Analyzing the Spatio-Temporal Domain: from View Synthesis to Motion Segmentation

THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

DORON FELDMAN

SUBMITTED TO THE SENATE OF THE HEBREW UNIVERSITY

SEPTEMBER 2006

This work was carried out under the supervision of

Prof. Daphna Weinshall.

Abstract

In this work I investigate spatio-temporal information in a video sequence. The advantage of considering a video sequence as a 3D spatio-temporal function with temporal continuity (rather than merely a discrete collection of 2D images) is demonstrated by two computer vision techniques which I have developed.

View Synthesis: Each frame of the video sequence is an intersection of the spatio-temporal video volume with a spatial plane. When a video sequence conforms to certain geometrical constraints, intersecting the video volume with other planes or surfaces can be used to easily produce new views of the scene. This powerful view synthesis technique is based solely on captured data and does not require scene reconstruction, as the constraint on the input camera motion make it invariant to the scene structure in some respects. This technique is demonstrated with real sequences, giving visually appealing results.

The technique gives rise to a novel projection model, the *Crossed-Slits projection*, that can be seen as a generalization of the perspective projection and several other models. A Crossed-Slits camera is defined by two lines which all rays must intersect. Here I study this new projection model and its epipolar geometry, which are shown to be quadratic equivalents of the perspective model.

Crossed-Slits images are not perspective, and thus they appear distorted. These distortions are studied, and two frameworks are developed for handling them: First, assuming that a coarse approximation of the scene structure is known (which is used to create a realtime omnidirectional virtual environment); Second, without any knowledge about the scene, based only on the set of rays. In both cases distortion is reduced by approximating the perspective projection.

The work on view synthesis and the Crossed-Slits projection, presented in Chapter 3 and 4, is based on work published in [1–6].

Motion Segmentation: Analysis of an *unconstrained* video sequence in general motion reveals a highly regular spatio-temporal structure, where moving objects appear as continuous structures in the temporal domain, broken by occlusion. Based on this observation, I developed a novel motion segmentation algorithm from a video sequence in general motion, which is based on differential properties in the spatio-temporal domain.

I present a differential occlusion detector, which detects corner-like features that are indicative of motion boundaries. Segmentation is achieved by integrating the response of this detector in scale space. The algorithm is shown to give good results on real sequences taken in general motion. Experiments with synthetic data show robustness to high levels of noise and illumination changes; the experiments also include cases where no intensity edge exists at the location of the motion boundary, or when no parametric motion model can describe the data

Next I describe two algorithms to determine depth ordering from two- and three-frame sequences based on observations about the scale space characteristics of the motion boundary. An interesting property of this method is its ability compute depth ordering from only two frames, even when no edge can be detected in a single frame.

Finally, experiments show that people, like my algorithm, can compute depth ordering from only two frames, even when the boundary between the layers is not visible in a single frame.

The work on motion segmentation and depth ordering, presented in Chapter 5, is based on [7, 8].

Contents

Chapter 1

Introduction

This introduction reviews previous work on multi-perspective mosaicing leading to results on the Crossed-Slits projection, and also provides introduction to motion segmentation.

The rest of this thesis is organized as follows: Chapter 2 summarizes the mathematical tools used in the inference of this work, including projection models, multiple view geometry and scale space. Next, my work on view synthesis and the Crossed-Slits projection is presented – in Chapter 3 I elaborate on the geometry and algebra of the Crossed-Slits projection, and in Chapter 4 I describe different approaches to Crossed-Slits view synthesis. Finally, in Chapter 5 my work on motion segmentation and depth ordering is presented.

1.1 View Synthesis and the Crossed-Slits Projection

Perspective projection forms the foundation of imaging. Since our eyes, as well as most of our cameras, observe the world through a pinhole (via a lens whose effects I shall ignore), we are used to viewing images that are generated by perspective projection. Therefore, techniques for the generation of new views are designed to achieve the effects of perspective projection. What happens if this requirement is relaxed? Can we do better computationally when not limited by the perspective projection, but are free to use other projection models?

In Chapter 3 I introduce and study an alternative projection model, defined by two slits - the

Crossed-Slits (X-Slits) projection. In the X-Slits model, the projection ray of every 3D point is defined by the line that passes through the point and intersects both slits. The image of a point will be the intersection of the projection ray with the image surface.

Curiously, a physical X-Slits camera was designed in the 19th century by one of the pioneers of color photography, Ducos du Hauron [33], under the title “transformisme en photographie”. Ducos du Hauron thought that X-Slits images would be used in the following (20th) century to “create visions of another world” [46]. A century later, Rudolf Kingslake reviewed this device in his book [33]; in his analysis Kingslake concludes that “the pair of slits working together thus constitutes a pinhole camera in which the image is stretched or compressed in one direction more than in the other”. This should make this exotic device rather useful for the new emerging technology of wide-screen cinematography. However, as I show below, the X-Slits projection does much more to images than horizontal stretching.

Independent of the physical device, I argue that the X-Slits projection model is useful and worthy of our attention. This is because new X-Slits images can be easily generated by mosaicing a sequence of images captured by a translating pinhole camera, and because those images look compelling and realistic.

1.1.1 Mosaicing and Multi-Perspective Images

Mosaicing is a technique by which strips are taken from frames of a video sequence and pasted together, creating a new image, or “mosaic”. Mosaics are usually used as a visual summary of the video [11, 28] or for 3D visualization [57]. In the simplest form, a mosaic is composed of vertical strips taken from the center of the frame where the input camera rotates about an axis that is perpendicular to its optical axis and passes through the optical center. Such mosaics constitute a perspective projection of the scene onto a cylinder, and are used for creating panoramic images.

Usually mosaics do not represent a perspective projection, i.e., they cannot be modelled as a projection through a single point, as different strips may be acquired from different camera locations. Such mosaics are known as *multi-perspective* images. A simple example of such mosaicing technique is the *linear pushbroom* which samples the central strip of each frame where the input camera trans-

lates sideways, thus "scanning" the scene with a line camera. This method is used to create highly detailed images of a large scene, typically in satellite and aerial imagery, where the camera motion is approximated as linear.

Another multi-perspective technique uses an input camera rotating *off-axis* so that its optical axis is perpendicular to the (circular) camera path. In [64], such mosaics based on concentric circular camera paths are used to construct a 3D plenoptic function for the purpose of image-based rendering (IBR). Sampling a certain strip from each frame for one mosaic and a different strip for another can be used to produce a panoramic *stereo pair* [29, 54]. Stereo mosaicing models are further investigated in [51, 61].

In [76], the input camera motion is unconstrained, and the mosaic is used to summarize a video captured by a moving robot. In [55], the strips are projected onto a manifold that is adapted to the motion trajectory.

The present work uses a similar mosaicing technique with one important difference: the mosaiced strips are sampled from varying positions in the input images. Thus it is called "non-stationary" mosaicing. Mosaic images obtained in this way are more similar to perspective images than traditional mosaic images. As I show in Chapter 4, using non-stationary mosaicing can be used to generate X-Slits images, where different strip-sampling functions correspond to different slit locations. Altering the parameters of the sampling function makes the generation of virtual walkthroughs possible.

In many image-based rendering techniques rays from a set of input images are collected and a new image is rendered by resampling the stored rays [22, 37, 41]. In order to create new perspective images of reasonable quality, the requirements become prohibitive: the number of stored rays becomes larger than available memory, and those rays are derived from a very large collection of carefully taken pictures. There are attempts to make IBR more efficient and more general [10, 17, 66], or to use such approximations as moving the camera in a lower dimensional space [64, 68].

The present work is mostly related to [17, 64, 66, 68] with several differences: First, rather than trying to approximate the perspective projection, I accurately define the projection geometry of the resulting images, and analyze the model limitations. Second, the rendering tool I present is very simple - slicing of the space-time volume obtained by a simple motion of a perspective camera. Consequently

the most important feature of my technique is the fact that ray-sampling for the generation of new views does not require detailed accounting of the parameters of the generating images. As I show below, if the camera's motion is sideways and constant, every vertical planar slice of the space-time volume gives *some* valid X-Slits image.

