consecutive viewpoints of the camera; i.e., the extrinsic parameters, is significant. In the case of uncalibrated wide baseline, the epipolar geometry is to be determined as a result of achieving point correspondence [Bau00]. Once the epipolar geometry is known, the search for further correspondences can be greatly facilitated.

5.2 Wide Baseline Matching of Obstacle Features

We may turn the problem of point matching across widely separated views into a stable marriage problem. In its formal definition, stable marriage problem states that, given two groups of the same number of men and women, and each member has expressed his/her ranking for the members in the opposite group, it is required to find a matching between both groups such that the marriages are stable. A marriage between a man m_i and a woman w_i is said to be stable if:

- whenever m_i prefers another woman w_j to w_i, w_j prefers her husband m_j to m_i; and
- whenever w_i prefers another man m_j to m_i, m_j prefers his wife w_j to w_i.

However, in our case, where the members of both groups are junctions detected in both images, we may have some assumptions:

- In general, the number of junctions is not the same in both groups.
- Also, generally, a member in one group does not provide a full ranking set. As consequences:
  - some members may end up with no matching at all (e.g., r_{n-1} in Figure 5.1);
  - others may not have a mutual acceptance (e.g., l_m and r_3 in Figure 5.1).

  both cases cannot be considered as perfect matches and should be rejected.

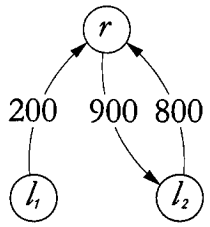


Figure 5.2: An example of stable marriage: rl_2 will be the accepted marriage even if l_1r has a higher cost.

To achieve a *stable marriage junction matching*, we used the following algorithm [Eli04]:

1. Detect a set of junctions in each image using the method described in Chapter 3.
2. Filter out those that belong to the ground plane using the method described in Chapter 4.
3. Calculate the fundamental matrix using the known approximate values for the camera parameters, following the equations given in Section 5.2.1.
4. Based on the estimated epipolar geometry and on the error margins of the camera parameters, determine the search area for matching. This is explained in Section 5.2.2.
5. Match each junction of the first image to the junctions, in the other image, lying inside the identified search area. The similarity between two junctions will be measured using homographic correlation as described in Section 5.3.
6. According to the stable marriage principle, the valid matches will be the ones having a mutual acceptance. For example, refer to Figure 5.2 on Page 87. Although the cost of the link l_1r is considerably lower than that of l_2r, but according to the stable marriage problem definition, r should not prefer l_1 to l_2.
7. Using the thus obtained match set, iteratively refine the fundamental matrix $\mathtt{F}$ using the procedure described in Section 5.5.

5.2.1 Fundamental Matrix Calculation Using Camera Parameters

There are many ways to calculate the fundamental matrix, $\mathtt{F}$. All are equivalent and must lead to the same matrix. We may start by calculating a projection matrix; e.g., $\mathtt{P}'$, using Equation (2.8). Hence assuming that the camera calibration matrices are identical for both view; i.e., $\mathtt{A} = \mathtt{A}'$, we have:

$$\mathtt{P}' = \mathtt{A}\mathtt{P}_c\mathtt{D}' \tag{5.3}$$

Refer to Section B.4 for the details of each term. This is followed by calculating an epipole; e.g., $\dot{\mathbf{e}}'$, using Equation (2.29). That is:

$$\mathbf{e}' = \begin{bmatrix} e'_1 \\ e'_2 \\ e'_3 \end{bmatrix} = \mathtt{P}'\mathbf{C} \tag{5.4}$$

According to Equation (2.34), the last row of $\mathtt{P}^+$ is $\mathbf{0}_3^T$, then:

$$\mathtt{P}'\mathtt{P}^+ = [\mathtt{AR}'][\mathtt{AR}]^{-1} \tag{5.5}$$

(Refer to Section B.5 for details of each term of AR (or equivalently AR'.) Equation (2.35) states that:

$$\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+ \tag{5.6}$$

Substituting the value of $\mathtt{P}'\mathtt{P}^+$ from Equation (5.5) in Equation (5.6), we get:

$$\mathtt{F} = [\mathbf{e}']_\times [\mathtt{AR}'][\mathtt{AR}]^{-1} \tag{5.7}$$

which represents the fundamental matrix that we may use in the upcoming matching process.

5.2.2 The Epipolar Line

Given one point, $\dot{\mathbf{m}}$, in an image, we can determine the equation of the corresponding epipolar line in the other image using Equation (2.36). That is:

$$\mathbf{l}' = \mathtt{F}\mathbf{m} \tag{5.8}$$

Notice that, theoretically, the corresponding point must lie on the epipolar line, $\mathbf{l}'$. However, due to the inaccuracy in camera parameters, the corresponding points are most likely to lie above or below this line in a strip whose shape may be determined through the error margins of the camera parameters; i.e., $\pm\Delta X$, $\pm\Delta Y$, $\pm\Delta Z$, $\pm\Delta\omega$, $\pm\Delta\phi$ and $\pm\Delta\kappa$. Thus, this strip should be considered as a range in our search for a suitable corresponding point.

In order to express the edges of this strip with respect to the location of the epipolar line, $\mathbf{l}'$ in Equation (5.8), we determine the points of intersections between this epipolar line, $\mathbf{l}'$, and both left and right borders of the image, $\mathbf{l}_l$ and $\mathbf{l}_r$, where:

$$\mathbf{l}_l = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{5.9}$$

and

$$\mathbf{l}_r = \begin{bmatrix} 1 \\ 0 \\ -w \end{bmatrix} \tag{5.10}$$

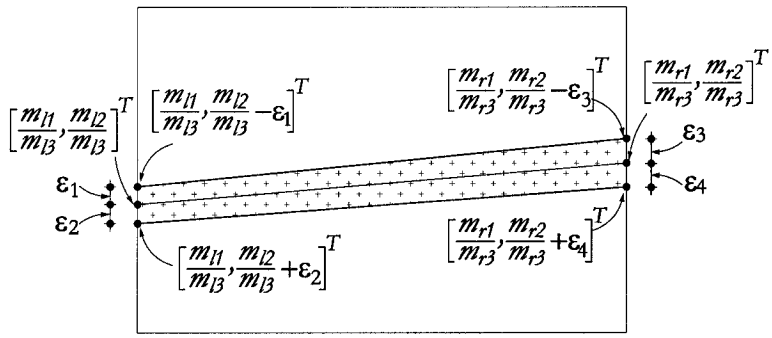


Figure 5.3: A strip is considered as a search range.

where w is the image width. Thus, these points are $\mathbf{m}_l$ and $\mathbf{m}_r$ and can be expressed as:

$$\mathbf{m}_l = \begin{bmatrix} m_{l1} \\ m_{l2} \\ m_{l3} \end{bmatrix} = \mathbf{l}' \times \mathbf{l}_l \tag{5.11}$$

and

$$\mathbf{m}_r = \begin{bmatrix} m_{r1} \\ m_{r2} \\ m_{r3} \end{bmatrix} = \mathbf{l}_r \times \mathbf{l}' \tag{5.12}$$

where $\times$ denotes the cross product.

Each parameter error margin contributes to a strip by itself. The four end points of each strip with the vertical lines, $[1, 0, 0]^T$ and $[1, 0, -w]^T$ are detected. The minimum and maximum coordinates with those vertical lines are detected among all candidates. This results in four points along the lines $\mathbf{m}_l$ and $\mathbf{m}_r$, mentioned in Equations (5.11) and (5.12), at distances ε_1, ε_2, ε_3 and ε_4 from the epipolar line, $\mathbf{l}'$, mentioned in Equation (5.8). See Figure 5.3 on Page 89. Then, the two lines, $\mathbf{l}'_a$ and $\mathbf{l}'_b$, representing the edges of this strip can be expressed as:

$$\mathbf{l}'_a = \begin{bmatrix} \frac{m_{r2}}{m_{r3}} - \frac{m_{l2}}{m_{l3}} + \varepsilon_1 - \varepsilon_3 \\ \frac{m_{l1}}{m_{l3}} - \frac{m_{r1}}{m_{r3}} \\ \frac{m_{r1}}{m_{r3}}\left(\frac{m_{l2}}{m_{l3}} - \varepsilon_1\right) - \frac{m_{l1}}{m_{l3}}\left(\frac{m_{r2}}{m_{r3}} - \varepsilon_3\right) \end{bmatrix} \tag{5.13}$$

and

$$\mathbf{l}'_b = \begin{bmatrix} \frac{m_{r2}}{m_{r3}} - \frac{m_{l2}}{m_{l3}} + \varepsilon_4 - \varepsilon_2 \\ \frac{m_{l1}}{m_{l3}} - \frac{m_{r1}}{m_{r3}} \\ \frac{m_{r1}}{m_{r3}}\left(\frac{m_{l2}}{m_{l3}} + \varepsilon_4\right) - \frac{m_{l1}}{m_{l3}}\left(\frac{m_{r2}}{m_{r3}} + \varepsilon_2\right) \end{bmatrix} \tag{5.14}$$

where ε_1, ε_2, ε_3 and ε_4 are the vertical distances from the epipolar line given by Equation (5.8). The previous derived equations are based on the fact that the lines both $\mathbf{l}_a$ and $\mathbf{l}_b$ are

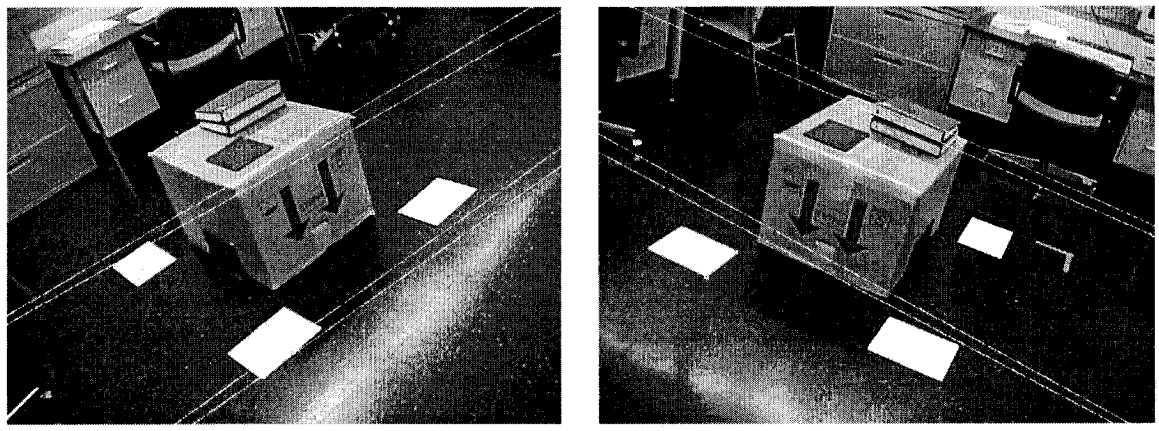


Figure 5.4: Two points selected in each image and their corresponding epipolar strips in the other image are shown. A strip is shaped according to the error margins of the camera parameters. The error margins are ±10 mm, ±10 mm, ±10 mm, ±0.8°, ±0.8° and ±0.8° for ΔX, ΔY, ΔZ, $\Delta\omega$, $\Delta\phi$ and $\Delta\kappa$ respectively.

passing through $[\frac{m_{l1}}{m_{l3}}, \frac{m_{l2}}{m_{l3}} - \varepsilon_1]^T$ and $[\frac{m_{r1}}{m_{r3}}, \frac{m_{r2}}{m_{r3}} - \varepsilon_3]^T$, and $[\frac{m_{l1}}{m_{l3}}, \frac{m_{l2}}{m_{l3}} + \varepsilon_2]^T$ and $[\frac{m_{r1}}{m_{r3}}, \frac{m_{r2}}{m_{r3}} + \varepsilon_4]^T$ respectively. Refer to Section C.1 for more explanation. Thus, to decide if a pixel $\dot{\mathbf{m}}' = [i, j]^T$ is included in this strip, the values:

$$\mathbf{l}_a'^T \mathbf{m} \tag{5.15}$$

and

$$\mathbf{l}_b'^T \mathbf{m} \tag{5.16}$$

must have different signs.

5.3 Matching Through Plane Homography

In order to match two junctions, regular correlation measure could be used. However, because of the important difference in the points of view of wide baseline images used here, this kind of approach is likely to fail. We, therefore, propose to transform the corresponding neighborhood prior to applying correlation according to a homography computed from the junction features. Indeed, assuming obstacles made of locally planar patches, every two corresponding 2-edge 2D junctions form a planar 3D junction in space; i.e., where two 3D lines intersect at a common point. Using the equation of the junction plane, the homography matrix relating the images of this plane can be calculated. Through the homography matrix, we can transform the plane of a given junction from one image to the other image and perform a correlation operation to check if matching happens. We will explain this idea in more detail throughout the rest of this section.

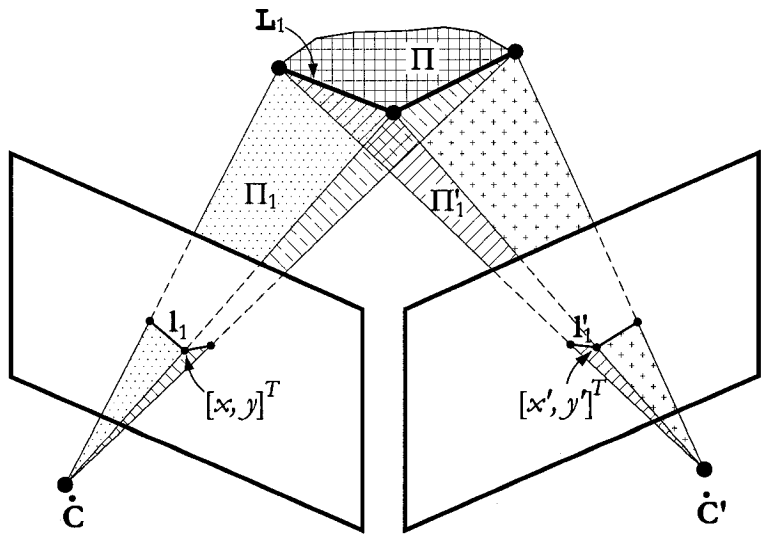


Figure 5.5: Every 3D line is the intersection of two planes formed by the optical centers and the projections of the line onto both images.

5.3.1 Line Reconstruction

As depicted in Figure 5.5 on Page 91, any 3D line, L_1, is the intersection of two planes formed by the optical centers and the projections of the line, l_1 and l'_1, onto both views. These planes are given by [HZ00]:

$$\Pi_1 = P^T l_1 \tag{5.17}$$

and

$$\Pi'_1 = P'^T l'_1 \tag{5.18}$$

where P and P′ are the projection matrices computed through the available camera parameters.

Suppose that the JUDOCA junction detector (discussed in Chapter 3) results in a pair of junctions at $[x, y]^T$ and $[x', y']^T$ with corresponding edges, l_1 and l'_1, whose inclination angles are μ_1 and μ'_1. Then, according to Equation (C.5), the equations of the lines in both images are given by:

$$l_1 = \begin{bmatrix} tan\ (\mu_1) \\ -1 \\ y - tan(\mu_1)x \end{bmatrix} \tag{5.19}$$

and

$$l'_1 = \begin{bmatrix} tan\ (\mu'_1) \\ -1 \\ y' - tan(\mu'_1)x' \end{bmatrix} \tag{5.20}$$

Hence, according to Equation (2.4), the line in 3D space can be expressed as a 2×4 matrix, L_1:

$$\mathrm{L}_1 = \begin{bmatrix} \mathrm{l}_1^T \mathrm{P} \\ \mathrm{l}_1'^T \mathrm{P}' \end{bmatrix} \tag{5.21}$$

5.3.2 Plane Reconstruction

Suppose that the lines $\mathrm{L}_1 = [\mathrm{l}_1^T \mathrm{P}, \mathrm{l}_1'^T \mathrm{P}']^T$ and $\mathrm{L}_2 = [\mathrm{l}_2^T \mathrm{P}, \mathrm{l}_2'^T \mathrm{P}']^T$ form a 3D junction. According to Equation (5.21), the equations of these lines are given by:

$$\mathrm{L}_1 \mathbf{M} = \begin{bmatrix} \mathrm{l}_1^T \mathrm{P}_1 \\ \mathrm{l}_1'^T \mathrm{P}_2 \end{bmatrix} \mathbf{M} = \mathbf{0} \tag{5.22}$$

and

$$\mathrm{L}_2 \mathbf{M} = \begin{bmatrix} \mathrm{l}_2^T \mathrm{P}_1 \\ \mathrm{l}_2'^T \mathrm{P}_2 \end{bmatrix} \mathbf{M} = \mathbf{0} \tag{5.23}$$

where $\dot{\mathbf{M}}$ is the location of the junction in 3D space. Equations (5.22) and (5.23) can be rewritten as:

$$\begin{bmatrix} \mathbf{N}_1^T | d_1 \\ \mathbf{N}_1'^T | d_1' \end{bmatrix} \mathbf{M} = \mathbf{0} \tag{5.24}$$

and

$$\begin{bmatrix} \mathbf{N}_2^T | d_2 \\ \mathbf{N}_2'^T | d_2' \end{bmatrix} \mathbf{M} = \mathbf{0} \tag{5.25}$$

where

- $\mathbf{N}_i$ and $\mathbf{N}_i'$, $i \in \{1, 2\}$, are 3D vectors representing the normals to the planes Π_i and Π_i'; and
- d_i and d_i', $i \in \{1, 2\}$, are the distances to the planes Π_i and Π_i' from the origin.

Any two lines that intersect at a point in space form a plane. As a consequence, the lines L_1 and L_2 form a plane in space. The normal to this plane can be calculated as follows:

$$\mathbf{N} = (\mathbf{N}_1 \times \mathbf{N}_1') \times (\mathbf{N}_2 \times \mathbf{N}_2') \tag{5.26}$$

where

- $\times$ denotes the cross product; and
- $\mathbf{N}$ should be normalized to unit magnitude.

Finally, the distance from the origin to the plane is calculated from the dot product of $\mathbf{N}$ and the 3D point representing the location of the junction (i.e., the intersection between L_1 and L_2). That is,

$$d = -\mathbf{N} \,.\, \dot{\mathbf{M}} \tag{5.27}$$

Hence, according to Equation (2.77), the 3D point, $\dot{\mathbf{M}}$, can be calculated as:

$$\hat{\dot{\mathbf{M}}} = \left([\mathtt{I} - \dot{\mathbf{M}}_\infty \dot{\mathbf{M}}_\infty^T] + [\mathtt{I} - \dot{\mathbf{M}}'_\infty \dot{\mathbf{M}}'^T_\infty]\right)^{-1} \left(\dot{\mathbf{C}} + \dot{\mathbf{C}}' - [\dot{\mathbf{C}}^T \dot{\mathbf{M}}_\infty]\dot{\mathbf{M}}_\infty - [\dot{\mathbf{C}}'^T \dot{\mathbf{M}}'_\infty]\dot{\mathbf{M}}'_\infty\right) \quad (5.28)$$

where

- $\dot{\mathbf{M}}$, $\dot{\mathbf{C}}$, $\dot{\mathbf{C}}'$, $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are the inhomogeneous representation of the point in space, the optical centers and the intersection of the rays with the plane at infinity respectively;
- $\mathtt{I}$ is a 3×3 identity matrix; and
- $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are normalized to unit magnitude.

Thus, the equation of the junction plane is given by:

$$\Pi = \begin{bmatrix} \mathbf{N} \\ d \end{bmatrix} \quad (5.29)$$

5.3.3 Plane Homography Correlation

Considering the local planar patches formed by two edges of every junction, we propose to use the homography transformation of the plane to perform the matching. Correlation techniques are performed between the warped image and the other image. This measure is invariant to projective transformation and hence is suitable for wide baseline matching where the neighborhoods of the corresponding feature points might vary considerably; e.g., see Figure 5.6 on Page 95.

Given the equation of the plane (Section 5.3.2), Equation (2.68) can be used to compute the plane homography. That is:

$$\mathtt{H} = \mathtt{A}' \left[\mathtt{R} - \frac{\mathbf{T}\mathbf{N}^T}{d}\right] \mathtt{A}^{-1} \quad (5.30)$$

However, this equation assumes that the world origin is at the first camera where $\mathtt{P} = \mathtt{A}[\mathtt{I}|\mathbf{0}]$ or, in other words, measurements are considered in the first camera coordinate system. Thus, we have to change our equations accordingly. According to Equation (2.27), this can be done through a 4×4 transformation matrix that transforms an object through a rotation matrix $\mathtt{R}'$ and a translation vector $\mathbf{T}'$. Thus, the new equation of the plane can be obtained as:

$$\Pi_{new} = \begin{bmatrix} \mathbf{N}_{new} \\ d_{new} \end{bmatrix} = \left[\Pi^T \begin{bmatrix} \mathtt{R}'^{-1} & -\mathtt{R}'^{-1}\mathbf{T}' \\ \mathbf{0}_3^T & 1 \end{bmatrix}\right]^T \quad (5.31)$$

Similarly, the new rotation matrix and the new translation vector can be calculated as:

$$[\mathtt{R}'_{new}|\mathbf{T}'_{new}] = \mathtt{A}'^{-1}\mathtt{P}' \begin{bmatrix} \mathtt{R}'^{-1} & -\mathtt{R}'^{-1}\mathbf{T}' \\ \mathbf{0}_3^T & 1 \end{bmatrix} \quad (5.32)$$

Then, by substituting the values of $\mathbf{N}_{new}$, d_{new}, $\mathtt{R}_{new}$ and $\mathbf{T}_{new}$ from Equations (5.31) and (5.32) instead of $\mathbf{N}$, d, $\mathtt{R}$ and $\mathbf{T}$ respectively in Equation (5.30), the homography can be calculated as:

$$\mathtt{H} = \mathtt{A}' \left[\mathtt{R}'_{new} - \frac{\mathbf{T}'_{new}\mathbf{N}^T_{new}}{d_{new}} \right] \mathtt{A}^{-1} \tag{5.33}$$

For example, the homography is calculated between the areas enclosed in the white squares of Figures 5.6(a) and (b) on Page 95 using the junctions at the centers of the squares. These areas are transformed using the homography matrix (and its inverse). Interpolation (discussed in Section 4.5) is used to calculate the intensity values of the established patch. The results are shown in Figures 5.6(e) and (f).

Now, through homographic SAD correlation, the neighborhoods of the two junctions can be correlated. This is by checking:

$$\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} |\mathcal{I}_l(x+m, y+n) - \mathcal{I}_r(\dot{\mathbf{k}}^T)| \leq t_{SAD} \tag{5.34}$$

where

- $N+1$ is the side length of the correlation window in pixels;
- $\mathbf{k} = \mathtt{H}[x+m, y+n, 1]^T$; and
- t_{SAD} is a threshold.

Then, the best choice is the one that results in a true condition with a minimum value among all candidates. Homographic VNC correlation can also be used:

$$\frac{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_l(x+m,y+n) - \overline{\mathcal{I}_l(x,y)}][\mathcal{I}_r(\dot{\mathbf{k}}^T) - \overline{\mathcal{I}_r(\dot{\mathbf{m}}'^T)}]}{\sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_l(x+m,y+n) - \overline{\mathcal{I}_l(x,y)}]^2} \sqrt{\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} [\mathcal{I}_r(\dot{\mathbf{k}}^T) - \overline{\mathcal{I}_r(\dot{\mathbf{m}}'^T)}]^2}} \geq t_{VNC} \tag{5.35}$$

where

- $N+1$ is the side length of the correlation window in pixels;
- $\mathbf{m}' = \mathtt{H}\mathbf{m} = \mathtt{H}[x, y, 1]^T$;
- $\overline{\mathcal{I}_l(x,y)}$ is the mean of the neighborhood surrounding $[x, y]^T$;
- $\mathbf{k} = \mathtt{H}[x+m, y+n, 1]^T$;
- $\overline{\mathcal{I}_r(\dot{\mathbf{m}}')}$ is the mean of the neighborhood surrounding $\dot{\mathbf{m}}'$; and
- t_{VNC} is a threshold.

Then, the best choice is the one that results in a true condition with a maximum value among all candidates.

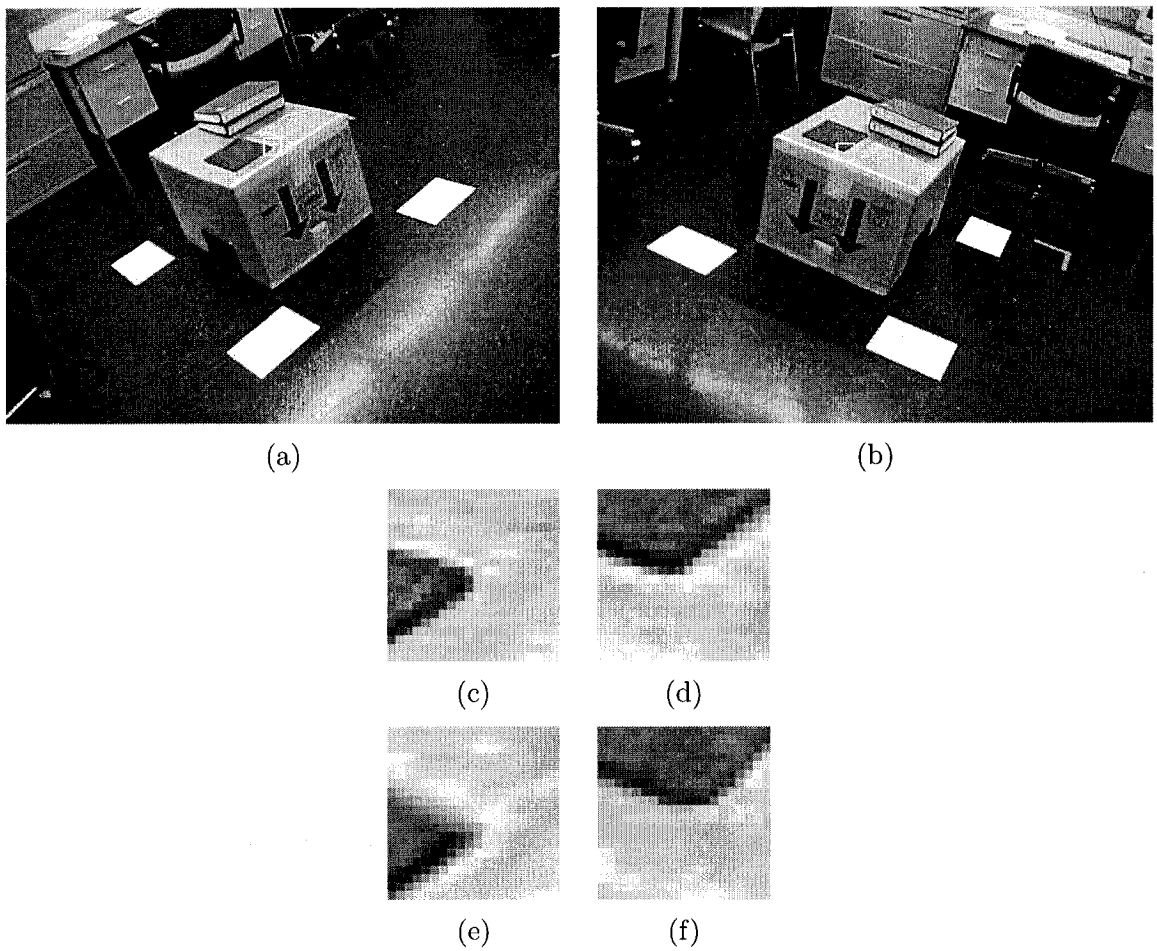


Figure 5.6: (a) and (b) The homography calculated is to be applied to the areas enclosed in the white square. (c) Original left image. (d) Original right image. (e) The homography applied from the left image to the right image. The intensity is fed from the right image. (f) The homography applied from the right image to the left image. The intensity is fed from the left image. The correlation is applied between (c) and (e) images or between (d) and (f) images. Images in the two bottom rows are magnified 4 times.

Notice that the computational cost to perform SAD correlation varies from that of VNC correlation. For example, if the correlation window side is $N+1$, the different operations to calculate the SAD case are as follows:

# operations	Description
$(N+1)^2$	Subtractions
$(N+1)^2$	Calculating the absolute value
$(N+1)^2$	Additions to calculate the correlation
$3(N+1)^2$	Total number of operations

In the VNC case, the different operations to calculate the correlation are as follows:

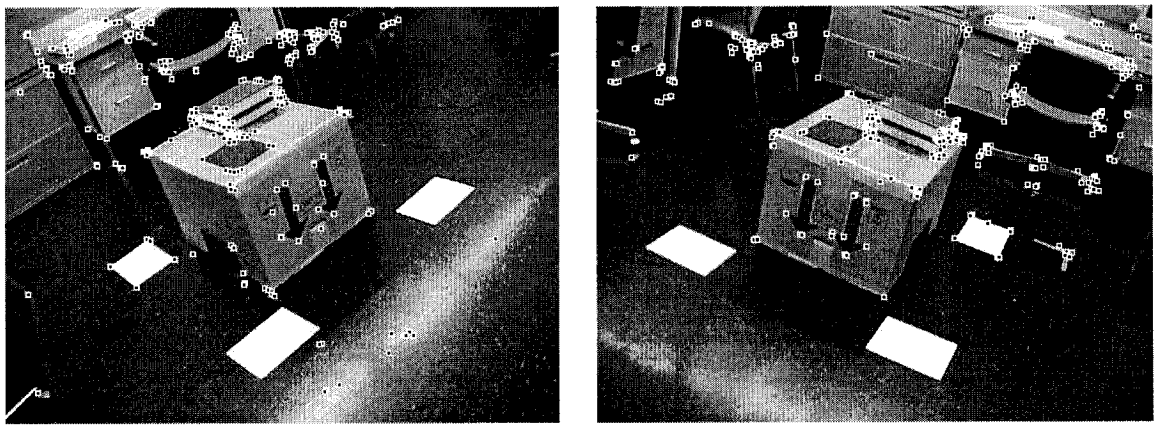
# operations	Description
$2(N+1)^2$	Additions to calculate the mean of the neighborhoods
$(N+1)^2$	Counting to calculate the mean of the neighborhoods
2	Division to calculate the mean of the neighborhoods
$2(N+1)^2$	Subtractions to calculate the correlation
$3(N+1)^2+1$	Multiplications to calculate the correlation
$3(N+1)^2$	Additions to calculate the correlation
2	Calculating the square roots
1	Division to calculate the correlation
$11(N+1)^2+6$	Total number of operations

It is obvious from the number of operations to be performed for SAD and VNC cases that it is much simpler and computationally cheaper to calculate the SAD case. In the following section, we will present some experimental results where both types of correlation mentioned above will be applied to pairs of wide baseline images in order to test the performance achieved by each of them.

5.4 Experimental Results

We applied our algorithm using junctions detected by the JUDOCA detector (Chapter 3) to the pair of gray-scale images shown in Figure 5.7 on Page 97. The algorithm explained in Section 4.9 was applied to that pair in order to exclude ground features. The candidate feature points are shown in Figure 5.7 as dotted white squares.

The camera parameters are listed in Table 5.1 on Page 98; and the world origin is located as shown in Figure 5.8 on Page 98 where the $XY-$plane coincides with the ground surface and the $Z-$axis is vertical. As a comparison, we applied the correlation techniques; non-homographic SAD, homographic SAD and homographic VNC, mentioned in Equations (5.1),



(a) (b)

Figure 5.7: A pair of images shown in Figure 3.8 on Page 47. (a) Left image. (b) Right image. The dotted white squares denote the locations of candidate junctions that are not detected as a part of the ground as explained in Section 4.9. The parameters for JUDOCA operator are $\sigma = 1.2$, $t_B = 5$, $\lambda = 13$ and $s_J = 0.45$. We call these images the "box" pair.

(5.34) and (5.35) respectively. A window size of 21×21 pixels is used for all cases. The threshold used for both non-homographic SAD and homographic SAD correlation cases, t_{SAD}, is 13000 (an average of 29.5 in intensity difference between every two corresponding pixels in the correlated windows). The threshold used for homographic VNC, t_{VNC}, is 0.75. Figures 5.9(a) and (b) on Page 99 show the non-homographic SAD correlation results where each match is represented by a line joining a given feature to the location of its corresponding feature in the other view. Figures 5.9(c) and (d) show the results of homographic SAD correlation while Figures 5.9(e) and (f) show the results of homographic VNC correlation. The percentage of correct matches are 23.8%, 96.0% and 100.0% for non-homographic SAD, homographic SAD and homographic VNC correlations respectively. It is obvious from the percentages of correct matches that using the homographic version of SAD technique to correlate two areas improves the outcome by a factor of more than 4 over that of non-homographic version. Using homographic VNC correlation improves the result even better. The large difference between the results of our homographic technique versus non-homographic version indicates that in case of wide baseline matching, non-homographic SAD correlation techniques do not result in a good outcome as expected because of the existence of perspective deformations. Figure 5.10 on Page 100 shows the behavior of the percentages of correct matches versus the number of matches.

The cumulative histograms for non-homographic SAD correlation, homographic SAD correlation and homographic VNC correlation are shown in Figures 5.11(a) and (b) on Page 101.

Image	X	Y	Z	ω	ϕ	κ
(a)	-1535.2119	1720.8447	1693.2169	-35.1616	-46.7723	-155.5747
(b)	-1694.5607	-142.2461	1585.1900	25.0285	-55.5578	-47.0806

Table 5.1: The "box" pair: The parameters for each camera. The world origin is located as shown in Figure 5.8 on Page 98 where the $Z-$axis is vertical or perpendicular to the ground plane. The units are in millimeters for X, Y, Z and in degrees for ω, ϕ and κ. Each image is 640 $\times$ 480 pixels. The focal length is 4.3 mm. The format size is 3.5311 $\times$ 2.7446 mm.

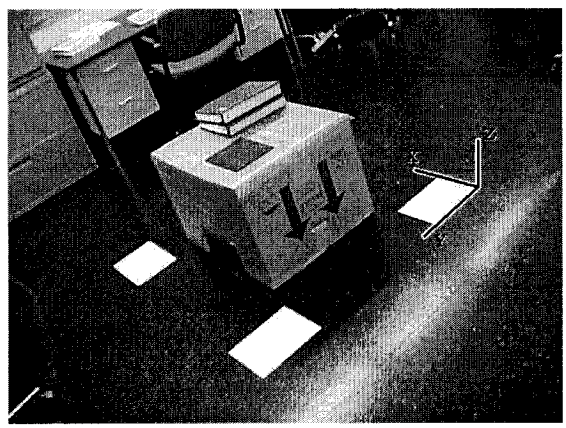


Figure 5.8: The "box" pair: The world coordinate system is shown where the world origin is located at a corner of a paper on the ground surface and where the $XY-$plane coincides with the ground surface and the $Z-$axis is vertical.

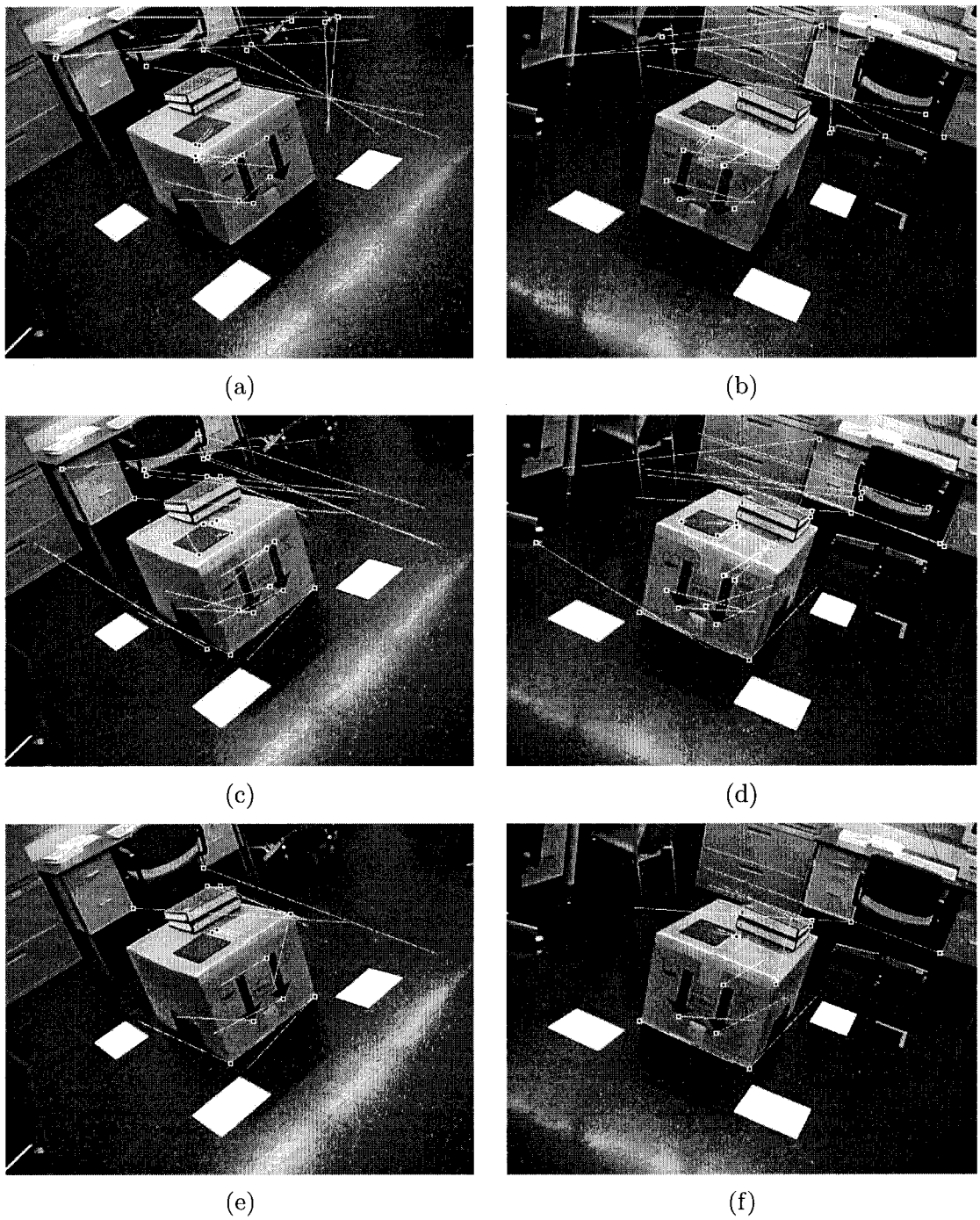


Figure 5.9: The "box" pair: Matches accepted are indicated by the dotted white squares. The correlation window size used is 21 × 21 pixels. Disparity vectors are shown in all images. (a) and (b) Non-homographic SAD correlation where the threshold, t_{SAD}, used is 1300. (c) and (d) Homographic SAD correlation where the threshold, t_{SAD}, used is 13000. (e) and (f) Homographic VNC correlation where the threshold, t_{VNC}, used is 0.75.