The study of X-Slits mosaicing presented in this work is based on work published in [1–3].

1.1.2 Crossed-Slits Geometry

While, as I show, the pinhole camera can be seen as a special case of the X-Slits projection, the geometry underlying the X-Slits projection in the general case is different from the geometry of the pinhole camera. I show that the pushbroom projection is also a special case of the X-Slits projection model, and in some sense it is the most distorted limiting case, i.e., the deviation from perspective projection in X-Slits images is maximal in the pushbroom limiting case.

In Section 3.2 I present an analysis and characterization of the epipolar geometry of the X-Slits projection. The motivation for doing this is twofold: First, for the purpose of visualization, it may give a theoretical foundation to X-Slits stereo pair generation. Second, understanding the epipolar geometry can aid image correspondence; in particular we would like to be able to match images of different kinds to each other, including X-Slits images, pushbroom images, perspective images with barrel distortion, and ideal perspective images. This can be used for such applications as 3D reconstruction, image warping, or animation.

Previously, the epipolar geometry of the pushbroom camera has been analyzed in [23]. A calibrated linear pushbroom camera was modeled as an orthographic projection in one axis (the input camera motion direction) and as a perspective projection in the other axis. It was encoded in a 3×4 non-homogeneous matrix with 11 degrees of freedom, allowing rotation, translation, scaling and rotation of the scanning plane. A study of camera retrieval, epipolar geometry and scene reconstruction manifests the quadratic nature of this projection, which I show to be a characteristic of the (more general) X-Slits model.

The epipolar geometry of X-Slits cameras resembles the pinhole epipolar geometry in some ways,

but has its own unique properties. In analogy with the pinhole case, there exists a 6×6 matrix which I call the *fundamental matrix* and denote $\mathbf{F}$. $\mathbf{F}$ gives a second order relation between corresponding image points, i.e., a correspondence between points image and second order epipolar curves (conics). Moreover, I show that the rank of $\mathbf{F}$ is 4; this result can be used to derive constraints on $\mathbf{F}$ to be used during the computation of its components.

The main novel feature of the X-Slits epipolar geometry is the fact that the epipolar conics do not usually match. In the pinhole case, all the image points on one epipolar line correspond to points on a *single* epipolar line in the second image. In the X-Slits case this is not generally the case; typically, each point on a certain epipolar conic in one image will induce a *different* epipolar conic in the second image. There are only two special cases where epipolar conics match each other uniquely in the X-Slits projection: (i) when the two cameras have a common slit, in which case the epipolar geometry is identical to the pinhole case, i.e., all epipolar curves are lines and all epipolar surfaces are planes; (ii) when the slits of the two cameras intersect each other in 4 distinct points, in which case the epipolar surfaces are quadratic but unique.

The study of X-Slits geometry presented in this work is based on work published in [4].

1.1.3 Omnidirectional Image-Based Rendering

New view generation is an emerging application which can benefit from both image-based techniques and graphics. The traditional approach to new view generation is to render a 3D model of the scene from different viewpoints. Unless the model is known a priori, this approach requires the recovery of the scene structure, which is a hard task. Moreover, the realistic synthesis of optical effects such as specularity, reflections and transparency is an involved problem.

The alternative approach, image-based rendering, advocates the use of raw images instead of 3D models. New views of the scene are generated based on a sequence of images, without a model of the scene, by sampling light rays. If the set of input images is dense, then the rays necessary for the synthesized image can be sampled from the input images, without knowledge about the scene and without attention to optical effects. However, the input images must be very precisely calibrated, and

together should contain all possible rays of the scene.

The set of all rays, as the plenoptic function [9], can be reasonably represented as a four-dimensional function. Using the full plenoptic function would require a very large amount of input data to produce synthetic views with good quality. The amount of data can be reduced using information about the scene or with some restrictions on the viewer's movement [17, 22, 30, 37, 41]. However, a more significant reduction (from 4D to 3D) can be obtained by using X-Slits mosaicing.

If, rather than generating perspective new views of the scene, we can settle for X-Slits views, we can do IBR with a much smaller plenoptic function. The input is taken from a perspective camera moving along a 1D path, and thus only a 3D subset of the 4D plenoptic function is sampled – all rays that intersect the camera path. X-Slits views of the scene can be rendered, with the horizontal slit at the input camera path, and the vertical "virtual" slit moving with the viewer. Although the images are not perspective, the sense of depth and occlusions is realistic and appealing.

In order to create a complete virtual environment, it is necessary to have rays in all directions, which leads to the choice of a circular camera path and panoramic perspective cameras [45]. I describe a setup in which a calibrated panoramic camera rotates off-axis in a circular path, so the set of rays thus collected is sufficient for generating X-Slits views with the vertical slit at *any* location inside the circle [13, 61, 64, 65, 68]. This is described in Section 4.2.

Since X-Slits images are not perspective, they may appear distorted. The main difference between X-Slits images and perspective images is that in the former the aspect ratio of fronto-parallel surfaces is not preserved, but rather it depends on depth. Thus scene objects may appear elongated or condensed depending on their depth and the virtual slit's location. This effect can be reduced by normalization – the image is transformed so as to cancel these distortions for all objects on a chosen surface; as long as this surface crudely approximates the scene structure, the amount of distortions decreases. Essentially, normalization provides a compromise between model-based and image-based rendering: we render based on a partial set of input rays, but approximate a perspective view using a coarse model of the scene.

Using an approximation surface was proposed in [67], which defines distortion in non-central im-

ages as the disparity between an image point and the corresponding point in an optimal perspective reprojection of an approximation surface. Given a non-central image, distortion is minimized by optimization.

It is often desirable to add virtual objects to the scene (e.g., for interaction or animation). When doing so, the objects must be rendered according to the same projection model as the image-based background, in order for them to blend into the scene geometrically. I show how to modify the geometry of the augmented objects to follow the distortions of the X-Slits projection. This allows us to use perspective rendering tools to augment the images, and still obtain consistent and compelling scenes.

The study of omnidirectional X-Slits rendering presented in this work is based on work published in [6].

1.1.4 Optimal Mosaicing

Mosaicing provides a method for imaging large scenes which may be impossible to capture in a single perspective image. However, multi-perspective images are inherently distorted (i.e., they are not perspective), and eliminating these distortions in the general case is only possible by perspective reprojection, which requires scene structure information. Retrieval of such information is computationally demanding and ill-posed (due to occlusion, reflections, transparency, etc.), therefore an alternative approach would be to try to minimize distortion without knowledge about the scene, based solely on the geometry of the multi-perspective mosaicing.

The liberty to choose any strip sampling scheme for mosaicing raises the following question: Given a sequence taken by a calibrated camera moving on a known trajectory, and an *unknown scene*, what is the best multi-perspective image that can be generated from it? In other words, which strips should be copied from the images and how should they be pasted into the mosaic, such that the result image will contain the maximal amount of visual information and minimal geometrical distortions? This subject is discussed in Section 4.3.

I define the necessary conditions for a good multi-perspective mosaic and a criterion quantifying the geometric distortions, and derive the least distorted mosaic under this criterion. The criterion is

justified theoretically as well as empirically. It turns out that the mosaic with the minimal distortion also has the maximal field-of-view. By minimizing the distortions of multi-perspective images, it is possible to generate a visually satisfying image with minor geometric distortions. These distortions may in many cases be practically negligible, especially in comparison with artifacts in perspective panoramas due to errors in depth estimation.

The study of optimal mosaicing presented in this work is based on work published in [5].

1.2 Motion Segmentation and Depth Ordering

The goal in motion-based segmentation is to partition images in a video sequence into segments of coherent motion. This problem, which is a fundamental layer in many vision tasks, has been studied extensively and several approaches to accomplish it have been suggested.

One popular approach is to assume that video motion is constrained by some global parametric motion model and the segmentation is according to values of the model parameters. In [49, 73], the image is partitioned into 2D layers assuming affine motion using MRF-based methods, while [32] takes advantage of the low dimensionality of the linear subspace of homographies representing planar motion and extracts layers by clustering. Other works, such as [50], model the video using 3D perspective geometry. Note that these two approaches represent different understandings of the term "segment", as multi-layer rigid objects would be considered a single segment by the latter while subdivided by the former.