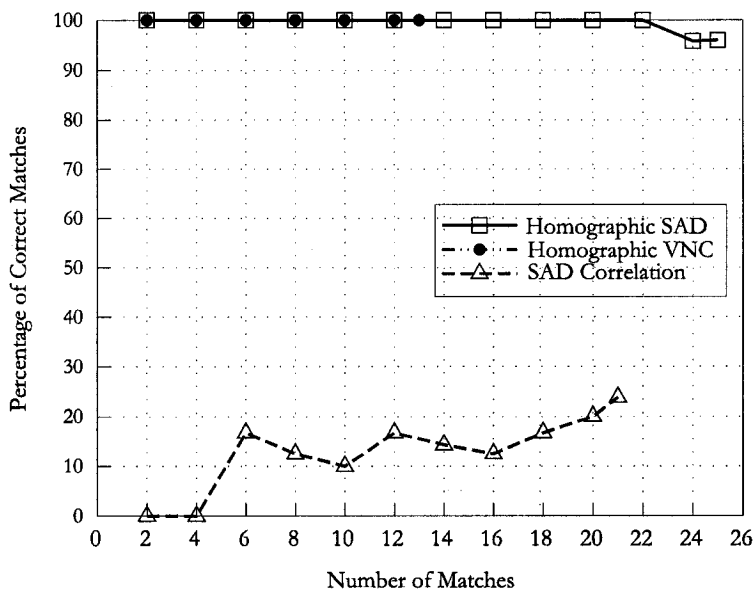
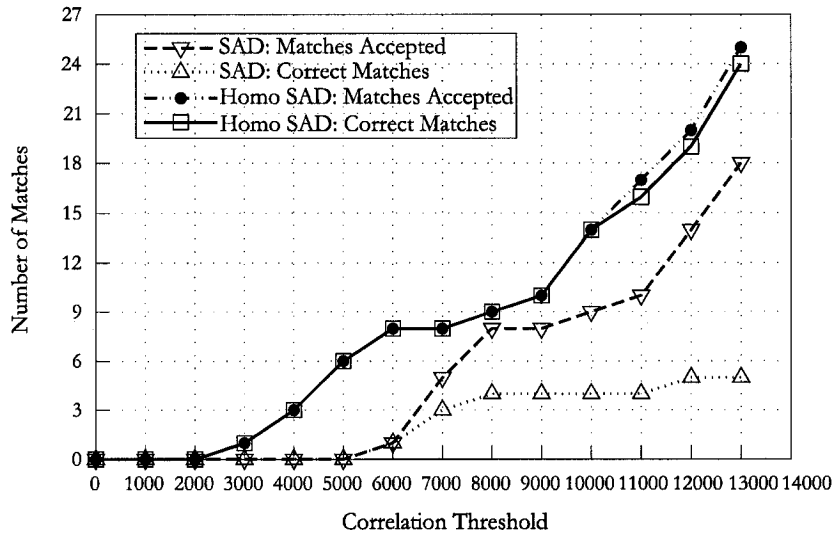


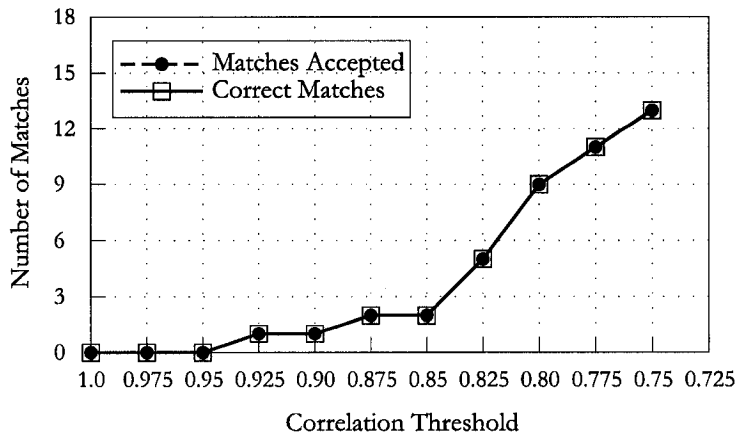
Figure 5.10: The "box" pair: The behavior of the percentage of correct matches versus the number of matches.

These graphs show the total number of matches corresponding to different correlation values. They are useful in determining the behavior of the number of matches versus different correlation thresholds. For example, this could be used to determine a threshold for a given number of matches to compare with another case with the same number of matches. Figure 5.11(a) shows that the number of matches increases in case of homographic SAD correlation over the non-homographic version for a given threshold, t_{SAD}, and at the same time the correlation value decreases for a given number of matches. This indicates that the correlated patches show more difference in the non-homographic case than the homographic case which, in turn, indicates that homographic transformation enhances the correlation results. Moreover, Figure 5.11(b) that shows the case of homographic VNC correlation indicates good correlation outcome as the descending gradual transition from -1.0 to 1.0 happens towards 1.0.

As a comparison, we added to each histogram the ideal case curve where the number of correct matches is 100.0%; where there are no mismatches. It is clear from Figure 5.11(a) that the real curve is closer to the ideal one in the homographic case than the non-homographic case of SAD correlation. This indicates that our proposed homographic SAD approach outperforms the non-homographic SAD correlation technique. Also, notice that the two curves representing the real case and that of correct matches coincide with each other in case of the homographic VNC correlation as shown in Figure 5.11(b) where the percentage of correct matches reaches 100.0% in this case.

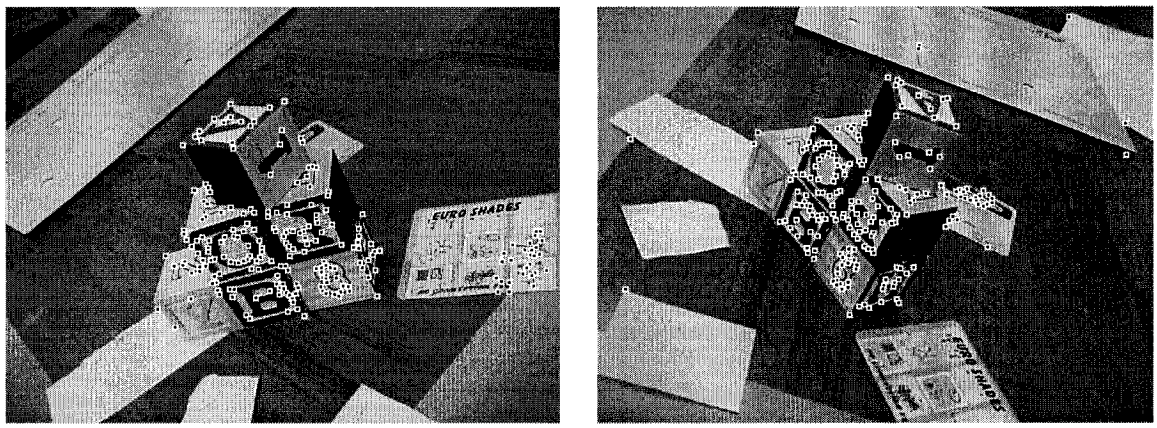


(a)



(b)

Figure 5.11: The "box" pair: (a) Homographic and non-homographic SAD correlation cumulative histograms. (b) Homographic VNC correlation cumulative histogram.



(a) (b)

Figure 5.12: A pair of images shown in Figure 5.12 on Page 102. (a) Left image. (b) Right image. The junctions that successfully passed the algorithm mentioned in Section 4.9 are shown as dotted white squares. The parameters used to detect junctions are $\sigma = 1.2$, $t_{\mathcal{B}} = 8$, $\lambda = 10$ and $s_J = 0.5$. We call these images the "cubes" pair.

Image	X	Y	Z	ω	ϕ	κ
(a)	-114.7469	398.9921	355.3564	-52.2783	-17.0877	162.6817
(b)	-206.6808	372.5378	351.6793	-49.5184	-27.0316	-132.0044

Table 5.2: The "cubes" pair: The parameters for each camera. The world origin is located as shown in Figure 5.13 on Page 103. The $Z-$axis is vertical or perpendicular to the ground plane. The units are in millimeters for X, Y, Z and in degrees for ω, ϕ and κ. Each image is 640 $\times$ 480 pixels. The focal length is 6.0 mm. The format size is 3.5311 $\times$ 2.7446 mm.

Another example is shown in Figure 5.12 on Page 102 where a pair of wide baseline color images is used. The junctions detected by the JUDOCA operator, shown in Figure 3.10 on Page 49, were gone through the algorithm mentioned in Section 4.9. This results in two sets of candidate junctions as it was shown in Figures 4.15(c) and (d). Those images are reproduced here in Figure 5.12. The camera parameters for the "cubes" pair are shown in Table 5.2 on Page 102. The world origin is located as shown in Figure 5.13 on Page 103 where the $XY-$plane coincides with the ground surface and the $Z-$axis is vertical.

As done in the previous example, and to compare between our approach and others, we applied the correlation techniques; non-homographic SAD, homographic SAD and homographic VNC, mentioned in Equations (5.1), (5.34) and (5.35) respectively. In this case, three color channels are used. The overall correlation result is computed as the average of the values cal-

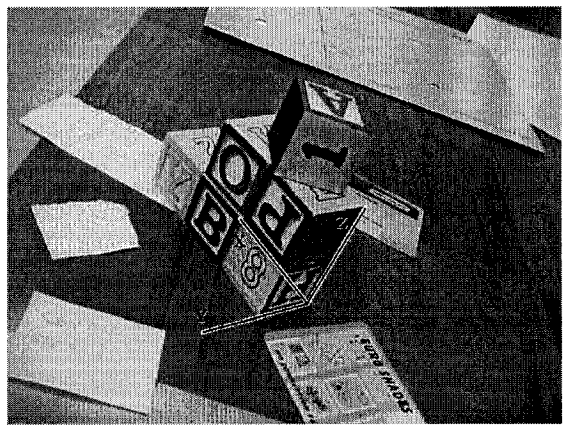


Figure 5.13: The "cubes" pair: The world origin is shown at the corner of a bottom cube where the $XY-$plane coincides with the ground surface and the $Z-$axis is vertical.

culated through the above-mentioned equations for each channel separately. Using a window size of 11 $\times$ 11 pixels for all cases, and correlation thresholds, t_{SAD} and t_{VNC}, of 4000 for non-homographic SAD, 2750 for homographic SAD and 0.75 for homographic VNC correlation, the results are shown in Figure 5.14 on Page 104. The percentage of correct matches are 55.0%, 100.0% and 100.0% for non-homographic SAD, homographic SAD and homographic VNC correlations respectively.

Figure 5.15 on Page 105 shows the overall performance for the non-homographic SAD, homographic SAD and homographic VNC. This graph shows that our proposed homographic correlation approach outperforms the non-homographic SAD correlation technique. It improves the outcome of the SAD correlation by a factor of 181.8%. This figure is yet another example that in case of wide baseline matching, our plane homographic approach results in a very good outcome that is preferred to the non-homographic SAD correlation technique.

The cumulative histograms for non-homographic SAD correlation, homographic SAD correlation and homographic VNC correlation are shown in Figures 5.16(a) and (b) on Page 106. As expected, Figure 5.16(a) shows that non-homographic SAD version overestimates the correlation value required to obtain a given number of matches. This can be explained as correlated pixels are quite different in their intensity as changes in illumination and severe perspective deformations result in large correlation value for the non-homographic SAD case. Similarly, for a given correlation value, the number of matches accepted in case of homographic SAD correlation is much greater than in case of non-homographic counterpart. For example, a threshold, t_{SAD}, of 2750 results in 19 matches in the homographic case (with success rate of 100.0%) while it results in 2 matches in the non-homographic case (with success

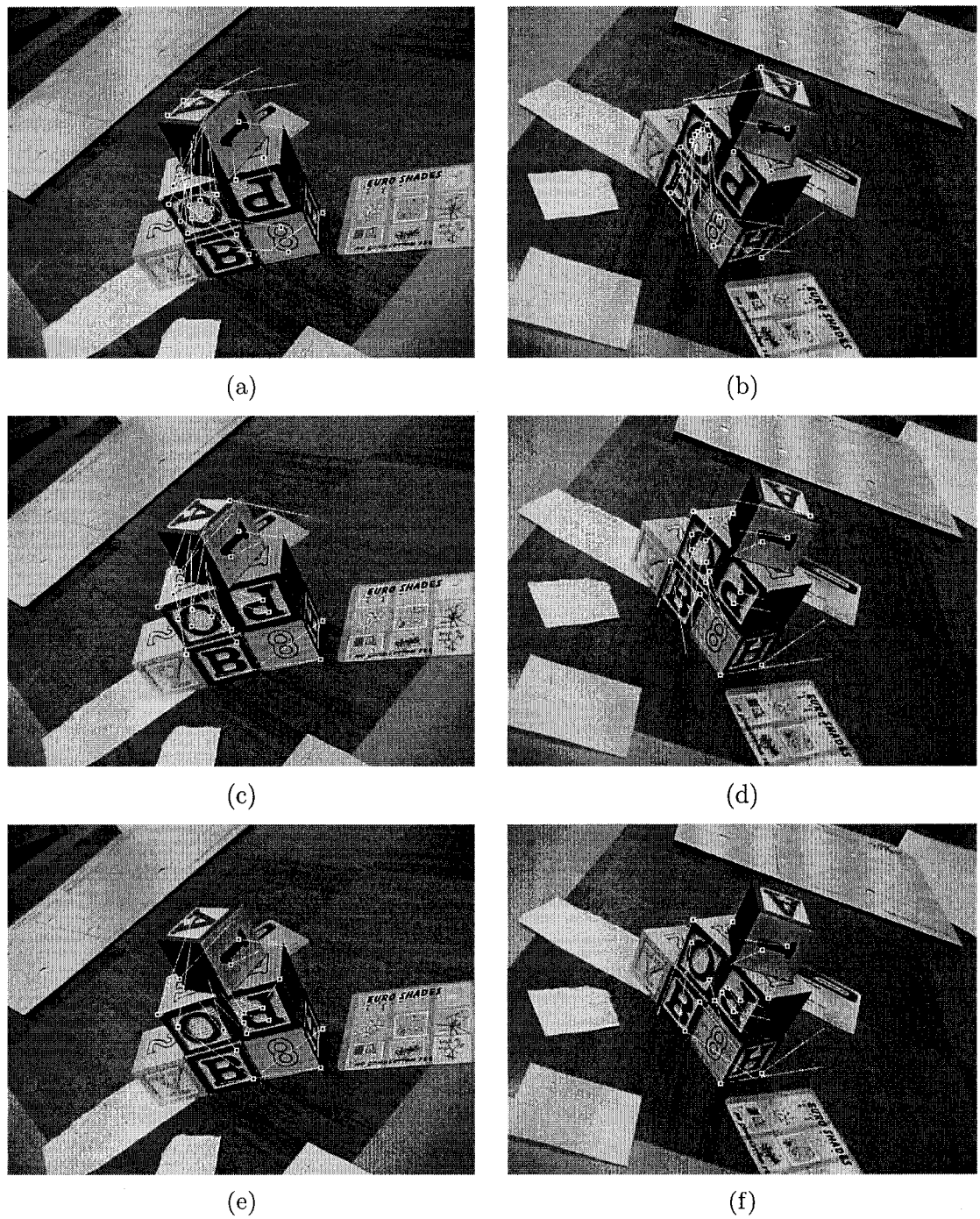


Figure 5.14: The "cubes" pair: Matches accepted are indicated by the dotted white squares. The correlation window size used is 11 × 11 pixels. Disparity vectors are shown in all images. (a) and (b) Non-homographic SAD correlation where the threshold, t_{SAD}, used is 4000. (c) and (d) Homographic SAD correlation where the threshold, t_{SAD}, used is 2750. (e) and (f) Homographic VNC correlation where the threshold, t_{VNC}, used is 0.75.

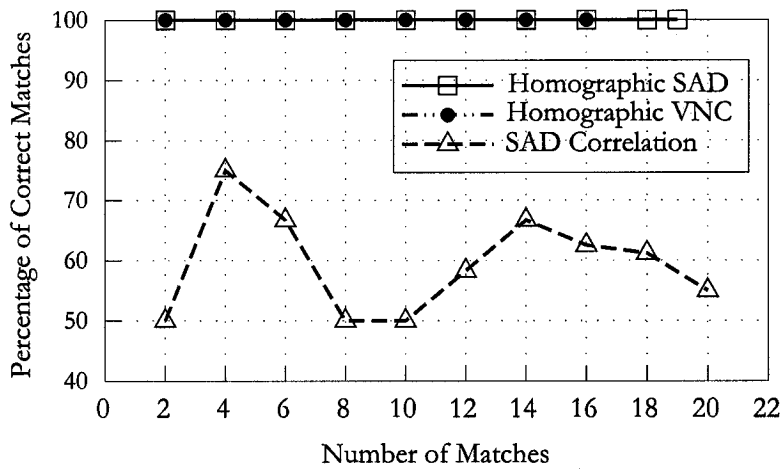
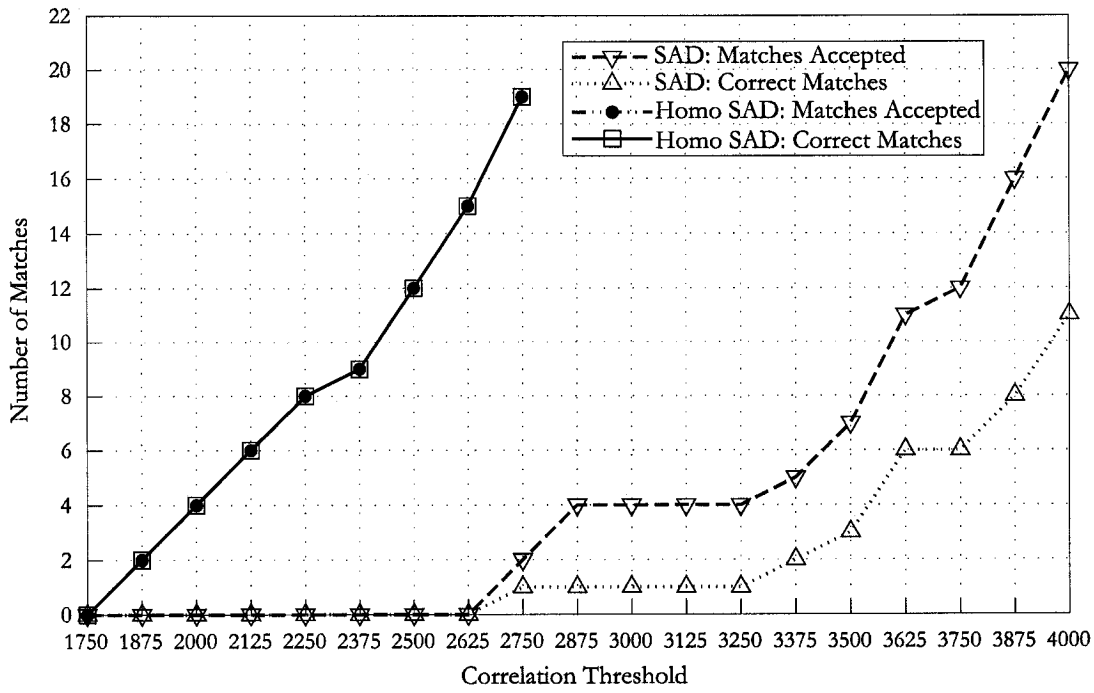


Figure 5.15: The "cubes" pair: The behavior of the percentage of correct matches versus the number of matches. Notice that the correlation thresholds are selected so that the number of matches obtained are close to each other.

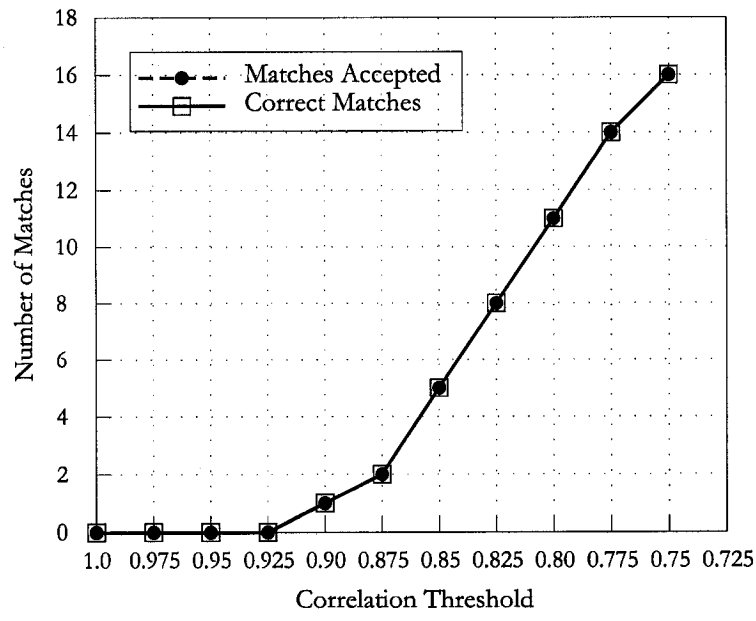
rate of 50.0%). In both homographic cases (SAD and VNC) the real curve and the ideal curve of correct matches coincide with each other where the percentage of correct matches reaches 100.0% in both cases.

In order to show the effect of the size of the correlation window and the correlation threshold as well on the outcome, we tested their impact on the percentage of correct matches. The results are listed in Table 5.3 on Page 107. Figure 5.17 on Page 107 shows the behavior of correct matches as affected by those two parameters. It is evident from the figure that the larger the correlation window and the correlation value, the better the results.

Figure 5.18 on Page 109 shows the results where a large correlation window of 17 × 17 pixels was used. The thresholds, t_{SAD} and t_{VNC} used are 7000 in the case of homographic SAD correlation and 0.7 for the homographic VNC correlation case. The percentages of correct matches were 100.0% in both cases. The same threshold used for homographic SAD correlation is used again with the same window size to check the effect on the non-homographic SAD correlation case but no matches were obtained. This happened due to two reasons. The first reason is that the threshold is too small comparing to the large difference in intensity that exists between two patches as we work on the image plane rather than the actual 3D plane; i.e., more perspective deformation exists. The second reason is that by definition, stable marriage requires that mutual acceptance must present in order to declare a good match. In other words, the difference in intensity must be lower than the threshold for both directions; i.e., left-right and right-left. This means that satisfying the threshold for only one direction



(a)



(b)

Figure 5.16: The "cubes" pair: (a) Homographic and non-homographic SAD correlation cumulative histograms. Notice that the thresholds used are different in this example. (b) Homographic VNC correlation cumulative histogram.

Window size	t_{VNC}	Total number	Number correct	% correct
11 × 11	0.50	44	40	90.9
11 × 11	0.55	40	37	92.5
11 × 11	0.60	35	33	94.3
11 × 11	0.65	27	25	92.6
11 × 11	0.70	23	21	91.3
11 × 11	0.75	16	16	100.0
13 × 13	0.50	42	36	85.7
13 × 13	0.55	38	34	89.5
13 × 13	0.60	35	32	91.4
13 × 13	0.65	28	27	96.4
13 × 13	0.70	23	22	91.7
13 × 13	0.75	17	17	100.0
15 × 15	0.50	40	36	90.0
15 × 15	0.55	34	33	97.1
15 × 15	0.60	33	33	100.0
15 × 15	0.65	28	28	100.0
15 × 15	0.70	20	20	100.0
15 × 15	0.75	18	18	100.0
17 × 17	0.50	38	36	94.7
17 × 17	0.55	35	35	100.0
17 × 17	0.60	33	33	100.0
17 × 17	0.65	26	26	100.0
17 × 17	0.70	20	20	100.0
17 × 17	0.75	18	18	100.0

Table 5.3: The effect of the correlation window size and the correlation threshold on the overall results of obstacle features matching.

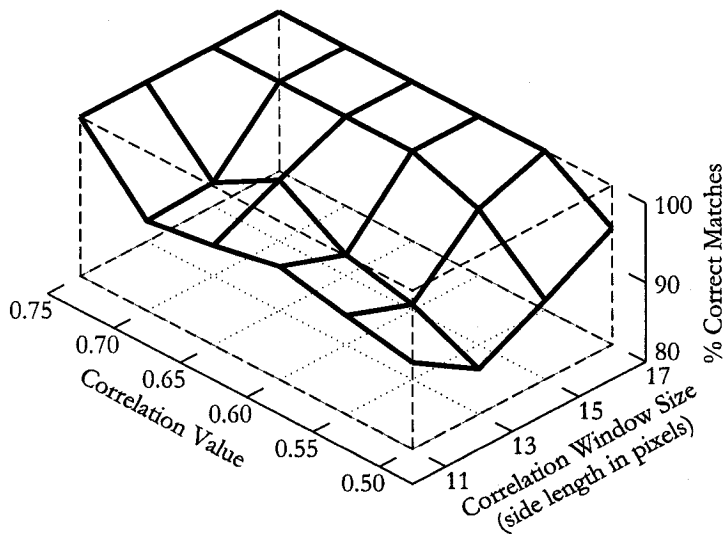


Figure 5.17: The behavior of correct matches as affected by the size of correlation window and the correlation threshold.

is not sufficient. This happened in this example to a mismatch pair.

Another comparison may be performed that is based on a vector of invariants of 8 components presented in [GMP98, MGD98]. This vector is represented as:

$$\mathbf{V}_{col}(\dot{\mathbf{m}}, \sigma) = \begin{bmatrix} R \\ |\nabla R|^2 \\ G \\ |\nabla G|^2 \\ B \\ |\nabla B|^2 \\ \nabla R.\nabla G \\ \nabla R.\nabla B \end{bmatrix} \tag{5.36}$$

where

- σ is the variance used to compute the gradient magnitude;
- $\dot{\mathbf{m}}$ is point under consideration;
- R, G and B are the three color channels; and
- $|\nabla R|$, $|\nabla G|$ and $|\nabla B|$ are the gradient magnitudes for the three channels.

The gradient magnitude images are computed through Gaussian filters as done in Chapter 3. Notice that the color intensities and the gradient magnitudes have different range of legitimate values. Thus, the values need to be resized prior to comparing if the distance is to be computed between two vectors. The matching process proceeds by comparing the values of a vector of invariant for one junction in one image with all vectors of junctions in the epipolar strip in the other image and choosing the junction with the closest values.

For simplicity, instead of resizing the values of the gradient magnitudes, we compare the values of a certain term in two vectors and if the difference is under a threshold, this pair receives one vote. we repeat this procedure with the other seven entries in vectors. This means that the maximum vote should be eight for a given pair. A pair with a vote value under a threshold should be rejected. This is to be repeated with all the candidates in the epipolar strip and pairs with maximum votes are considered matches.

We applied the above-mentioned idea to our example with $\sigma = 1.0$ where the error margins and window size are the same as the above values. (The error margins were ± 10 mm, ± 10 mm, ± 10 mm, $\pm 0.8°$, $\pm 0.8°$ and $\pm 0.8°$ for ΔX, ΔY, ΔZ, $\Delta\omega$, $\Delta\phi$ and $\Delta\kappa$ respectively.) Figure 5.19 on Page 110 shows the results of this approach where the success rate is 45.5%. The minimum threshold of votes to be accepted as a match was 3.

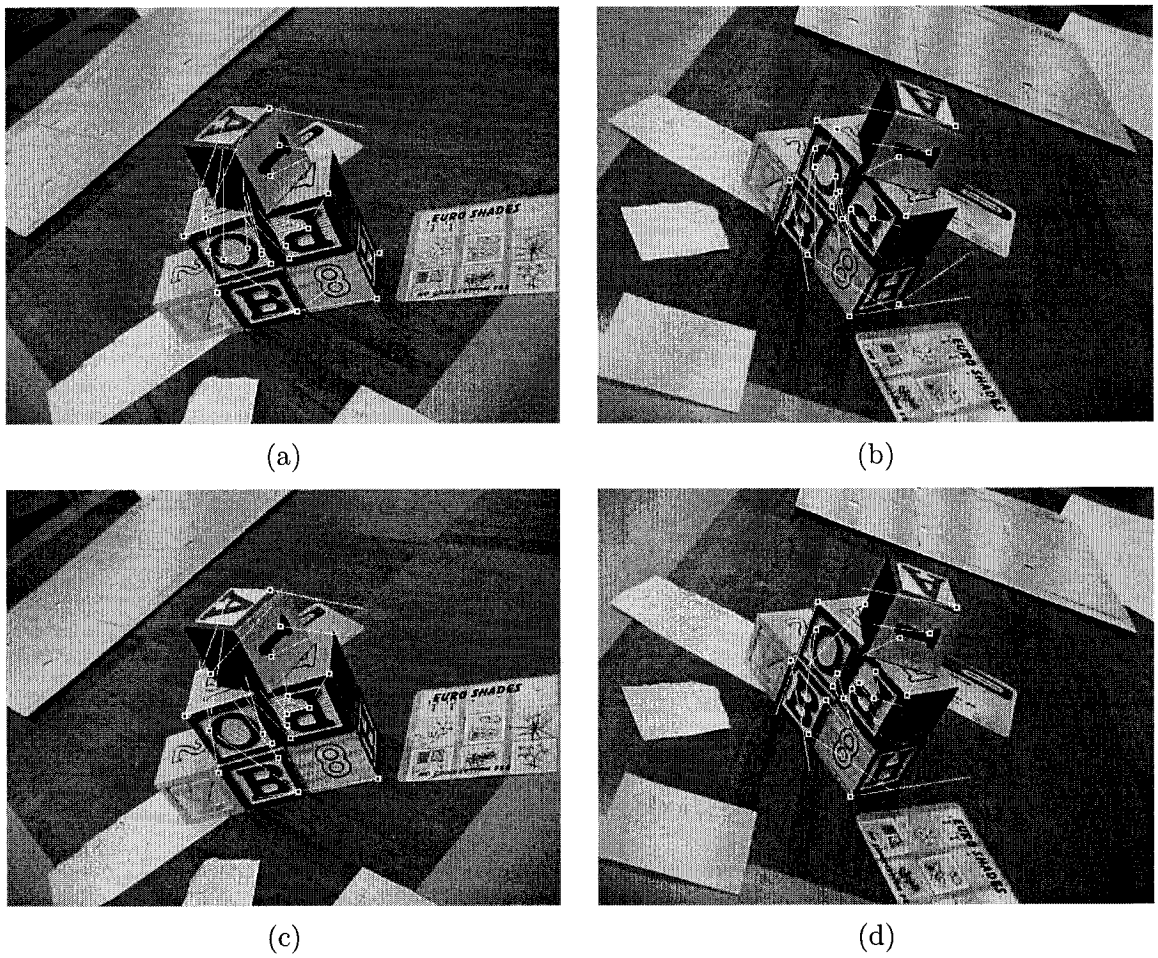
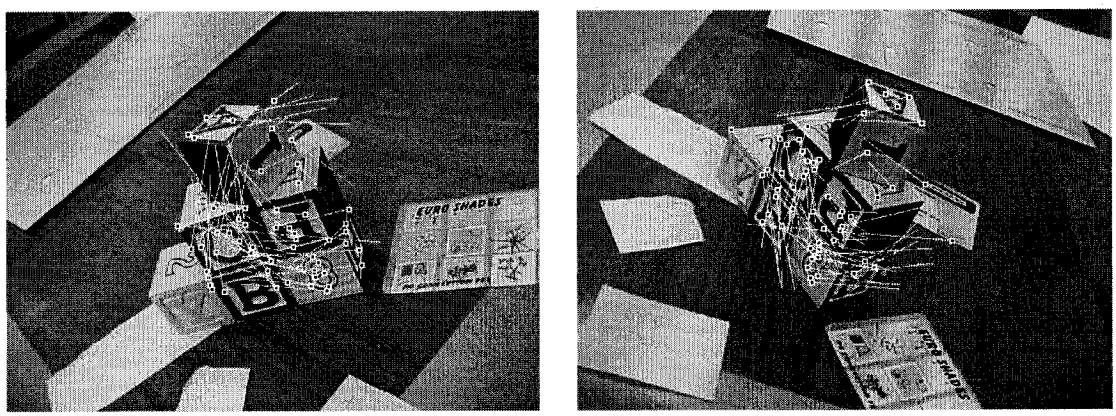


Figure 5.18: The "cubes" pair: Matches accepted are indicated by the dotted white squares. The correlation window size used is 17 × 17 pixels. Disparity vectors are shown in all images. (a) and (b) Homographic SAD correlation where the threshold, t_{SAD}, used is 7000. (c) and (d) Homographic VNC correlation where the threshold, t_{VNC}, used is 0.7. Non-homographic SAD correlation resulted in no matching at all.



(a) (b)

Figure 5.19: The "cubes" pair: Applying the vector of invariant approach results in a success rate of 45.5%.

5.5 Epipolar Constraint Enforcement

At this point, we may end up with some mismatches. In order to enhance the matching results, we use RANSAC scheme to recompute a more accurate fundamental matrix, F, and reject candidates that do not agree with that matrix. The steps are as follows [RW00, VL01]:

1. Seven points are randomly selected.
2. The fundamental matrix is computed from this set where:

$$\mathbf{m}'^{T}\mathbf{F}\mathbf{m} = 0 \tag{5.37}$$

3. Pairs that satisfy the computed F are selected to the support set. The pair is satisfying the matrix F if the point in one image lies within some threshold (e.g., 1 pixel) from the computed epipolar line. That is:

$$d(\mathbf{F}\mathbf{m}, \mathbf{m}') < t_{\mathrm{F}} \tag{5.38}$$

   where

   - $d()$ represents the distance from the point $\dot{\mathbf{m}}'$ to the epipolar line $\mathbf{l}'$; and
   - t_{F} is a threshold.

   If (5.38) results in a true condition, then the pair ($\dot{\mathbf{m}}$, $\dot{\mathbf{m}}'$) agrees with the matrix F.

4. The preceding steps are repeated and the largest support set along with its fundamental matrix, F, are returned.

As an example, the above-mentioned method is applied to the homographic SAD correlation results mentioned in Section 5.4 for the "box" pair. After enforcing the epipolar constraint through a threshold, t_{F}, of 3.0, the results are shown in Figures 5.20(a) and (b) on Page 112. The overall percentages increased from 96.0% to 100.0% in this case. Finally, Figure 5.20(c) shows the behavior of the percentage of correct matches versus the number of matches.

5.6 Parameters and Thresholds

There are some parameters and thresholds controlling the process of wide baseline matching on arbitrary planes. The parameters include the intrinsic and extrinsic parameters of the camera as detected by the available sensors in addition to the error margin of each of them.

Another parameter is the correlation window size. As in the last chapter, our experiments showed that large correlation window generally produce better results than smaller windows.

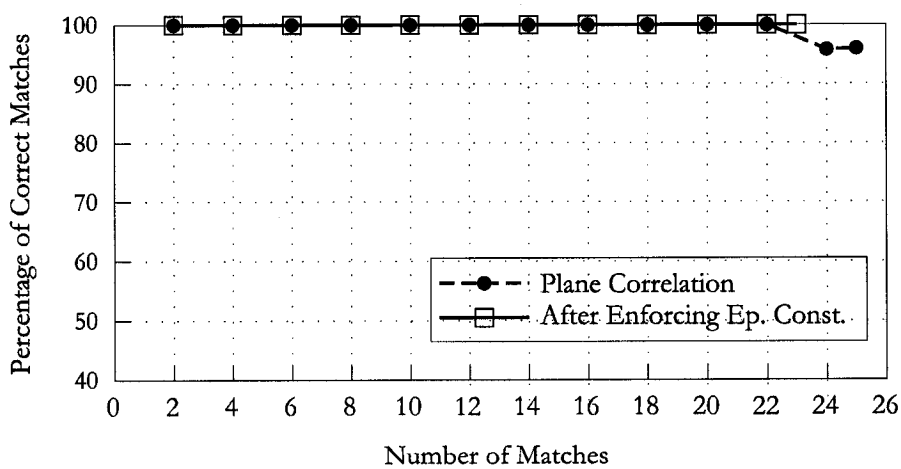
The correlation thresholds t_{SAD} and t_{VNC} determine how close two patches are through SAD and VNC correlations respectively. The threshold t_{SAD} does not have a predetermined range but its range depends on how large the correlation window is. Assume that the correlation window side is $N+1$ pixels, then the legitimate range resides in the interval $[0, 255(N+1)^2]$ with 0 represents the perfect case and $255(N+1)^2$ represents the worst case. In case of VNC correlations, t_{VNC} has an interval [-1.0, 1.0] with -1.0 represents the worst case and 1.0 represents the perfect case. From our experiments, a value around 0.7 would result in a good outcome.

As the threshold t_{H} used in the last chapter with RANSAC to calculate more accurate homography matrices, the threshold t_{F} in this chapter is used with RANSAC to represent the deviation of a corresponding point from the epipolar line. Finally, Section A.4 summarizes the role of each of the above-mentioned parameters and thresholds.

5.7 Summary

In the preceding chapter, we discussed different formulations of the matching, or point correspondence, problem. The techniques developed to solve this problem can be divided into two main categories. These categories are area-based matching and feature-based matching. Another formulation of the problem is possible that is based on the length of the baseline between cameras and on whether the calibration information is available or not. Our focus in this thesis is concerned with the wide baseline case where camera parameters are known approximately.

(a) (b)



(c)

Figure 5.20: The "box" pair: Results of matching through homographic SAD correlation after enforcing the epipolar constraint where matches accepted are indicated by the dotted white squares. Disparity vectors are shown on both left and right image, (a) and (b). (c) The behavior of the percentage of correct matches versus the number of matches after enforcing the epipolar constraint.

Our approach to perform matching could be split into two distinctive phases. The first phase was to find matches on a specific plane. The second phase was to find other matches on arbitrary planes. Chapter 4 proposed a homographic correlation technique to perform the first phase; i.e., to find matches on the ground plane. Through the match set obtained from that phase, candidates for the second phase could be identified. In Chapter 5, we proposed a novel technique to solve the wide baseline matching problem through homographic transformation. Unlike that of Chapter 4, the algorithm in this chapter was designed to work on arbitrary planar pathes rather than on a specific plane. We utilized the advantage of knowing the camera parameters approximately. We used the attributes of the junctions detected by the JUDOCA operator that was discussed in Chapter 3. We suggested two versions of the proposed solution based on SAD and VNC correlations. We showed by examples and analysis that both of them outperform the traditional SAD correlation technique. The output match set will be used in Chapter 6 to reconstruct points in 3D space. The results of this chapter have been published in [Eli04].

Chapter 6

Volumetric Representation of Obstacles

Up to this point, we end up with a correspondence match set that may be transformed into a set of 3D points. Our goal is to use this set of space points to localize obstacles so that a robot may navigate safely in the environment. A box that bounds every obstacle may be sufficient for the purpose of robot navigation. Indeed, a bounding box should enclose the whole volume of the obstacle. On the other hand, the whole volume of the bounding box might not be occupied leaving sub-volume parts that could be added to the rest of the environment. Hence, for some applications, a more accurate volumetric representation may be requested. This chapter discusses both issues; i.e., representing obstacles as bounding boxes and obtaining a more accurate volumetric representation of the obstacle.

One way to construct an object in space could be by Delaunay triangulating[1] the points in one image; e.g., the left image and determining the corresponding triangles in the other image. This is followed by locating, in 3D space, the locations of points that surround every triangle and hence determining the projection of every triangle in the synthesized resulting image. Finally, mapping is performed between a source triangle in an original image (e.g., the left image) to the destination triangle in the synthesized image through a 3×3 affine transformation matrix. Indeed, an affine transformation matrix has 6 degrees of freedom and can be fully defined through 3 pairs of points. This step fills in the gaps among feature points.

However, the above mentioned method may be affected dramatically by the success of the feature detection and the matching procedures. For example, one mismatch may result in a

[1]Delaunay triangulation of a point set is defined as the triangulation of this set with the property that no point in the point set falls in the interior of the circumcircle where a circumcircle is the circle that passes through all three vertices. This is not to be confused with triangulation that we will use later in this chapter to locate points in space. This term refers to the fact that a pair of corresponding points with their corresponding 3D point in space form a triangle.

shapeless structure in the synthesized resulting image. Also, failing to detect some feature points especially at the boundaries of the obstacle may result in an incomplete representation of the obstacle. Hence, we need to find a procedure that is more tolerant to errors that may exist as a result of the previous procedures; i.e., feature detection and matching.

We developed an algorithm that we claim to be better than the above mentioned method. The main steps of this algorithm can be summarized in the following points:

1. Using the match set we have, we estimate the positions of the points in space (Section 6.1).
2. We cluster points in space into groups where each group could possibly belong to an object in space. This is done according to some proximity criteria. Thus, we end up with bounding boxes surrounding obstacles (Section 6.2). This step localizes the obstacles and can be used as a final output by itself [HEL03, HEL04]. If more accurate volumetric representation for the obstacle is required, we proceed with the next step.
3. Using a voxel occupancy scheme and a number of images, we can volumetrically represent obstacles in space (Section 6.4).

In the next sections, we will explain each of these steps.

6.1 3D Estimation of Points

Every two corresponding points are contributing to two rays that should intersect precisely in space at the location of the point as depicted in Figure 2.7 on Page 31. At this point, we utilize the match set obtained in the previous chapter along with the available camera parameters to triangulate[2] the points in order to estimate their locations in space. However, due to the inaccuracy of camera parameters, the two rays are likely to be skew. Then, the 3D point can be estimated as the middle point of the perpendicular to both rays as shown in Figure 2.8 on Page 33. Hence, using Equation (2.77), the 3D point can be estimated as:

$$\hat{\dot{\mathbf{M}}} = \left([\mathtt{I} - \dot{\mathbf{M}}_\infty \dot{\mathbf{M}}_\infty^T] + [\mathtt{I} - \dot{\mathbf{M}}'_\infty \dot{\mathbf{M}}'^T_\infty]\right)^{-1} \left(\dot{\mathbf{C}} + \dot{\mathbf{C}}' - [\dot{\mathbf{C}}^T \dot{\mathbf{M}}_\infty]\dot{\mathbf{M}}_\infty - [\dot{\mathbf{C}}'^T \dot{\mathbf{M}}'_\infty]\dot{\mathbf{M}}'_\infty\right) \quad (6.1)$$

where

- $\hat{\dot{\mathbf{M}}}$ is a 3D vector representing the estimated inhomogeneous coordinates of the point in space;
- $\mathtt{I}$ is a 3×3 identity matrix;
- $\dot{\mathbf{C}}$ and $\dot{\mathbf{C}}'$ are 3D vectors representing the inhomogeneous coordinates of the optical centers of both cameras; and

[2]Triangulation here means locating the position of the point in 3D space through its two corresponding projected pair.

- $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are 3D vectors representing the inhomogeneous coordinates of the intersection of the rays with the plane at infinity. $\dot{\mathbf{M}}_\infty$ and $\dot{\mathbf{M}}'_\infty$ are normalized to unit magnitude.

Consequently, for every match pair, we obtain a 3D point. The 3D point set is then clustered. Through 3D point clustering, outliers may be detected and removed as they will likely result in isolated points in space. Moreover, if isolated true matches that belong to the ground leaked off the step mentioned in Section 4.9, clustering may discard these cases. As a result of point clustering, a 3D bounding box for every obstacle can be obtained which may serve as a localization information. As mentioned above, for some applications, a bounding box may be sufficient information for robot navigation. For others, a more accurate occupancy model may be required.

The next section handles the concept of hierarchical processing, which we use to perform point clustering in space and discusses the structure that we propose to cluster points.

6.2 Hierarchical Clustering

A *hierarchy* is defined as a set of layers; each of them corresponds to a given level of abstraction of the information being processed [JR94, Eli99a]. From the Computer Vision point of view, the bottom layer of the hierarchy, which is called the *base*, is usually fed with the input image. Gathering more global information is done as we proceed to the top of the hierarchy until reaching the highest layer; i.e., the top layer, which is called the *apex*. Information flows up and down through the hierarchy and is transformed and combined between layers. Because there are two directions of the flow of information, there are two different kind of processes. These are *bottom-up* and *top-down* processes. A bottom-up process, which we address below, is a hierarchical propagation of information from bottom to the top of the hierarchy. This type of processes allows the transformation of local and distributed sets of data, into global data.

Based on the concept of hierarchical processing, we propose a hierarchical structure to cluster points in space [Eli03a]. However, unlike what is usually done in Computer Vision research, the base of our hierarchical structure is not an image but a number of space points. The rest of this section presents the hierarchical structure we propose and shows experiments applied to real-world scenes.

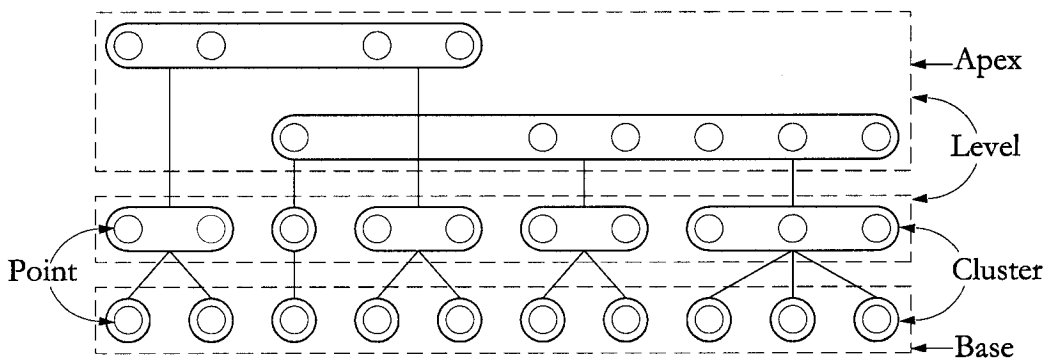


Figure 6.1: The hierarchical structure.

6.2.1 Initialization

Each level of the structure is represented as a graph. At the base, the graph consists of a number of cluster nodes where each node shares an edge[3] with every other node and where every node contains only one space point. At the upper levels, a cluster node may contain more points while the number of clusters is reduced at that level comparing to its predecessor. At the apex, the information will be at the highest level of abstraction. In our case, the set of points contained in each cluster at the apex represents one object in space. In [EL99, JM91], such a set is called the *receptive field* of a node and is defined as the set of all descendants located at the base of the hierarchical structure. An example of this hierarchical structure is shown in Figure 6.1.

6.2.2 Hierarchical Rules

As in many hierarchical structures, there are some rules that control the creation and the shape of the next level and how the inter-level links are established. These rules are:

- Two neighbors may both survive at the next level if and only if some binary variable is set to zero during the decimation process. Note that this is different from the case of the adaptive and disparity pyramids [JM91, EL99];
- For each nonsurviving node, there exists at least one surviving node in its neighborhood. This is true in the case of the adaptive and disparity pyramids.

6.2.3 Building a New Level

Suppose that the set of clusters at a given level, $\mathcal{L}_{(i)}$, is:

$$\mathcal{L}_{(i)} = \{\mathcal{C}_{(i,1)}, \mathcal{C}_{(i,2)}, ..., \mathcal{C}_{(i,n)}\} \tag{6.2}$$

[3]Edge here is used in graph terms. This is not to be confused with the term "edge" that is used frequently in Image Processing to indicate sharp transitions in intensity levels.

where

- i is the level number;
- n is the number of clusters at a level, $\mathcal{L}_{(i)}$; and
- $\mathcal{C}_{(i,j)}$, where $j \in \{1, ..., n\}$, is a cluster consisting of a number of space points.

Furthermore, a cluster $\mathcal{C}_{(i,j)}$ is defined as:

$$\mathcal{C}_{(i,j)} = \{\dot{\mathbf{M}}_{(i,j,1)}, \dot{\mathbf{M}}_{(i,j,2)}, ..., \dot{\mathbf{M}}_{(i,j,m)}\} \tag{6.3}$$

where

- i is the level number;
- j is the cluster number, $j \in \{1, ..., n\}$;
- m is the number of points in cluster; and
- $\dot{\mathbf{M}}$ is a vector whose length depends on the dimension of space (3 in our case of 3D space).

Then, for every two clusters, $\mathcal{C}_{(i,j)}$ and $\mathcal{C}_{(i,k)}$, at level $\mathcal{L}_i$, we reset a binary variable q to 0 and perform the following check among the points contained in these clusters. This is by checking:

$$||\dot{\mathbf{M}}_{(i,j,a)} - \dot{\mathbf{M}}_{(i,k,b)}|| < t_{\dot{\mathbf{M}}} \tag{6.4}$$

where

- i is the level number;
- j and k are the cluster numbers;
- a and b are the point numbers;
- $||.||$ represents the norm of the difference between the two vectors where Euclidean or Manhattan metric may be used; and
- $t_{\dot{\mathbf{M}}}$ is a threshold.

If (6.4) results in a true condition, then we break the search for the current clusters and set the variable q to 1; otherwise, q remains 0. At this step, we have the following possibilities:

- If $q = 1$ and one cluster; e.g., j, has a parent while the other cluster, k, does not, then we link the cluster k to the parent of cluster j.
- If $q = 1$ and neither cluster j nor cluster k has a parent, then a new node at the next level is to be created and linked to both clusters.
- If $q = 1$ and each cluster (j and k) has a different parent created previously, then we link both to one parent and delete the other.

- If $q = 0$ and one cluster; e.g., j, has a parent while the other cluster, k, does not, then a new node is to be created at the next level and linked to cluster k.
- If $q = 0$ and neither cluster j nor cluster k has a parent, then a new node at the next level is to be created for each one and linked to the corresponding cluster.
- The last possibility happens when $q = 0$ and each cluster (j and k) has a different parent created previously. In this case, no action should be taken.

6.2.4 Roots at the Apex

The procedure explained in Section 6.2.3 is repeated iteratively until all clusters become roots at the apex. This is a common property to this structure as well as the adaptive and disparity pyramids [JM91, EL99]. In our case, a root is defined as a cluster that is at a distance $\geq t_{\dot{M}}$, where $t_{\dot{M}}$ is a threshold mentioned in (6.4), from any other cluster. In this case; i.e., at the apex, one cluster should exist for each object in space.

6.2.5 Experimental Results

Consider the images (b) and (f) of the scene shown in Figure 6.2 on Page 120. The junctions were detected using the JUDOCA operator, discussed in Chapter 3. These were shown in Figure 3.8 on Page 47. The junctions were filtered through the algorithm explained in Chapter 4. This left two sets of candidate junctions as shown in Figure 4.10 on Page 74. The matching algorithm described in Chapter 5 is applied to those sets. Through that algorithm, we ended up with a set of match pairs. These were shown in Figures 5.9(c) and (d) on Page 99. We used the results of homographic SAD correlation before enforcing the epipolar constraint to show that the clustering algorithm we propose may exclude points that correspond to mismatches. Following the reconstruction of points in 3D space, the 3D point clustering algorithm mentioned above was applied to the set of 3D points with a threshold, $t_{\dot{M}}$, of 600 mm. The clustering result is shown in Figure 6.3. Notice that the bounding boxes overlap in the side views as the distances in the direction of projection are equal or exceed the threshold, $t_{\dot{M}}$, mentioned in (6.4). The roots of all clusters were detected at the second level of the structure.

Figure 6.4 on Page 123 shows another scene. Consider the pair (e) and (f). The same procedures mentioned above were applied again to this example. The junctions detected were shown in Figure 3.10 on Page 49. The junctions that survived the procedure of ground feature exclusion are shown in Figure 4.15 on Page 80. Following the matching algorithm, the match set used is shown in Figure 5.18 on Page 109. A threshold, $t_{\dot{M}}$, of 60 mm is used to cluster the set of 3D points obtained by triangulation. The result is shown in Figure 6.5(a) on Page 124. It is clear that the matching process for that pair of images does not result in a sufficient

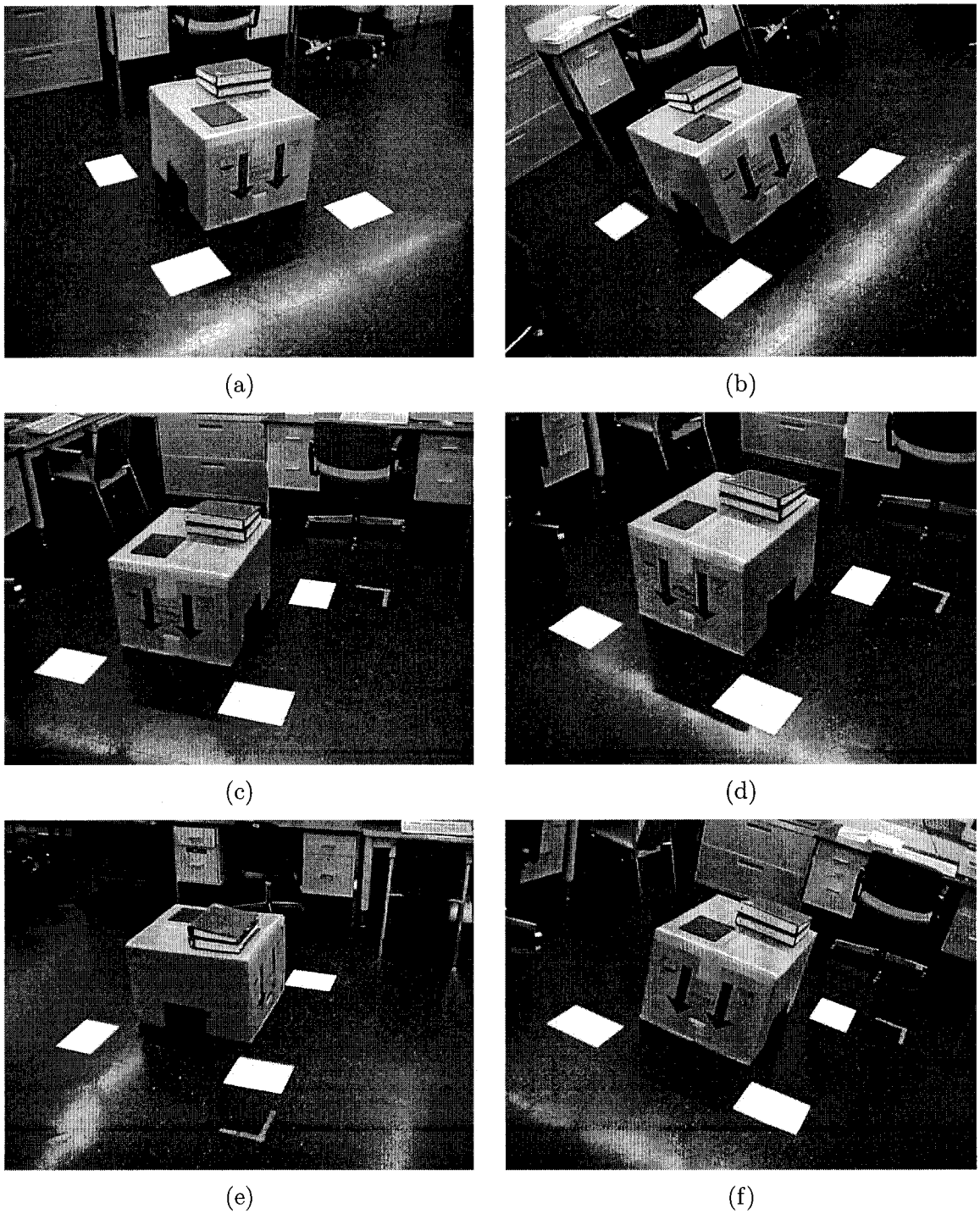


Figure 6.2: Images of the scene.

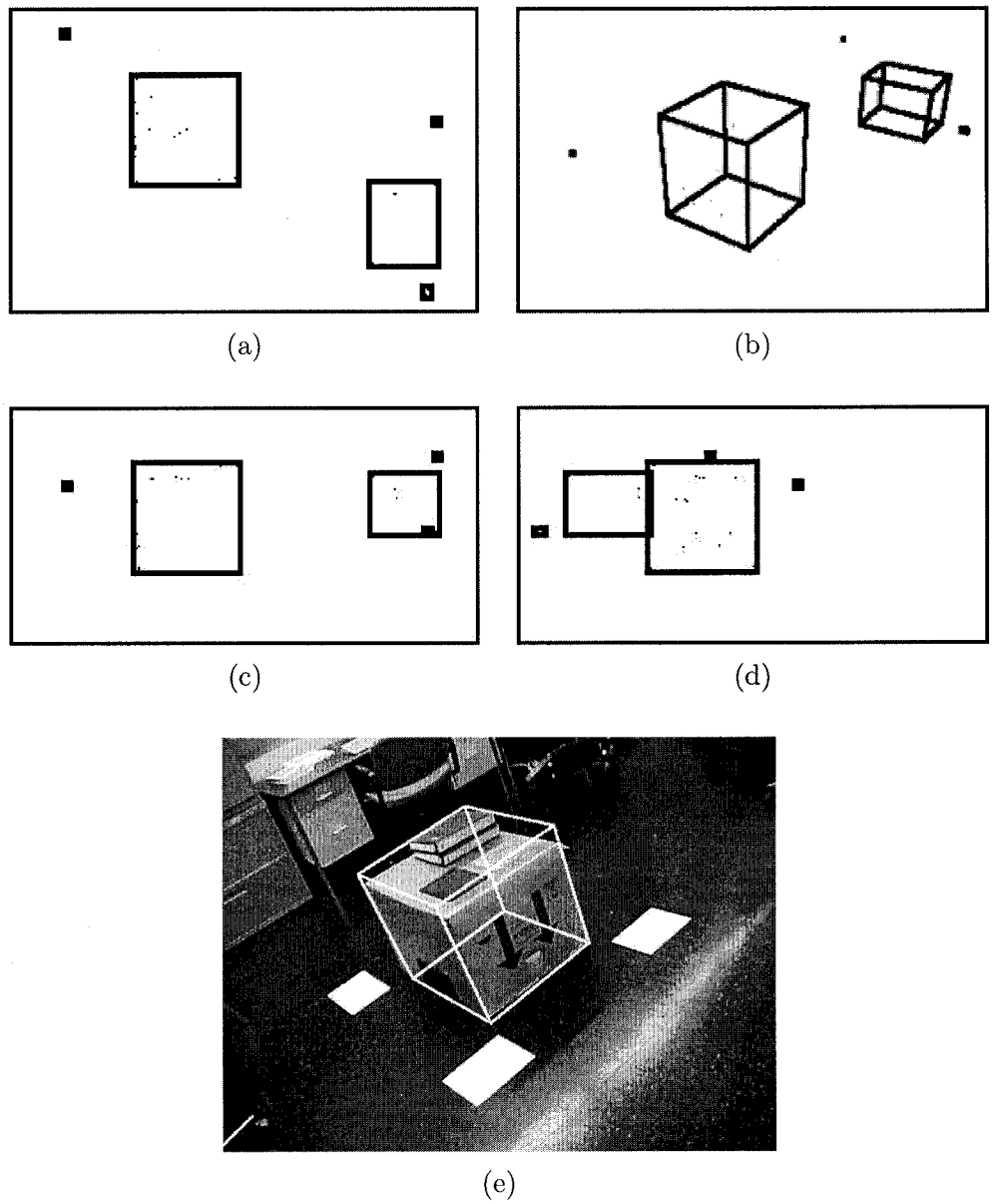


Figure 6.3: Results for clustering points in a 3D scene through a threshold of 600 mm. (a) Top projection. (b) Perspective view. (c) and (d) Side views. All clusters are shown. (e) The cluster with the maximum number of points superimposed on the original image as a bounding box.

number of match pairs that are well distributed on the surface of the obstacle. Consequently, 3D point clustering step results in a bounding box that does not contain the whole obstacle volume. In order to overcome this problem and since we have more views of the scene, more image pairs can be used. For example, we added the 3D points coming from the pair (a) and (b) of Figure 6.4 to the 3D point set to be clustered. Figure 6.5(b) shows the result of using the two pairs with a threshold, t_{M}, of 80 mm. The root of the cluster shown happened at the third level of the hierarchical structure.

Notice that the choice of the world coordinate system plays a major role in determining the bounding box and its orientation as the surfaces of the bounding box are considered parallel to the main axes. For example, the world coordinate systems of the previous examples are shown in Figure 6.6 on Page 124 (these are reproductions of Figures 5.8 and 5.13 on Pages 98 and 103 respectively). In each of these examples, the world frame is carefully chosen so that the surfaces of the resulting bounding boxes are parallel to the main surfaces of the obstacle. This is an important point affecting the volume of the bounding box, which, in turn, affects the following voxelization step. In other words, if different world frames were used, the orientations of the resulting bounding boxes might be different and the volumes of these boxes might get larger. But this is not a problem for the final representation since the voxelization step to follow will extract the volume occupied by the obstacle; however, this can be done at a cost of more running time as the volume increases and the number of voxels increases. An example showing the case where a bounding box is not aligned with the surfaces of the obstacle is handled in Section 7.9.

Up to this point, the bounding boxes obtained can be used to localize obstacles, an operation important for robot navigation (as we presented in [HEL03, HEL04]). However, as seen in the last example, parts of the bounding box were not occupied at all. Hence, the need may arise to obtain a more accurate volumetric representation for obstacles. The remaining part of this chapter explains the voxel occupancy technique we use to achieve a volumetric representation for objects in space. The input to that step is the bounding boxes obtained by the above-mentioned hierarchical structure.

6.3 3D Reconstruction

Techniques developed to determine 3D object reconstruction can be divided into two main classes [NW97]: *active methods* that use laser scanners, and *passive methods* that use groups of images taken by cameras. We will restrict ourselves to the second class, which we address in this thesis. Furthermore, reconstruction from images can be categorized into four categories [CZ00b]:

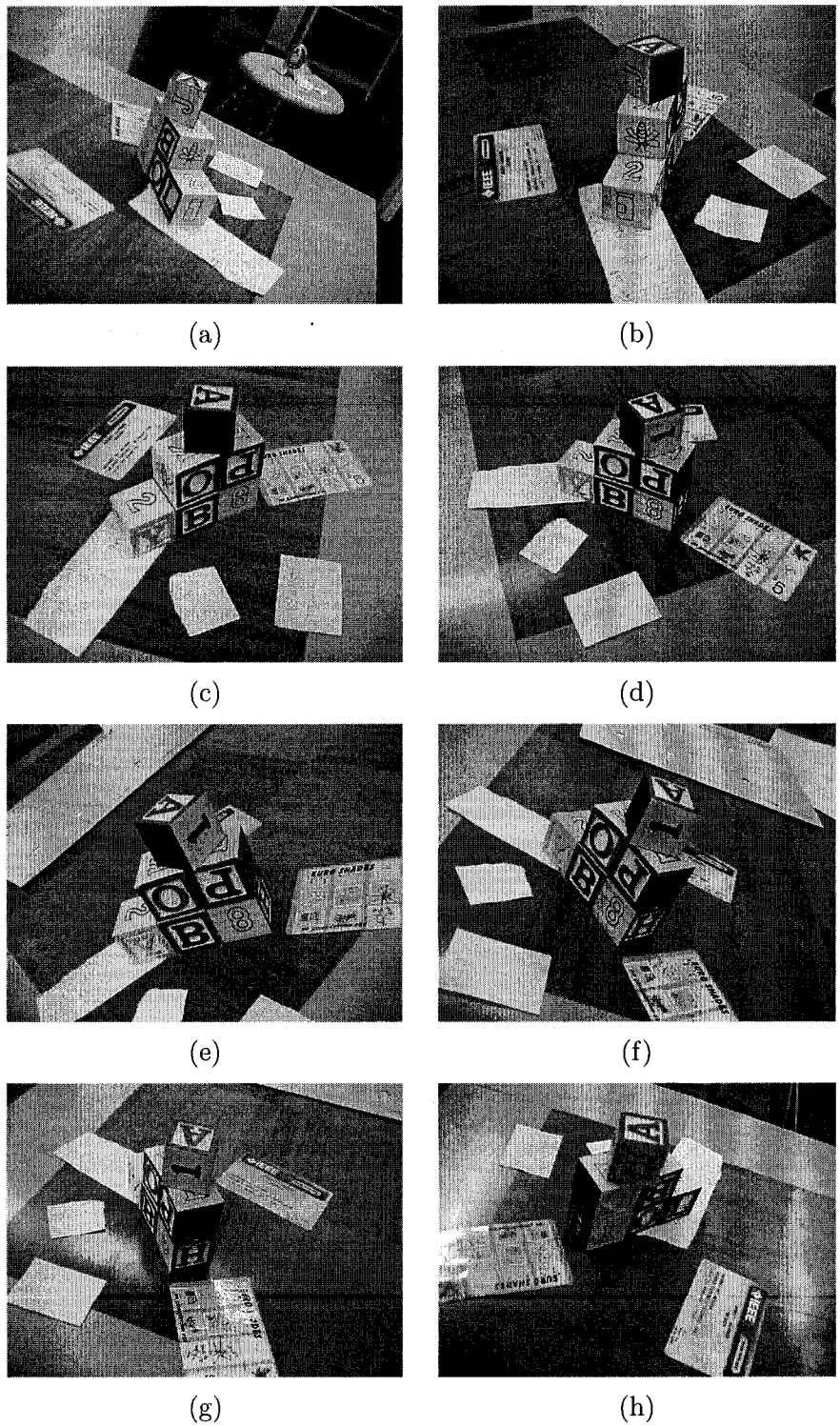
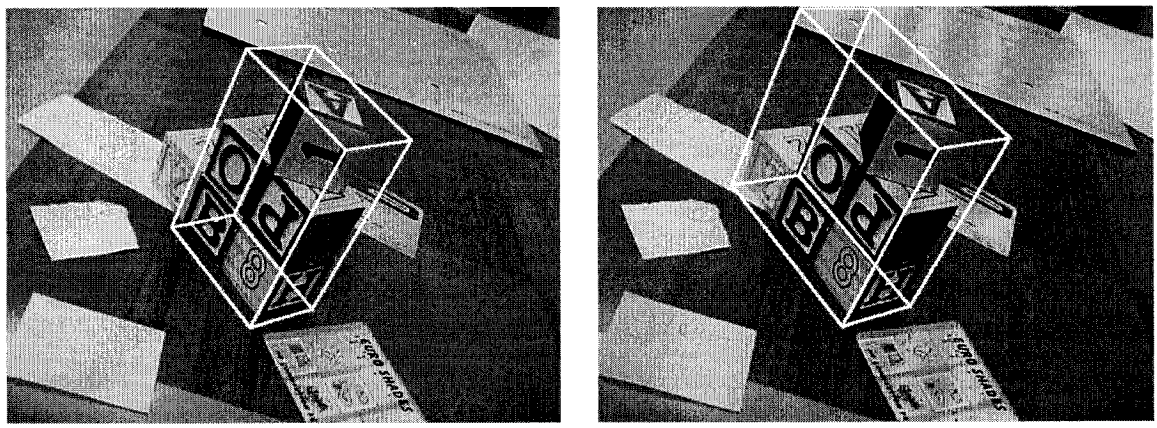


Figure 6.4: Images of another scene.



(a) (b)

Figure 6.5: Results for clustering points in a 3D scene. (a) Using only one pair of images results in a bounding box that does not contain the whole obstacle. (b) Using one more pair results in a bounding box that better represents the volume.

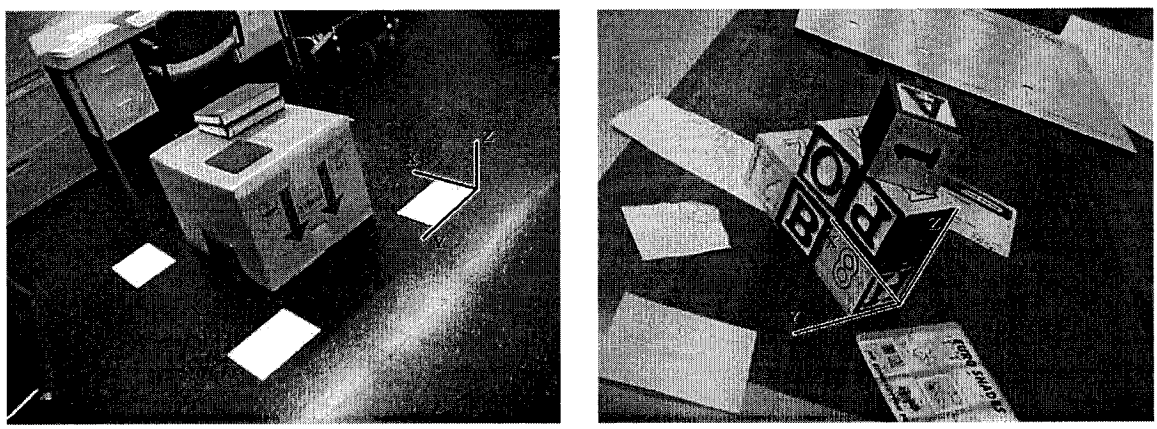


Figure 6.6: The world coordinate systems of the examples shown in Figures 6.2 and 6.4. Those are reproductions of Figures 5.8 and 5.13.

1. Reconstruction from contours;
2. Texture correlation;
3. Feature based matching; and
4. Space carving.

In [NW97], an example for the first category, Niem *et al.* suggested using a technique that consists of two phases. Those phases are shape reconstruction and texture mapping. In order to achieve shape reconstruction, they suggested using what is called *shape from silhouettes* or *method of occluding contours* [CA86, Sze93]. In other words, this is done by using borders of the silhouettes. The idea can be split into two steps. In the first step, a convex hull representing the bounding pyramid is formed. This is determined by the lines of sight from the focal point of the camera and the contours of the silhouette. In the second step, the pyramids are intersected resulting in an approximate bounding volume. Then, a mesh growing algorithm is used to transform that bounding volume into a surface model that consists of a mesh of triangles. Finally, texture mapping is used to render each mesh triangle patch.

Another algorithm based on the *shape from silhouette* approach is presented in [MTSA97]. This algorithm uses a turntable that is used along with an object of known shape to acquire camera parameters as a first step. This step is followed by image capturing, silhouette extraction, shape modeling and texture acquisition. They presented a new idea to extract silhouettes. They called it *multi-level extraction.* In the ordinary silhouette extraction scheme, a pixel-level subtraction between the object image and the background image is to be performed. However, they proposed to add a new stage to this one. They called it *region-level extraction* where region segmentation is performed and the average of the absolute subtraction at each pixel is calculated. According to a threshold, the whole region may be decided if it is a part of the silhouette. This may avoid misjudging pixels similar to its background image while its neighborhood differs. Related works can be found in [GW87, MA83, ZK92].

An example for the second category, texture correlation, is presented in [Koc95] where Koch used a stereoscopic image sequence. The steps used to achieve 3D surface reconstruction are image correspondence phase that uses cross correlation of a small image patch as a similarity measure; disparity estimation that results in a disparity map and hence a depth map; surface segmentation according to surface orientation; depth interpolation; surface approximation by a triangular mesh; and finally texture mapping. Related works to this category can be found in [FK98, BB81].

An example for the third category is presented in [AF87a]. In this work, Ayache and Faugeras used three cameras mounted on a robot to recognize the environment. The triplet of images is analyzed by a program presented in [AL87, AF87b]. This program outputs 3D line segments expressed in a coordinate system attached to the three cameras. Thus, for n robot positions, we have n line segments. (According to the authors, combining coherently various sources of information at different times and from different places is called *Visual Fusion problem.*) They applied transformations to relate various 3D line segments for the same real line at different robot locations in order to express all segments in one coordinate system attached to a given robot position. Through some representation of uncertainty of both 3D line segment positions and motions among various positions, they could combine various representations to decide if they are for the same segment and hence enhance the accuracy and abandon the redundant information.

The last recent development is *space carving* [CZ00b] or *volumetric scene reconstruction* [Dye01]. While *shape from silhouette* approach is usually used with binary silhouette images, *shape from photo consistency* combined with visibility testing is for color images. In this category, the scene is often discretized into voxels and a voxel occupancy scheme is to be developed to determine the volumetric model from silhouette or from color information. Photometric information resides in color or gray-scale images can be used to enhance the model achieved.

In [CZ00b], Cross and Zisserman presented an approach to reconstruct a surface of an object from a number of views. They combined information from contours and surface texture and used a voxelization technique. Through this, they could overcome deficiencies of using one source by the strength of the other and hence developed an efficient algorithm. Related works to this category can be found in [KS99, SD97].

The approach that we propose below combines three categories in order to achieve good results. These categories are reconstruction from contours, texture correlation, and space carving.

6.4 Voxel Occupancy Approach

The hierarchical structure we presented provides a bounding box for every obstacle in the scene according to the features extracted. However, this bounding box is built according to the overall maximum and minimum coordinates of the feature points detected on that obstacle along the three axes. In other words, an obstacle can be contained in the bounding box but may not occupy its whole volume. Hence, the need may arise to determine the sub-volume

that could better represent the obstacle. In order to reach this goal, we propose an algorithm that is based on a voxelization scheme. In this approach, we proceed as follows [EL04]:

1. Use RANSAC scheme to get updated versions of the homography matrices among all pairs as explained in Section 4.9.
2. Use RANSAC scheme to get updated versions of the fundamental matrices as explained in Section 5.5. Hence, the projection matrices may be updated accordingly (Section 6.4.1).
3. Extract silhouettes for the obstacles (Section 6.4.2).
4. Determine the occupied voxels through silhouettes, projection matrices and original images (Section 6.4.3).
5. Determine hidden and occluded voxels (Section 6.4.4).

The following sections describe these steps in more details.

6.4.1 Enhancement of Matrices

As done in Sections 4.9 and 5.5, RANSAC scheme may be applied to all pairs to get updated versions for the homography matrices, $\mathtt{H}_g$, as well as the fundamental matrices, $\mathtt{F}$, for all images. Consequently, more accurate projection matrices, $\mathtt{P}$, can be obtained.

6.4.2 Silhouette Extraction

Using the enhanced set of homography matrices and the projections of the bounding boxes on all images in the set, we can extract silhouette-like images for the obstacles corresponding to every image in the set. The idea is as follows. Every bounding box, b, is surrounded by eight 3D corners, $\dot{\mathbf{M}}_{ib} = [X_{ib}, Y_{ib}, Z_{ib}]^T$ where $i \in \{1, 2, ..., 8\}$, and twelve 3D edges connecting them. Using the above projection matrices, we project those edges with their corners on every image plane. For a given image j, this is done by using Equation (2.6):

$$\mathbf{m}_{ijb} = \mathtt{P}_j \mathbf{M}_{ib} \tag{6.5}$$

where

- i is the corner number such that $i \in \{1, 2, ..., 8\}$;
- j is the image number such that $j \in \{1, 2, ..., j, ..., g\}$ where g is the number of images; and
- b is the bounding box (or obstacle) number such that $b \in \{1, 2, ..., b, ..., o\}$ where o is the number of obstacles.

At this point, we end up with eight 2D points for an obstacle b on image j, $\dot{\mathbf{m}}_{ijb} = [x_{ijb}, y_{ijb}]^T \mid i \in \{1, 2, ..., 8\}$, $j \in \{1, 2, ..., j, ..., g\}$ where g is the number of images and $b \in \{1, 2, ..., b, ..., o\}$ where o is the number of obstacles. We determine the convex hull of them. This could be done through any well-known algorithm; e.g., Graham Scan. However, we have a small number of points. Moreover, the lines that represent the edges of the convex hull are a subset of the projections of the edges of the bounding box. This makes it simpler to obtain the convex hull. In order to check if an edge is a part of the convex hull, its linear equation may be used. This equation can be easily obtained through its two end points (refer to Sections 2.1.2 and C.1). Thus, applying the coordinates of the other projected points (only 6 points) to this linear equation can determine whether this edge is a part of the convex hull or not. This is to check the sign of:

$$\mathbf{m}_{ijb}^T \mathbf{l}_{pq} = \mathbf{m}_{ijb}^T [\mathbf{m}_{pjb} \times \mathbf{m}_{qjb}] = [x_{ijb}\ y_{ijb}\ 1] \begin{bmatrix} y_{qjb} - y_{pjb} \\ x_{pjb} - x_{qjb} \\ y_{pjb}x_{qjb} - y_{qjb}x_{pjb} \end{bmatrix} \tag{6.6}$$

where

- $\mathbf{l}_{pq}$ is the equation of the line connecting points $\dot{\mathbf{m}}_{pjb}$ and $\dot{\mathbf{m}}_{qjb}$; and
- $\dot{\mathbf{m}}_{ijb}$, $\dot{\mathbf{m}}_{pjb}$ and $\dot{\mathbf{m}}_{qjb}$ are the projections on image j of three different corners of the bounding box, b; i.e., $i, p, q \in \{1, 2, ..., 8\}$ and $i \neq p \neq q$.

If the sign of the result of Equation (6.6) is the same for all $i \in \{1, 2, ..., 8\}$ such that $i \neq p \neq q$, then the line $\mathbf{l}_{pq}$ is an edge of the convex hull; otherwise, it lies inside its boundaries. For the image shown in Figure 6.7(a) on Page 129, the convex hull of the bounding box projected on this image is shown in Figure 6.7(b). The last point to mention at this step is that we keep track of the sign of the internal direction of each edge of the convex hull as this will be used in the next step.

Knowing the edges of the convex hull, we determine if a pixel lies inside its boundaries. As in Equation (6.6), applying the pixel coordinates to the linear equations of all edges of the convex hull determines if this pixel lies inside the convex hull if all signs are consistent with the corresponding signs of the last step. To reduce consuming time, the smallest rectangle enclosing the convex hull may serve as a range for candidate pixels. Figure 6.7(c) on Page 129 shows a binary image showing all points that lie inside the convex hull of the bounding box projected on the image shown in Figure 6.7(a).

The homography of the ground plane between two images j and k can be obtained through the concatenation of the available set of homography matrices. Thus, given a set of homography matrices $\{\mathtt{H}_1, \mathtt{H}_2, ..., \mathtt{H}_j, ..., \mathtt{H}_k, ..., \mathtt{H}_g\}$ where $\mathtt{H}_j$ represents the ground plane homography

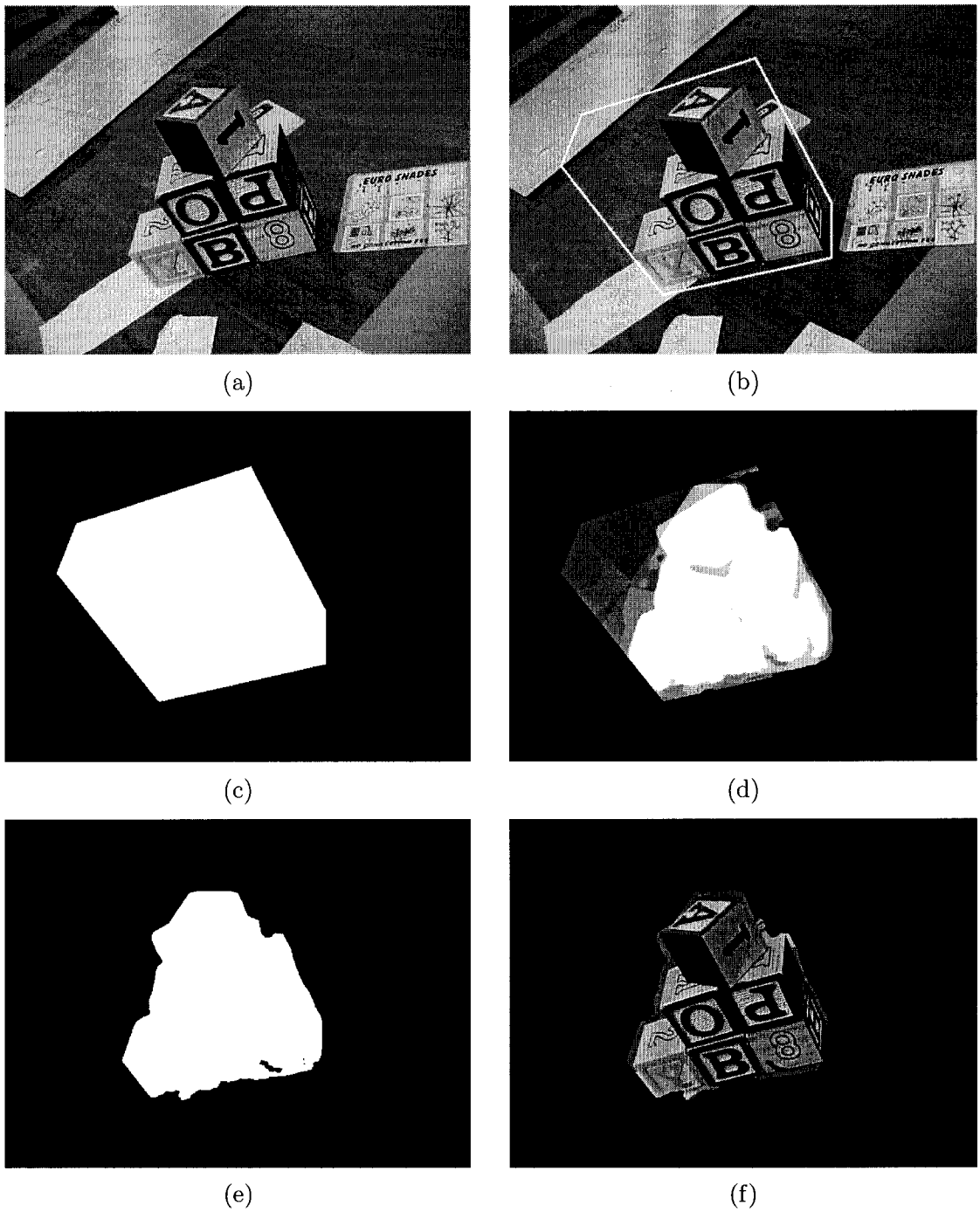


Figure 6.7: (a) An image shown in Figure 6.4(e). (b) The convex hull of the bounding box projected onto the Image (a). (c) A binary image showing all points that lie inside the convex hull of the bounding box projected on the image (d) A gray-scale version of the silhouette where the brighter the pixel the closer it gets to be an obstacle point. (e) The silhouette of the obstacle as a binary version of (d). (f) The obstacle detected using (e).

between images j and $j+1$, and in order to calculate the homography from image j to image k, $\mathtt{H}_{jk}$, we use the relation:

$$\mathtt{H}_{jk} = \prod_{h=j}^{k-2} \mathtt{H}_{h+1}\mathtt{H}_h \tag{6.7}$$

For a given pixel inside the convex hull boundaries in an image, j, we use the homography matrix, $\mathtt{H}_{jk}$, that relates the ground plane projection between images j and k, where $j \in \{1, 2, ..., g\}$, $k \in \{1, 2, ..., g\}$ and $j \neq k$. This is to check whether the intensities of the corresponding pixel neighborhoods are close to each other. That is, for a given pixel $[x_{jb}, y_{jb}]^T$ belonging to obstacle b in an image j and with respect to another image k, we perform the following SAD correlation check:

$$\sum_{m,n=-\frac{N}{2}}^{\frac{N}{2}} |\mathcal{I}_j(x_{jb}+m, y_{jb}+n) - \mathcal{I}_k(\dot{\mathbf{k}}^T)| > t_{\mathcal{S}} \tag{6.8}$$

where

- $N+1$ is the side length of the correlation window in pixels;
- $\mathbf{k} = \mathtt{H}_{jk}\ [x_{jb}+m, y_{jb}+n, 1]^T$; and
- $t_{\mathcal{S}}$ is a threshold.