An alternative approach assumes motion is piecewise smooth, so that motion within a segment is smooth and motion discontinuity occurs only at boundaries between segments. [72] uses a prior distribution on flow fields to obtain smooth dense flow fields for all segments. The MRF-based technique of [16, 34] finds an optimal flow assignment which penalizes for unsmooth motion between neighboring pixels with similar color. The tensor voting technique [47] extracts smooth structures in space-time-velocity domain, yielding a piecewise-smooth flow field.

Assuming only piecewise smoothness is a potentially more general approach, compared with explicit geometrical constraints, and it lies at the base of the method proposed here. Note that piecewise

smoothness represents a different understanding of "segments" – a non-rigid moving object would be subdivided by a geometry-based algorithm while outlined as a single segment by a piecewise smoothness algorithm; on the other hand, a self-occluding object would be subdivided by a piecewise smoothness algorithm while outlined as a single segment by a geometry-based algorithm.

Motion discontinuities can be identified by clustering a previously computed motion field (e.g., [50, 73]). The problem is that such discontinuities are found at exactly those locations where the computation of the motion field is least reliable: since all optical flow algorithms rely on the analysis of a region around a point (even if only to compute first-order derivatives), the optical flow must be continuous within the region to support reliable computation. This chicken-and-egg problem makes motion segmentation particularly challenging. On the other hand, the successful computation of motion discontinuities can be useful for a number of applications, including motion computation (by highlighting those areas where the computation should be considered unreliable) and object segmentation from multiple cues. Here I propose a motion segmentation method that does not require a reliable optical flow to begin with.

Having segmented the image, we next want to determine the occlusion order of objects in the image, as the first step in 3D scene understanding and object recognition. In principle, any depth-retrieval algorithm (e.g., [34]) would also provide depth ordering. However, full 3D reconstruction is usually only practical in static scenes, and it relies on accurate geometric calibration which remains a hard task. In this work I present a method to compute depth ordering from occlusion cues without explicit scene reconstruction. The most surprising characteristic of this method is its ability compute depth ordering from only two frames.

The problem of depth ordering is similar to figure/ground segregation, an issue which has been studied extensively in the context of Gestalt psychology. Many possible spatial cues may contribute to figure perception from a single image, including *convexity* [52], *junctions* [59], and *familiar configurations* [58]). However, depth ordering from a single image may be subjective and prone to ambiguities, whereas motion gives a very powerful and usually unambiguous cue.

Given an image sequence, the accretion and deletion of texture elements [31], as well as the *com-*

mon fate of texture and edge [19, 75], have long been recognized as cues for depth ordering. There are several methods for depth ordering from three frames or more, e.g., by tracking disappearing texture elements [44], optical flow filling [50], detecting T-junctions in space-time [12, 48], matching the motion of surface and boundary [18,20,70] and localization of errors in flow computation w.r.t. monocular segmentation [15].

However, when given only two frames, it is impossible to determine depth ordering from motion alone, without additional assumptions or prior knowledge. This is because the motion of pixels that become occluded cannot be determined, and thus they may belong to either side of the motion edge, leading to more than one valid order assignment. One solution would be to assume that the occluded pixels belong to the layer that is more similar in appearance; i.e., determine depth ordering by matching the motion of color and motion edges [71]. However, color edges are often unreliable as edges between layers, since the figure and ground may have similar colors.

1.2.1 Motion Segmentation

My work is based on the extraction of motion boundaries, which are defined *locally* as boundaries between different motions (since many real video sequences do not obey any global motion model). While some methods are based on color or texture boundaries between the moving object and the background (e.g., [21, 34, 69]), I restrict myself to solutions which do not rely on such boundaries. This is motivated by humans' ability to segment objects from motion alone (e.g., in random dot kinematograms), and by the need to avoid over-segmentation of objects whose appearance includes varying color and textures. Finally, I only consider local properties of the temporal profile of motion, in order to be able to deal with pairs of frames or stereo pairs (but see, for example, [63]).

In my approach I start by considering the video sequence as a spatio-temporal intensity function, where the goal is to extract information from this spatio-temporal structure. Video sequences have highly regular temporal structure, with regions of coherent motion forming continuous tube-like structures. These structures break where there is occlusion, creating spatio-temporal corner-like features. Using a differential operator that detects such features, I develop an algorithm that extracts motion

boundaries.

Specifically, my algorithm is based on the occlusion detector described in Section 5.1, which is used to extract a motion boundary at any given scale. Since different scales may be appropriate for different parts of the image, a cross-scale optimal boundary is computed, based on the response of the detector. At the end, a closed contour is built along the most salient boundary fragments to provide the final segmentation. The algorithm was evaluated on three challenging real sequences, as well as several synthetic examples which are particularly difficult for some commonly used algorithms, in order to demonstrate the robustness of this method. Some results from other algorithms, whose implementation was made available by the authors, are provided for comparison. In Section 5.3 I analyze the behavior and mathematical properties of the algorithm.

The motion segmentation algorithm presented in this work is based on [7, 8].

1.2.2 Depth Ordering

My computational approach to the problem of ordinal depth from two frames utilizes the principle of common fate of texture and boundary, though without attempting to extract the boundary explicitly. The spatio-temporal partial derivatives in each frame are affected by both the motion of the layers (i.e., their texture), and the motion of the motion boundary. When using my occlusion detector, which relies on these derivatives, a bias towards the occluded side appears. The bias depends on the density gap between the two layers (this bias disappears when the layers have the same local density). Moreover, when measuring this bias in scale space, it can be seen to increase as the scale is increased.

From this observation I derive an algorithm in Section 5.4, which computes the ordinal depth of two layers based on the trend of the bias in scale space. With some minor modifications, I show that the same algorithm can be applied to three-frame sequences, without relying on local differences of density between the layers. The algorithms are shown to perform well on real sequences. The performance of the algorithms is compared with the performance of human subjects on two- and three-frame sequences of random-dot textures of varying density and to an ideal observer model in Section 5.5.

The depth ordering algorithm presented in this work is based on [8].

Chapter 2

Methodology

This chapter summarizes the mathematical tools used in the inference of this work. In Section 2.1 I give an introduction to the geometry and algebra of projection models, and in Section 2.2 I introduce epipolar geometry and the fundamental matrix. Section 2.3 gives a brief introduction to scale space.

2.1 Projection Models

For simplicity, I describe a camera as a device that projects a 3D scene onto an image surface. Each scene point has one ray that passes through it, and the image of the scene point is the intersection of this ray with the image surface. Thus, all points on a ray are projected onto the same image point, which also implies that every image point is the projection of *some* scene point. A well-defined camera has exactly one ray passing through every image point (although in this work I tolerate cameras that have a few singular image points that do not fulfil this requirement).

2.1.1 Pinhole Camera

The most commonly-used cameras, as well as our eyes, can be modeled by the *pinhole (*or *perspective) camera* model. For such a camera, all rays intersect a single point (the "pinhole") and the image surface is usually a plane (although panorama photography often uses cylinders or spheres). For this presentation I ignore the issues of occlusion and camera direction and the effect of the lens and aperture.

A simple algebraic representation of the pinhole camera is as follows: a 3D scene point (X, Y, Z) is projected onto the 2D image point (x, y) given by

$$(x, y) \quad = \quad (X/Z, Y/Z) \tag{2.1}$$

In *homogeneous* coordinates, this is expressed as

$$p \propto \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{p} \tag{2.2}$$

where $p \in \mathcal{P}^2$ is the image of the scene point $\mathbf{p} \in \mathcal{P}^3$. In this representation, referred to as the *calibrated* pinhole camera, the Z axis is referred to as the *optical axis* of the camera, and the origin is the *optical center*. The most general representation of a pinhole camera, or the *uncalibrated* pinhole camera, is given by

$$p \propto \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T t \\ 0 & 1 \end{bmatrix} \mathbf{p} = \mathbf{K} \left[\mathbf{R}^T; -\mathbf{R}^T t\right] \mathbf{p} = \mathbf{M}_{3\times 4}\mathbf{p} \tag{2.3}$$

where $\mathbf{K}_{3\times 3}$ is the *internal calibration matrix*, and $\mathbf{R}_{3\times 3}$ and $t \in \mathbf{R}^3$ represent, respectively, the rotation of the camera and the translation from the origin.

The internal calibration consists of five parameters that represent characteristics of the camera:

- f_x (resp. f_y) is the ratio between the *focal length* and the horizontal (resp. vertical) pixel size. The ratio f_x/f_y is the *aspect ratio*, and is usually 1.
- $s = \tan\alpha$, where α is the *skew angle*, which is usually 0.
- (c_x, c_y) is the *principal point*, which is usually $(0, 0)$.

The *external calibration* consists of $\mathbf{R}$ and t, which determine the camera location and orientation in the scene coordinate system. $\mathbf{R}$ is a rotation matrix, i.e., $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. Thus, the uncalibrated pinhole camera is determined by 11 independent parameters. Any non-degenerate matrix $\mathbf{M}$ represents some pinhole camera (note that $\mathbf{M}$ is a homogeneous entity).

Given two scene points $\mathbf{p}_1, \mathbf{p}_2 \in \mathcal{P}^3$, the set of all points on the line joining $\mathbf{p}_1, \mathbf{p}_2$ is given by $\{\alpha_1\mathbf{p}_1+\alpha_2\mathbf{p}_2|\alpha_1, \alpha_2 \in \mathcal{R}\}$. Thus the projection of this line is given by $\{\alpha_1\mathbf{M}\mathbf{p}_1+\alpha_2\mathbf{M}\mathbf{p}_2|\alpha_1, \alpha_2 \in \mathcal{R}\}$, which is the line that joins $\mathbf{M}\mathbf{p}_1, \mathbf{M}\mathbf{p}_2$ in the image. Therefore, the pinhole camera projects scene lines onto image lines.

2.1.2 Affine Camera

The *affine (*or *orthographic) camera* is a camera where all rays are parallel. The calibrated affine camera can be represented simply by

$$(x, y) = (X, Y) \tag{2.4}$$

and in the general (uncalibrated) case by

$$p \propto \begin{bmatrix} \mathbf{A}_{2\times 3} & b \\ 0 & 1 \end{bmatrix} \mathbf{p} \tag{2.5}$$

It is determined by 8 parameters. It is often used as an approximation of the pinhole camera, especially for large focal lengths. Algebraically, it can be seen as a special case of the pinhole camera (where the bottom row of $\mathbf{M}$ is $(0, 0, 0, 1)$).

2.2 Multiple View Geometry

2.2.1 Homographies

If scene point $\mathbf{p} \propto (p_\pi^T; 1)^T$ lies on a plane $\Pi \propto (\pi^T; 1)^T$, then $\Pi^T\mathbf{p} = 0$. Thus, the scene point can be expressed in homogeneous coordinates as

$$\mathbf{p} \propto \begin{bmatrix} \mathbf{I}_{3\times 3} \\ -\pi^T \end{bmatrix} p_\pi \tag{2.6}$$

and it is projected by the camera $[\mathbf{A}_{3\times 3}; a]$ onto

$$p \propto (\mathbf{A} - a\pi^T)p_\pi = \mathbf{H}p_\pi \tag{2.7}$$

where $\mathbf{H}$ is a homography, i.e., it describes a mapping from one plane to another (in this case, from Π to the image plane).

Denoting the homography that maps points from plane Π to camera i as $\mathbf{H}_i$, one can map image points from camera i to camera j through the reference plane with the homography

$$\mathbf{H}_{ij} = \mathbf{H}_j \mathbf{H}_i^{-1} \tag{2.8}$$

It is easy to see that when the cameras have the same optical center, this homography is independent of Π; in other words, when there is no translation between two given cameras, there is a mapping that transforms one image to the other.

2.2.2 Epipolar Geometry

Next I shall derive the relation between corresponding image points in two uncalibrated cameras based on concepts introduced in [25]. By definition, every ray of camera i passes through its optical center. Denoting e_j^i to be the projection of the optical center of camera i in camera j, it follows that every ray of camera i is projected by camera j onto a line that passes through e_j^i. Given a point p_i in camera i, the line that joins it with the epipole e_i^j is given by $l_i = e_i^j \times p_i$. It can be shown that the plane given by $\Pi \propto \mathbf{M}_i^T l_i$ passes through the optical center of camera i and the projection of every point on Π lies on l_i. If $\mathbf{p}$ is a scene point whose image in i is p_i, then

$$0 = (\mathbf{M}_i \mathbf{p})^T l_i = \mathbf{p}^T \mathbf{M}_i^T l_i = \mathbf{p}^T \Pi \tag{2.9}$$

i.e., $\mathbf{p}$ lies on Π. In the same way, it can be shown that the optical center of camera j also lies on Π. Thus, Π, which is called an *epipolar plane*, joins the optical centers of both cameras and the scene point. It is easy to see that $\Pi \propto \mathbf{M}_j^T (e_j^i \times p_j) = \mathbf{M}_j^T l_j$, where p_j corresponds to p_i in camera j (i.e., they are both projections of the same scene point $\mathbf{p}$). This means that corresponding points in two different cameras have a single epipolar plane.

The lines l_i, l_j are called epipolar lines. From the discussion above it follows that every point on the line l_i in camera i corresponds to a point that lies on l_j in camera j. Denoting the pseudoinverse of

$\mathbf{M}$ as $\mathbf{M}^+$, it can be shown that

$$l_j \propto \left(\mathbf{M}_j^T\right)^+ \mathbf{M}_i^T l_i = \left(\mathbf{M}_j^T\right)^+ \mathbf{M}_i^T [e_i^j]_\times p_i \equiv \mathbf{F} p_i \tag{2.10}$$

where $[e_i^j]_\times$ denotes the skew-symmetric matrix representing the cross product. It follows that for every pair of corresponding points $p_j^T \mathbf{F} p_i = 0$. The matrix $\mathbf{F}$ is called the *fundamental matrix*. It maps every point in one camera to a line in the other camera, which any corresponding point must lie on, and vice versa. The fundamental matrix is a homogeneous 3×3 matrix, and since $\mathbf{F} e_i^j = 0$, and thus $\det \mathbf{F} = 0$, it follows that $\mathbf{F}$ has 7 degrees of freedom.

2.3 Scale Space

Scale space theory deals with representation and analysis of signals at different detail levels. The notion of *scale* determines the level of details, so that fine structures are progressively suppressed at coarser scales.

A canonical scale space representation that is commonly used in signal processing and computer vision is the *linear scale space*, presented and discussed extensively in [39]. The linear scale space representation of a d-dimensional signal $I(x_1, \ldots, x_d)$ is defined as

$$L(x_1, \ldots, x_d; s) = I * g_s \tag{2.11}$$

where $s \in [0, \infty)$ denotes scale and $*g_s$ denotes the convolution with the d-dimensional Gaussian function with variance s (or "smoothing"). Thus, the scale space representation is $(d+1)$-dimensional – it has an additional scale dimension. Linear scale space is the solution of the diffusion equation

$$\frac{\partial L}{\partial s} = \frac{1}{2} \nabla^2 L \qquad\qquad L(x_1, \ldots, x_d; 0) = I(x_1, \ldots, x_d) \tag{2.12}$$

The intuition behind this choice is easily observed in 1D signals – as we progress towards higher (coarser) scales, concave regions are decreased and convex regions are increased. In fact, in 1D signals it can be shown that local extrema are not created as scale is increased.

This stems from the notion of *causality*, which requires that features in a coarse scale originate from features in a finer scales, or in other words, features are not created in coarser scales. In signals of higher

dimension it is not necessarily the case that local extrema are not created; instead, causality requires that local extrema are not enhanced. Other features of linear scale space include shift invariance, rotational invariance, scale invariance and of course linearity. Throughout this work by "scale space" I will refer to linear scale space.