There are a few possibilities at this point. If the colors of two corresponding pixels are close to each other; i.e., false condition in (6.8), the examined pixel may be a part of the ground plane. If the colors are far from each other according to $t_{\mathcal{S}}$; i.e., true condition in (6.8), then there is a good chance that the examined pixel is a part of the silhouette. Another possibility may happen where the location of the corresponding pixel lie outside the image.

The last step is to be repeated between that given pixel and its corresponding location in every other possible image. Different counters are to be initialized to record every comparison resulting in the above three possibilities; i.e., being part of the ground plane (Σ_{grd}), being part of the obstacle (Σ_{obs}), or being outside range (Σ_{out}), in addition to the total number of comparisons, Σ_{all}. The outcome can be summarized in Table 6.1.

Hence, an approximation of the silhouette for each obstacle can be obtained on each image plane. For the image shown in Figure 6.7(a), a gray-scale version of the silhouette is shown in Figure 6.7(d) where the brighter the pixel the closer it gets to the obstacle. A binary version of the silhouette of the obstacle is shown in Figure 6.7(e). The threshold, $t_{\mathcal{S}}$, used is 6200 with a window size of 21 $\times$ 21 pixels (i.e., $N = 20$ pixels).

No.	Case	Interpretation
1	$\Sigma_{out} = \Sigma_{all}$	no info
2	$\Sigma_{obs} = \Sigma_{all}$	obstacle
3	$\Sigma_{obs} > \Sigma_{grd}$	obstacle
4	$\Sigma_{obs} = \Sigma_{grd}$	obstacle or ground
5	$\Sigma_{obs} < \Sigma_{grd}$	ground
6	$\Sigma_{grd} = \Sigma_{all}$	ground

Table 6.1: Possible cases to decide if a pixel is a part of the ground plane or obstacle. Σ_{grd} is being part of the ground plane. Σ_{obs} is being part of the obstacle. Σ_{out} is being outside range. Σ_{all} is the total counter. In our application, we added cases "1" and "4" to the ground plane.

6.4.3 Voxel Occupancy Determination

At this stage, we reconstruct a 3D model that should represent each obstacle through voxelization. First and according to some resolution ϵ, we dynamically allocate space for voxels representing each obstacle. This is a 3D array $[\ \lceil \frac{max_x - min_x}{\epsilon} \rceil, \lceil \frac{max_y - min_y}{\epsilon} \rceil, \lceil \frac{max_z - min_z}{\epsilon} \rceil\ ]$ where $[min_x, min_y, min_z]^T$ and $[max_x, max_y, max_z]^T$ are the coordinates of the two extreme points of the bounding box of an obstacle and ϵ is a resolution factor. Thus, for every voxel, $\dot{\mathbf{V}} = [V_x, V_y, V_z]^T$, we perform the following. Using the updated projection matrices determined above, we project the voxel onto its corresponding location in the silhouette images. As in Equations (6.5) and (2.6), the voxel projection, $\mathbf{v}$, can be obtained from:

$$\mathbf{v}_{jb} = \mathtt{P}_j \mathbf{V}_b \tag{6.9}$$

where

- j is the image number such that $j \in \{1, 2, ..., j, ..., g\}$ where g is the number of images; and
- $\mathbf{V}_b$ is a 4D vector representing the center of the voxel belonging to an obstacle, b, in homogeneous coordinates.

If the voxel projection is not part of the silhouette in $t_{n\mathcal{S}}$ or more images, then this voxel is not occupied. Otherwise, there is a good chance that this voxel is a part of the obstacle and hence continue with the next steps. The value of intensity or color of the voxel is calculated through interpolation (Section 4.5). Thus, if the number of images is g, then we end up with g intensity values for the same voxel ($3g$ in case of using color images). At this point, we perform a color consistency test by resetting g counters to zeros; i.e., $\Sigma_i = 0$ where $i \in \{1, 2, ..., g\}$. Then, for every two intensity values $\mathcal{I}_j(\dot{\mathbf{v}}_{jb}^T)$ and $\mathcal{I}_k(\dot{\mathbf{v}}_{kb}^T)$ (corresponding to images j and k), we check:

$$|\mathcal{I}_j(\dot{\mathbf{v}}_{jb}^T) - \mathcal{I}_k(\dot{\mathbf{v}}_{kb}^T)| \leq t_{\mathcal{I}} \tag{6.10}$$

where $t_{\mathcal{I}}$ is a threshold. If (6.10) results in a true condition, we increment the counters Σ_j and Σ_k. The voxel is considered a part of the obstacle if and only if the value of one counter meets or exceeds another threshold, $t_{n\mathcal{I}}$. That is by checking:

$$\Sigma_i \geq t_{n\mathcal{I}} \text{ for any } i \in \{1, 2, ..., g\} \tag{6.11}$$

This is to determine the minimum number of images showing that voxel. If a voxel is occupied, a binary flag must be set to indicate occupancy. The color of the voxel is the average intensity of all its projections. Notice that the thresholds $t_{n\mathcal{I}}$ and $t_{n\mathcal{S}}$ are different. This is due to the fact that the projection of a voxel could be part of the silhouette while it is vacant. This happens when the projection of another occupied voxel coincides with that of the vacant one. In this case, the color consistency test provided by Equation (6.10) and the threshold $t_{n\mathcal{I}}$ of Equation (6.11) will have the impact rather than the threshold $t_{n\mathcal{S}}$.

6.4.4 Hidden Voxels Determination

There are two points pertinent to hidden or occluded voxel determination issue. The first point is that some voxels may be occluded all the time. This happens when the six sides of the voxel are occupied by other voxels. That is, for a voxel $\dot{\mathbf{V}} = [V_x, V_y, V_z]^T$, we check if $[V_x \pm 1, V_y, V_z]^T$, $[V_x, V_y \pm 1, V_z]^T$ and $[V_x, V_y, V_z \pm 1]^T$ are occupied. If so, and in order to avoid further calculation with this voxel, another binary flag is set to exclude this voxel from further consideration.

The second point is that sometimes, some voxels may be hidden behind others. In other words, voxel $\dot{\mathbf{V}}_m$ may be occluded by another voxel $\dot{\mathbf{V}}_n$ for some virtual camera viewpoint. That is, if the line segment connecting the center of voxel $\dot{\mathbf{V}}_m$ and the optical center of the virtual camera, $\dot{\mathbf{C}}$, intersects the voxel $\dot{\mathbf{V}}_n$, then $\dot{\mathbf{V}}_m$ occurs after $\dot{\mathbf{V}}_n$ in order. This should be checked. We proceed as follows:

1. Calculate the projection matrix of the synthesized image according to the desired virtual camera parameters. This is done through Equation (2.8):

$$\mathtt{P} = \mathtt{A}\mathtt{P}_c\mathtt{D} \tag{6.12}$$

2. Since visibility testing is dependent only on the labels of voxels visited previously, keep track of the location of the voxel contributing to every pixel in the synthesized image.

3. If a pixel is chosen more than once by $\dot{\mathbf{V}}_m$ and $\dot{\mathbf{V}}_n$, we check:

$$||\dot{\mathbf{V}}_m - \dot{\mathbf{C}}|| < ||\dot{\mathbf{V}}_n - \dot{\mathbf{C}}|| \tag{6.13}$$

where

- $\dot{\mathbf{V}}_m$ and $\dot{\mathbf{V}}_n$ are the centers of the voxels in space; and

- $\dot{\mathbf{C}}$ is the virtual optical center.

If (6.13) results in a true condition, then the intensity of the pixel is supplied from the voxel $\dot{\mathbf{V}}_m$; otherwise, the voxel $\dot{\mathbf{V}}_n$ is used instead.

6.4.5 Experimental Results

Our algorithm is applied to the bounding box of the obstacle shown in Figure 6.3. The set of images used are shown in Figure 6.2 on Page 120. The parameters used are 58 for the intensity threshold, $t_{\mathcal{I}}$, 7 mm for the voxel side length, ϵ, and 4 for the number of images, $t_{n\mathcal{I}}$. The reconstructed object is shown in Figure 6.8 on Page 134 as bounding box and as voxels from different viewpoints and angles. Notice that the sides shown in Figures 6.8(c) through (g) are the only sides of the object that appear in 4 or more images which is the minimum number of images, $t_{n\mathcal{I}}$, required in this example.

The algorithm mentioned-above is applied to the other set of images shown in Figure 6.4 on Page 123. The parameters used are 60 for the intensity threshold, $t_{\mathcal{I}}$, 1 mm for the voxel side length, ϵ, 4 for the number of images, $t_{n\mathcal{I}}$ and 2 for the number of silhouette pixels, $t_{n\mathcal{S}}$. Through these parameters, the total number of voxels in this example is 947376. The number of occluded voxels that have neighbors along all their six sides is 484788, which represents 51.2% of the total number of voxels. Thus, the test mentioned in Section 6.4.4 helps save space when storing or processing information about the final object. Moreover, the number of voxels declared vacant only by checking the silhouettes generated in Section 6.4.2 is 295250, which represents 31.2% of the total number of voxels. This releases some pressure off the color consistency step and proves the usefulness of the silhouette extraction step. The top view of this scene is generated using the method explained in Appendix D. Figure 6.9(a) on Page 135 shows a part of that view. The reconstructed object included in this scene is shown in Figure 6.9(b) as bounding box and in Figures 6.9(c) through (g) as voxels from different viewpoints and orientations.

Notice that the sides shown in Figures 6.9(c) through (g) are the only sides of the object that appear in 4 or more images which is the minimum number of images required in this example. Notice also that a small number of vacant voxels were declared occupied. There are two reasons explaining this issue. This happened because, for some viewpoints, some other occupied voxels were projected at the same positions onto their respective images. This makes the incorrectly declared voxels a part of the silhouette and hence possible occupied candidates. The second reason concerns the other main test; i.e., color consistency. The area where the problem happens has very few features and the color is almost the same all over the area. Consequently, this results in incorrect occupancy in this particular area.

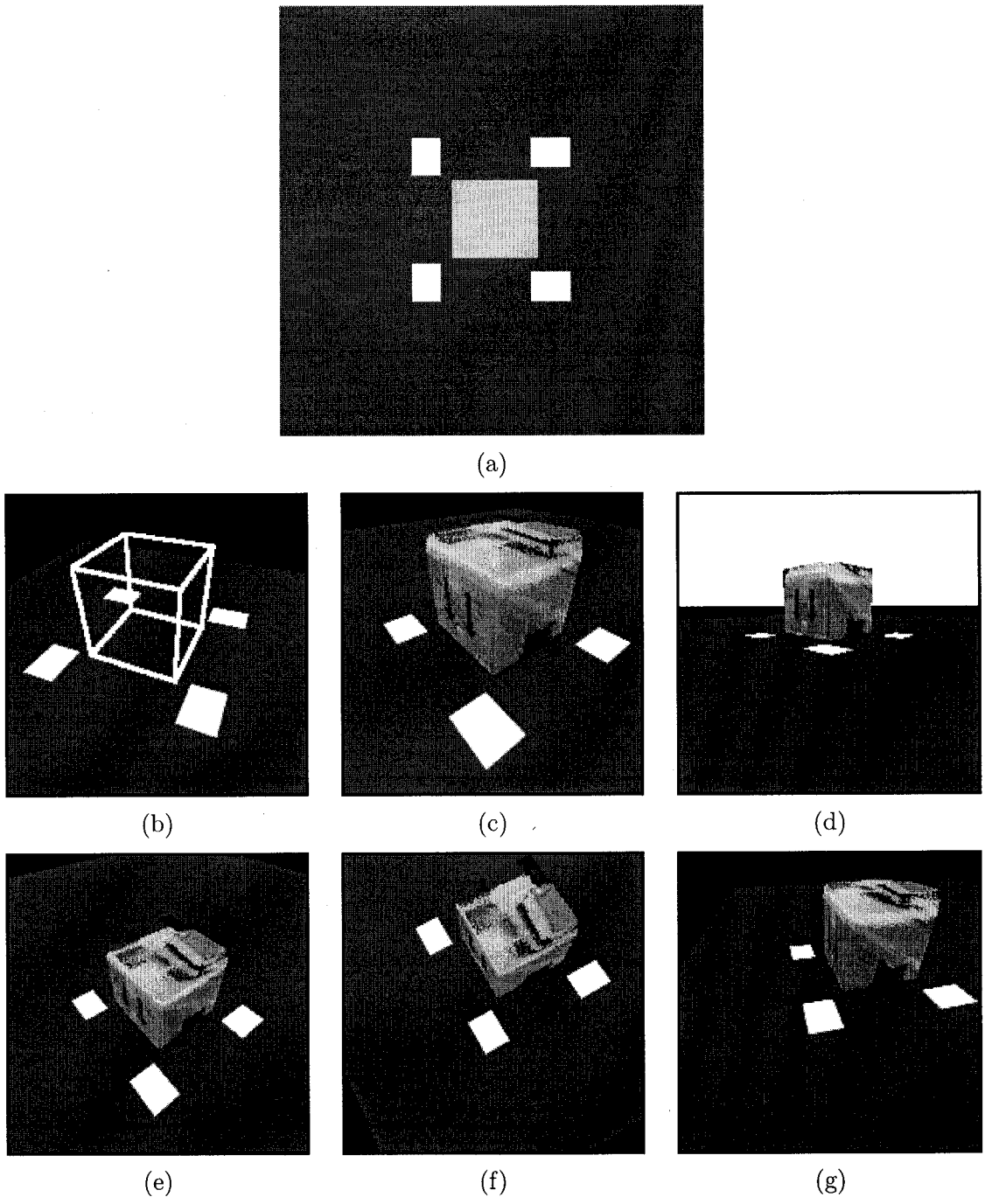


Figure 6.8: (a) A part of the top view of the scene shown in Figure 6.2. (b) The obstacle represented as a bounding box. (c) through (g) The reconstructed object represented as voxels from different viewpoints and orientations.

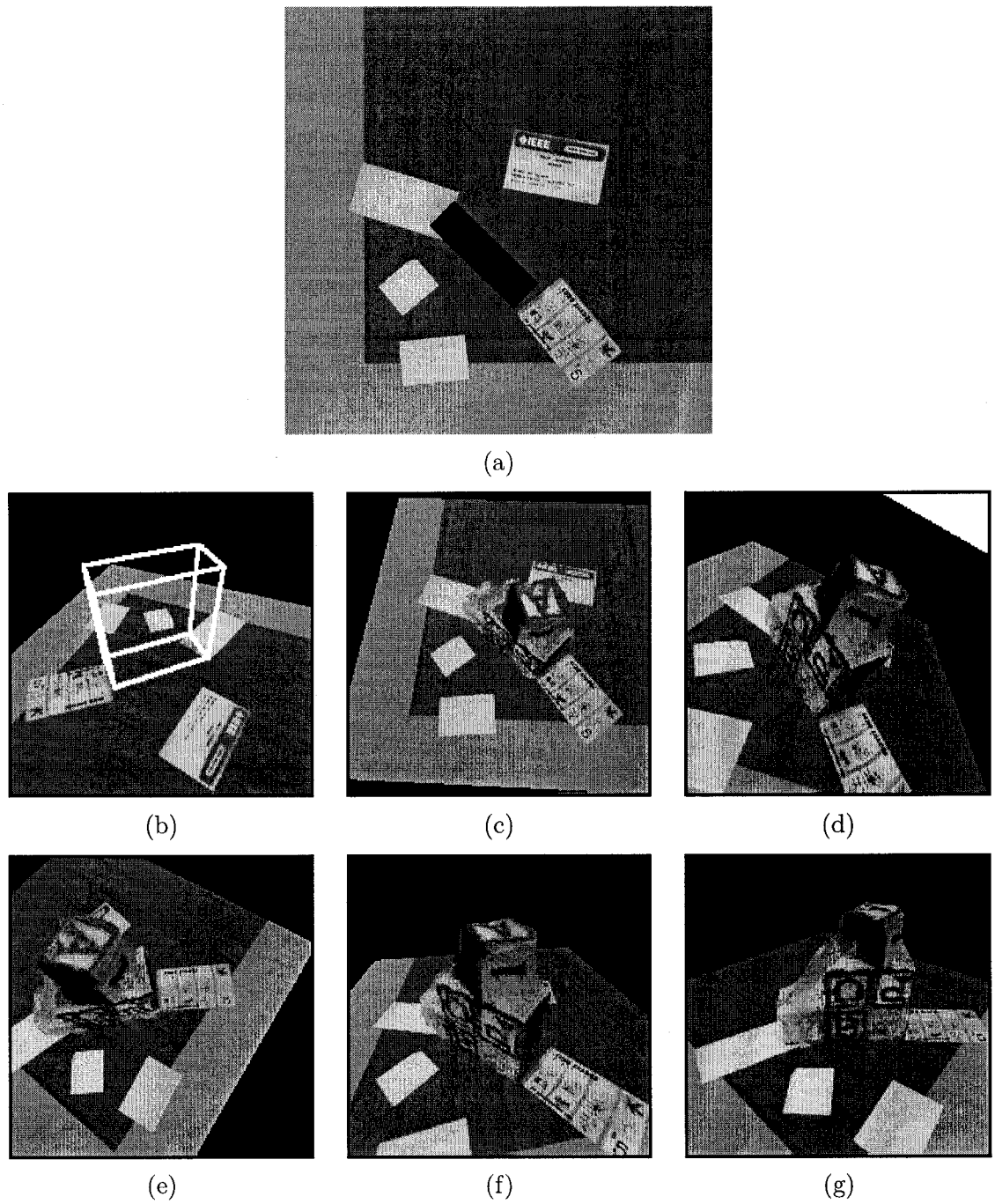


Figure 6.9: (a) A part of the top view generated for the scene shown in Figure 6.4. (b) The obstacle represented as a bounding box. (c) through (g) The reconstructed object represented as voxels from different viewpoints.

6.5 Parameters and Thresholds

This chapter dealt with a number of parameters and thresholds through different stages of the computations. In order to reconstruct points in 3D space, the original camera parameters (intrinsic and extrinsic) detected through available sensors are used. At the clustering stage, the threshold $t_{\dot{\mathbf{M}}}$ is used to test if two points are close enough to belong to the same cluster. The legitimate range for this threshold varies considerably according to the scene under consideration and also according to the measuring units of the camera parameters. For example, if the translation of each camera is measured in millimeters, the locations of 3D points will be produced in millimeters and the threshold $t_{\dot{\mathbf{M}}}$ should reflect that. The threshold value would differ for the same scene if inches were used instead.

In order to extract silhouettes, the threshold $t_{\mathcal{S}}$ is used. This threshold measures SAD correlation utilized to extract silhouettes. Assume that the correlation window side is $N+1$ pixels, then the legitimate range for the threshold $t_{\mathcal{S}}$ resides in the interval $[0, 255(N+1)^2]$ with 0 represents the perfect case and $255(N+1)^2$ represents the worst case.

At the voxelization step, more parameters and thresholds are used. These are ϵ, $t_{n\mathcal{S}}$, $t_{\mathcal{I}}$ and $t_{n\mathcal{I}}$. The parameter ϵ determines the size of a voxel. In other words, it determines the resolution at which a bounding box may split and the number of the resulting voxels. Selecting too low value for ϵ will certainly increase the accuracy of the voxelization; however, this comes with the cost of more space to be utilized and more execution time to be consumed. The value of ϵ may get affected by some factors. For example, if the resolution of the resulting synthesized image is too high, one may have to lower the value of this parameter to increase accuracy. Other values; e.g., the focal length of the virtual camera, may have a similar role affecting the choice of the parameter ϵ.

Another threshold used with the color consistency test at the voxelization step is $t_{\mathcal{I}}$. This test checks how close two voxels are in terms of intensity or color. Thus, the range of this threshold can be represented by the interval $[0, 255]$ with 0 represents identical intensities and 255 represents the maximum difference possible. In our experiments, we found that setting this threshold around 50 to 60 would result in a good outcome.

Two more thresholds used at the voxelization step are $t_{n\mathcal{S}}$, and $t_{n\mathcal{I}}$. The threshold $t_{n\mathcal{S}}$ restricts the maximum number of images allowed for the projection of a voxel to deviate from the silhouette if the voxel is considered occupied. The threshold $t_{n\mathcal{I}}$ determines the minimum number of images where the color or intensity is consistent for a given voxel. It is obvious that the maximum number of images available represents the upper limit for the last

two thresholds. However, the ideal value for each of these thresholds differs according to the poses of the cameras.

In addition to the above-mentioned parameters and thresholds, virtual camera parameters (intrinsic and extrinsic) are used in this chapter to produce the final synthesized image. Finally, Sections A.5, A.6 and A.7.1 summarize the role of each of the above-mentioned parameters and thresholds.

6.6 Summary

In this chapter we started by explaining a method to reconstruct an object in space and showing the drawbacks of that approach. We presented the main steps of the algorithm we proposed to represent an object in space volumetrically. These steps include reconstructing points in space, clustering points and determining the volume of the object through voxelization. We explained the details of every step of our approach. We utilized the match set obtained in Chapter 5 to reconstruct points in 3D space following the materials presented in Chapter 2. We proposed a hierarchical structure to cluster points in space and build a bounding box containing each object. These bounding boxes can be used to localize obstacles. Finally, we discussed how to apply a voxel occupancy technique to reconstruct the object in space. Results for different stages were presented. The results shown in this chapter have been published in [Eli03a, EL04].

Chapter 7

Planar Representation of Obstacles – Sequence Generation

The goal of our obstacle modeling approach is to obtain a 3D representation of the environment inside which a robot will move. As presented in Chapter 6, volumetric representation is certainly an interesting solution; however, the voxelization step may be time and space consuming especially if the resolution of the discretization is too high and the bounding box volume is too big. But, on the other hand, it is the choice when it comes to volumetric accuracy.

In this chapter, we propose an alternative way to save time and space by considering representation of obstacles as planar patches where each of these patches is showing an image of the corresponding obstacle. The proposed approximate representation might be sufficient for an operator to get a good understanding of the scene structure through a sequence of images generated along a specified virtual path. In this chapter, we explore this method. In order to present the ideas, we consider an example of a real-world scene that contains an obstacle. We will use the planar perspective matrix discussed in Chapter 4 to represent the ground plane; i.e., the plane on which the obstacles reside. The bounding boxes proposed in Chapter 6 will be used to locate obstacles. Also, silhouette images presented in Chapter 6 will be used as opacity maps in order to add realism to the synthesized image. Alternatively, as explained later, the projections of the bounding boxes may be used as opacity maps instead. This chapter also includes the steps to generate a sequence of images for a given virtual path. The last part of this chapter discusses a complete example where the different proposed representations are considered.

7.1 Obstacles as Planar Patches

As an example, we consider the scene used in the previous chapter and shown in Figure 7.1 on Page 140. This scene contains an obstacle and in compliance with our basic assumption, the surface, on which the obstacle resides, is flat.

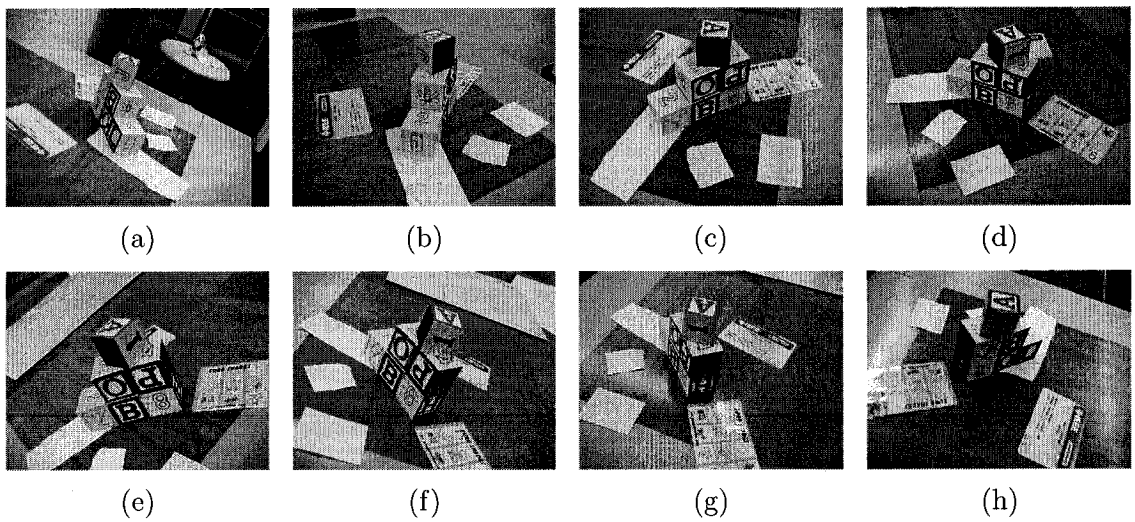
In order to generate the sequence, we assume that the top view of the site is given. A synthesized version of the top view for our example is shown in Figure 6.9(a) on Page 135. We, also, assume that camera parameters of the original images are known approximately. These parameters are listed in Table 7.1 on Page 141. Figure 7.1(i) on Page 140 shows the locations and orientations of the cameras. Moreover, the location and dimensions of the bounding box containing the obstacle have been computed through junction detection, matching and clustering operations as explained in the previous chapters. On the top view, an operator can specify a path along which he requests the robot to move under his control. (An example of such a path is shown in Figure 7.2.) Before executing the proposed task, a virtual sequence, showing what the robot would see when moving along the path, is required. Our challenge is to produce such a virtual sequence using the information available.

The ground surface of the site can be displayed by applying the planar perspective matrix (Chapter 4) to the top view of that site. Representing the obstacle volumetrically has been considered in Chapter 6. In this chapter, we consider a planar representation alternative where each obstacle is represented as a planar patch on which an image of the obstacle is mapped to.

The following steps are used in order to create the requested sequence.

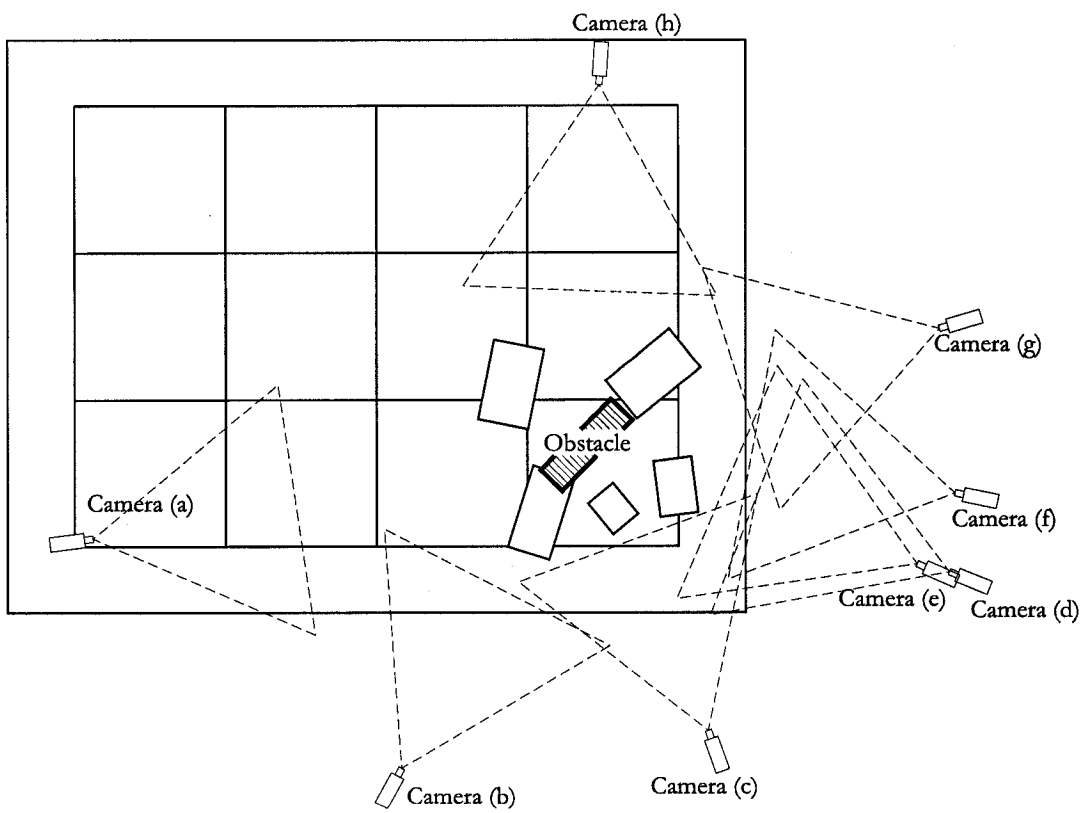
1. Unify the systems of units (Section 7.2).
2. Determine the parameters of the virtual camera along the path (Section 7.3).
3. For every frame, do the following:
   (a) Determine the intermediate virtual parameters through interpolation (Section 7.3).
   (b) Determine the boundaries of the planar patch for every obstacle (Section 7.4).
   (c) Select the appropriate view for mapping (Section 7.5).
   (d) Perform mapping between the source and the target image areas for every obstacle (Section 7.6).

The next sections explain the details of the procedure.



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

(e) (f) (g) (h)



(i)

Figure 7.1: (a) through (h) Images of a scene. (i) The plan of the scene showing the positions and orientations of Cameras (a) to (h).

Image	X	Y	Z	τ	ρ	ψ
(a)	555.2359	-325.3561	431.4782	58.4021	-53.0427	25.941
(b)	493.0843	132.5351	420.7934	48.8943	-105.8211	-0.3921
(c)	203.3010	362.9578	451.3856	39.7229	-155.3958	-4.9299
(d)	-134.3809	429.7352	466.3157	43.145	156.3501	0.3174
(e)	-114.7469	398.9921	355.3564	54.2098	158.7623	-30.1229
(f)	-206.6808	372.5378	351.6793	54.6702	146.1463	26.7937
(g)	-334.0451	220.6162	461.5761	43.5479	122.0288	9.8471
(h)	-258.2932	-274.7759	472.1151	41.5558	48.6178	20.7127

Table 7.1: The parameters for each camera of the set shown in Figure 6.4. The world origin is shown in Figure 5.13. The rotation angles used are tilt, τ, pan, ρ, and swing, ψ. The units are in millimeters for X, Y, Z and in degrees for τ, ρ and ψ. Each image is 640 $\times$ 480 pixels. The focal length is 6.0 mm. The format size is 3.5311 $\times$ 2.7446 mm.

Figure 7.2: The top view image of the previous scene showing the path of the sequence requested by the operator. Keyframes shown were chosen so that the largest number of images might be used. Viewing directions are shown as arrows. Numbers (i) through (xii) represent the views seen by the robot at each keyframe viewpoint. Those views are shown in Figure 7.5. In this example, Camera (c) is used for Keyframes (i) through (iii); Camera (d) is used for Keyframes (iv) through (vi); Camera (e) is used for Keyframes (vii) through (ix); and Camera (f) is used for Keyframes (x) through (xii).

7.2 Unit System

Before proceeding with generating the sequence, we should point out that the units measuring the top view image might be different from those measuring the locations of the viewpoints and the bounding boxes; e.g., pixels and millimeters. Moreover, the origin and the orientation of the coordinate systems in both cases might be different. Thus, we need to unify both systems of units into one system; e.g., pixels. In these two systems, only ratio of lengths and angles are preserved in both cases constituting invariant measures, which are the invariant properties pertinent to similarity transformation. A similarity transformation matrix has 4 degrees of freedom (DOF); hence, can be fully defined through two pairs of corresponding points. Let us determine the two points; e.g., the origin of the world coordinate system, $\dot{\mathbf{m}}_0 = [x_0, y_0]^T$, and another point along the $X-$axis on the top view image, $\dot{\mathbf{m}}_1 = [x_1, y_1]^T$ where the points $[x_0, y_0]^T$ and $[x_1, y_1]^T$ are expressed in the top view pixel coordinate system. As done in the previous chapters, we assume that the world $XY-$plane resides on the ground plane; hence, $z_0 = z_1 = 0$. This is followed by determining the scale by assigning the real length; e.g., in millimeters, to the distance between both points. Suppose that the length of the line segment $< \dot{\mathbf{m}}_0, \dot{\mathbf{m}}_1 >$ is d mm. Consequently, the scale factor, s, is calculated as:

$$s = \frac{||\dot{\mathbf{m}}_0 - \dot{\mathbf{m}}_1||}{d} = \frac{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}{d} \tag{7.1}$$

and the angle of inclination, ϑ, is calculated as:

$$\vartheta = tan^{-1}\left(\frac{y_1 - y_0}{x_1 - x_0}\right) \tag{7.2}$$

Hence, the transformation between both systems on the planar ground surface is carried out through a rotation matrix, $\mathtt{R}$; a translation vector, $\mathbf{t}$; and a scaling factor, s. Those are defined as:

$$\mathtt{R} = \begin{bmatrix} cos(\vartheta) & -sin(\vartheta) \\ sin(\vartheta) & cos(\vartheta) \end{bmatrix}, \mathtt{C} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{7.3}$$

For example, a point $[X, Y, Z]^T$ expressed in millimeters may be transformed to pixel coordinate system as:

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{bmatrix} = \begin{bmatrix} \mathtt{RC}\begin{bmatrix} X \\ Y \end{bmatrix} + \mathbf{t} \\ s\ Z \end{bmatrix} \tag{7.4}$$

The similarity matrix, $\mathtt{S}$, can be expressed homogeneously as done in Equation (2.62):

$$\mathtt{S} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}_2^T & 1 \end{bmatrix} \tag{7.5}$$

Hence, a point $[X, Y, Z]^T$, where $\dot{\mathbf{m}} = [X, Y]^T$ on the ground plane expressed in millimeters, may be transformed to pixel coordinate system as:

$$\mathbf{m}_{new} = \mathtt{S}\mathbf{m} \tag{7.6}$$

then

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{m}}_{new} \\ s\ Z \end{bmatrix} \tag{7.7}$$

The following section discusses creating the sequence as a whole. This is followed by showing how one frame of the sequence can be generated as a planar patch.

7.3 Specifying the Sequence

The operator should be able to specify a few points along the virtual path on the top view. These points will constitute keyframes in the sequence to be generated (as depicted in Figure 7.2 on Page 141). According to the total length of the path and the total number of steps determined by the operator, intermediate camera viewpoints are to be established. We used the following steps:

1. Calculate the Euclidean distance, $d(\dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1})$, between every two consecutive points in the specified path, $\dot{\mathbf{m}}_k = [x_k, y_k]^T$ and $\dot{\mathbf{m}}_{k+1} = [x_{k+1}, y_{k+1}]^T$, where $k \in \{1, 2, ..., m-1\}$ and m is the number of points.

$$d(\dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1}) = ||\dot{\mathbf{m}}_k - \dot{\mathbf{m}}_{k+1}|| = \sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2} \tag{7.8}$$

2. According to the total number of steps, n, calculate the step size, δ.

$$\delta = \frac{\sum_{k=1}^{m-1} d(\dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1})}{n} \tag{7.9}$$

3. The viewing direction can be determined so that the camera points along the direction of the motion. That is, for every segment, use the two end points, $\dot{\mathbf{m}}_k$ and $\dot{\mathbf{m}}_{k+1}$, to determine the direction of the motion, $\rho_{(k,k+1)}$, along the path by calculating:

$$\rho_{(k,k+1)} = tan^{-1}\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \tag{7.10}$$

   Note that, in this case, $\rho_{(k,k+1)}$ is the viewing direction as well as the direction of the motion.

4. Using the step size, δ, and the direction of the motion, $\rho_{(k,k+1)}$, determine the coordinates of the intermediate points.

$$\begin{aligned} x_n &= x_p \pm \delta\ cos(\rho_{(k,k+1)}) \\ y_n &= y_p \pm \delta\ sin(\rho_{(k,k+1)}) \end{aligned} \tag{7.11}$$

   where $[x_n, y_n]^T$ and $[x_p, y_p]^T$ are the next and the previous points respectively along the line segment $< \dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1} >$.

5. As shown in Figure 7.2, the viewing direction may be specified for the keyframes so that the intermediate frames may inherit the angle among keyframes by interpolation. Thus, if the viewing directions of the points $\dot{\mathbf{m}}_k$ and $\dot{\mathbf{m}}_{k+1}$ are $\rho_{(k)}$ and $\rho_{(k+1)}$ respectively, then the viewing direction of an intermediate point $\dot{\mathbf{m}}_n = [x_n, y_n]^T$ along the line segment $< \dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1} >$ can be determined as:

$$\rho_{(n)} = \rho_{(k)} + \frac{(\rho_{(k+1)} - \rho_{(k)})\ d(\dot{\mathbf{m}}_k, \dot{\mathbf{m}}_n)}{d(\dot{\mathbf{m}}_k, \dot{\mathbf{m}}_{k+1})} \tag{7.12}$$

Note that we may apply the above-mentioned steps to create sequences based on bounding boxes, volumetric or planar representations as well.

7.4 Planar Patch Boundaries

In order to determine the location of a planar patch in the synthesized image, the corners of the bounding box of each obstacle are projected onto the image plane. Thus, according to Equation (2.5), a corner $\dot{\mathbf{M}}_i = [X_i, Y_i, Z_i]^T$ is projected as:

$$\mathbf{m}_i = \mathtt{P}_v \mathbf{M}_i \tag{7.13}$$

where

- $\mathbf{M}_i$ where $i \in \{1, 2, ..., 8\}$ are the 3D corner points of the bounding box;
- $\mathbf{m}_i$ where $i \in \{1, 2, ..., 8\}$ are the 2D corner points projected onto the image plane; and
- $\mathtt{P}_v$ is the virtual camera projection matrix calculated according to Section B.4.

According to the minimum and maximum coordinates of these points, $\mathbf{m}_i$, two points representing a target rectangle may be determined.

At this step, the projection of the bounding boxes onto the original images are used to determine a bounding rectangle for each obstacle according to the minimum and maximum coordinates as done above. This rectangle represents the source area to be mapped. In order to add realism to the generated images, the binary images representing the silhouettes may used as opacity maps in order to eliminate the background features present in the source images. However, silhouette extraction procedure, as explained in Chapter 6, is time consuming. Consequently, the direct use of the projections of the bounding boxes might be preferred.

Finally, mapping is performed between the areas determined in the source image and the target image. The next section discusses which source image should be selected for mapping to the target view. This is followed by discussing the mapping operation.

7.5 Selecting the Appropriate View

One important question that may arise here is how to select the closest view possible for the obstacle to use for mapping. In order to select an obstacle view for each viewpoint, we use the pan angles of each camera. Therefore, we seek to minimize the difference:

$$\text{Selected view} = \arg\min_{j=1}^{g} |\rho_{(c)} - \rho_{(j)}| \tag{7.14}$$

where

- g is the number of available views;
- $\rho_{(c)}$ is the pan angle (or the viewing direction) of the current target frame; and
- $\rho_{(j)}$ is the pan angle (or the viewing direction) of the $j-$th camera.