2.3.1 Feature Detection

Since differentiation commutes with convolution, differentiating L can be done simply by applying

$$\frac{\partial^n L(\cdot;s)}{\partial x_i^n} = \frac{\partial^n (I * g_s)}{\partial x_i^n} = \left(\frac{\partial^n g_s}{\partial x_i^n}\right) * I \tag{2.13}$$

Using derivatives of the image signal, there are several feature types that can be detected, such as edges and corners. Applying the same differential operators to the image representation at different scales would give different results – fine scales tend to be more detailed and accurate, and perhaps more noisy; while coarser scales tend to represent only dominant features, albeit with a less accurate localization.

Consider edges in an image. As suggested in [38], one way to define an edge is points where the gradient magnitude is maximal in the gradient direction. This can be expressed as a differential operator as

$$\begin{cases} I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy} = 0 \\ I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x I_y^2 I_{xyy} + I_y^3 I_{yyy} < 0 \end{cases} \tag{2.14}$$

This operator can be applied to $L(\cdot;s)$ for any scale s. Since there are advantages and disadvantages to both fine and coarse scales, it is desirable to develop an adaptive scale-selection mechanism, which would determine the optimal scale for edge extraction in every region of the image. An image may contain sharp edges as well as diffuse edges, and their appearence changes considerably with scale – sharp edges may become smooth or even disappear at coarse scales, while diffuse edges may bifurcate at fine scales.

The first problem is how to compare operator values between different scales. Since the amplitude of the n derivatives at decreases with scale by a factor of $s^{n/2}$, a common heuristic principle is to use

scale-normalized derivatives, given by

$$\frac{\partial^n}{\partial x_i^n}^{(s)} \equiv s^{n\gamma/2} \cdot \frac{\partial^n}{\partial x_i^n} \tag{2.15}$$

for some positive normalization parameter γ. In the case of edge detection, a good choice for this parameter is $\gamma = 1/2$. To see this, suppose we have a synthetic image with a step edge, diffused by a Gaussian with variance s_0, given by

$$I(x, y) = h_{s_0}(x) \equiv \int_{-\infty}^{x} g_{s_0}(x')dx' \tag{2.16}$$

It is easy to see that the operator (2.14) will detect the edge correctly at $x = 0$ at all scales for any γ. If we consider the magnitude of the gradient as a measure of edge intensity, then it can be shown that this measure assumes maximum at scale $s = \frac{\gamma}{1-\gamma} s_0$. If $\gamma = 1/2$, then the edge is detected at $s = s_0$ – the "scale" of the edge. In other words, this mechanism enables us to automatically select the scale where the edge "resides" – sharp edges would be detected at fine scales and diffuse edges would be detected at coarse scales.

To summarize, edge detection in scale space is done by intersecting the set of points detected by (2.14) over all scales, with the set of points where the edge strength measure assumes maximum over scale, given by

$$\begin{cases} \frac{\partial}{\partial s}(I_x^{2(s)} + I_y^{2(s)}) = 0 \\ \frac{\partial^2}{\partial s^2}(I_x^{2(s)} + I_y^{2(s)}) < 0 \end{cases} \tag{2.17}$$

Other edge strength measures have given similar results. Note that the introduction of the scale normalization does not affect the outcome of the edge operator (2.14). This property of the edge operator is present in a large class of operators that are used for detecting various image and video features such as corners, blobs and ridges.

Chapter 3

The Crossed-Slits Projection

In this chapter I present the Crossed-Slits (X-Slits) projection and study its properties. The X-Slits projection is defined by two non-intersecting lines which all rays intersect. This projection is generally non-perspective, as there is no single point which lies on all rays, though the perspective projection can be seen as a special case of the X-Slits model. A concise algebraic formulation is given, from which various properties of this projection model can be inferred. This representation is then used to derive the epipolar geometry.

3.1 Projection Model

Fig. 3.1a shows the basic design of the X-Slits camera as built by Ducos du Hauron in 1888 [33]. A more general design is shown in Fig. 3.1b. A X-Slits camera has two slits $\mathbf{l}_1, \mathbf{l}_2$ which should be two different lines in 3D space, and an image plane Π that does not contain any of the slits. For every 3D point not lying on either of the slits there is a single ray which connects the point with both slits simultaneously. The intersection of this ray with the image plane defines the projected image of the 3D point. The camera in Fig. 3.1a is a special case of the X-Slits camera, where the two slits are orthogonal to each other and parallel to the image plane. I call this special arrangement the Orthogonal X-Slits camera.

The X-Slits model is a valid 3D to 2D projection, defining a many-to-one mapping from the 3D

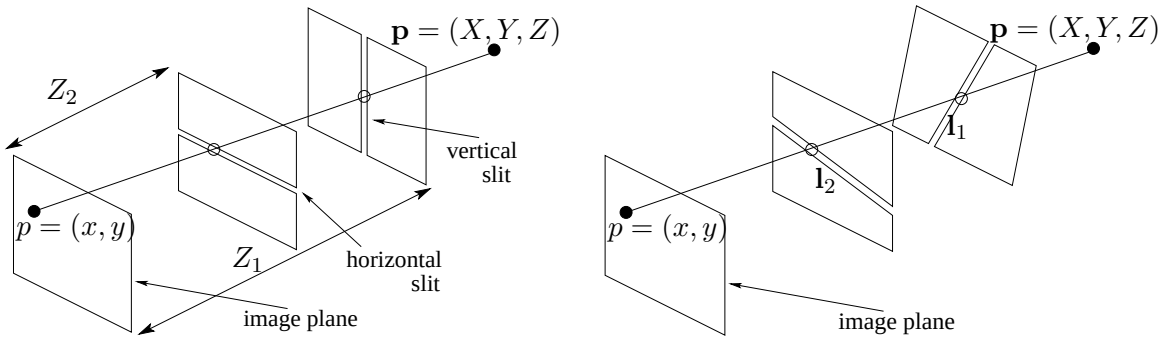


Figure 3.1: (a) A design of a X-Slits camera where the slits are orthogonal to each other and parallel to the image plane (Orthogonal X-Slits camera). The projection ray of a 3D point $\mathbf{p} = (X, Y, Z)$ is shown, with circles showing its intersection points with the 2 slits. (b) A general X-Slits design, with two arbitrary slits $\mathbf{l}_1, \mathbf{l}_2$. Note that the camera is defined by the specific configuration of the two slits and the image plane; any of these three factors can change independently, giving rise to a different X-Slits camera.

world to the 2D image plane. Below I develop the specific equations of the camera mapping as a function of the slits $\mathbf{l}_1, \mathbf{l}_2$ and the image plane Π and discuss this model's properties.

3.1.1 Definition

Consider the camera configuration as shown in Fig. 3.1b. The *projection ray* of a point $\mathbf{p} \in \mathcal{P}^3$ (represented by homogeneous coordinates) intersects the two camera slits $\mathbf{l}_1, \mathbf{l}_2$. It is thus the intersection of two planes, defined by joining the point $\mathbf{p}$ with each of the slits. The image of $\mathbf{p}$ is the intersection of this projection ray with the image plane. Furthermore, all points on a projection ray will project to the same image point (unless the ray lies on the image plane).

More formally, let $\mathbf{u}_i, \mathbf{v}_i$ denote two planes in $\mathcal{P}^3$ which contain slit $\mathbf{l}_i$, for $i \in \{1, 2\}$ respectively. For any α, [1] the plane given by $\mathbf{u}_i + \alpha\mathbf{v}_i$ also contains the slit $\mathbf{l}_i$; and vice versa, every plane that contains the slit can be described by $\mathbf{u}_i + \alpha\mathbf{v}_i$ for some α. Thus, for each scene point $\mathbf{p} \in \mathcal{P}^3$ and $i \in \{1, 2\}$, there exists $\alpha \in \mathcal{R}$ such that $(\mathbf{u}_i + \alpha\mathbf{v}_i)\mathbf{p} = 0$. Eliminating α, we get the following

[1] α denotes a projective parameter as in [62] page 43.

expression for the plane:

$$\mathbf{u}_i - \frac{(\mathbf{u}_i^T \mathbf{p})}{(\mathbf{v}_i^T \mathbf{p})}\mathbf{v}_i \quad \propto \quad (\mathbf{v}_i^T \mathbf{p})\mathbf{u}_i - (\mathbf{u}_i^T \mathbf{p})\mathbf{v}_i \quad = \quad (\mathbf{u}_i \mathbf{v}_i^T - \mathbf{v}_i \mathbf{u}_i^T)\mathbf{p} \tag{3.1}$$

Let us define the skew-symmetric matrix

$$\mathbf{S}_i^* = \mathbf{u}_i \mathbf{v}_i^T - \mathbf{v}_i \mathbf{u}_i^T \tag{3.2}$$

Matrix $\mathbf{S}_i^*$ is actually the dual Plücker matrix representation of the line $\mathbf{l}_i$ [26]. It can be shown to be independent of the choice of $\mathbf{u}_i, \mathbf{v}_i$. The elements of such matrices must satisfy one non-linear constraint (since a line has only 4 degrees of freedom up to scale).

From (3.1) and (3.2) it follows that the plane which contains point $\mathbf{p}$ and slit $\mathbf{l}_i$ is $\mathbf{S}_i^*\mathbf{p}$. The projection ray of the scene point $\mathbf{p}$ is therefore defined by the intersection of the two planes $\mathbf{S}_1^*\mathbf{p}$, and $\mathbf{S}_2^*\mathbf{p}$ defined in (3.2).

Next, observe that the image of scene point $\mathbf{p}$ is the intersection of the projection ray with the image plane Π. Let $\mathbf{m} \in \mathcal{P}^3$ denote a point on plane Π, and let $\mathbf{j}, \mathbf{k} \in \mathcal{P}^3$ denote two distinct points at infinity which also lie on plane Π. Every point on Π can be expressed as $x\mathbf{j} + y\mathbf{k} + w\mathbf{m}$. The projection ray of $\mathbf{p}$ intersects plane Π at a certain point $\mathbf{q} \propto x\mathbf{j} + y\mathbf{k} + w\mathbf{m}$ such that $(\mathbf{S}_1^*\mathbf{p})^T\mathbf{q} = 0$ and $(\mathbf{S}_2^*\mathbf{p})^T\mathbf{q} = 0$. This gives us a set of linear equations in x, y and w (the homogeneous image coordinates of point $\mathbf{q}$), namely:

$$\begin{bmatrix} \mathbf{p}^T\mathbf{S}_1^*\mathbf{j} & \mathbf{p}^T\mathbf{S}_1^*\mathbf{k} & \mathbf{p}^T\mathbf{S}_1^*\mathbf{m} \\ \mathbf{p}^T\mathbf{S}_2^*\mathbf{j} & \mathbf{p}^T\mathbf{S}_2^*\mathbf{k} & \mathbf{p}^T\mathbf{S}_2^*\mathbf{m} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} = 0 \tag{3.3}$$

Define the following skew-symmetric matrices

$$\mathbf{Q}_1 = \mathbf{k}\mathbf{m}^T - \mathbf{m}\mathbf{k}^T \qquad \mathbf{Q}_2 = \mathbf{m}\mathbf{j}^T - \mathbf{j}\mathbf{m}^T \qquad \mathbf{Q}_3 = \mathbf{j}\mathbf{k}^T - \mathbf{k}\mathbf{j}^T \tag{3.4}$$

the solution of the linear system (3.3) is[2]

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \propto \begin{pmatrix} \mathbf{p}^T\mathbf{S}_1^*\mathbf{Q}_1\mathbf{S}_2^*\mathbf{p} \\ \mathbf{p}^T\mathbf{S}_1^*\mathbf{Q}_2\mathbf{S}_2^*\mathbf{p} \\ \mathbf{p}^T\mathbf{S}_1^*\mathbf{Q}_3\mathbf{S}_2^*\mathbf{p} \end{pmatrix} \tag{3.5}$$

[2]The solution is a unique point unless $\mathbf{p}$ resides on the line joining the intersections of the two slits with the image plane.

and $\mathbf{p}$ is projected to $(\frac{x}{w}, \frac{y}{w})$. Note that $\mathbf{S}_i^*$ depends only on the slit $\mathbf{l}_i$, while $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3$ depend only on the plane of projection Π. $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ are the Plücker matrix representations of the image X axis, the image Y axis and the image line at infinity, respectively (all in 3D). Finally, note that the choice of $\mathbf{j}$ and $\mathbf{k}$ to be points at infinity limits the internal calibration to an affine transformation of the image.

The camera projection model in (3.5) is a quadratic mapping from $\mathcal{P}^3$ to $\mathcal{P}^2$, and therefore a X-Slits camera cannot be described by a 3×4 projection matrix over $\mathcal{P}$. Let $\nu : \mathcal{P}^3 \rightarrow \mathcal{P}^9$ denote the Veronese map given for $\mathbf{p} = (p_1, p_2, p_3, p_4)^T$ by

$$\nu(\mathbf{p}) = (p_1^2, p_1p_2, p_1p_3, p_1p_4, p_2^2, p_2p_3, p_2p_4, p_3^2, p_3p_4, p_4^2)^T \tag{3.6}$$

Using this notation, the transformation in (3.5) can be concisely written as

$$p \propto \mathbf{A}\nu(\mathbf{p}) \tag{3.7}$$

where $\mathbf{A}$ is a 3×10 matrix determined by the two slits and the image plane (or, equivalently, the five camera matrices $\mathbf{S}_1^*,\ \mathbf{S}_2^*,\ \mathbf{Q}_1,\ \mathbf{Q}_2,\ \mathbf{Q}_3$).

3.1.2 Orthogonal X-Slits camera

A configuration of special interest is when the slits are orthogonal to each other and parallel to the image plane. I will refer to this configuration as *Orthogonal X-Slits* (see Fig. 3.1a).

Without loss of generality I fix the slits by assigning $\mathbf{u}_1 = (1, 0, 0, 0)^T$, $\mathbf{v}_1 = (0, 0, 1, -Z_1)^T$, $\mathbf{u}_2 = (0, 1, 0, 0)^T$, and $\mathbf{v}_2 = (0, 0, 1, -Z_2)^T$. This defines a vertical slit at $X = 0$, $Z = Z_1$ and a horizontal slit at $Y = 0$, $Z = Z_2$. I also denote $\Delta = Z_2 - Z_1$. Plane Π is the $X - Y$ plane at $Z = 0$, and therefore I assign $\mathbf{m} = (0, 0, 0, 1)^T$, $\mathbf{j} = (1, 0, 0, 0)^T$, $\mathbf{k} = (0, 1, 0, 0)^T$. From (3.5) it follows that

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -Z_1 \frac{X}{Z - Z_1} \\ -Z_2 \frac{Y}{Z - Z_2} \end{pmatrix} \tag{3.8}$$

This projection equation is identical to the model first analyzed in [1], where it was called bi-centric projection.

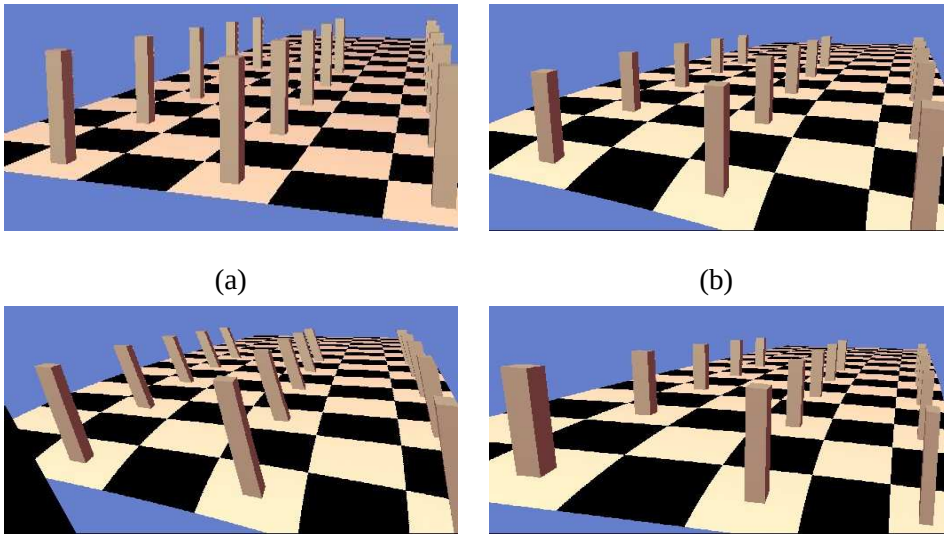


Figure 3.2: Simulated X-Slits images of an artificial scene. (a) A regular pinhole image. (b) Orthogonal X-Slits image using the projection equation from (3.8). (c) Same as (a), but with the vertical slit rotated about the Z axis. (d) Same as (a), but with the vertical slit rotated about the X axis.