Thus, the view that results in the minimum value for the distance (7.14) is the view that should be selected. Also, the angle of inclination, ϑ of Equation (7.2), should be taken into account to get the correct transformation. Hence, a transformed pan angle, $\rho_{(j)}$, of the $j-$th camera can be computed as:

$$\rho_{(j)} = \begin{cases} \rho - \vartheta + 360.0 & \text{if } \rho < 0.0 \\ \rho - \vartheta & \text{if } \rho \geq 0.0 \end{cases} \tag{7.15}$$

where

- $\rho_{(j)}$ is the transformed pan angle of the $j-$th camera;
- ρ is the pan angle of the $j-$th camera appearing in Table 7.1; and
- ϑ is the angle of inclination of the system.

This will result in different original images selected for different views. For instance, for the points of the path shown in Figure 7.2, the images selected are shown in Figure 7.5 on Page 151. In this example, the image of Camera (c) is used for Keyframes (i) through (iii); the image of Camera (d) is used for Keyframes (iv) through (vi); the image of Camera (e) is used for Keyframes (vii) through (ix); and the image of Camera (f) is used for Keyframes (x) through (xii).

As shown in the last column of Table 7.1, the swing affects a few images of those shown in Figure 7.1. In order to compensate and neutralize that effect, rotation as a special case of affine transformation is applied to the images. As in Equation (7.3), the rotation matrix is used [FPWW00]:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) \end{bmatrix} \begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{7.16}$$

where

- φ is the rotation angle;
- $[x_0, y_0]^T$ is the center of rotation. For the image shown in Figure 7.3(b) on Page 147, this point is the center of the image;
- $[x_1, y_1]^T$ is the input point; and
- $[x_2, y_2]^T$ is the output point where interpolation (Section 4.5) is used to get the final color.

The rotation angle, φ, is the inclination angle of the line $< [\frac{max_x+min_x}{2}, \frac{max_y+min_y}{2}, min_z]^T$, $[\frac{max_x+min_x}{2}, \frac{max_y+min_y}{2}, max_z]^T >$ as shown in Figure 7.3(b) where $[min_x, min_y, min_z]^T$ and $[max_x, max_y, max_z]^T$ are two points determining the bounding box of the obstacle.

7.6 Mapping Between Source and Target Images

The mapping problem can be split into two problems:

1. Determining the target points using the corner points; and
2. Determining the location – in the source image – that corresponds to a given target point so as to compute the target point intensity or color value.

The direction of mapping is carried out from the destination to the source image as depicted in Figure 7.4 on Page 148.

7.6.1 Determining the Target Points

An easy way to determine the target pixels is by considering only the signs of points applied to the equations of lines of the four-sided target area. That is, for each line joining two points, we can determine the sign of the internal side using one of the other two points. Every line forms a region by one of its sides. The intersection of the four regions determine all target pixels.

7.6.2 Determining the Source Points

It is required to compute a 3 × 3 plane projective transformation matrix, or homography matrix, H, that maps a source point into another destination one. That is:

$$\mathbf{m}' = \mathtt{H}\mathbf{m} \tag{7.17}$$

where $\mathbf{m}$ is the source point and $\mathbf{m}'$ is the destination one. Equation (7.26) can be rewritten as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7.18}$$

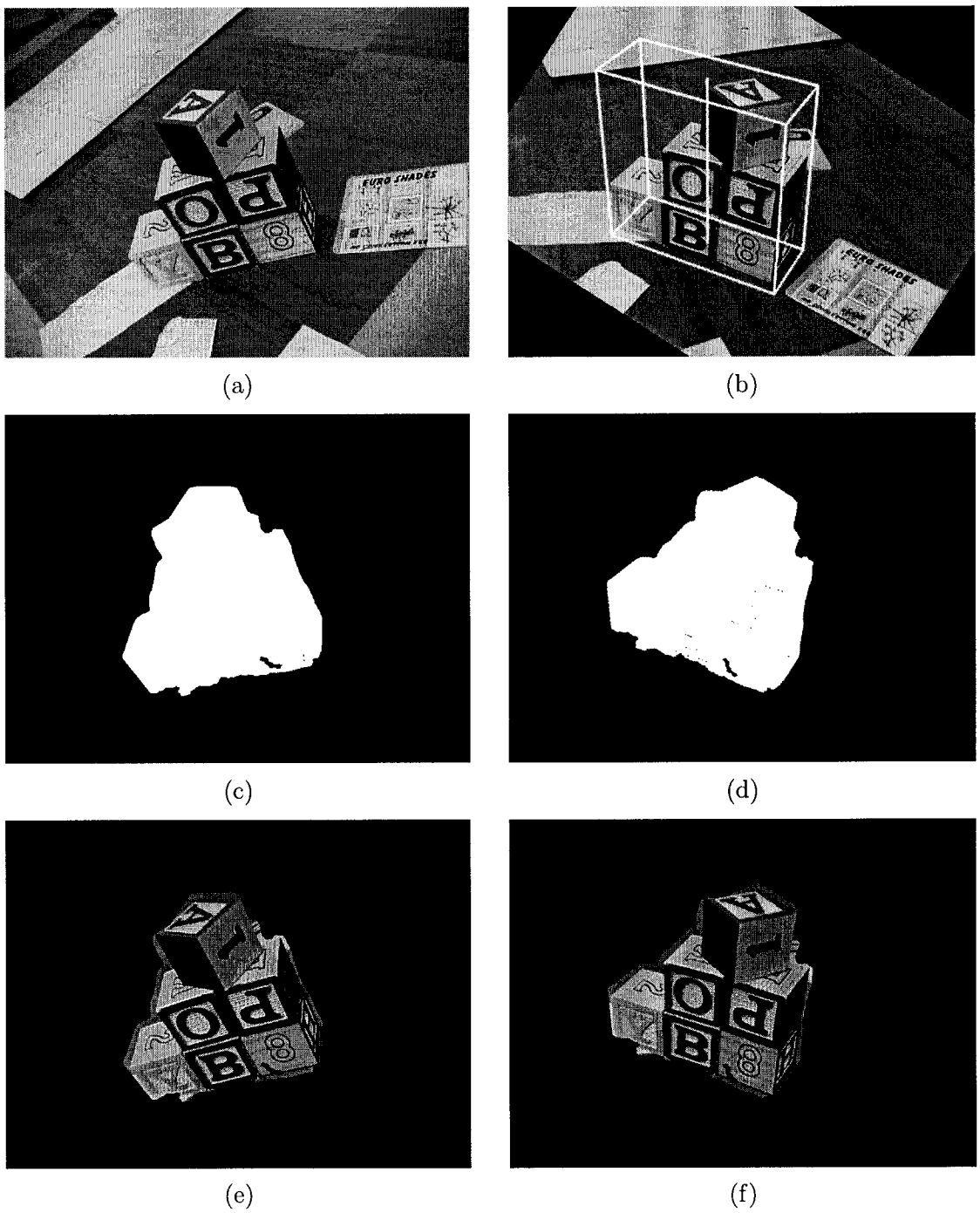


Figure 7.3: (a) An image shown in Figure 6.4(e). (b) The rotated version of Image (a). (c) The silhouette extracted from Image (a). (d) The silhouette extracted from Image (b) and used as an opacity map. (e) and (f) The obstacle extracted using the silhouettes.

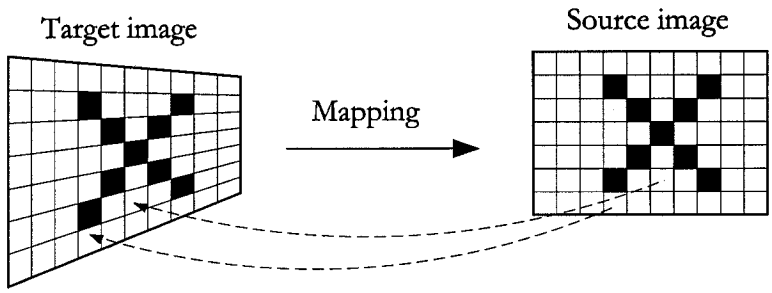


Figure 7.4: Mapping is performed from the target image to the source image.

Each pair of corresponding points contributes to two linear equations for the elements of H. Removing the scale factor by dividing by the third component, we get [Fis97a]:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{7.19}$$

and

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{7.20}$$

Consequently,

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13} \tag{7.21}$$

and

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23} \tag{7.22}$$

The homography matrix, H, has 8 degrees of freedom and can be defined through 4 corresponding pairs. Having four correspondences results in eight linear equations, which are sufficient to solve for H up to a scale factor. We choose one matrix term to have a certain value; e.g., $h_{33} = 1$. Thus, the eight equations can be written as:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} \tag{7.23}$$

or

$$\mathbf{Mh} = \mathbf{x} \tag{7.24}$$

Thus, a solution can be obtained as:

$$\mathbf{h} = \mathtt{M}^{-1}\mathbf{x} \tag{7.25}$$

The homography transformation, H, is then applied to the planar patches to render the image of the obstacle. If the location in the synthesized target image is $\dot{\mathbf{m}}'$, then the location in the source image, $\dot{\mathbf{m}}$, is given by:

$$\mathbf{m} = \mathtt{H}^{-1}\mathbf{m}' \tag{7.26}$$

Interpolation (Section 4.5) is used at this point to get the intensity of the synthesized image at the location $\dot{\mathbf{m}}'$.

7.7 Experimental Results

Consider the scene shown in Figure 7.1 on Page 140. In this example, there are 8 camera viewpoints {a, b, ..., h} and 1 obstacle. The plan of this scene is shown in Figure 7.1 on Page 140. Also, that figure shows the locations and orientations of the cameras. (The origin of the world coordinate system was shown in Figure 5.13.) Table 7.1 on Page 141 lists the parameters of each camera. It is required to generate a synthesized sequence of images seen by a robot moving along the path defined in Figure 7.2 on Page 141 where the viewing directions for the keyframes are indicated by the arrows. For Keyframes (ii) and (iii), the parameters are: height, t_z of 200 pixels; tilt, τ, of 50°; swing, ψ, of 0°; and focal length, f, of 200 pixels. For the rest of the keyframes, the parameters are: height, t_z, of 280 pixels; tilt, τ, of 35°; swing, ψ, of 0°; and focal length, f, of 250 pixels. The other parameters; i.e., the location, t_x and t_y, and the pan angle, ρ, differ from one keyframe to the other. The locations and the viewing directions are shown in Figure 7.2.

In order to select the appropriate view, we seek to minimize the distance expressed in Equation (7.14). The angles compared must take into account the inclination angle of the world coordinate system with respect to the overhead image coordinates. In this example, the angle of inclination of the world coordinate system, ϑ of Equation (7.2), is $-43.5°$. As done Equation (7.15), this angle is used to transform the pan angle of every camera viewpoint in order to perform distance minimization. This is a simple mapping operation whose results are shown in the third row of Table 7.2 on Page 150 where the second row represents the original pan angles that appear in Table 7.1. The next step is the distance minimization expressed by Equation (7.14). The minimum result amongst all camera viewpoints determines the original image to use for mapping. The rest of Table 7.2 lists the distances measured between every camera viewpoint ({a, b, ..., h} of Figure 7.1) versus every viewing direction ({i, ii, ..., xii} of Figure 7.2). The minimum distance for each keyframe appears in boldface numbers. The camera viewpoints used are those that correspond to these numbers.

After determining the appropriate original image to use by minimizing angle difference, mapping is performed. The keyframes of that sequence are shown in Figure 7.5 on Page

	Cam.→	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
	$\rho \rightarrow$	-53.0427	-105.8211	-155.3958	156.3501	158.7623	146.1463	122.0288	48.6178
frame↓	$\rho_{(j)} \rightarrow$	350.5071	297.7287	248.1540	199.8999	202.3121	189.6961	165.5786	92.1676
(i)	230.0	120.5071	67.7287	**18.1540**	30.1001	27.6879	40.3039	64.4214	137.8324
(ii)	230.0	120.5071	67.7287	**18.1540**	30.1001	27.6879	40.3039	64.4214	137.8324
(iii)	240.0	110.5071	57.7287	**8.1540**	40.1001	37.6879	50.3039	74.4214	147.8324
(iv)	200.0	150.5071	97.7287	48.1540	**0.1001**	2.3121	10.3039	34.4214	107.8324
(v)	200.0	150.5071	97.7287	48.1540	**0.1001**	2.3121	10.3039	34.4214	107.8324
(vi)	195.0	155.5071	102.7287	53.1540	**4.8999**	7.3121	5.3039	29.4214	102.8324
(vii)	205.0	145.5071	92.7287	43.1540	5.1001	**2.6879**	15.3039	39.4214	112.8324
(viii)	210.0	140.5071	87.7287	38.1540	10.1001	**7.6879**	20.3039	44.4214	117.8324
(ix)	210.0	140.5071	87.7287	38.1540	10.1001	**7.6879**	20.3039	44.4214	117.8324
(x)	185.0	165.5071	112.7287	63.1540	14.8999	17.3121	**4.6961**	19.4214	92.8324
(xi)	185.0	165.5071	112.7287	63.1540	14.8999	17.3121	**4.6961**	19.4214	92.8324
(xii)	180.0	170.5071	117.7287	68.1540	19.8999	22.3121	**9.6961**	14.4214	87.8324

Table 7.2: Determining the minimum distance between every camera viewpoint and synthesized keyframe view. The images used for mapping are those that correspond to the minimum distances appearing in boldface numbers.

151. Intermediate viewpoints and viewing directions can be determined by interpolating the parameters of the camera between keyframes. For example, Figure 7.6 on Page 152 shows intermediate views generated between Keyframes (iii) and (iv) of Figure 7.5.

Notice that, after applying the rotation to compensate for the swing effect for Cameras (e) and (f), their corresponding images could successfully be used here in six synthesized images. This results in four different images selected in this example for mapping. Those images correspond to Cameras (c), (d), (e) and (f) where Camera (c) is used for Keyframes (i) through (iii); Camera (d) is used for Keyframes (iv) through (vi); Camera (e) is used for Keyframes (vii) through (ix); and Camera (f) is used for Keyframes (x) through (xii).

Finally, if silhouette extraction step is preferred to be skipped so more execution time can be saved, the projections of the bounding boxes can be used as opacity maps instead of the silhouette images. The results of using bounding boxes are shown in Figure 7.7 on Page 153. However, note that, in this case, undesirable artifacts composed of ground patterns are introduced to the synthesized images.

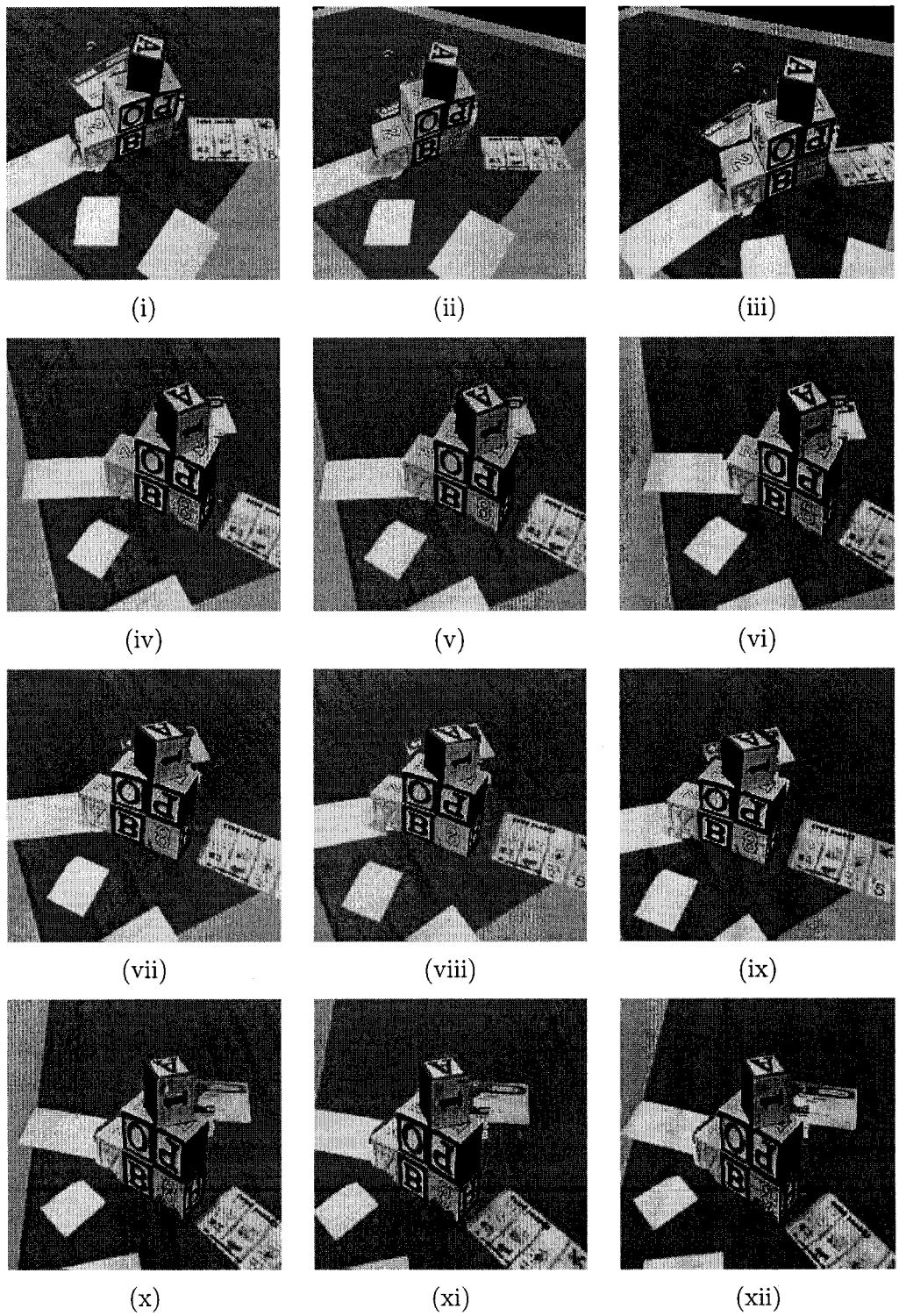
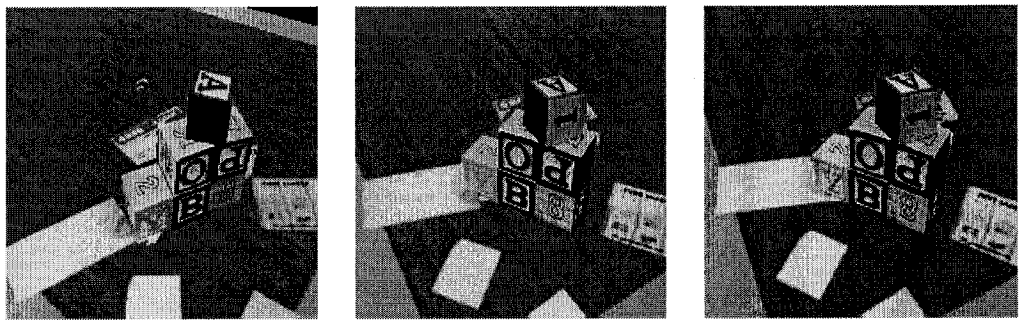


Figure 7.5: Keyframes of the synthesized sequence shown in Figure 7.2 generated by the virtual camera where silhouette images are used as opacity maps.



(i) (ii) (iii)

Figure 7.6: Intermediate views can be generated. Those views are generated between Keyframes (iii) and (iv) of Figure 7.5.

7.8 Parameters and Thresholds

Virtual camera parameters are used to represent obstacles as planar patches. The pan angles of the original cameras are used to minimize distance with the virtual viewing direction and select the appropriate view for mapping. Of course, the virtual camera parameters are also used to generate sequences for any of the representations; i.e., volumetric or planar. Also, in order to generate a sequence, the parameter n representing the number of steps of the sequence should be determined. This parameter determines the step size among successive frames. If its value is too high, the transition among frames will be smooth; however, this comes at the price of more execution and running time. If the value of n is too low, the transition from one frame to another will be abnormal. Notice that the value of n should proportional to the total length of the virtual path. Other parameters are needed to generate the sequence. These parameters are two points on the top view of the scene in addition to the real distance between them. Those are used to determine the similarity matrix that transforms between the real world measurements to the top view pixel coordinates. Finally, Sections A.7.2 and A.8 summarize the role of each of the above-mentioned parameters.

The rest of this chapter presents a complete application where sequences are to be generated for a path defined by the operator considering different methods of representation; volumetric as bounding boxes and as voxels; and planar where silhouette and alternatively bounding boxes projections are to be used as opacity maps.

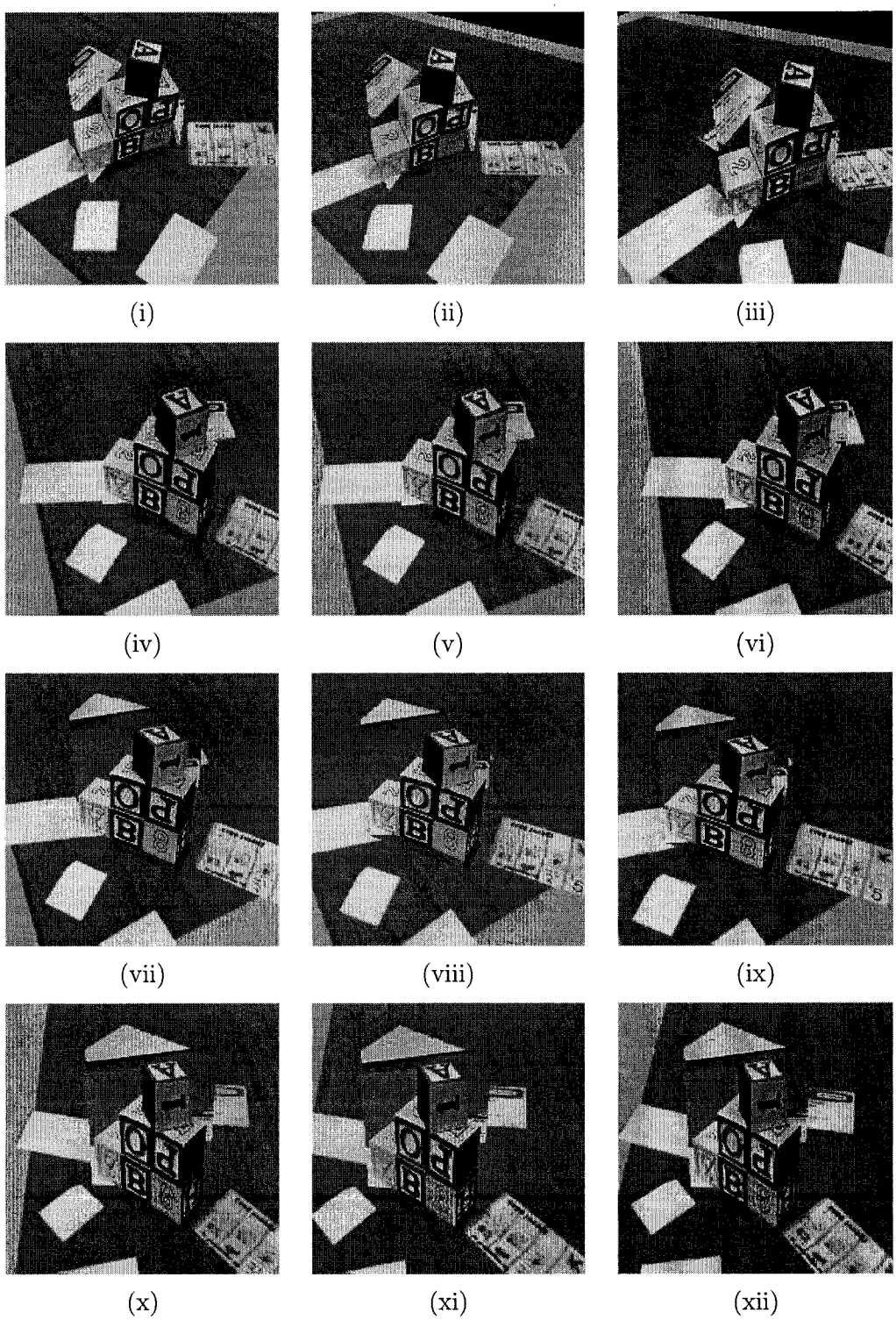


Figure 7.7: Keyframes of the synthesized sequence shown in Figure 7.2 generated by the virtual camera where the projections of the bounding boxes are used as opacity maps.

7.9 A Complete Application

In this section, we will go through the details of the algorithms proposed in this thesis by considering a complete application. In particular, four different alternatives of generating a virtual sequence for a specified path will be presented. Our procedure can be summarized as follows. We start with a group of images. Our proposed junction detection operator, JUDOCA, is to be applied to those images. Features on the ground are to be detected so that the remaining features are considered to belong to obstacles. Our wide baseline matching approach is to be applied to candidate pairs. The result will be a list of corresponding points; i.e., a match set. Then, the 3D location of every point in space can be estimated by triangulation. At this point, we will end up with a cloud of 3D points. Through our hierarchical structure we proposed, points in 3D space are to be clustered. Thus, a bounding box can be obtained for each obstacle (which can be used as a final representation for the obstacle in some applications). Through a space carving scheme, every bounding box can be split into voxels. Occupied and vacant voxels can be determined through shape from silhouette and color consistency observations. This will result in a 3D structure representing each obstacle. Another approximate representation of obstacles can be done via planar mapping. Two variants will be shown where silhouettes are to be used as opacity maps and alternatively the projections of the bounding boxes are to be used instead.

Now, let us consider the site shown in Figure 7.8 on Page 155 where eight images are available. Table 7.3 on Page 156 lists the camera parameters of each image using ω, ϕ and κ angles where the world origin is placed as shown in Figure 7.9(a) on Page 157. The error margins are assumed to be ± 10 mm, ± 10 mm, ± 10 mm, $\pm 1.0°$, $\pm 1.0°$ and $\pm 1.0°$ for ΔX, ΔY, ΔZ, $\Delta\omega$, $\Delta\phi$ and $\Delta\kappa$ respectively. The overhead view is generated using the method described in Appendix D. Different patches are mapped at their respective locations through triangulation as done in Equation (2.77). The final result of the overhead view is shown in Figure 7.9(b) on Page 157.

Let us consider the pair (f)-(g) shown in Figure 7.8. Junctions detected by the JUDOCA operator, described in Chapter 3, are shown in Figure 7.10 on Page 158. The features that putatively belong to the ground are detected using the algorithm explained in Chapter 4. This step results in a match set that putatively belong to the ground plane. This set is shown in Figures 7.11(a) and (b) on Page 159. Applying RANSAC scheme to this match set provides an updated version of the ground homography matrix. Consequently, more features that belong to the ground plane but do not represent match pairs may be identified. These features are shown in Figures 7.11(c) and (d).

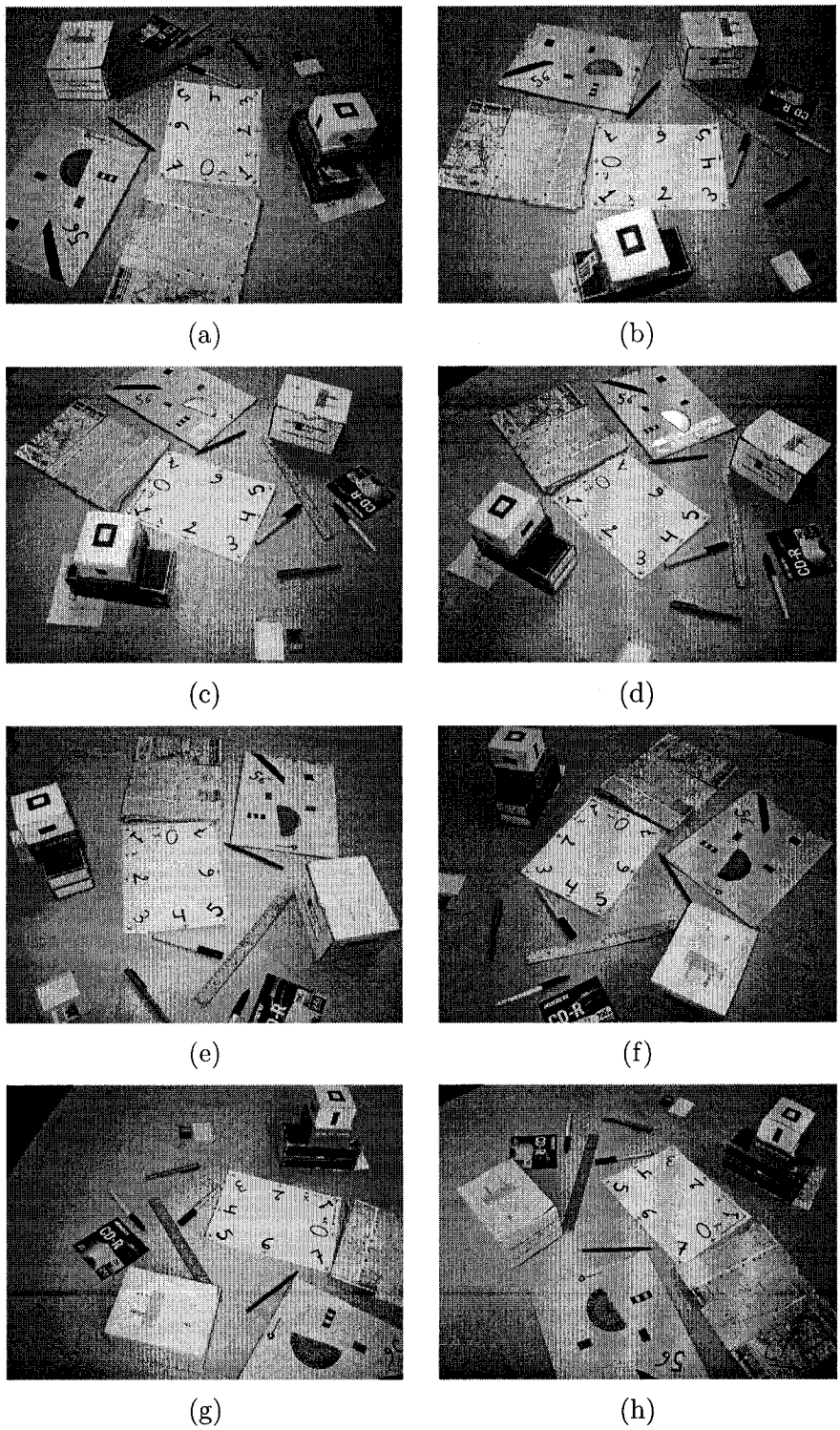


Figure 7.8: Images of the scene (courtesy of H. Hajjdiab).

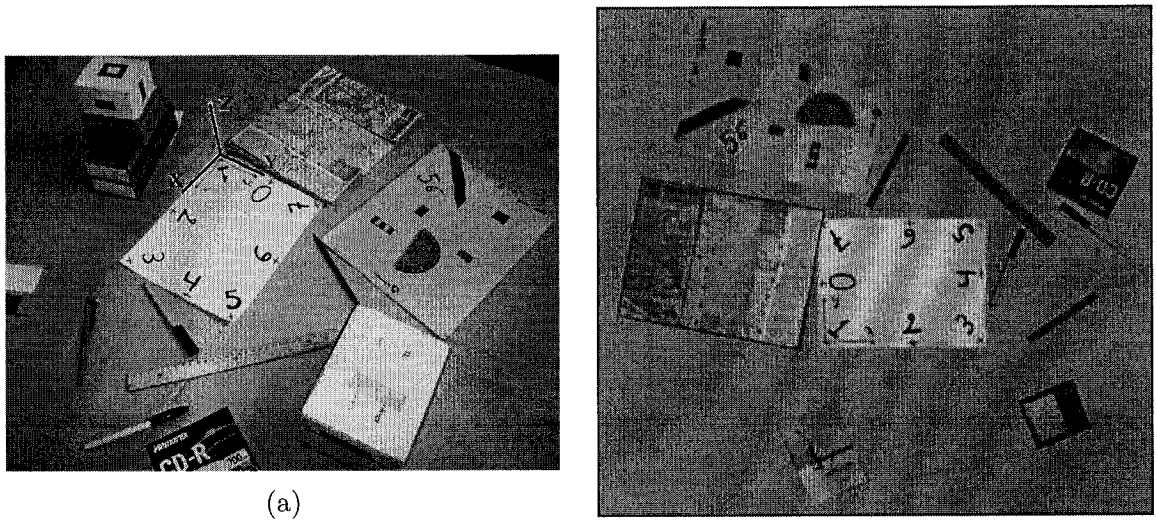
Image	X	Y	Z	ω	ϕ	κ
(a)	-386.8721	91.3538	761.4270	2.5962	-30.9155	-83.5224
(b)	83.8084	-317.5293	763.2127	30.5821	-0.6745	0.3239
(c)	351.8765	-349.7548	749.8407	29.6836	12.3346	21.0880
(d)	509.2151	-238.6600	742.8595	24.9641	23.0747	40.9618
(e)	620.7139	77.2922	742.3834	7.5144	32.0715	82.2833
(f)	577.4111	422.3900	753.2084	-15.2744	28.9333	125.6392
(g)	170.9942	657.6221	752.5581	-31.5844	-7.1256	-165.1466
(h)	-220.2339	531.4125	755.2014	-21.5670	-22.6891	-130.9459

Table 7.3: The parameters for each camera of the set shown in Figure 7.8. The world origin is at point "1" at the corner of the central paper, the $X-$axis is along "1" - "3" direction and the $Y-$axis is along "1" - "7" direction. Thus, the $XY-$plane coincides with the ground surface. The $Z-$axis is vertical or perpendicular to the ground plane. The units are in millimeters for X, Y, Z and in degrees for ω, ϕ and κ. Each image is 640 $\times$ 480 pixels. The focal length is 3.6 mm. The format size is 3.5311 $\times$ 2.7446 mm.

The operation of extracting the ground features leaves two sets of candidate points to be wide baseline matched. Those sets are the complement of the feature sets shown in Figures 7.11(c) and (d) where the universal sets are those junctions detected by the JUDOCA operator. Wide baseline matching algorithm explained in Chapter 5 is applied to candidate junctions. Using a window of 15 $\times$ 15 pixels, homographic SAD is applied. The threshold used to match features on obstacles is, t_{SAD}, 6200. The results of the above pair are shown in Figures 7.11(e) and (f) on Page 159 with a success rate of 83.3%. Of course, applying another threshold may improve the overall percentage; however, we want to show the ability of the clustering procedure to exclude outliers that might occur.

The match set obtained above is turned into 3D point set through triangulation using Equation (6.1). Due to the inaccuracy of camera parameters, some 3D points may be estimated within a narrow range under the ground plane. If this case happens, the third term of their vectors is then put to 0 so that all points become on or above the ground surface.

The algorithm discussed in Chapter 6 to cluster points in 3D space is applied. Using the hierarchical structure we presented – to the outcome set of 3D points obtained from the pair (f)-(g) – resulted in the bounding box shown in Figure 7.12(a) on Page 160. Four clusters turned roots at the third level of the hierarchical structure. One of these clusters contains 9 points while each one of the other 3 contains only one point. The difference between the correct and wrong clusters can be distinguished by the difference in number of points of each cluster where clusters that represent obstacles are of larger number of points. The other



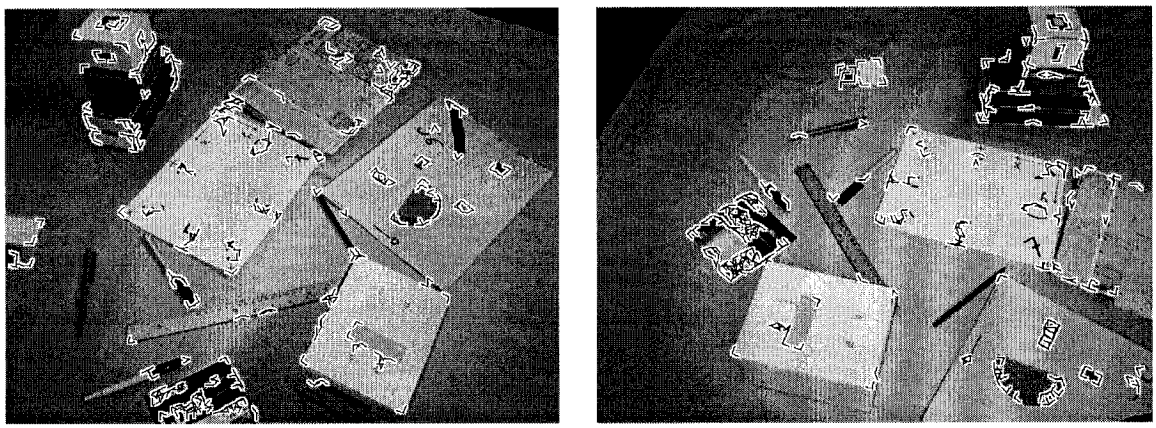
(a) (b)

Figure 7.9: (a) The world coordinate system is shown where the world origin is located at a corner of a paper on the ground surface and where the $XY-$plane coincides with the ground surface and the $Z-$axis is vertical. (b) The overhead view generated using the method described in Appendix D.

min_x	max_x	min_y	max_y	min_z	max_z
-24.12	178.79	-193.87	-53.21	0.00	189.48
206.97	369.81	366.25	498.56	6.52	112.90

Table 7.4: The dimensions and locations of the bounding boxes for the obstacles using the couple of pairs (f)-(g) and (h)-(a).

clusters represent isolated points. Thus, a threshold that restricts the minimum number of points in clusters may exclude these incorrect clusters (we used 5 in this example). Notice that, in this point set, only one match is detected on one of the obstacles; hence, no bounding box is obtained for that obstacle. Also, notice that a part of the other obstacle could not be included in its corresponding bounding box as the point set did not cover its whole surface area. In order to improve the obstacle localization results, the output of another image pair can be added prior to clustering. This is what has been done in Figure 7.12(b) where the pair (h)-(a) has been utilized. The dimensions and locations of the bounding boxes are listed in Table 7.4 on Page 157.



(a) (b)

Figure 7.10: Examples for detecting junctions where the JUDOCA operator is applied to Images (f) and (g). Junctions are superimposed on image. The parameters used are $\sigma = 0.9$, $t_B = 9$, $\lambda = 10$ and $s_J = 0.5$.

7.9.1 Sequence Generation

Suppose that the operator requests to generate the sequence as seen by a virtual camera moving along the path shown in Figure 7.13 on Page 160 where the black circle represents the start point and the arrow represents the end. The viewing direction, is determined so that the camera points along the direction of the motion as in Equation (7.10).

From the results obtained above, a virtual sequence can be generated. Different alternatives may be obtained. These alternatives are presented in the next subsections.

7.9.1.1 Volumetric Representation

There are two possible volumetric representations. These are: representing each obstacle as a bounding box; and alternatively representing it more accurately through voxelization. Using the bounding boxes shown in Figure 7.12 along with the overhead view generated and shown in Figure 7.9(b), a sequence of images seen by a moving robot along the path shown in Figure 7.13 is generated. Figure 7.14 on Page 161 shows 20 frames of that sequence.

Using a window size of 15 × 15, silhouettes are extracted for all images through thresholds, t_S, ranging from 3150 to 8875. Following the silhouette extraction, voxelization process begins. The two bounding boxes are split into voxels. The resolution, or voxel side length, ϵ, used is 2 mm. Through this resolution and the sizes of the boxes listed in Table 7.4, the total number of voxels resulted are 687990 and 296676 for Obstacles (1) and (2) respectively. A voxel is declared vacant if its projection onto $t_{nS} = 1$ images falls out of the corresponding silhouette.