3.1.3 Properties

It follows from (3.5) that the X-Slits projection is defined by three 4×4 matrices or a $3 \times 4 \times 4$ tensor. It is a homogeneous object, so it is defined up to a scale. For any matrix $\mathbf{A}$ and any anti-symmetric matrix $\mathbf{B}$

$$\mathbf{p}^T(\mathbf{A} + \mathbf{B})\mathbf{p} = \mathbf{p}^T\mathbf{A}\mathbf{p} \tag{3.9}$$

and therefore the anti-symmetric part of the matrices of (3.5) has no effect on the projection. This means that the projection is defined only up to its symmetric part - by 29 variables . However, the number of degrees of freedom is much smaller - each slit has 4 degrees of freedom up to a scalar, the plane has 3 degrees of freedom, and there are additional 6 degrees of freedom[3] for the choice of the origin and the axes in the image plane, yielding a total of 17 degrees of freedom.

To get some intuition for X-Slits images, Fig. 3.2 shows examples of X-Slits images as compared with pinhole images of the same scene. I also show the effects of varying the relative geometry of the

[3]The three points $\mathbf{j}, \mathbf{k}, \mathbf{m} \in \mathcal{P}^{\mathbf{3}}$ are confined to a plane, leaving 2 degrees of freedom for each, up to scaling.

slits and image plane. Next I demonstrate algebraically a few interesting properties of X-Slits images.

Projection of Lines

In X-Slits images, the image of a 3D line is a conic. Given a line represented by the Plücker matrix $\mathbf{L}$ that does not coincide with either of the slits, consider the quadric surface $\mathbf{S}_1^*\mathbf{L}\mathbf{S}_2^*$. The projection ray of a point $\mathbf{p} \in \mathcal{P}^3$ not on either of the slits is the intersection of the planes $\pi_i = \mathbf{S}_i^*\mathbf{p}$ for $i = 1, 2$, and the line $\mathbf{L}$ intersects π_i at $\mathbf{L}\pi_i$. Therefore, $\mathbf{p}$ lies on the quadric $\mathbf{S}_1^*\mathbf{L}\mathbf{S}_2^*$ if and only if its projection ray intersects $\mathbf{L}$.

In other words, this quadric is the surface of all projection rays of points on $\mathbf{L}$. Since the intersection of a quadric with a plane is a conic, the X-Slits image of a line $\mathbf{l}$ is always a conic.

If $\mathbf{L}$ intersects the slit $\mathbf{l}_2$ at some point $\mathbf{q}$, then it can be shown that $\mathbf{L}\mathbf{S}_2^*\mathbf{p} = \mathbf{q}$ for any $\mathbf{p} \in \mathcal{P}^3$. Therefore, the quadric degenerates to the plane $\mathbf{S}_1^*\mathbf{q}$. Equivalently, if $\mathbf{L}$ intersects the slit $\mathbf{l}_1$ at $\mathbf{q}$, then the quadric degenerates to the plane $\mathbf{S}_2^*\mathbf{q}$. Hence, the projection of a line that intersects one of the slits is a line.

In the special case of the Orthogonal X-Slits camera, if the 3D line is perpendicular to the Z axis, i.e., it is of the form $(a + c\lambda, b + d\lambda, e)$, then its image is the curve $(x, y) = \left(-Z_1\frac{a+c\lambda}{e-Z_1}, -Z_2\frac{b+d\lambda}{e-Z_2}\right)$, which is a line; if the 3D line is *not* perpendicular to the Z axis, i.e., it is of the form $(a+c\lambda, b+d\lambda, \lambda)$, then its image is the curve

$$(x, y) = \left(-Z_1\frac{a + c\lambda}{\lambda - Z_1}, -Z_2\frac{b + d\lambda}{\lambda - Z_2}\right) \tag{3.10}$$

Solving for λ, we find that the line is projected to the hyperbola given by

$$(Z_1 - Z_2)xy + Z_2(b + Z_1d)x - Z_1(a + Z_2c)y + Z_1Z_2(bc - ad) = 0 \tag{3.11}$$

For $Z_1 \neq Z_2$, this hyperbola degenerates to a line if and only if $a + Z_1c = 0$ or $b + Z_2d = 0$, that is, only if the line intersects one of the slits.

The distortion of straight lines is illustrated in Fig. 3.2b. In practice this distortion is not very disturbing, as can be seen in the examples in Section 4.1.4. In the various scenes I have experimented

with, this distortion was rather minor. Many scenes, particularly natural scenes, do not have very dominant straight lines, in which case this distortion is hardly noticed. Furthermore, people are accustomed to similar effects caused by lens distortions.

Aspect Ratio Distortions

The most apparent aspect of the distortion in Orthogonal X-Slits images is the variation of aspect ratio (especially in pushbroom images [23]). The apparent aspect ratio of objects in the image depends on their depth. This is unlike the perspective model, in which the distortion in aspect ratio is constant for all objects.

From (3.8) it follows that an object at depth Z with aspect ratio of 1 would appear on the image plane to have an aspect ratio of

$$\frac{\Delta y}{\Delta x} = \frac{Z_2}{Z_1} \cdot \frac{Z - Z_1}{Z - Z_2} \tag{3.12}$$

In practice, I found this distortion to be typically rather insignificant. If the range of depth values of scene objects is not too large, we can normalize the image to compensate for this distortion by scaling. Specifically, if we cancel the aspect ratio distortion for some intermediate depth value Z_0, the distortion at depth Z would be

$$\frac{Z - Z_1}{Z - Z_2} \cdot \frac{Z_0 - Z_2}{Z_0 - Z_1} \tag{3.13}$$

To demonstrate the magnitude of the distortion, consider the following example: Suppose the depth range of objects in the scene is $3 - 5$ meters (measured from the horizontal slit at Z_1, i.e., $3\text{m} < Z - Z_1 < 5\text{m}$), and assume that the images are normalized so that objects at the depth of 3.84m appear undistorted (i.e., $Z_0 - Z_1 = 3.84\text{m}$). If the vertical slit is behind the horizontal slit at $\Delta = -2.5\text{m}$, the aspect ratio distortion would not exceed 10%.

Tilted Slit

Departing from the Orthogonal X-Slits camera, suppose we tilt the vertical slit sideways. Assigning all substitutions as with the Orthogonal X-Slits configuration, except that $\mathbf{u}_1 = (1, a, 0, 0)^T$ for some

$a \in \mathcal{R}$, we obtain

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -Z_1 \frac{X+aY}{Z-Z_1} + a\left(Z_2 \frac{Y}{Z-Z_2}\right) \\ -Z_2 \frac{Y}{Z-Z_2} \end{pmatrix} \tag{3.14}$$

By substituting $x' = x + ay$, $y' = y$ and $X' = X + aY$, $Y' = Y$, $Z' = Z$, we get the simple model of (3.8) for the projection of (X', Y', Z') to (x', y'). In other words, we get a skewed image of a skewed scene. Such a projection is demonstrated in Fig. 3.2c (note that the skew is 3-dimensional).