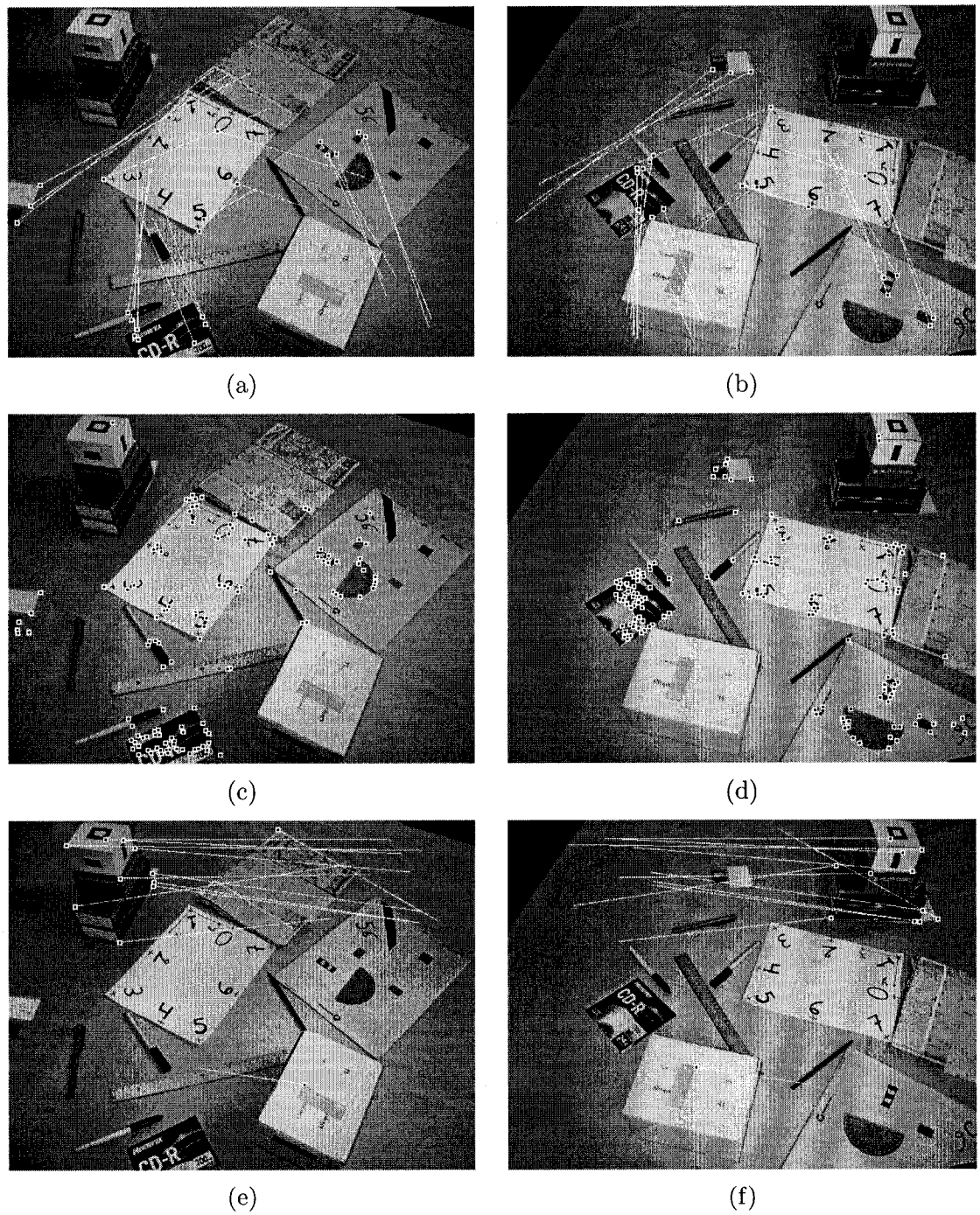
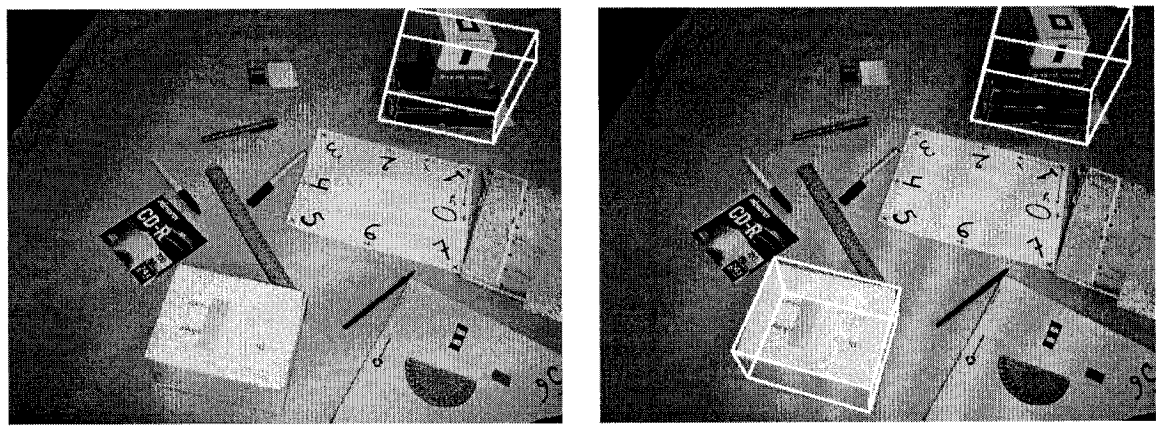


Figure 7.11: Using Images (f) and (g) of Figure 7.8. (a) and (b) The features detected on the ground that represent matches. This result is obtained using a window of 25 × 25 pixels where the parameters are $t_g = 0.6$ $t_h =$ is 20.0. (c) and (d) The features putatively belong to the ground surface after applying RANSAC and obtaining the updated homography matrix with parameters $t_{\text{H}} = 3.0$ and $t_g = 0.5$. (e) and (f) Homographic SAD correlation results of obstacle features. The parameters are $N = 15$ and $t_{SAD} = 6200$.



(a) (b)

Figure 7.12: (a) The result of clustering obtained using the pair (f)-(g). The minimum distance used, $t_{\dot{M}}$, is 200 mm with minimum of 5 points. (b) The result obtained after adding the pair (h)-(a). The minimum distance used, $t_{\dot{M}}$, is 150 mm with minimum of 5 points.

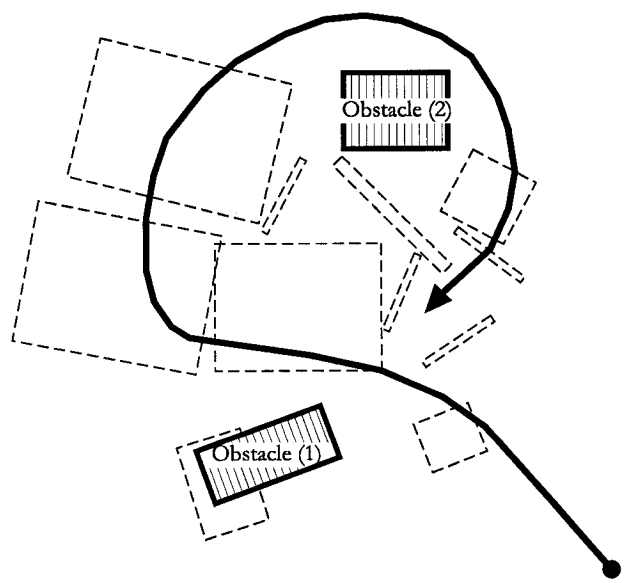


Figure 7.13: Representation of the top view of the scene under consideration where a path is determined by the operator to generate a sequence of image as seen by a robot moving along this path. The parameters of that sequence are $t_z = 280$, $f = 150$, $\tau = 40.0°$ and $\psi = 0.0°$ where distances are measured in pixels and angles are measured in degrees. The locations of the moving camera on the ground plane; i.e., t_x and t_y vary according to the position along the path. The pan, ρ, is determined so that the camera points along the direction of the motion as in Equation (7.10). The frame size requested is 200 $\times$ 200 pixels and the number of steps, n, is 200.

Figure 7.14: Twenty frames of a sequence requested by the operator. This sequence of images considers volumetric representation using bounding boxes.

Through this test 375479 and 7291 voxels are declared vacant, which represent percentages of 54.6% and 2.5% for Obstacles (1) and (2) respectively. Color consistency test threshold, $t_{\mathcal{I}}$, used is 50 while the minimum number of images, $t_{n\mathcal{I}}$, to be used for color consistency test is 3. By checking the connectivity of voxels, 220656 and 260484 of them where declared hidden in Obstacles (1) and (2) respectively representing percentages 32.1% and 87.8% for both obstacles. Figure 7.15 on Page 163 shows 20 frames of the sequence using voxelization method. These frames correspond to those of Figure 7.14.

7.9.1.2 Planar Representation

Again, a sequence of images seen by a moving camera along the path of Figure 7.13 is to be generated using planar representation. In this representation, we may use the silhouette images extracted before as opacity maps to extract obstacles out of their background and add more realism to the final appearance. Figure 7.16 on Page 164 shows 20 frames of the sequence using planar representation method. These frames correspond to those of Figures 7.14 and 7.15. Different original images were used for mapping and generating this sequence. However, extracting silhouettes may be time consuming. So, in order to save more time, silhouette extraction step may be skipped and the projections of the bounding boxes may be used for mapping instead. Figure 7.17 on Page 165 shows the corresponding 20 frames using this approach.

7.10 Summary

Volumetric representation presented in Chapter 6 could be time and space consuming especially if the resolution of discretization is too high and the bounding box volume is too big. It may not be suitable in cases where restrictions on time and space are present. Hence, we may need to find a compromise among time, space and final appearance through an alternative representation. We claim that a planar representation could be the solution.

This chapter enclosed two parts. In the first part, we discussed an approach to represent obstacles in a scene. It is an approximation to the appearance of the obstacle by a planar patch mapped onto the synthesized image. The planar patch was selected from the original set of images. The view was chosen as the closest possible one to the virtual camera viewing direction. Mapping through homography matrix was used to transform the image of the obstacle to its location in the synthesized image. The source area in the original image and the target area in the synthesized image were determined through the minimum and maximum coordinates of the projection of the bounding box of a given obstacle. (Those bounding boxes were obtained as proposed in Chapter 6.) Silhouette images might be used as opacity maps in order to exclude the background that might be present and add realism

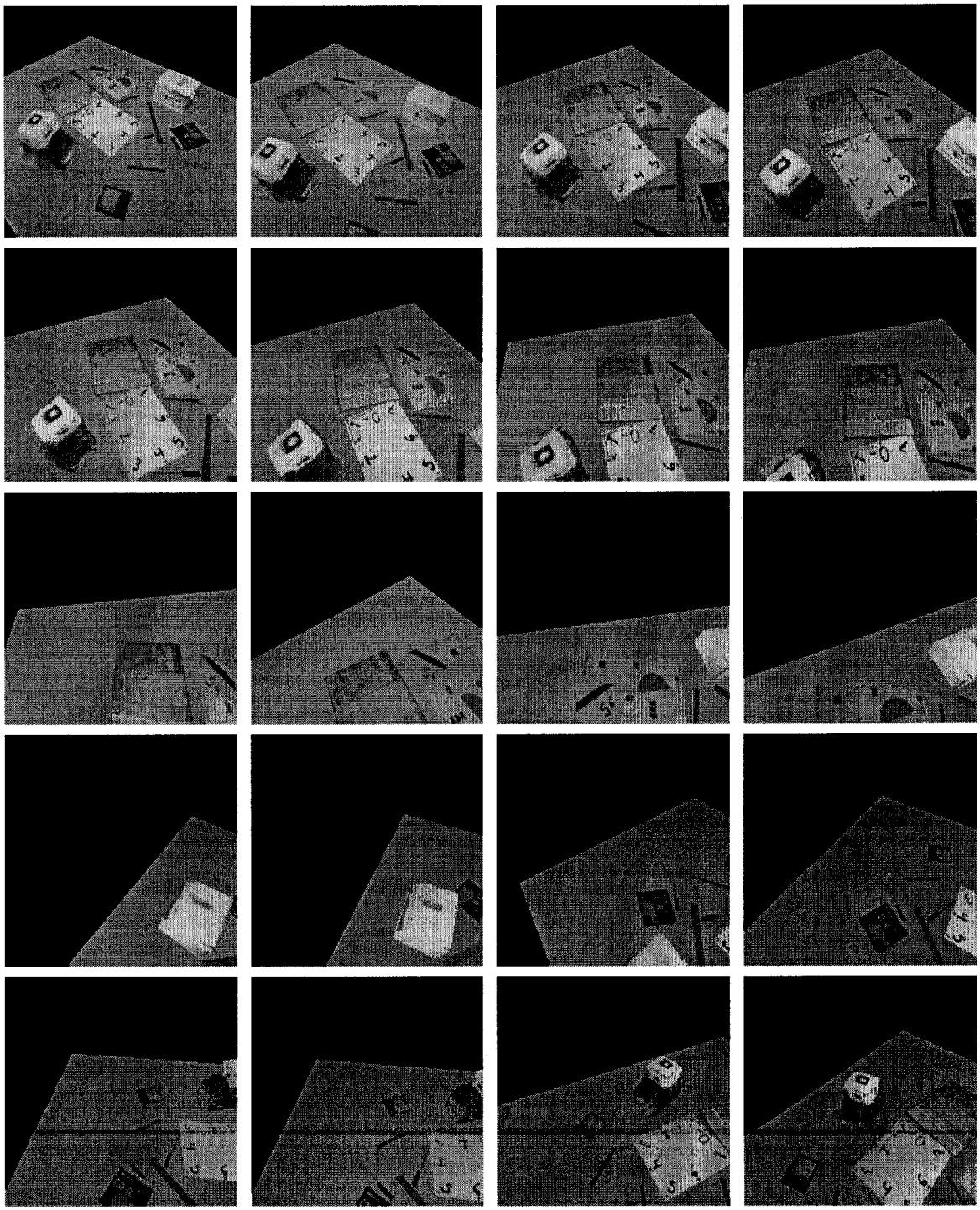


Figure 7.15: Twenty frames of a sequence requested by the operator. This sequence of images considers volumetric representation through voxelization.

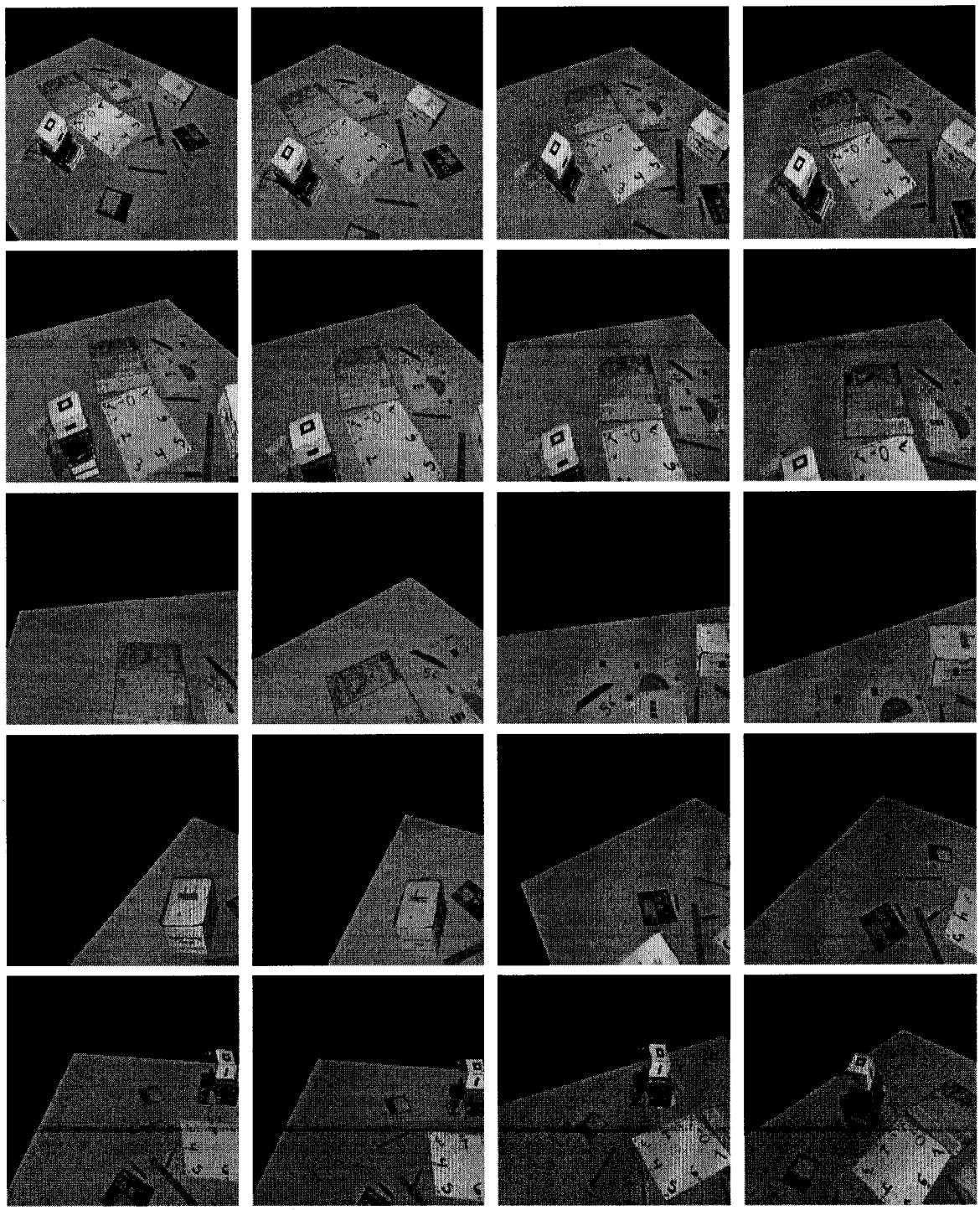


Figure 7.16: Twenty frames of a sequence requested by the operator. This sequence of images considers planar representation where the silhouette images were used.

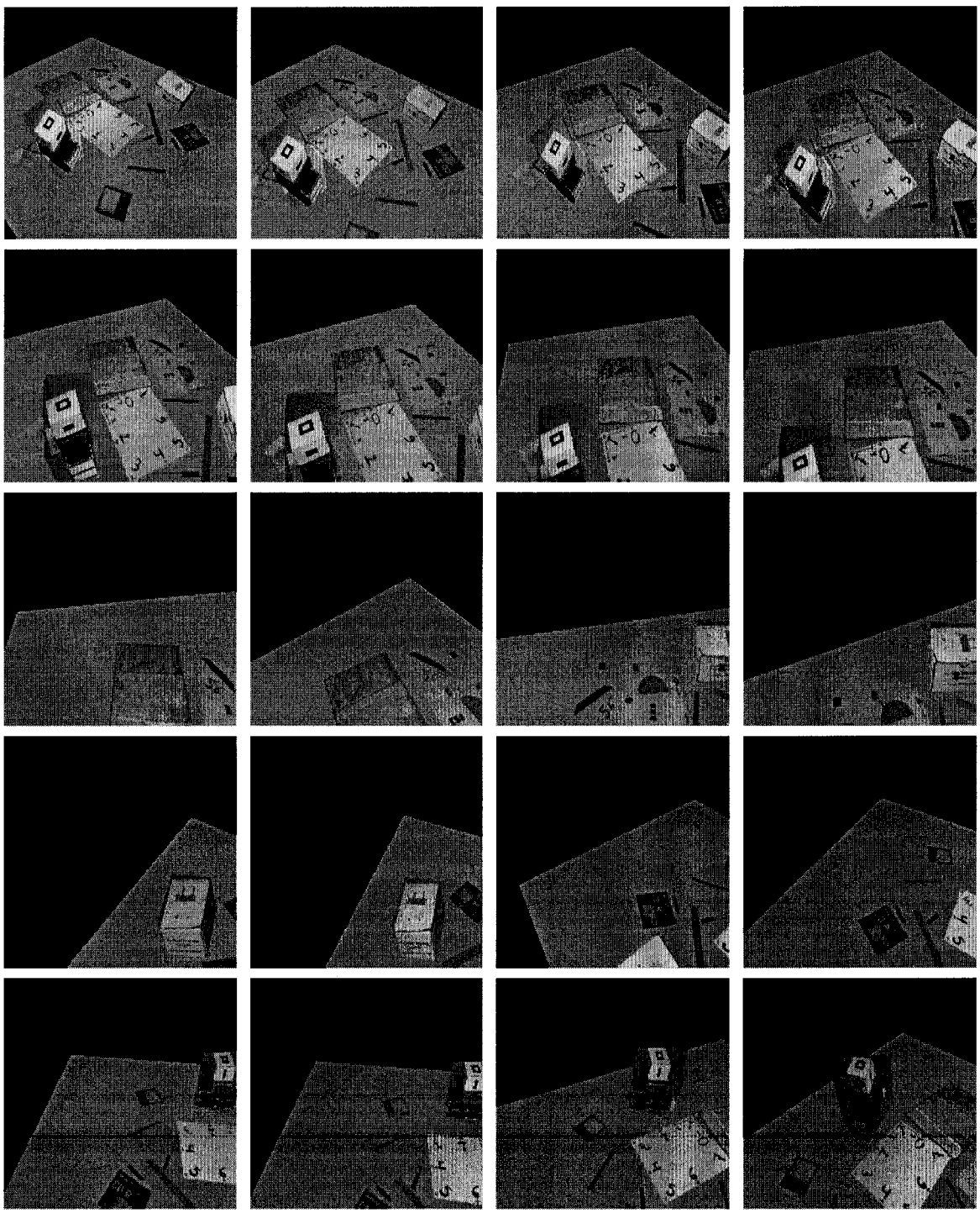


Figure 7.17: Twenty frames of a sequence requested by the operator. This sequence of images considers planar representation where the projections of the bounding boxes were used.

to the final synthesized image. Instead, the projections of the bounding boxes might be used in order to save time consumed by the silhouette extraction step.

The second part of this chapter presented an application where a site containing two obstacles was considered and a sequence of images, seen by a virtual camera moving along a predefined path, was requested. We went through the steps of our modeling system as applied to this example. Two different volumetric representations were considered; as bounding boxes and as voxels. Planar representations displaying obstacles as planar patches were also considered where silhouette images and bounding box projections were used as opacity maps. Sequences of images requested were obtained.

Chapter 8

Conclusions

In this thesis, we address the problem of scene modeling of a work site through different approaches. We assume the existence of a planar ground surface as well as a few obstacles that must be detected and avoided by a moving robotic vehicle. Moreover, a few wide baseline images are available for the work site. This implies that a small number of scene points may be visible in more than two views. In addition, camera parameters; i.e., positions and orientations, are assumed to be obtained through the use of sensors such as GPS, inertial devices or other measuring tools. According to the error margins of the sensor used, the parameters read should lie within some range. A sequence of synthesized images of the work site is required by the operator. Every synthesized image should contain a representation of each obstacle as well as the ground plane.

8.1 Summary of the Proposed Algorithms

We suggest a number of approaches to model the scene under consideration. Each of these approaches starts with detecting junction features in images using an operator we proposed and called JUDOCA [LE, LE04]. This operator defines junctions as points where linear ridges in the gradient domain intersect. It provides the location of the junction, the number of edges forming it as well as their orientations accurately for our task. The scene modeling algorithm then proceeds by determining the approximate inter-image ground plane homography between every pair of images. We suggest two different ways to estimate the homography matrix that relates the planar ground surface between a pair of images. The first method utilizes the approximate parameters of the camera to calculate what we call the planar perspective matrix and hence the inter-image homography matrix can be estimated. The second uses the equation of the ground plane; e.g., $[0, 0, 1, 0]^T$ assuming that the $Z-$axis is perpendicular to the ground. Hence, through this information and the possible error margins of the camera parameters, we should be able to detect matches that belong to the ground plane. Then, using a RANSAC scheme, we get an updated version of the ground plane homography matrix for

every pair of images. Consequently, we can identify other junctions on the ground plane where matches are not available as presented in [Eli04]. This step leaves features that putatively belong to obstacles. This facilitates the matching of feature points on obstacles that comes later.

Every 2-edge junction contributes to a plane in space. Using the information of the detected junctions and the approximate camera parameters, we calculate the equation of the plane on which the corresponding 3D junction should lie. Hence, the homography matrix that relates the images of that plane can be estimated. Through homographic correlation, matching of feature points that putatively belong to obstacles may be performed. Obeying the epipolar constraint as well as utilizing the error margins of camera parameters available, the search for matches is restricted to a strip surrounding the epipolar lines. Our homographic correlation approach has an impact comparing to non-homographic correlation as presented in [Eli04].

At this point, we end up with a set of point pairs or a match set. For every pair, we estimate the 3D location of the point in space by triangulation. Then, clustering of 3D points in space according to some proximity criterion can be performed through a hierarchical structure we propose that is based on the concept of irregular pyramids [Eli03a]; hence, a bounding box for each obstacle can be obtained. All approaches we suggest share the algorithm up to this point. For some applications, a bounding box may be sufficient to represent the obstacle, which is the first volumetric representation we suggest as an output for the sequence.

We suggest another volumetric representation that is based on a voxel occupancy scheme as proposed in [EL04]. Again, applying RANSAC to every match set, we get an updated version for the fundamental matrix, F, for every pair of images. Then, more accurate projection matrices, P, may be estimated. Using the above set of homography matrices and the projections of the bounding boxes, we extract silhouette-like images for the obstacles corresponding to every image in the set. We split the bounding boxes obtained previously into voxels according to some resolution and allocate space for these voxels dynamically. Finally, we use the projection matrices and the silhouette images to determine the voxel occupancy through color consistency test and shape from silhouette approach. Hence, another volumetric representation for each obstacle can be obtained.

We suggest another approximate method that represents obstacles as planar patches. The location, size and boundaries of those patches, in the synthesized image, are determined using the projections of the bounding boxes obtained previously. Mapping between the original images and the synthesized image is performed to render the images of the obstacle. Choosing which image to render is done through minimizing the distance among all candidate images between their corresponding pan angles and the viewing direction of the synthesized image.

The projections of the bounding boxes may be used as opacity maps at this step. Alternatively, silhouette images may be used as opacity maps in order to exclude the projection of the ground plane and add more realism to the synthesized image.

One last point related to all approaches suggested is representing the planar ground surface. The ground plane is displayed through what we called the planar perspective matrix. This 3×3 homography matrix can be estimated through the virtual camera parameters. The estimation of this matrix through different rotation systems was discussed in the thesis and in [EL03] as well. In addition to displaying the ground plane, this matrix may be used to relate the overhead view of the work site to the image plane. Hence, it may be used to relate the projections of the ground plane between a pair of images as mentioned earlier.

8.2 Complexity Analysis

As mentioned in Section 1.4, we may categorize the steps of our algorithm in the following stages.

Step 1: Representing the ground plane;

Step 2: Junction detection;

Step 3: Ground feature matching;

Step 4: Matching on arbitrary planes;

Step 5: Point reconstruction;

Step 6: Point clustering; and

Step 7: Volumetric and planar representation.

In what follows, we will estimate the time to be consumed by each of the above-mentioned steps.

Step 1: Obviously, it takes constant time; i.e., $O(1)$, to turn the virtual camera parameters into the planar perspective matrix to display the ground surface in the resulting synthesized image.

Step 2: If the number of pixels in the input image is N_p and every pixel is processed to determine if a junction is present at that location, then the time consumed by this junction detection step is linear. In other words, the cost of this step is $O(N_p)$.

Step 3: Junctions detected in two images form two disjoint sets that contribute to a bipartite graph. At the ground feature matching step, a link is sought between two candidates; each from a different set. If the number of junctions in the left image is N_l and the number

of junctions in the right image is N_r, then the time spent by this step is $O(N_l N_r)$ taking into consideration that a number of operations is required to be performed for each pair of candidates as explained in Chapter 5.

Step 4: Matching on arbitrary planes step has the same time complexity as the previous step; i.e., $O(N_l N_r)$ taking into consideration the number of operations to be performed for SAD and VNC as explained in Chapter 5.

Step 5: At this step, constant time; i.e., $O(1)$, is required to reconstruct a point in space through its projections onto images. If the number of match pairs is N_m, then the time consumed by this step is $O(N_m)$.

Step 6: The cost of merging two clusters into one larger one is quadratic since all points in one cluster are checked against all points in the other cluster. Suppose that the number of points in clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ are $N_{\mathcal{C}1}$ and $N_{\mathcal{C}2}$, then the time cost will be $O(N_{\mathcal{C}1} N_{\mathcal{C}2})$ taking into account that several clusters may be present at each level of the hierarchical structure.

Step 7: For the silhouette extraction phase, if the number of images is N_i and the number of pixels to be processed is N_p, then the time required to extract a silhouette is $O(N_p N_i)$ taking into account the number of operations to be performed for correlation. For the voxelization phase, if the number of images is N_i and the number of voxels is N_v, then the time required to process all voxels is $O(N_i N_v)$. In case of planar representation, it takes constant time to determine the projections of the bounding box corners and calculate the matrix needed for mapping.

It is evident from the complexity analysis that the steps with quadratic time complexity are the dominating factors; i.e., those that require the largest amount of processing. However, one should notice that inputs to different steps are quite diverse. While the inputs to the junction detection step are pixels of input images and the inputs to the matching step are sets of junctions and pairs of images, the inputs to the point reconstruction step are locations of junctions and for the point clustering step is a set of 3D points. Finally, the inputs to the voxelization step are a group of images in addition to bounding boxes; i.e., a variable number of voxels. Thus, the amount of time consumed by every step depends not only on the complexity but on the type of inputs as well. For example, although the complexity to merge two clusters is quadratic, all clusters together can be processed in real time in most cases since usually the number of points is not large at this step. Also, while the time complexity of Step 2 is linear, this operation can be performed in real time as we work on two binary images, which reduce the time spent considerably. Moreover, although the complexities of Step 3 and Step 4 are the same, the time consumed by Step 3 is expected to be more than that of Step 4 since at Step 3, all junctions are to be processed while only a selective set of the junctions is processed in Step 4.

8.3 Discussion

Algorithms developed may work on both gray-scale and color images resulting in different success rates. Different examples were presented throughout the thesis to show the effectiveness of the techniques we presented.

Binary, gray-scale and color images: As the position and shape of the any junction can be detected through the edges forming it, binary versions of the edge maps are used at the junction detection stage to accelerate the process. In order to perform matching, gray-scale as well as color images are used to collect information from the intensity and color channel values. Color channels may be used in the wide baseline matching phase that often produce better match set comparing to using the gray-scale versions of the images. However, the following stage; i.e., clustering, may not be affected if gray-scale or color images are used as the information inferred from the images are irrelevant in this case. Color images play a major and key role at the voxel occupancy determination stage. For presentation purposes, we used color images for volumetric and planar final representations of the obstacles.

Number of images: At the junction detection stage, individual images are to be operated on separately. Pairs of images are used at the homographic correlation stage to detect both ground and obstacle matches. Notice that, because of the wide baseline constraint, a small number of feature points may appear in more than two views. One of our concerns was to obtain the largest number of matches through the minimum processing time we could. Thus, a minimum number of images is used to match features in order to obtain the largest match set to be used for reconstruction. Considering this fact, pairs – rather than triplets – of images are considered at the matching phase. Finally, after this stage and locating the points in the Euclidean 3D space, all available images are exploited in order to achieve a good volumetric or planar representation for a given obstacle.

Feature detection: We explained that in order to facilitate the matching process, interest points should be detected as a primary step. A corner detector can provide information on the locations of interest points in images. Any corner point detector may be used as this stage to find feature points. However, such a detector concerns mostly with the positions of those points. Hence, we developed a junction detector, called JUDOCA, that not only detects the locations of such points but provides information on the shape of the areas surrounding them as well. We assumed that the locations of those points happen where ridges in the gradient domain of the image intersect. The development of our operator made the matching process much easier.

Ground feature detection: Through the approximate camera parameters along with their corresponding error margins provided by the sensor or the measuring device used, we can remove most of the possible matches that belong to the ground plane. Although at this stage, it is not our primary goal to detect match points on the ground plane, the existence of such a step relieves some pressure off the next step; i.e., the matching step. Few of the ground features may leak to the next step. There are some reasons why this step may not detect all ground features. One reason is that some of the real matches might be occluded behind objects in the other image or having their correspondences outside the boundaries of the other image. Also, the inaccuracy in camera parameters reflects on the homography matrix calculation. This may cause the correlation value to trigger the given threshold and hence abandon right matches. Another reason is that some feature points might not have correspondences because no junctions could be detected at their corresponding locations. Hence, a further RANSAC phase is used afterwards to update the homography matrix and remove other ground feature points that do not constitute match pairs. Moreover, using the updated versions of the homography matrices may solve the problem of triggering the threshold due to the inaccuracy in camera parameters. In our experiments, we found that enlarging the correlation window at this stage as well as applying conservative thresholds may enhance the results. Another point to be mentioned here is the use of triangulation at this stage. We found that triangulation and checking the height of the point in space may avoid detecting mismatches at the occluded areas.

Homographic correlation: As perspective deformation is likely to happen in case of wide baseline correlation, the homographic SAD correlation we propose gives better results than the non-homographic SAD correlation to match wide baseline pair of images. This measure is invariant to projective transformation, which dominates the case of wide baseline imaging. The achievement reached by homographic VNC correlation improves the correlation approach within a limited range according to the variance in lighting conditions among views. Also, color images tend to give better results where more information supplied through the three color channels is utilized instead of only one gray-scale channel. Experimental tests show that the larger the correlation window, the better the results. This is expected as the correlation of a mismatch pair tends to diverge significantly with the extension of the correlated area through homography matrix.

SAD versus VNC homographic correlation: It is evident that the number of operations to be performed to calculate SAD and VNC varies substantially. It is much easier, simpler and faster to use the SAD correlation. However, there are a number of factors to consider. The first consideration is that the use of VNC homographic correlation outperforms the SAD version in situations like changing in lighting conditions. Another consideration is that with the advances of today's hardware, the speed of the processors may reduce the execution

time to perform VNC calculation and make the difference between SAD and VNC negligible in many cases.

Hierarchical clustering: The concept of irregular pyramids, that is used usually on images, is used in a different way in this thesis on a set of space points. Clustering of space points is done hierarchically according to a proximity criterion. This results in a more global information pertinent to each obstacle separately. Hence, this stage may be used to localize obstacles by itself and may be used as a final result for a robot to navigate safely in its environment.

Voxelization: Note that the error margins of camera parameters may be used with the initial rough estimates of the ground homography matrix to extract the silhouettes by searching for the right match within a specific range. However, extracting silhouettes this way might be expensive in terms of run time. Instead, finding matches that belong to the ground plane and hence applying RANSAC to get an accurate version of a homography matrix before using it to extract silhouettes may save time. Moreover, at this stage, the whole bounding box volume may digitized into voxels and checked for color consistency. However, we find that the silhouette extraction step plays a key role in determining the occupancy of a voxel by assuring whether a voxel is vacant. Finally, experiments show that checking the connectivity of a voxel may considerably save space and hence memory when processing and displaying the final synthesized image.

Silhouette extraction: The process of silhouette extraction relies on how accurate the homography matrix of the ground is estimated. This matrix, calculated through RANSAC procedure, relies, in turn, on the matches detected behind the obstacles in the different views. For example, if no matches were detected in the background of an obstacle due to occlusion or lack of junctions detected, the accuracy of the homography matrix may be affected. One solution to this problem can be done by stretching the parameters of the JUDOCA operator in order to increase the number of junctions detected and hence increase the chance to detect matches and obtain accurate homography matrix.

Updating calculations: RANSAC scheme is used twice in this thesis; once to update the homography matrices of the ground plane and another time to update the fundamental matrices following the matching process. The enhanced homography matrices are used twice; once to detect the ground features prior to matching features on obstacles and another time to extract silhouettes. Updated fundamental matrices may also be used twice; once to exclude mismatches that might occur at the matching stage and another time to update the projection matrices for subsequent voxelization and silhouette extraction steps. In our experiments, skipping RANSAC step applied to the match set to get enhanced fundamental matrices may

not affect the clustering process as clustering space points may exclude possible mismatches as those mismatches will likely result in isolated 3D points.

Performance issues: Execution time varies from one phase to another in our system. Using a Pentium 4 machine at running 1.8 GHz, we measured 1.01 and 0.93 seconds to detect 322 and 301 junctions for the pair shown in Figure 3.10 on Page 49 (both images are of size 640×480 pixels). We measured an execution time of 32.50 seconds to detect ground feature matches between the same pair of images. Further processing with the updated homography matrix consumes 15.66 seconds while homographic VNC correlation takes 4.25 seconds to execute both directions. Alternatively, homographic SAD correlation takes 1.61 seconds to execute both directions as well. Execution time for clustering the output of 2 pairs; that is 23 points, is 0.38 seconds.

Of course, at the voxelization stage, the resolution plays a key role resulting in different number of voxels and hence affects the execution time. Using a size of 1 mm for a voxel side length in the same example results in 947376 voxels, which takes 59.62 seconds to detect all surface voxels. The most expensive step in voxelization is the silhouette extraction stage. Again, depending on the area of the projected bounding box onto the image, the execution time for silhouette extraction may vary significantly. For example, the execution time for the image shown in Figure 3.10(a) is 838.39 seconds. Unlike the previous steps, the correlation here is applied to all pixels, which results in extended execution time. Hence, this step may be ignored in the planar representation option where the whole projection of the bounding box may be used as done in Figure 7.17 on Page 165. (Execution time to detect the projection of the whole bounding box for the same image takes only 0.51 seconds.)

Which representation is the best? In this thesis, we presented several methods to model the scene under consideration. These methods include volumetric representations as bounding boxes and as voxels; and planar representations where the projections of the bounding boxes were used or alternatively silhouette images might be used as opacity maps to add more realism to the output sequence. These ideas may result in 4 different final appearances for the same sequence. So, which one is the best amongst them? This depends on the target goal. For example, if the main concern is to avoid collision, bounding boxes representation might be enough. There is a very important issue relating to this representation which is the time consumed to output a sequence of that representation that is the fastest amongst all. However, if more accurate volumetric representation is required, then voxelization might be deployed with the cost of more time to be consumed but, at the same time, with the privilege of more details to be available. In contrast to bounding boxes representation, voxelization provides images rendered with different accuracy levels depending on the resolution used. Planar representation provides rendered images where the voxelization step is skipped and

hence time consumed by it might be saved. If it is required to be save more time while preserving the rendering option, then planar representation might be used where the silhouette images are replaced by the projections of the bounding boxes.

8.4 Future Research

One important topic for future research is using the available information to automatically adjust the parameters and thresholds. For example, the information on camera parameters like the height of the camera and focal length gives an idea about the volume seen and how large it is. Through this information, the threshold $t_{\dot{M}}$, used with the proposed hierarchical structure to cluster points in space, may be adjusted. Also, knowing the bounding box sizes and the dimensions of the requested synthesized sequence frame may be used to determine the voxel size by adjusting the parameter ϵ.

Several other research ideas have emerged out of our work in this thesis. We have implemented some of them partially and are planning to pursue the rest of them in the near future. We may categorize those ideas in the following topics.

3D reconstruction: In some application, we may use the plane equations constructed by junctions to build the surfaces by grouping or clustering planes with close equations according to some proximity criterion and hence getting average plane equations. Then texture mapping is performed to render the area enclosed by each patch and defined by a homography matrix. This may be a good idea for scenes where planar surfaces dominates the environment.

Another idea pertinent to planar equations comes after clustering points in 3D space to obtain bounding boxes for obstacles. The bounding boxes may be split into parallel planes with limited areas. The equation of each planar patch can then be calculated and homographic correlation is to be applied to the planar patches to extract the occupied voxels. However, those ideas may work better in case where considerable planar patches may exist; e.g., indoor scenes or architecture models and buildings.

We have noticed through our work with voxelization that some holes in the resulting object may appear that we might need to fill in. As well, some boundary voxels may be present and might be needed to erode away. As morphological operators; e.g., erosion and dilation, could be used on 2D binary images, we suggest using 3D morphological operators on voxels. In Image Processing, the structuring element is usually a 2D array characterizing the effect of the operation. In *3D voxel morphology*, a 3D array should be used as a structuring element. There are two states when working on binary images; i.e., foreground and background. The corresponding two states for 3D voxel morphology is being occupied or vacant.

Stereo matching: In this thesis, we have used the information about approximate camera parameters to facilitate the matching process between pairs of images by constructing the planes of junctions – detected by JUDOCA operator – in space and performing homographic correlation. In cases where no camera parameters are known, we suggest applying JUDOCA operator to both images. This is followed by slicing the circular section enclosed by every 2-edge junction into radial rays radiating from the position of the junction according to some factor. SAD correlation of Equation (5.1) or VNC correlation of Equation (5.2) is to be applied to corresponding rays in both images. The overall correlation value for the junction should indicate the best match. We have implemented several steps of this approach. The results have shown promise.

Given the same conditions mentioned above, we suggest another correlation approach. We start by applying JUDOCA again to both images. SAD correlation of Equation (5.1) or VNC correlation of Equation (5.2) is applied to the triangular area enclosed by the position of the junction and the CA points through a 3×3 affine transformation matrix. Affine transformation matrix has 6 DOF's and can be fully defined through using 3 pairs of points.

Stereo compression: Following the idea of hierarchical processing, our work in Chapter 6 and [Eli03a], we suggest using the concept of irregular pyramids for stereo compression. In [EL99], we presented an approach based on the concept of irregular pyramids to segment a stereo pair according to disparity vectors. We propose to extend this idea by using its segmentation results to store one image and information on the second. This information may include disparity vectors and region boundaries according to the segmentation results. The boundaries may be represented through run length coding (RLC).

Image transmission: Another idea that utilizes the concept of irregular pyramids may be suggested to hierarchically transmit an image through a region growing algorithm. This algorithm segments an image into regions according to the intensity levels. In ordinary hierarchical transmission, the average of a squared region may be transmitted along different levels of a pyramid. Here, we suggest that irregular regions shaped according to the segmentation results are to be transmitted along different levels. Only the difference between each two consecutive levels is to be transmitted. This results in crisp boundaries along different levels of abstraction comparing to blurred regions in ordinary hierarchy.

Segmentation of the ground plane: We presented a new method for extracting silhouettes based on determining the projections of the bounding boxes on the original images and applying correlation among candidate images. The same idea may be applied in case that the ground surface is required to be segmented and extracted from the scene.

Parameters recovery: An idea to recover accurate extrinsic camera parameters, if the focal length and the height of the camera are given, may be presented. Its algorithm starts with detecting junction using the JUDOCA operator. Matching is then applied through homographic transformation. RANSAC scheme is applied to obtain an updated fundamental matrix. Consequently the fundamental matrix is transformed into the essential matrix through the calibration matrix. Singular value decomposition (SVD) is applied to decompose the essential matrix into a rotation matrix and a translation vector. The rotation angles can be recovered from the rotation matrix as explained in Chapter 4. In the decomposition, the translation vector will be normalized; however, assuming the height of the camera, the other two parameters of the translation vector can be recovered.

Vector graphics: In order to convert from raster images; e.g., `.pgm` format, to vector drawings; e.g., `.dwg` format, we suggest using the circumferential anchors idea that has been used in our junction detector. At the end of each scanned line, we assign a weight to indicate the length and angle of inclination of that line. After scanning the whole image, we combine the lines according to their weights. (Of course, this idea will approximate curves into line segments.)

Virtual environments: In the SMART project, we have applied the planar perspective matrix we derived in Chapter 4 to the overhead view mosaics of the scene as we have done in [EL00]. In order to create a virtual tour for existing objects, we suggest a similar approach. We start by acquiring a set of images that are to be grouped by mosaicing to get a panoramic image for the scene. Then spherical mapping is applied to the top semi-sphere or the dome on top of the virtual camera. We have used the same idea to render the sky image in [Eli99b]. Java language could be used for easy use with a Web browser.

Appendix A

Parameters and Thresholds

This appendix lists the parameters and thresholds used in our implementations throughout the thesis. Those parameters and thresholds are categorized by the operation they perform.

A.1 Ground Plane Representation

In order to estimate the planar perspective matrix, intrinsic and extrinsic camera parameters should be known. Note that this matrix is used to display the perspective projection of the ground plane as well as to compute the original inter-image homography of the ground plane. In case of displaying the ground plane, virtual camera parameters are used while camera parameters measured through sensors are used to compute the inter-image homography. In both cases, the parameters are as follows:

f	The focal length of the camera.
$[u_0, v_0]^T$	The principal point of the camera.
X	The camera location along the $X-$axis.
Y	The camera location along the $Y-$axis.
Z	The camera location along the $Z-$axis.
ω	The first rotation about the $X-$axis in (ω, ϕ, κ) system.
ϕ	The second rotation about the Y_1-axis in (ω, ϕ, κ) system.
κ	The third rotation about the Z_2-axis in (ω, ϕ, κ) system.
ρ	The first rotation about the $Z-$axis in (ρ, τ, ψ) system.
τ	The second rotation about the X_1-axis in (ρ, τ, ψ) system.
ψ	The third rotation about the Z_2-axis in (ρ, τ, ψ) system.

A.2 Junction Detection

The parameters and thresholds of the JUDOCA operator are the following:

λ	The length of the radial ray used to detect junctions through JUDOCA operator.
σ	The variance value that specifies the size of the Gaussian filter used to detect junctions through JUDOCA operator.
s_J	A threshold imposed to filter the strength of a junction.
$t_{\mathcal{B}}$	A threshold imposed on the gradient image $\mathcal{B}$ to get its binary version.

A.3 Junction Selection

At this phase, there are different parameters and thresholds ranging from intrinsic and extrinsic camera parameters (Section A.1) to other parameters and thresholds. These are as follows:

ΔX	A scalar quantity representing an error margin for the parameter X.
ΔY	A scalar quantity representing an error margin for the parameter Y.
ΔZ	A scalar quantity representing an error margin for the parameter Z.
$\Delta\omega$	A scalar quantity representing an error margin for the parameter ω.
$\Delta\phi$	A scalar quantity representing an error margin for the parameter ϕ.
$\Delta\kappa$	A scalar quantity representing an error margin for the parameter κ.
t_g	A threshold used to detect feature points on the ground plane.
t_h	A threshold used to exclude points that are too high or too low.
t_{H}	A threshold used with RANSAC to obtain an enhanced homography matrix.
N	Determines the size of the correlation window.

A.4 Wide Baseline Matching

The parameters and thresholds at this phase include intrinsic and extrinsic camera parameters and associated error margins in addition to the window size $(N+1)^2$. Those parameters are the same as in Section A.3. Moreover, the following thresholds are present at this stage:

t_{SAD}	A threshold used with the SAD correlation.
t_{VNC}	A threshold used with the VNC correlation.
t_{F}	A threshold used with RANSAC to obtain enhanced fundamental matrix.

A.5 Three-Dimensional Point Reconstruction

The parameters are those of the camera (intrinsic and extrinsic) as mentioned in Section A.1.

A.6 Three-Dimensional Point Clustering

The hierarchical structure we proposed to cluster points in space controls the operation through the following threshold:

$t_{\dot{M}}$	A threshold used with the proposed hierarchical structure to cluster points in space.

In addition to the above threshold, another threshold may be imposed on the number of points in the resulting clusters in order to exclude isolated point that might result from outliers.

A.7 Obstacle Representation

A.7.1 Voxelization

In addition to the window size $(N+1)^2$ and the parameters of the virtual camera, there are other thresholds at the voxelization stage. Those are listed below.

ϵ	A scalar quantity used to determine the resolution of the discretized voxels.
$t_{n\mathcal{I}}$	A threshold used to determine the minimum number of images for color consistency test at the voxelization phase.
$t_{n\mathcal{S}}$	A threshold used to determine the minimum number of images for a voxel projection to be part of the silhouette.
$t_{\mathcal{I}}$	A threshold used to determine the minimum difference for color consistency test at the voxelization phase.
$t_{\mathcal{S}}$	A threshold used with SAD correlation to obtain silhouette images.

A.7.2 Planar Representation

In order to represent the obstacles as planar patches, the virtual camera parameters (intrinsic and extrinsic) are used. In particular, the virtual pan angle is checked against the viewing direction in order to select the appropriate view for mapping.

A.8 Sequence Generation

At this stage, the virtual camera parameters (intrinsic and extrinsic) are used. Also, the size of each frame is determined at this stage. In addition, we have the following parameters:

n	The number of steps that determines δ, which is the step size for creating the sequence.
d	The real distance between two specified points in order to determine the scale factor.
$[x_0, y_0]^T$	The origin of the world coordinate system expressed in the top view pixel coordinates.
$[x_1, y_1]^T$	A point on the $X-$axis of world coordinate system expressed in the top view pixel coordinates.

Appendix B

Matrices: Details

This appendix presents the details of the planar perspective matrix, $\mathtt{H_z}$, the projection matrix, P, and the calibration rotation matrix, AR. Section B.1 of this appendix introduces the notation used. Sections B.3, B.4 and B.5 state the details of each term of these matrices.

B.1 Notation

$\mathtt{H_z}$	The planar perspective matrix.
h_{ij}	The term ij in $\mathtt{H_z}$.
P	The projection matrix.
$\dot{\mathbf{P}}_z$	The third column in P.
AR	The calibration rotation matrix.
f_x	The focal length of the camera along the $x-$direction.
f_y	The focal length of the camera along the $y-$direction.
ω	The angle of rotation about the $X-$axis (see Figure 4.3(a) on Page 58).
ϕ	The angle of rotation about the Y_1-axis (see Figure 4.3(b)).
κ	The angle of rotation about the Z_2-axis (see Figure 4.3(c)).
ρ	The pan angle of rotation (see Figure 4.4 on Page 60).
τ	The tilt angle of rotation (see Figure 4.4).
ψ	The swing angle of rotation (see Figure 4.4).
t_x	The translation of the camera along the $X-$axis.
t_y	The translation of the camera along the $Y-$axis.
t_z	The translation of the camera along the $Z-$axis.
$[u, v]^T$	The location, on the retinal plane, of the intersection between the optical axis and the retinal plane (see Figure 2.3 on Page 19).

B.2 The Translation Vector

According to Equation (4.20), the translation vector can be expressed as:

$$\tilde{\mathbf{T}} = \begin{bmatrix} -cos\ \phi\ cos\ \kappa\ t_x \\ -\ sin\ \omega\ sin\ \phi\ cos\ \kappa\ t_y - cos\ \omega\ sin\ \kappa\ t_y \\ +\ cos\ \omega\ sin\ \phi\ cos\ \kappa\ t_z - sin\ \omega\ sin\ \kappa\ t_z \\ \\ cos\ \phi\ sin\ \kappa\ t_x \\ +\ sin\ \omega\ sin\ \phi\ sin\ \kappa\ t_y - cos\ \omega\ cos\ \kappa\ t_y \\ -\ cos\ \omega\ sin\ \phi\ sin\ \kappa\ t_z - sin\ \omega\ cos\ \kappa\ t_z \\ \\ -sin\ \phi\ t_x \\ +\ sin\ \omega\ cos\ \phi\ t_y \\ -\ cos\ \omega\ cos\ \phi\ t_z \end{bmatrix} \tag{B.1}$$

$$= \begin{bmatrix} -\ cos\ \psi\ cos\ \rho\ t_x\ -\ sin\ \psi\ cos\ \tau\ sin\ \rho\ t_x \\ +\ cos\ \psi\ sin\ \rho\ t_y\ -\ sin\ \psi\ cos\ \tau\ cos\ \rho\ t_y \\ -\ sin\ \psi\ sin\ \tau\ t_z \\ \\ sin\ \psi\ cos\ \rho\ t_x\ -\ cos\ \psi\ cos\ \tau\ sin\ \rho\ t_x \\ -\ sin\ \psi\ sin\ \rho\ t_y\ -\ cos\ \psi\ cos\ \tau\ cos\ \rho\ t_y \\ -\ cos\ \psi\ sin\ \tau\ t_z \\ \\ sin\ \tau\ sin\ \rho\ t_x \\ +\ sin\ \tau\ cos\ \rho\ t_y \\ -\ cos\ \tau\ t_z \end{bmatrix}$$

B.3 The Planar Perspective Matrix

We can write the planar perspective matrix as:

$$\mathtt{H_z} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{B.2}$$

The following are the values of each term separately.

$$\begin{aligned}
h_{11} &= f_x \cos\kappa \cos\phi + u \sin\phi \\
&= f_x \cos\psi \cos\rho + f_x \sin\psi \cos\tau \cos\rho - u \sin\tau \sin\rho \\
h_{12} &= f_x \sin\kappa \cos\omega + f_x \cos\kappa \sin\phi \sin\omega - u \cos\phi \sin\omega \\
&= -f_x \cos\psi \sin\rho + f_x \sin\psi \cos\tau \cos\rho - u \sin\tau \cos\rho \\
h_{13} &= -f_x \cos\phi \cos\kappa t_x - f_x \sin\omega \sin\phi \cos\kappa t_y \\
&\quad - f_x \cos\omega \sin\kappa t_y + f_x \cos\omega \sin\kappa t_z - f_x \sin\omega \sin\kappa t_z \\
&\quad - u \sin\phi t_x + \sin\omega \cos\phi t_y - \cos\omega \cos\phi t_z \\
&= -f_x \cos\psi \cos\rho t_x - f_x \sin\psi \cos\tau \sin\rho t_x \\
&\quad + f_x \cos\psi \sin\rho t_y - f_x \sin\psi \cos\tau \cos\rho t_y - f_x \sin\psi \sin\tau t_z \\
&\quad + u \sin\tau \sin\rho t_x + u \sin\tau \cos\rho t_y - u \cos\tau t_z \\
h_{21} &= -f_y \sin\kappa \cos\phi + v \sin\phi \\
&= -f_y \sin\psi \cos\rho + f_y \cos\psi \cos\tau \sin\rho - v \sin\tau \sin\rho \\
h_{22} &= f_y \cos\kappa \cos\omega - f_y \sin\kappa \sin\phi \sin\omega - v \cos\phi \sin\omega \\
&= f_y \sin\psi \sin\rho + f_y \cos\psi \cos\tau \cos\rho - v \sin\tau \cos\rho \\
h_{23} &= f_y \cos\phi \sin\kappa t_x - f_y \sin\omega \sin\phi \sin\kappa t_y - f_y \sin\omega \cos\kappa t_z \\
&\quad - f_y \cos\omega \sin\phi \sin\kappa t_z - f_y \sin\omega \cos\kappa t_z - v \sin\phi t_x \\
&\quad + v \sin\omega \cos\phi t_y - v \cos\omega \cos\phi t_z \\
&= f_y \sin\psi \cos\rho t_x - f_y \cos\psi \cos\tau \sin\rho t_x \\
&\quad - fy \sin\psi \sin\rho t_y - f_y \cos\psi \cos\tau \cos\rho t_y - f_y \cos\psi \sin\tau t_z \\
&\quad + v \sin\tau \sin\rho t_x + v \sin\tau \cos\rho t_y - v \cos\tau t_z \\
h_{31} &= \sin\phi \\
&= -\sin\tau \sin\rho \\
h_{32} &= -\cos\phi \sin\omega \\
&= -\sin\tau \cos\rho \\
h_{33} &= -\sin\phi t_x - \sin\omega \cos\phi t_y - \cos\omega \cos\phi t_z \\
&= \sin\tau \sin\rho t_x + \sin\tau \cos\rho t_y - \cos\tau t_z
\end{aligned}$$

B.4 The Projection Matrix

Inserting

$$\begin{aligned}\dot{\mathbf{P}}_z &= \begin{bmatrix} f_x \sin\kappa \sin\omega - f_x \cos\kappa \sin\phi \cos\omega + u \cos\phi \cos\omega \\ f_y \cos\kappa \sin\omega + \sin\kappa \sin\phi \cos\omega + v \cos\phi \cos\omega \\ \cos\phi \cos\omega \end{bmatrix} \\ &= \begin{bmatrix} f_x \sin\psi \sin\tau + u \cos\tau \\ f_y \cos\psi \sin\tau + v \cos\tau \\ \cos\tau \end{bmatrix}\end{aligned} \tag{B.3}$$

after the second column would transform the previous 3×3 matrix to the 3×4 projection matrix. That is,

$$\begin{aligned}\mathtt{P} &= \begin{bmatrix} h_{11} & h_{12} & f_x \sin\kappa \sin\omega - f_x \cos\kappa \sin\phi \cos\omega + u \cos\phi \cos\omega & h_{13} \\ h_{21} & h_{22} & f_y \cos\kappa \sin\omega + f_y \sin\kappa \sin\phi \cos\omega + v \cos\phi \cos\omega & h_{23} \\ h_{31} & h_{32} & \cos\phi \cos\omega & h_{33} \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} & f_x \sin\psi \sin\tau + u \cos\tau & h_{13} \\ h_{21} & h_{22} & f_y \cos\psi \sin\tau + v \cos\tau & h_{23} \\ h_{31} & h_{32} & \cos\tau & h_{33} \end{bmatrix}\end{aligned} \tag{B.4}$$

B.5 The Calibration Rotation Matrix

The first three columns of the matrix P are the calibration rotation matrix AR. That is:

$$
\begin{aligned}
\mathtt{AR} = &\begin{bmatrix}
f_x \cos\kappa \cos\phi + u \sin\phi & f_x \sin\kappa \cos\omega + f_x \cos\kappa \sin\phi \sin\omega - u \cos\phi \sin\omega & f_x \sin\kappa \sin\omega - f_x \cos\kappa \sin\phi \cos\omega + u \cos\phi \cos\omega \\
-f_y \sin\kappa \cos\phi + v \sin\phi & f_y \cos\kappa \cos\omega - f_y \sin\kappa \sin\phi \sin\omega - v \cos\phi \sin\omega & f_y \cos\kappa \sin\omega + f_y \sin\kappa \sin\phi \cos\omega + v \cos\phi \cos\omega \\
\sin\phi & -\cos\phi \sin\omega & \cos\phi \cos\omega
\end{bmatrix} \\
= &\begin{bmatrix}
f_x \cos\psi \cos\rho + f_x \sin\psi \cos\tau \sin\rho - u \sin\tau \sin\rho & -f_x \cos\psi \sin\rho + f_x \sin\psi \cos\tau \cos\rho - u \sin\tau \cos\rho & f_x \sin\psi \sin\tau + u \cos\tau \\
-f_y \sin\psi \cos\rho + f_y \cos\psi \cos\tau \sin\rho - v \sin\tau \sin\rho & f_y \sin\psi \sin\rho + f_y \cos\psi \cos\tau \cos\rho - v \sin\tau \cos\rho & f_y \cos\psi \sin\tau + v \cos\tau \\
-\sin\tau \sin\rho & -\sin\tau \cos\rho & \cos\tau
\end{bmatrix}
\end{aligned}
\tag{B.5}
$$

Appendix C

Formula Derivations

This appendix lists the derivations of some formulae listed in the thesis.

C.1 Equation of a Line Passing Through Two Points

Suppose that the end points of a line are $[x_1, y_1]^T$ and $[x_2, y_2]^T$ then the equation of this line is:

$$\begin{array}{c} \frac{y-y_1}{x-x_1} = \frac{y_2-y_1}{x_2-x_1} \\ (y_2 - y_1)x - (x_2 - x_1)y + y_1x_2 - x_1y_2 = 0 \end{array} \tag{C.1}$$

Thus, using the form $ax + by + c = 0$, we can express the coefficients as:

$$\begin{array}{rcl} a & = & y_2 - y_1 \\ b & = & x_1 - x_2 \\ c & = & y_1x_2 - y_2x_1 \end{array} \tag{C.2}$$

Using the notation of Chapter 2, this line can be expressed as:

$$\mathbf{l} = \begin{bmatrix} y_2 - y_1 \\ x_1 - x_2 \\ y_1x_2 - y_2x_1 \end{bmatrix} \tag{C.3}$$

C.2 Equation of a Line of Known Slope and Passing Through One Point

The eqaution of a line of slope m and passing through a point $[x_1, y_1]^T$ is:

$$\begin{array}{c} \frac{y-y_1}{x-x_1} = m \\ mx - y + y_1 - mx_1 = 0 \end{array} \tag{C.4}$$

Using the notation of Chapter 2, this line can be expressed as:

$$\mathbf{l} = \begin{bmatrix} m \\ -1 \\ y_1 - mx_1 \end{bmatrix} \tag{C.5}$$

C.3 Equation of a Vertical Line Passing Through One Point

The eqaution of a vertical line passing through a point $[x_1, y_1]^T$ is:

$$x = x_1 \tag{C.6}$$

Using the notation of Chapter 2, this line can be expressed as:

$$\mathbf{l} = \begin{bmatrix} 1 \\ 0 \\ -x_1 \end{bmatrix} \tag{C.7}$$

Similarly, the equation of a horizontal line passing through $[x_1, y_1]^T$ is:

$$\mathbf{l} = \begin{bmatrix} 0 \\ 1 \\ -y_1 \end{bmatrix} \tag{C.8}$$

Notice that this is the same result we get using Equation (C.5) where m=0.

C.4 Equation of a Line Passing Through One Point and Parallel to Another Line

Consider the line passing through $[x_1, y_1]^T$ and parallel to another line with equation $ax + by + c = 0$ (Figure C.1 on Page 188). If the slope of this line is $-\frac{a}{b}$, then the equation of the parallel line passing through $[x_1, y_1]^T$ is:

$$\begin{gathered} \frac{y-y_1}{x-x_1} = -\frac{a}{b} \\ ax + by - ax_1 - by_1 = 0 \end{gathered} \tag{C.9}$$

Using the notation of Chapter 2, this line can be expressed as:

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ -(ax_1 + by_1) \end{bmatrix} \tag{C.10}$$

where $[a, b, c]^T$ is the parallel line.

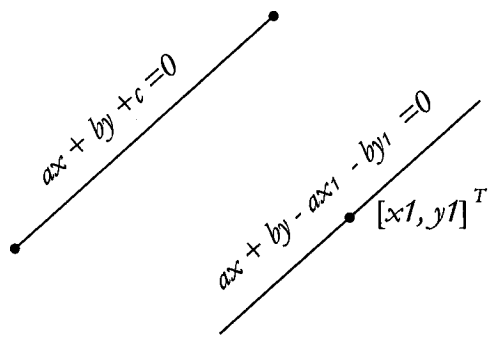


Figure C.1: The equation of a line parallel to $ax+by+c=0$ and passing through $[x_1, y_1]^T$ is $ax+by-ax_1-by_1=0$.

Appendix D

Three-Dimensional Modeling: Computer Graphics Viewpoint

In this appendix, we show how to build a 3D model of an object using concepts of computer graphics. As an example, we will build the scene shown in Figure D.1 on Page 190. The whole operation centers on two main stages:

- three-dimensional wire-frame modeling;
- assigning materials and rendering.

In the following sections, we will discuss each of these stages briefly.

D.1 Three-Dimensional Wire-Frame Modeling

This step requires several aspects to be modeled including:

- a geometrical description of the object we want to construct. The level of details depends on:
  - the distance between the object under consideration and the camera;
  - the size of the final frame; the less the resolution, the less the level of details is required;
- the properties of the material which will be assigned to the object; for example, a single object may contain of more than one texture; however, an object should be assigned only one material. In this case, while building the wire-frame model of that object, we should split it into parts in such a way that each part can be assigned one material later;
- the position and the intensity of the light source in the scene.

In what follows, we present some techniques used to achieve complicated 3D solid models. Although these operations seem to be primitive, the results made out of them are excellent.

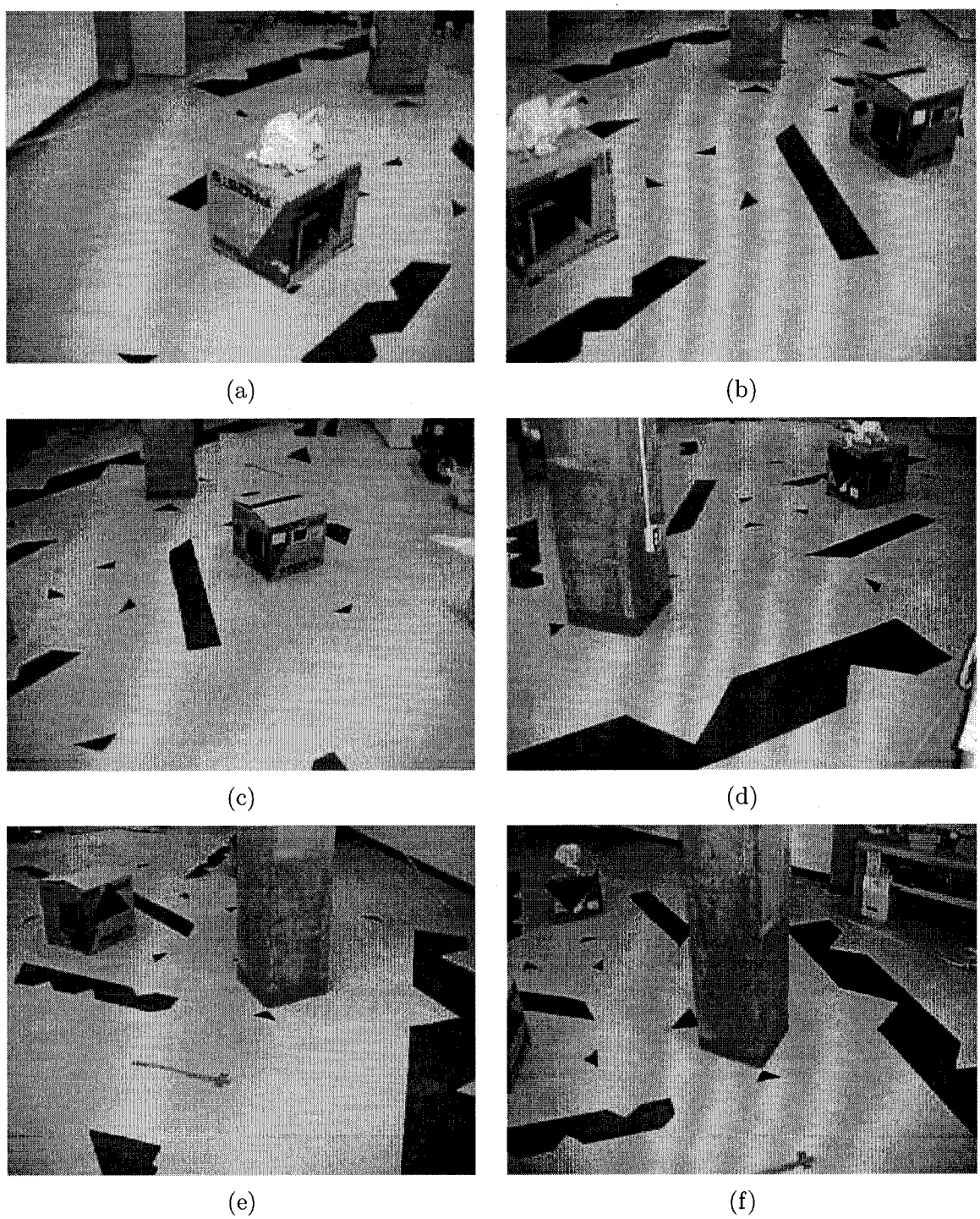


Figure D.1: Images of the scene.

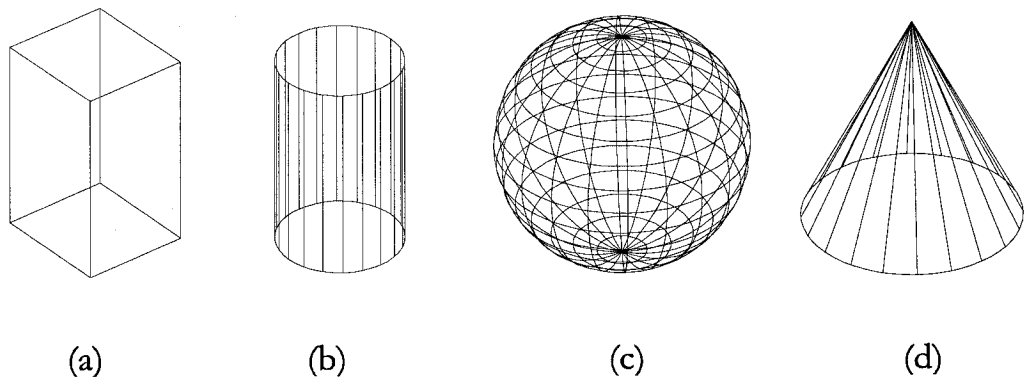


Figure D.2: Primitive solids: (a) Box. (b) Cylinder. (c) Sphere. (d) Cone.

D.1.1 Solids

Solid primitives such as box or cube, sphere, cylinder, cone and wedge might be used to built models in 3D space (Figure D.2 on Page 191). A primitive can be described through a set of parameters. For example, a sphere can be described by its center and radius and a cylinder may be described by the center of its base along with radius and height. In order to built more complex 3D models, a user may use a combination of those primitives. Boolean operations such as union, subtraction and intersection are performed to unify objects and create more complicated solids. *union* operation combines two or more different solids into one composite solid as depicted in Figure D.3 (b) on Page 192. The resulting solid includes the total volume enclosed by all solids forming it [Inc03a]. The *intersection* operation calculates the common (overlapping) volume of two or more solids [Inc03a]. This is shown in Figure D.3 (c). The resulting solid is not affected by the order of processing the intersected solids as the overlapping volume remains the same. The third operation is the *subtraction* or the *difference*. It calculates the difference volume resulting from subtracting two or more solids. Difference or subtraction operator is order dependent [Zei94]. In other words, the order of processing the solids under consideration is very important as it results in quite different solids; e.g., consider Figure D.3 (d) and Figure D.3 (e) on Page 192. Processed solids for all three operations may lie in any number of arbitrary planes.

A modeling method that performs such Boolean procedures is called *Constructive Solid Geometry* (CSG) [FvDFH96]. Using what is called *CSG tree*, an object is stored as a tree data structure with operators at the internal or branch nodes and simple primitives at the leaves. (Some nodes represent Boolean operators whereas others perform translation, rotation and scaling.) To determine physical properties, we must be able to combine the properties of the leaves to obtain properties of the root [Pet99]. In CSG, the intersecting boundaries are mathematically implied [Mat02].

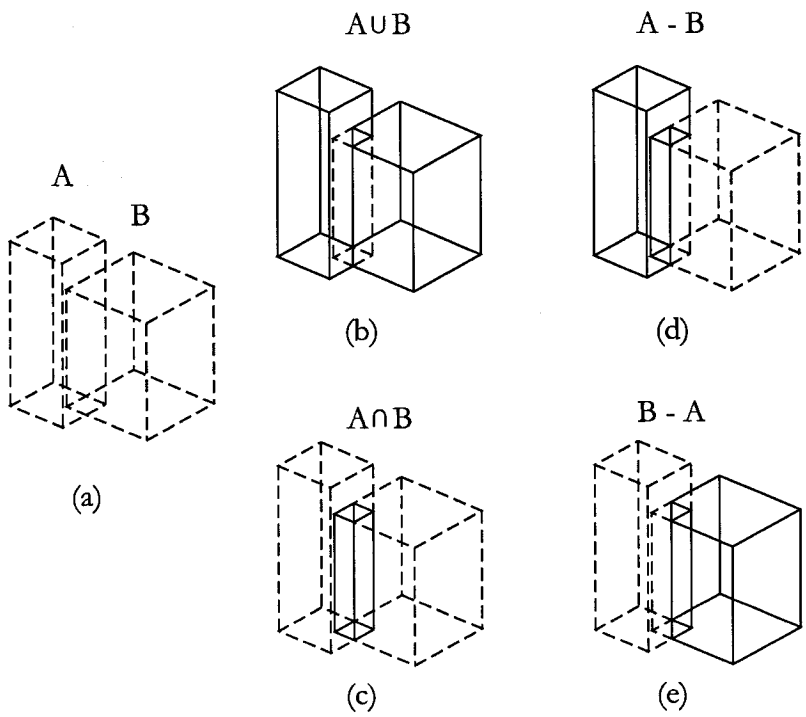


Figure D.3: Boolean operations: (a) Two solids (A and B) before performing the operation. (b) Solid union: One combined solid. (c) Solid intersection: One solid representing the overlapping area after Intersection operation. (d) Solid subtraction: One solid representing the difference (A-B). (e) Solid subtraction: One solid representing the difference (B-A).

Some technologies do not maintain CSG trees; e.g., Spatial Technologies' ACIS Modeler, where data are stored in *boundary representation (B-rep)* [Cor03a], does not maintain such a tree. In this case, an object is represented by its boundaries; i.e., faces, edges and vertices [Rot01]. Data structures are used where vertices, edges and faces that represent the solid are linked so that they result in a closed volume [Woo]. Such an ACIS kernel is used in software packages such as AutoCAD R13 and later. Yet another solid modeling kernel that is based on boundary models is Unigraphics Solutions' Parasolid. Parasolid is used in products such as MicroStation/J.

There are other methods to represent solids [Sam89, Rot01]. They include *spatial enumeration* where the space is split into equal small cubic voxels. Voxels that belong to the solid are flagged. Another variation is called *cell decomposition* where the sizes of the cells may not be equal as the previous case. Others include sweep methods that may be split into translational and rotational sweeps and *primitive instancing*, which contains a set of pre-defined object types that may be characterized by some parameters to create instances of a given type.

D.1.2 Extrusion versus Revolution

Extrusion, a *translational sweep* [Rot01] is an important operation that converts a 2D shape located on a plane into a 3D solid model by adding thickness to it. In order to extrude an object, we need a plan for this object, a cross-section or even an elevation. Usually, the cross-sections (or the elevations) used are orthogonal projections of the object under consideration. Extrusion is useful in cases where lots of details are present [Inc03a] so that using primitive solids with Boolean operations may be more complicated. We may create a profile using lines or arcs, and then join them together into a single polyline object. (For example, a plan of the scene in Figure D.1 may be created prior to use extrusion as in Figure D.1.2.) Then, the height of extrusion, which is usually (but not necessarily) perpendicular to the plane of our profile, may be specified to give thickness to the object. Refer to Figure D.4 (b) on Page 194. Although a lot of details can be embedded in the profile prior to extrude the object, needs may arise to use Boolean operations afterwards to subtract cavities or add more 3D features; e.g., cavities of windows in buildings.

Similar to extrusion, *revolution*, a *rotational sweep* [Rot01], converts a 2D shape into a 3D solid by revolving it about an axis. This is useful in case of the existence of many details in the revolved 2D object. Figure D.4 (c) depicts this concept. (*Lathing* is a similar operation where a curve can rotate about an axis.)

Consider the skirting surrounding the column in Figure D.1 on Page 190. As depicted in Figure D.6 on Page 195, we may get it by subtracting two solids, or unionizing four other

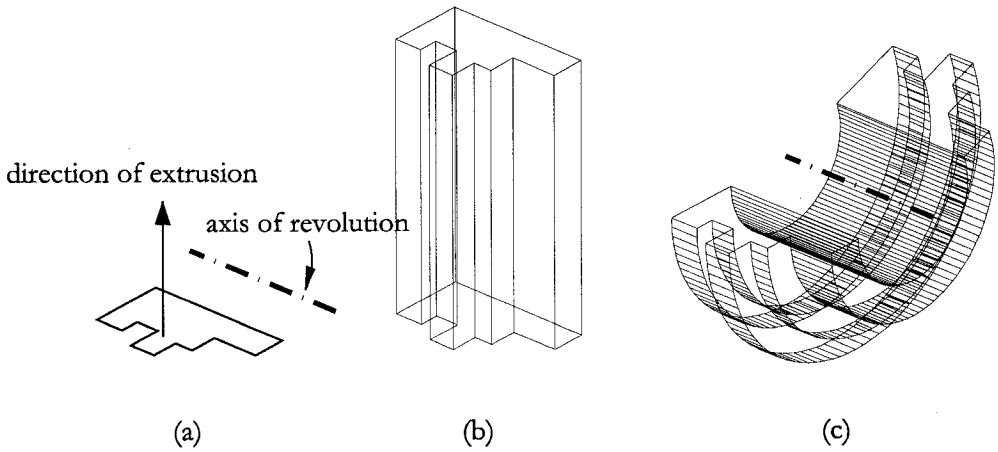


Figure D.4: (a) A 2D shape showing the direction of extrusion and the axis of revolution. (b) The same shape after extrusion results in a 3D solid model. (c) Revolving the 2D shape about the axis of revolution shown in (a) results in different solid.

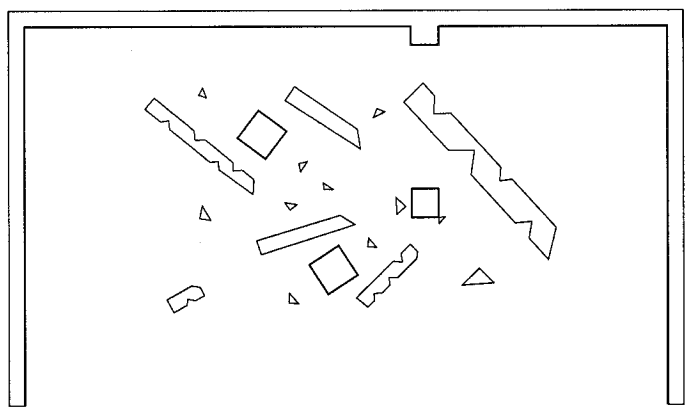


Figure D.5: The plan of the scene shown in Figure D.1.

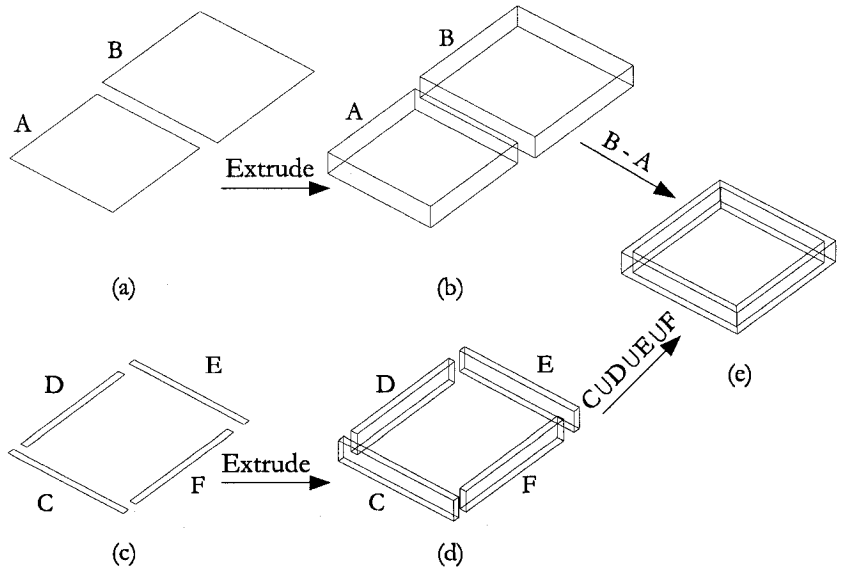


Figure D.6: Example: (a) Two regions, A and B, before extrusion; (b) Two solids, A and B, after extrusion; (c) Four regions, C, D, E ,and F, before extrusion; (d) Four solids C, D, E, and F, after extrusion; (e) Final solid as $B - A$ or $C \cup D \cup E \cup F$.

solids. All solids that we use to subtract or unionize are results of extrusion.

A wire frame model of the scene under consideration is shown in Figure D.7 on Page 196.

D.2 Illuminating and Rendering the Scene

This rest of this appendix deals with the second phase in building a 3D model for an object. The following sections describe the role of each step.

D.2.1 Illuminating the Scene

After building the wire-frame model, a material is to be assigned to every single object included in the scene. In order to view this material, virtual light sources, which simulates as accurate as possible real source characteristics, must be added to the scene. Some radiosity illumination models may be used to calculate diffuse inter-reflections between surfaces [CBP95].

Different types of light sources can be used ranging from a point source of illumination that shoots out in all directions, a source that casts a beam of light infinitely before and after it and a spotlight that casts a focused beam of light. Software packages; e.g., 3D Studio MAX

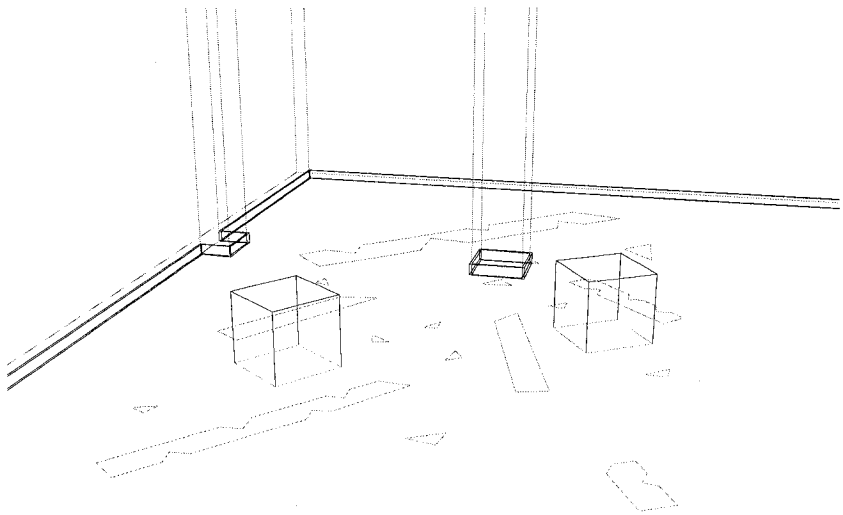


Figure D.7: Images of the scene.

and REALS, apply light types with some variations [Kin98, Cor03b]. Using a database, the city, date and time may be determined and hence sun light may be simulated [Cor03b].

D.2.2 Rendering the Images

There are a number of ways to render or shade the scene. *Constant* or *flat shading*, as it appears from its name, provides constant shading for a given surface. *Gouraud shading* simulates shading by interpolation and smoothes the edges between faces [Inc03a]. *Phong shading* uses the normals of adjacent faces for interpolation [Kin98]. It examines every pixel and results in more realistic highlights [Inc03a]. The results of *Blinn shading* is similar to the previous with better quality. Other types may include *metal*, *ray tracing* and *radiosity*.

D.2.2.1 Assigning Materials

A *material* is data to be assigned an object controlling its appearance; e.g., color, illumination, etc. The color of a material depends on different components. The *ambient* component is the color of the object while in shadow [Kin98]. The *diffuse* component is the color the object reflects when illuminated [Kin98]. The *specular* component controls the color of the highlight on a material [Dxy01]. In addition to the different components of color, the final appearance of an object might be affected significantly by bitmap images that mapped to its surfaces. The following section discusses mapping and some of its important types.

D.2.2.2 Mapping

The concept of mapping centers around applying real-world images to add realism to computer-generated rendered images [HS93]. Those images are referred to as *maps* [Kin98]. Maps add new characteristics to the material and hence to the object under consideration. Attributes controlling the appearance of the maps are often called *mapping coordinates* [Kin98].

Orthogonal projections of faces are often used as maps. Homography transformation explained in Section 7.6 may be used to accurately estimate of transformation of a planar patch. Affine matrix is another simpler type of transformation that could be used. Affine transformation matrix has 6 degrees of freedom and can be fully estimated using 3 pairs of points; i.e., mapping one triangle into another. Suppose that a triangle $< [x_1, y_1]^T, [x_2, y_2]^T, [x_3, y_3]^T >$ is to be mapped to another triangle $< [x'_1, y'_1]^T, [x'_2, y'_2]^T, [x'_3, y'_3]^T >$. In other words, we need to find a 3×3 transformation matrix H that maps a point in the source triangle to another in the destination triangle. Thus:

$$\mathbf{m}' = \mathtt{H}\mathbf{m} \tag{D.1}$$

where $\mathbf{m}$ is the source point and $\mathbf{m}'$ is the destination one. We can rewrite Equation (D.1) as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{D.2}$$

In this case, we have three pairs of points. Then, we can write:

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix} \mathbf{h}_1 \tag{D.3}$$

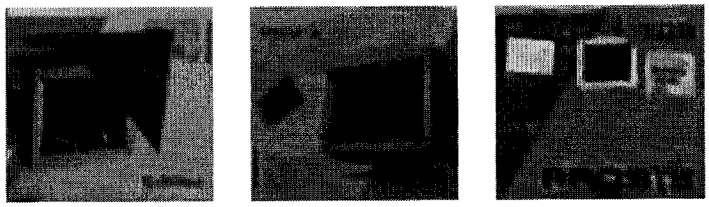
where $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ are the three source vertices. Then, the first row in matrix H can be written as:

$$\mathbf{h}_1 = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix}^{-1} \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} \tag{D.4}$$

Similarly, the second and the third rows can be expressed as:

$$\mathbf{h}_2 = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix}^{-1} \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix} \tag{D.5}$$

$$\mathbf{h}_3 = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{D.6}$$



(a) (b) (c)

Figure D.8: Examples of planar patches transformed through affine matrix.

Then, the transformation from a destination point to its corresponding source can be expressed as:

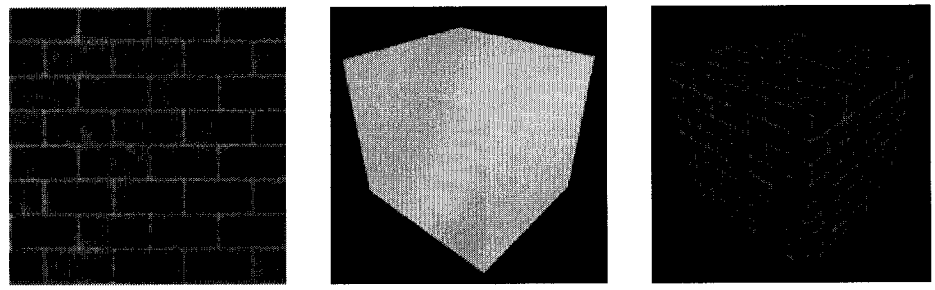
$$\mathbf{m} = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix}^{-1} \mathbf{m}' \tag{D.7}$$

Equation (D.7) has been used to generate the planar patches shown in Figure D.8 on Page 198 by splitting the four-sided area into two triangles and then mapping through affine transformation thereafter. The source areas used are supplied from Figure D.1.

There are many types of mapping; maps that affect the main color components and maps that add new characteristics to the material. For example, *ambient* and *specular mapping* are used where the map is applied to the ambient and specular areas of the object respectively [Kin98]. Other mapping types may include *shininess mapping* that affects the locations of the highlights where lighter areas in the shininess map result in shiny regions; *self-illumination mapping* that lets us use a map to affect the intensity in different areas of the self-illuminated surface according to the intensity levels of the map where lighter areas of the map result in self-illuminated regions on the surface; and *filter color mapping* that applies a transparent color to the object under consideration where the appearance of this color depends on the intensities of the map pixels [Kin98]. Below are some other important types of mapping.

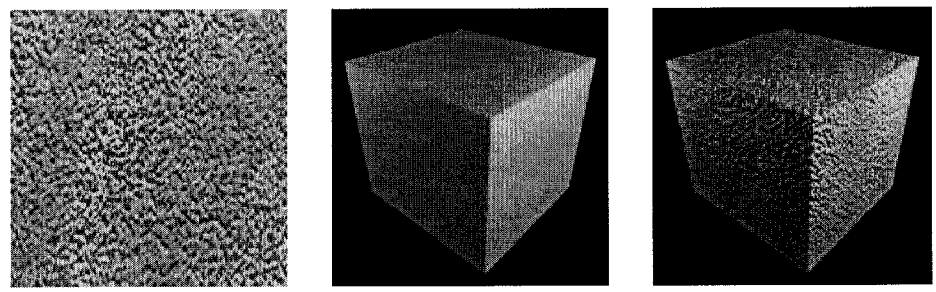
- **Diffuse mapping:** Diffuse mapping affects the basic appearance of a material [Dxy01]. It may alter the original color assigned previously to a material into an image. Applying a diffuse map is like painting an image on the surface of the object [Kin96]. For example, to render a wall made out of brick, we can apply a brick bitmap image as a diffuse map. This is shown in Figure D.9 on Page 199.

- **Bump mapping:** This important concept is used to add roughness or wrinkles to a smooth surface [Bli78, Kug98]. It helps make features on surfaces look raised or sunk; i.e., it alters the height appearance, without altering the geometry of the object under consideration. In order to do that, it uses the brightness values of an image. Bright regions will look



(a) (b) (c)

Figure D.9: Using diffuse mapping to paint an image of a brick wall on an object. (a) A bitmap image [Kin98]. (b) A cube. (c) The image in (a) diffuse mapped on cube (b).

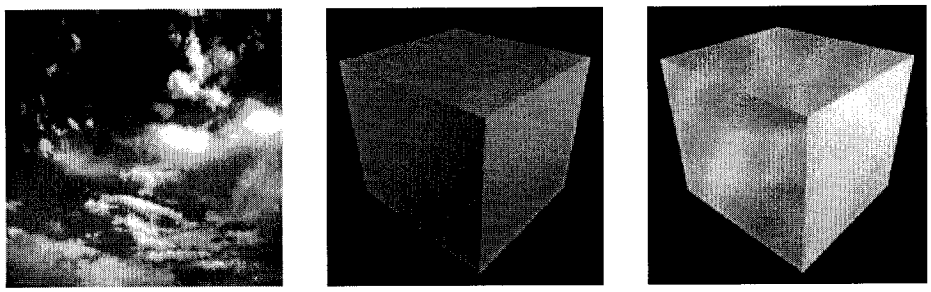


(a) (b) (c)

Figure D.10: Using bump mapping to provide the coarse texture. (a) A bitmap image [Kin98]. (b) A cube. (c) The image in (a) bump mapped on cube (b). Notice that Image (a) only affected the texture while the color remains unchanged.

raised while dark regions look sunk [Dxy01]. This is done by perturbing the normal to the surface vector by another vector that depends on the values in the bump map space. This mapping type could be used to obtain coarse texture in walls for instance. Refer to Figure D.10 on Page 199. Thus, although the appearance of coarse surface, the surface itself remains smooth. This is in contrast to *displacement mapping* where the surface is displaced or modified.

- **Reflection mapping:** Also known as *environment mapping*, reflection mapping calculates the reflection direction of a ray from the viewer to the point being shaded [Kug98]. This type of mapping creates the effect of a scene reflected on the surface of a shiny or reflective object; e.g., mirror, glass, or brass. For example, this mapping type can be used to achieve the effect of reflection of the sky on windows as seen in Figure D.11 on Page 200. Note that with this type of mapping, if the reflective surface moves in the



(a) (b) (c)

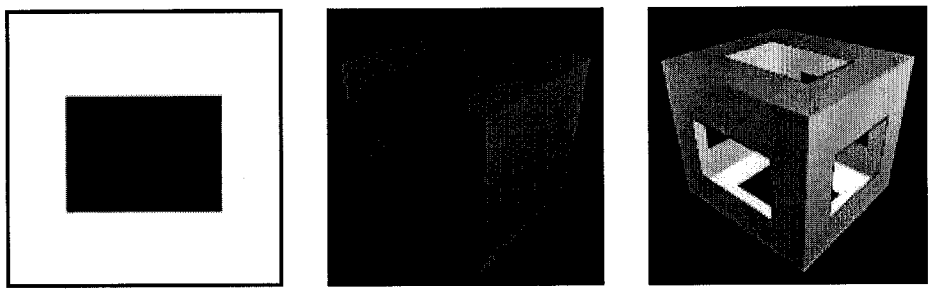
Figure D.11: Using reflection mapping to obtain the effect of the sky. (a) A bitmap image [Kin98]. (b) A cube. (c) The image in (a) refection mapped on cube (b).

(a) (b) (c)

Figure D.12: Using refraction mapping lets us perceive the map as if it is seen through the sphere. (a) A bitmap image [Kin98]. (b) A sphere. (c) The image in (a) refraction mapped on sphere (b).

environment, the image reflected on it does not move with the surface. However, the map does move with changes in viewpoints [Kin98]. This is different from diffuse mapping for example where the map is attached to the surface and moves with it [EKW+98].

- **Refraction mapping:** Instead of reflecting the map off the surface as in the previous type, refraction mapping allows the map to be perceived as if it is seen through the object [Dis02]. This is illustrated in Figure D.12 on Page 200. Similar to reflection mapping, refraction mapping depends on the viewpoint rather than the movement of the object.
- **Opacity mapping:** This type of mapping affects the opacity or transparency of objects [Cor03b]. It uses the intensity levels of the opacity map image to alter the opacity of a surface [Kin96]. For example, dark areas when mapped onto an object result in more translucent regions while lighter areas result in more opaque regions. This is illustrated in Figure D.13 on Page 201.



(a) (b) (c)

Figure D.13: Using opacity mapping to alter the opacity of a surface. (a) A bitmap image. (b) A cube. (c) The image in (a) used as an opacity map results in a transparent region on the faces of the cube.

The above-mentioned method is applied to model the scene in Figure D.1 on Page 190. The results are shown in Figure D.14 on Page 202. Compare these images to those of Figure D.1.

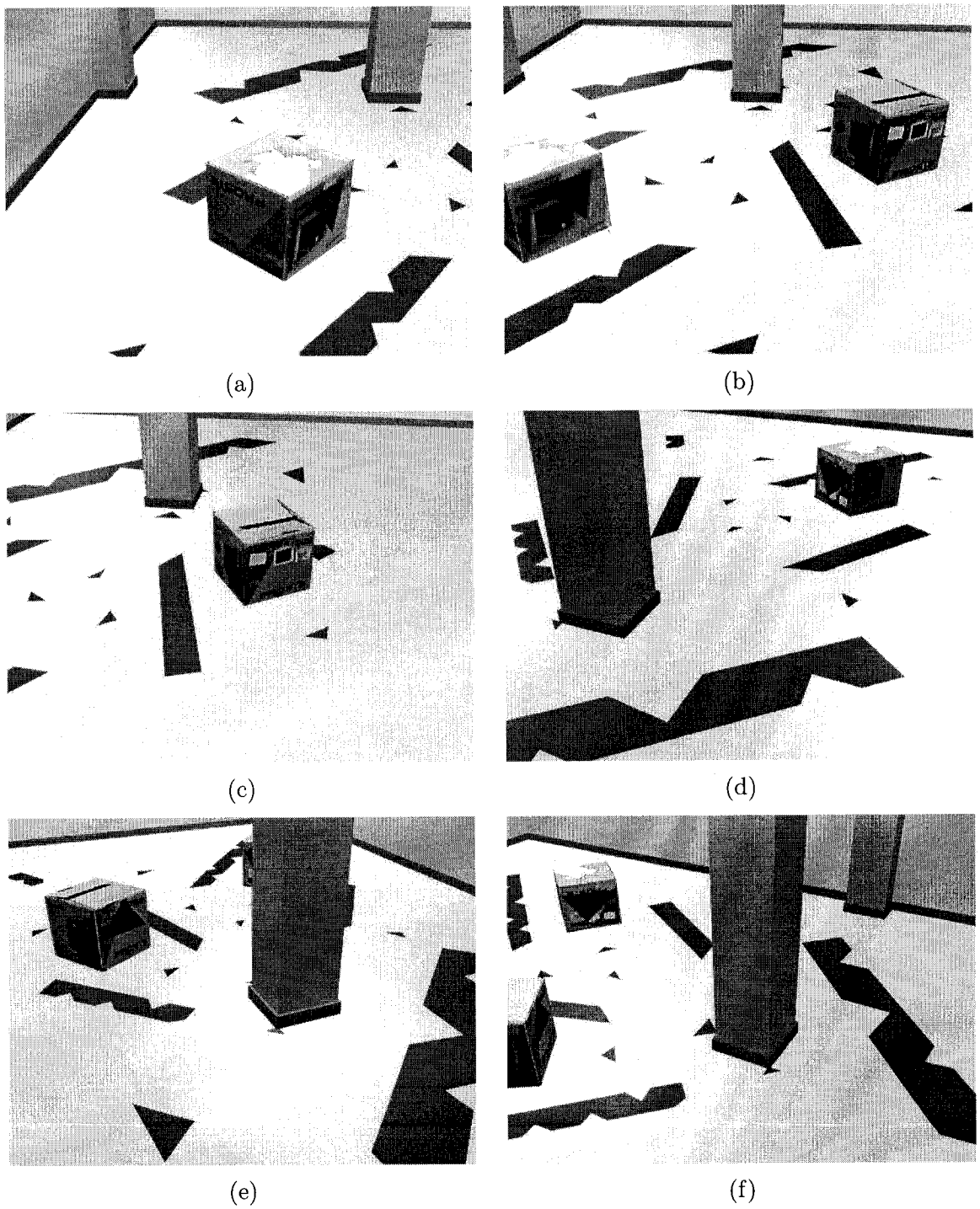


Figure D.14: Synthesized images of the scene. Compare these images with those of Figure D.1.

Bibliography

[AAH+] G. Agam, M. Ahmadi, J.-Y. Hervé, R. Hurteau, and P. Cohen. SMART: Sensori-Motor Augmented Reality for Telerobotics. *World Wide Web*, page http://www.ai.polymtl.ca/ webActivities/SMART/Overview/Overview.html.

[AF87a] N. Ayache and O. D. Faugeras. Building a consistent 3d representation of a mobile robot environment by combining multiple stereo views. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 808–810, 1987.

[AF87b] N. Ayache and B. Faverjon. Efficient Registration of Stereo Images by Matching Graph Descriptions of Edge Segments. *International Journal on Computer Vision*, 1(2), April 1987.

[AL87] N. Ayache and F. Lustman. Trinocular Stereovision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1987.

[AN95] M. Accame and F. G. B. De Natale. An adaptive size block-matching approach for stereo image coding. In *Proceedings of the International Workshop on Stereoscopic and Three Dimensional Imaging*, pages 70–74, 1995.

[aODF89] T. S. Huang abd O. D. Faugeras. Some properties of the E-matrix in two view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1310–1312, 1989.

[Bau00] A. Baumberg. Reliable Feature Matching Across Widely Separated Views. In *Computer Vision and Pattern Recognition*, pages 774–781, 2000.

[BB81] H. H. Baker and T. O. Binford. Depth from edge and intensity based stereo. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 631636, 1981.

[BCF+99] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[BDU+99] E. Bachmann, I. Duman, U. Usta, R. McGhee, X. Yun, and M. Zyda. Orientation Tracking for Humans and Robots Using Inertial Sensors. In *Proceeding of the International Symposium on Computational Intelligence in Robotics and Automation, CIRA'99*, 1999.

[Bey91] D. J. Beymer. Finding Junctions using the Image Gradient. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 720–721, 1991.

[Bli78] J. Blinn. Simulation of wrinkled surfaces. *Computer Graphics*, 12(3):286–292, 1978.

[BP01] W. A. Barrett and Kevin D. Petersen. Houghing the Hough: Peak Collection for Detection of Corners, Junctions and Line Intersections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 302–309, 2001.

[BRM97] K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3D motion trajectories and their use in prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):219–223, March 1997.

[BS91] J. Bloomenthal and K. Shoemake. Convolution Surfaces. *Computer Graphics, Proceedings of SIGGRAPH'91*, 25(4):251–256, July 1991.

[BZM97] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential Update of Projective and Affine Structure from Motion. *International Journal of Computer Vision*, 23(3):235–259, 1997.

[CA86] C. H. Chien and J. K. Aggarwel. Identification of 3D Objects from Multiple Silhouettes Using Quadtrees/Octees. *Comp. Vision, Graphics and Image Processing*, 36:256–273, 1986.

[CBP95] C. Chevier, Belblidia, and J. C. Paul. Composing Computer-generated Images and Video Films: An Application for Visual Assessment in Urban Environments. *Computer Graphics: Developments in Virtual Environments*, pages 115–125, 1995.

[CC92] L. D. Cohen and I. Cohen. Deformable Models for 3D Medical Images Using Finite Element and Balloons. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR'92*, pages 592–598, June 1992.

[CJZ03] D. Cobzas, M. Jagersand, and H. Zhang. A Panoramic Model for Remote Robot Environment Mapping and Predictive Display. In *Proceedings of Vision Interface*, 2003.

[Cor03a] Webnox Corp. HyperDictionary. *World Wide Web*, page http://www. hyperdictionary.com/, 2003.

[Cor03b] Kajima Corporation. REALS on-line help. 2003.

[CRZ99] A Criminisi, I. Reid, and A Zisserman. A Plane Measuring Device. *Image and Vision Computing*, 17(8):625–634, 1999.

[CZ00a] D. Cobzas and H. Zhang. 2D Robot Localization with Image-Based Panoramic Models Using Vertical Line Features. In *Proceedings of Vision Interface*, 2000.

[CZ00b] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In A. Leonardis, F. Solina, and R. Bajcsy, editors, *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics, Ljubljana, Slovenia*, pages 25–47, 2000.

[DeS97] R. DeSantis. Modeling and Path Tracking for a Load-Haul-Dump Mining Vehicle. *Journal of Dynamic Systems, Measurement and Control*, 119, March 1997.

[Dis02] Discreet. 3D Studio MAX on-line help. 2002.

[DK01] A. Davison and N. Kita. 3D Simultaneous Localisation and Map-Building Using Active Vision for a Robot Moving on Undulating Terrain. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[DTS94] M. G. Strintzis D. Tzovaras and H. Sahinoglu. Evaluation of multiresolution block matching techniques for motion and disparity estimation. *Signal Processing: Image Communication*, pages 59–67, 1994.

[Dxy01] Dxyner. Custom Materials for 3DS MAX. *World Wide Web*, page http://dxyner2000.com/tutorials/, 2001.

[Dye01] C. R. Dyer. *Foundations of Image Understanding – Volumetric Scene Reconstruction From Multiple Views*, chapter 16, pages 469–489. Kluwer, Boston, 2001.

[DZLF94] R. Deriche, Z. Zhang, Q. Luong, and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In *European Conference on Computer Vision*, pages 567–576, 1994.

[EKW+98] G. Eckel, R. Kempf, L. Wennerberg, D. Galgani, M. Levine, A. Clardy, B. Cabral, M. Hopcroft, J. Lim, K. Lin, Z. Liu, L. Petrovic, T. Roy, and J. Yen. OpenGL Optimizer Programmer's Guide: An Open API for Large-Model Visualization. *World Wide Web*, page http://www.parallab.uib.no/SGI_bookshelves/SGI_Developer/books/Optimizer_PG/sgi_html/index.html, 1998.

[EL99] R. Elias and R. Laganière. The Disparity Pyramid: An Irregular Pyramid Approach for Stereoscopic Image Analysis. In *Proceedings of Vision Interface, VI'99*, pages 352–359, 1999.

[EL00] R. Elias and R. Laganière. Sensori-Motor Augmented Reality for Telerobotics (SMART) Project: Path Specifications using Top View Mosaics. *Institute for Robotics and Intelligent Systems, IRIS/PRECARN Conference Demo*, page http://www.site.uottawa.ca/~relias/demos/smart/#IRIS2000, May 2000.

[EL01a] R. Elias and R. Laganière. Projective Geometry for Three-Dimensional Computer Vision. Technical Report TR-2001-08, VIVA Lab, School of Information Technology and Engineering, SITE, University Of Ottawa, Ottawa, Ontario, Canada, Nov. 2001.

[EL01b] R. Elias and R. Laganière. Sensori-Motor Augmented Reality for Telerobotics (SMART) Project: Modeling of Obstacles. *Institute for Robotics and Intelligent Systems, IRIS/PRECARN Conference Demo*, page http://www.site.uottawa.ca/~relias/demos/smart/#IRIS2001, June 2001.

[EL02] R. Elias and R. Laganière. Cones: A New Approach Towards Corner Detection. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, CCECE'02*, volume 2, pages 912–916, 2002.

[EL03] R. Elias and R. Laganière. The Planar Perspective Matrix: Derivation Through Different Rotation Systems. Technical Report TR-2003-02, VIVA Lab, School of Information Technology and Engineering, SITE, University Of Ottawa, Ottawa, Ontario, Canada, Feb. 2003.

[EL04] R. Elias and R. Laganière. Volumetric Representation of Obstacles Through Wide Baseline Set of Images. In *Proceedings of the 35th International Symposium on Robotics, ISR'04 (accepted)*, 2004.

[Eli99a] R. Elias. The Disparity Pyramid: An Irregular Pyramid Approach for Stereoscopic Image Analysis. *M.C.S. thesis*, 1999.

[Eli99b] R. Elias. Virtual Tour at the University of Ottawa. *World Wide Web*, page http://www.site.uottawa.ca/~relias/campus.html, April 1999.

[Eli03a] R. Elias. Clustering Points in nD Space Through Hierarchical Structures. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, CCECE'03*, volume 3, pages 2079–2081, 2003.

[Eli03b] R. Elias. Projective Geometry for Three-Dimensional Computer Vision. In *Proceedings of the 7th Multiconference on Systemics, Cybernetics and Informatics, SCI'03*, volume v, pages 99–104, 2003.

[Eli04] R. Elias. Wide Basleine Matching Through Homographic Transformation. In *Proceedings of the 17th International Conference on Pattern Recognition, ICPR'04 (accepted)*, 2004.

[Fal94] L. Falkenhagen. Depth estimation from stereoscopic image pairs assuming piecewise contineous surface. In *Image Processing for Broadcast and Video Production*, pages 115–127, 1994.

[Fau96] O. Faugeras. *Three-Dimentional Computer Vision, A Geometric Viewpoint.* The MIT Press, Cambridge, Massachusetts, 1996.

[FB81] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM*, 24:381–395, 1981.

[FCZ98] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D Model Construction for Turn-Table Sequences. In *Lecture Notes in Comuter Science 1506, 3D Structure from Multiple Images of Large-Scale Environments, European Workshop, SMILE'98*, pages 155–170, 1998.

[FD00] D. Ziou F. Deschenes. Detection of Line Junctions in Gray-level Images. In *International Conference on Pattern Recognition*, volume 3, pages 754–757, 2000.

[Fis97a] B. Fisher. Computing Plane Projective Transformations. *World Wide Web*, CVonline: Geometry and Mathematics:http://www.dai.ed.ac.uk /CVonline /LOCAL_COPIES/EPSRC_SSAZ/node11.html, April 1997.

[Fis97b] B. Fisher. Homogeneous Coordinates. *World Wide Web*, CVonline: Vision Geometry and Mathematics:http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/BEARDSLEY/node1.html, November 1997.

[Fis97c] B. Fisher. How to Establish Correspondences. *World Wide Web*, CVonline: Geometry and Mathematics:http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node26.html, April 1997.

[FK98] O. Faugeras and R. Keriven. Complete dense stereovision using level set methods. In *Proceedings of the 5th European Conference on Computer Vision*, page 379393, 1998.

[FPWW94] B. Fisher, S. Perkins, A. Walker, and E. Wolfart. *Hybermedia Image Processing Reference.* Department of Artificial Intelligence, University of Edinburgh, United kingdom, 1994.

[FPWW00] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Rotate. *World Wide Web*, Image Processing Learning Resources:http: //www.dai.ed.ac.uk/HIPR2/rotate.htm, 2000.

[FvDFH96] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley Publishing Company, 1996.

[GMP98] V. Gouet, P. Montesinos, and D. Pelé. Stereo matching of color images using differential invariants. In *Proceedings of the IEEE International Conference on Image Processing*, pages 152–156, Chicago, Etats-Unis, 1998.

[GW87] P. Giblin and R. Weiss. Reconstruction of surfaces from profiles. In *International Conference Comp. Vision*, pages 136–144, 1987.

[HA98] A. Heyden and K. Astrom. Minimal conditions on intrinsic parameters for euclidean reconstruction. In *Lecture Notes in Comuter Science 1352, Computer Vision - ACCV'98, Third Asian Conference on Computer Vision*, volume II, pages 169–176, 1998.

[HEL03] H. Hajjdiab, R. Elias, and R. Laganière. Wide Baseline Obstacle Detection and Localization. In *Proceedings of the IEEE Seventh International Symposium on Signal Processing and its Applications, ISSPA'03*, volume i, pages 21–24, 2003.

[HEL04] H. Hajjdiab, R. Elias, and R. Laganière. Obstacle Localization from Ground Plane Sparse Views. *Robotics and Autonomous Systems (submitted)*, 2004.

[HS93] P. Haeberli and M. Segal. Texture Mapping as a Fundamental Drawing Primitive. In *Fourth Eurographics Workshop on Rendering*, pages 259–266, 1993.

[HS94] R. I. Hartley and P. Sturm. Triangulation. *American Image Understanding Workshop*, pages 957–966, 1994.

[HS95] R. I. Hartley and P. Sturm. Triangulation. In *Lecture Notes in Comuter Science 970, Computer Analysis of Images and Patterns CAIP'95, 6th International Conference*, pages 190–197, 1995.

[HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[IB95] S. Intille and A. Bobick. Closed-World Tracking. In *Proceeding of the International Conference on Computer Vision*, pages 672–678, 1995.

[Inc03a] Autodesk Inc. AutoCAD on-line help. 2003.

[Inc03b] Precarn Incorporated. Web site. *World Wide Web*, page http://www.precarn.ca, 2003.

[Jai89] A. K. Jain. *Fundamental of Digital Image Processing.* Prentice Hall, 1989.

[JM91] J. M. Jolion and A. Montavert. The Adaptive Pyramid: A Framework for 2D Image Analysis. *CVGIP: Image Understanding*, 55(3):339–348, 1991.

[JR94] J. M. Jolion and A. Rosenfeld. *A Pyramid Framework for Early Vision.* Kluwer Academic Publishers, 1994.

[Kin96] Kinetex. 3D Studio MAX on-line help. 1996.

[Kin98] Kinetex. 3D Studio MAX on-line help. 1998.

[Koc95] R. Koch. 3D Surface Reconstruction from Stereoscopic Image Sequences. In *International Conference Computer Vision*, pages 109–114, 1995.

[KS99] K. Kutulakos and S. Seitz. A theory of shape by space carving. In *Proceedings of the 7th International Conference on Computer Vision*, page 307314, 1999.

[Kug98] A. Kugler. IMEM: An Intelligent Memory for Bump- and Reflection-Mapping. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 259–266, 1998.

[LAC93] F. Argenti L. Alparone and V. Capellini. A robust coarse-to-fine least squares stereo matching. In *Proceedings of the 4th European Workshop on Three Dimensional Television*, pages 171–177, 1993.

[Lag00] R. Laganière. Composing a bird's view mosaic. In *Proceedings Vision Interface'00*, pages 382–386, 2000.

[LE] R. Laganière and R. Elias. JUDOCA: A Fast Junction Detection Operator. *Pattern Recognition Letters, (pending revision).*

[LE04] R. Laganière and R. Elias. The Detection of Junction Features in Images. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'04 (accepted)*, 2004.

[LF94] S. Laveau and O. Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. Technical Report 2205, Institut National de Recherche en Informatique et Automatique, INRIA, Unité de Recherche INRIA-Sophia Antipolis, February 1994.

[LF96] Q.-T. Luong and O. D. Faugeras. The fundamental matrix: theory, algorithms and stability analysis. *The International Journal of Computer Vision*, 17(1):43–76, 1996.

[LH81] H. C. Longuet-Higgins. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293:133–135, 1981.

[LL96] R. Laganière and F. Labonté. Stereokineopsis: A survey. Technical Report CRPR-RT-9603, École Polytechnique de Montréal, Groupe de Recherche en Perception et Robotique, July 1996.

[LO98] M. I. A. Lourakis and S. C. Orphanoudakis. Visual Detection of Obstacles Assuming a Locally Planar Ground. In *Lecture Notes in Comuter Science 1352, Computer Vision - ACCV'98, Third Asian Conference on Computer Vision*, volume II, pages 527–534, 1998.

[LP02a] B. Liang and N. Pears. Ground Plane Segmentation from Multiple Visual Cues. In *Second International Conference on Image and Graphics (SPIE series)*, 2002.

[LP02b] B. Liang and N. Pears. Visual Navigation using Planar Homographies. In *IEEE International Conference on Robotics and Automation, ICRA'02*, 2002.

[LRS00] L. Lee, R. Romano, and G. Stein. Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):758–767, 2000.

[LV94] Q.-T. Luong and T. Viéville. Canonic Representations for the Geometries of Multiple Projective Views. *Lecture Notes in Computer Science, Computer Vision - ECCV '94*, 800:589–599, 1994.

[LV02] R. Laganière and É. Vincent. Wedge-based Corner Model for Widely Separated Views Matching. In *Proceedings of the 16th International Conference on Pattern Recognition, ICPR'02*, volume 3, 2002.

[LZ98] D. Liebowitz and A. Zisserman. Metric Rectification for Perspective Images of Planes. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[MA83] W. N. Martin and J. K. Affarwal. Volumetric description of objects from multiple views. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[Mat02] Mathdata.com. Knowledgebase - Solid Modeling Glossary. *World Wide Web*, page http://www.mathdata.com/faqs/solid_modeling_glossary.html, 2002.

[MBL+91] J. V. Miller, D. E. Breen, W. E. Lorenson, R. M. O'Bara, and M. J. Wosny. Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics (SIGGRAPH'91 Proceedings)*, pages 217–226, July 1991.

[MGD98] P. Montesinos, V. Gouet, and R. Deriche. Differential Invariants for Color Images. In *Proceedings of* 14^{th} *International Conference on Pattern Recognition*, pages 838–840, Brisbane, Australia, 1998.

[MS98] F. Mokhtarian and R. Suomela. Robut image corner detection through curvature scale space. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20(12):1376–1381, 1998.

[MTSA97] Y. Matsumoto, H. Terasaki, K. Sugimoto, and T. Arakawa. A Portable Three-dimensional Digitizer. In *Proceedings of the IEEE International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 197–204, 1997.

[MZ92] J. L. Mundy and A. Zisserman. *Geometric invariance in computer vision.* The MIT Press, Cambridge, Massachusetts, 1992.

[NH95] H. V. Nguyen and M. Hanajik. 3-D Scene Reconstruction from Image Sequences. In *Lecture Notes in Comuter Science 970, Computer Analysis of Images and Patterns CAIP'95, 6th International Conference*, pages 182–189, 1995.

[NW97] W. Niem and J. Wingbermuhle. Automatic Reconstruction of 3D objects Using Mobile Monoscopic Camera. In *Proceedings of the IEEE International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 173–180, 1997.

[Pea] N. Pears. Multi-Cue Vision. *World Wide Web*, pages http://www–users.cs.york.ac.uk/~nep/research/research.html.

[Pea02] N. Pears. Robust, Adaptive Visual Navigation using Multiple Diverse Visual Cues. *World Wide Web*, IGR Report:http://www–users.cs.york.ac.uk/~nep/ research/igr.ps, 2002.

[Pet99] E. Petriu. Virtual Environment. *University of Ottawa Lecture Notes*, Winter 1998-1999.

[PGH98] L. Parida, D. Geiger, and R. Hummel. Junctions: Detection, Classification, and Reconstruction. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20:687–698, 1998.

[PKVG98] M. Pollefeys, R. Koch, M. Vergauwen, and L. V. Gool. Metric 3D Surface Reconstruction from Uncalibrated Image Sequences. In *Lecture Notes in Comuter Science 1506, 3D Structure from Multiple Images of Large-Scale Environments, European Workshop, SMILE'98*, pages 139–154, 1998.

[PL01] N. Pears and B. Liang. Ground Plane Segmentation for Mobile Robot Visual Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS01*, pages 1513–1518, 2001.

[PL02] N. Pears and B. Liang. Robust Visual Navigation from Multiple Visual Cues. *World Wide Web*, pages http://www–users.cs.york.ac.uk/~nep/research/vn.html, 2002.

[PS91] A. Pentland and S. Sclaroff. Generalized implicit functions for computer graphics. *Computer Graphics (SIGGRAPH'91 Proceedings)*, pages 247–250, July 1991.

[PZ98] P. Pritchett and A. Zisserman. Wide Baseline Stereo. In *Proceedings of the International Conference on Computer Vision*, pages 754–759, 1998.

[RBYL93] N. A. Thacker R. B. Yates and R. A. Lane. Stretch-correlation for real-time disparity evaluation of stereo pairs. In *Proceedings of the 4th European Workshop on Three Dimensional Television*, pages 149–155, 1993.

[RC01] P. Ridley and P. Croke. Autonomous Control of an Underground Mining Vehicle. In *Proceedings of Australian Conference on Robotics and Automation*, pages 26–31, 2001.

[RCF95] C. Rothwell, G. Csurka, and O. Faugeras. A Comparison of Projective Reconstruction Methods for Pairs of Views. Technical Report 2538, Institut National de Recherche en Informatique et Automatique, INRIA, Unité de Recherche INRIA-Sophia Antipolis, April 1995.

[RFC95] C. Rothwell, O. D. Faugeras, and G. Csurka. Different Paths towards Projective Reconstruction. In R. Mohr and W. Chengke, editors, *Europe-China workshop on Geometrical Modelling and Invariants for Computer Vision*, Xi'an, China, 1995. Xidan University Press.

[Roh92] K. Rohr. Modelling and identification of characteristic intensity variations. *Image Vision Computing*, 10:66–76, 1992.

[Rot01] F. Rottensteiner. Semi-automatic extraction of buildings based on hybrid adjustment using 3d surface models and management of building data in a tis. *Ph.D. thesis*, 2001.

[RT01] M. A. Ruzon and C. Tomasi. Edge, Junction and Corner Detection using Color Distributions. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 23(11):1281–1295, November 2001.

[RW00] G. Roth and A. Whitehead. Using Projective Vision to Find Camera Positions in an Image Sequence. In *Proceedings Vision Interface 2000*, pages 255–232, 2000.

[Sam89] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1989.

[SB95] S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995.

[SD97] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 10671073, 1997.

[SG92] V. Seferidis and M. Ghanbari. Generalized block matching motion estimation. In *Proceedings of the SPIE: Visual Communication and Image Processing, VCIP'92*, volume 1818 pt.1, pages 110–119, 1992.

[SK79] J. Semple and G. Kneebone. *Algebric Projective Geometry.* Oxford University Press, 1979.

[Sze93] R. Szeliski. Rapid Octee Construction from Image Sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.

[TBB+00] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.

[TG01] T. Tuytelaars and L. Van Gool. Matching Widely Separated Views based on Affinely Invariant Neighbourhoods. *Submitted to International Journal on Computer Vision*, July 2001.

[TGDK99] T. Tuytelaars, L. Van Gool, L. D'haene, and R. Koch. Matching of Affinely Invariant Regions for Visual Servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1601–1606, 1999.

[THF+03] S. Thruny, D. Hahnelyz, D. Fergusony, M. Montemerloy, R. Triebely, W. Burgardz, C. Bakery, Z. Omohundroy, S. Thayery, and W. Whittakery. A System for Volumetric Robotic Mapping of Abandoned Mines. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2003.

[VL01] É. Vincent and R. Laganière. Detecting Planar Homographies in an Image Pair. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis, ISPA 2001*, pages 182–187, 2001.

[VLR00] É. Vincent, R. Laganière, and G. Roth. Widely Separated View Matching: A Literature Reviews. *Submitted to ??*, ?? 200?

[WD00] P. Wolf and B. Dewitt. *Elements of Photogrammetry.* McGraw Hill, 2000.

[Woo] J. Woodwark. Set - theoretic kernel modeller. *World Wide Web*, page http://www.johnwoodwark.com/inge/svlis/.

[WY98] G. Sommer W. Yu, K. Danilidis. Rotated Wedge Averaging Method for Junction Characterization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–395, 1998.

[ZDFL94] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. *Institut National De Recherche En Informatique Et En Automatique*, 1994.

[Zei94] J. Zeijl. Constructive Solid Geometry. *World Wide Web*, page http://graphics.stanford.edu/~ cek/rayshade/doc/guide/section2_7_4.html, 1994.

[Zha95] Z. Zhang. An Automatic and Robust Algorithm for Determining Motion and Structure from Two Perspective Images. In *Lecture Notes in Comuter Science 970, Computer Analysis of Images and Patterns CAIP'95, 6th International Conference*, pages 174–181, 1995.

[Zha98] Z. Zhang. A New Multistage Approach to Motion and Structure Estimation by Gradually Enforcing Geometric Constrains. In *Lecture Notes in Comuter Science 1352, Computer Vision - ACCV'98, Third Asian Conference on Computer Vision*, volume II, pages 567–574, 1998.

[Zie92] M. Ziegler. Disparity estimation using variable block size. In *Proceedings of European Workshop on 3D-TV Signal Processing*, 1992.

[ZK92] J. Y. Zheng and F. Kishino. 3D models from contours: Further identification of unexposed areas. In *International Conference Pattern Recognition*, pages 349–353, 1992.

[ZLA98] A. Zisserman, D. Liebowitz, and M. Armstrong. Resolving Ambiguities in Auto-Calibration. *Royal Society Typescript*, 1998.

Index