Relation to Other Projection Models

The pinhole camera model is a special case of the X-Slits model, obtained when the two slits intersect. Let there be three planes $\mathbf{u}, \mathbf{v}_1, \mathbf{v}_2$ defining the slits so that $\mathbf{S}_i^* = \mathbf{u}\mathbf{v}_1^T - \mathbf{v}_1^T\mathbf{u}$ (i.e., both planes lie on plane $\mathbf{u}$). Since $\mathbf{Q}_i$ (3.4) are anti-symmetric, $\mathbf{u}^T\mathbf{Q}_i\mathbf{u} = 0$ and therefore

$$\begin{aligned} \mathbf{p}^T\mathbf{S}_1^*\mathbf{Q}_i\mathbf{S}_2^*\mathbf{p} &= \mathbf{p}^T(\mathbf{u}\mathbf{v}_1^T\mathbf{Q}_i\mathbf{u}\mathbf{v}_2^T - \mathbf{u}\mathbf{v}_1^T\mathbf{Q}_i\mathbf{v}_2\mathbf{u}^T + \mathbf{v}_1\mathbf{u}^T\mathbf{Q}_i\mathbf{v}_2\mathbf{u}^T)\mathbf{p} = \\ &= (\mathbf{p}^T\mathbf{u}) \cdot (\mathbf{v}_1^T\mathbf{Q}_i\mathbf{u}\mathbf{v}_2^T - \mathbf{v}_1^T\mathbf{Q}_i\mathbf{v}_2\mathbf{u}^T + \mathbf{u}^T\mathbf{Q}_i\mathbf{v}_2\mathbf{v}_1^T)\mathbf{p} \end{aligned} \tag{3.15}$$

and thus (3.5) becomes

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \propto \begin{pmatrix} \mathbf{v}_1^T\mathbf{Q}_1\mathbf{u}\mathbf{v}_2^T - \mathbf{v}_1^T\mathbf{Q}_1\mathbf{v}_2\mathbf{u}^T + \mathbf{u}^T\mathbf{Q}_1\mathbf{v}_2\mathbf{v}_1^T \\ \mathbf{v}_1^T\mathbf{Q}_2\mathbf{u}\mathbf{v}_2^T - \mathbf{v}_1^T\mathbf{Q}_2\mathbf{v}_2\mathbf{u}^T + \mathbf{u}^T\mathbf{Q}_2\mathbf{v}_2\mathbf{v}_1^T \\ \mathbf{v}_1^T\mathbf{Q}_3\mathbf{u}\mathbf{v}_2^T - \mathbf{v}_1^T\mathbf{Q}_3\mathbf{v}_2\mathbf{u}^T + \mathbf{u}^T\mathbf{Q}_3\mathbf{v}_2\mathbf{v}_1^T \end{pmatrix} \mathbf{p} \tag{3.16}$$

which is a projective transformation, and the pinhole is the intersection of the slits. Note that this is true only when $\mathbf{p}^T\mathbf{u} \neq 0$, i.e., when $\mathbf{p}$ does not lie on plane $\mathbf{u}$. If $\mathbf{p}$ does lie on $\mathbf{u}$, then its projection is $(0, 0, 0)^T$, which is not a point. Geometrically, the projection of $\mathbf{p}$ is the line where the image plane intersects plane $\mathbf{u}$.

The linear pushbroom projection is also a special cases of the (Orthogonal) X-Slits projection, obtained when the vertical slit resides on the plane at infinity. Indeed, setting $Z_1 = \infty$, (3.8) becomes

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X \\ -Z_2 \frac{Y}{Z-Z_2} \end{pmatrix} \tag{3.17}$$

It was shown in [23] that a 3D line is projected by the linear pushbroom projection model to a hyperbola in the image. This is a special case of the result shown in Section 3.1.3.

3.2 Multiple View Geometry

Next I study the relation between two X-Slits images, taken by two X-Slits cameras $\mathbf{T}$ and $\mathbf{T}'$.

3.2.1 Image Plane Transformation

Let us consider two X-Slits images which were obtained by the same slit configuration but with different projection planes Π and Π'. Note that the set of rays intersecting the image plane are determined by the two slits. Thus, there exists a mapping between the two image planes which is invariant to the 3D structure of the viewed scene, similarly to the case of rotation in pinhole cameras.

Specifically, I derive the relation between a point (x', y', w') on Π' and a point (x, y, w) on Π, both of which are the projection of the same scene point $\mathbf{p}$. Recall that the 3D point which corresponds to (x, y, w) on Π is $\mathbf{q} \propto x\mathbf{j} + y\mathbf{k} + w\mathbf{m}$. Similarly, (x', y', w') on Π' corresponds to $\mathbf{q}' \propto x'\mathbf{j}' + y'\mathbf{k}' + w'\mathbf{m}'$. By definition $\mathbf{q}$ lies on the planes $\mathbf{S}_1^*\mathbf{p}$ and $\mathbf{S}_2^*\mathbf{p}$, and therefore $\mathbf{q}$ is also projected to (x', y', w') on Π'. Denoting

$$\mathbf{M} = \begin{bmatrix} \mathbf{j} & \mathbf{k} & \mathbf{m} \end{bmatrix} \qquad\qquad p = (x, y, w)^T \tag{3.18}$$

it follows that $\mathbf{q} = \mathbf{M}p$. We now project $\mathbf{M}p$ on Π' to obtain

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \propto \begin{pmatrix} p^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{Q}'_1\mathbf{S}_2^*\mathbf{M}p \\ p^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{Q}'_2\mathbf{S}_2^*\mathbf{M}p \\ p^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{Q}'_3\mathbf{S}_2^*\mathbf{M}p \end{pmatrix} \tag{3.19}$$

Note that $\mathbf{M}$ depends only on plane Π, $\mathbf{S}_1^*$ and $\mathbf{S}_2^*$ depend only on slits $\mathbf{l}_1$ and $\mathbf{l}_2$ respectively, and $\mathbf{Q}'_1, \mathbf{Q}'_2, \mathbf{Q}'_3$ depend only on plane Π'.

Now, similarly to the notation used in (3.7), we can write this transformation as

$$p' \propto \mathbf{H}\nu(p) \tag{3.20}$$

where ν denotes the Veronese map $\nu : \mathcal{P}^2 \rightarrow \mathcal{P}^5$ and $\mathbf{H}$ is a 3×6 matrix.

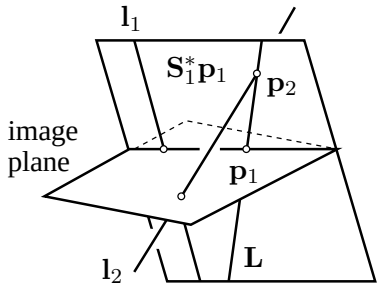


Figure 3.3: Illustration for the derivation of the fundamental matrix for two X-Slits cameras.

3.2.2 Fundamental Matrix

I now study the relation between two X-Slits images, taken by two arbitrary X-Slits cameras $\mathbf{T}$ and $\mathbf{T}'$. Given a point $p \in \mathcal{P}^2$ on the image of camera $\mathbf{T}$, the 3D point on the image plane corresponding to p is given by $\mathbf{p}_1 = \mathbf{M}p$, where $\mathbf{M}$ is the 4×3 matrix defined in (3.18). The plane that joins $\mathbf{p}_1$ to slit $\mathbf{l}_1$ is given by $\mathbf{S}_1^*\mathbf{p}_1$, see Fig. 3.3. This plane intersects $\mathbf{l}_2$ at the point $\mathbf{p}_2 = \mathbf{S}_2\mathbf{S}_1^*\mathbf{p}_1$ (note that $\mathbf{S}_2$ is the Plücker matrix representing $\mathbf{l}_2$ and $\mathbf{S}_1^*$ is the *dual* Plücker matrix representing $\mathbf{l}_1$). The point $\mathbf{p}_2$, which is not seen by $\mathbf{T}$ since it lies on $\mathbf{l}_2$, lies on the ray $\mathbf{L}$ that projects $\mathbf{p}_1$ to p. This ray is given by the Plücker matrix

$$\mathbf{L} = \mathbf{p}_1\mathbf{p}_2^T - \mathbf{p}_2\mathbf{p}_1^T = \mathbf{p}_1\mathbf{p}_1^T\mathbf{S}_1^*\mathbf{S}_2 - \mathbf{S}_2\mathbf{S}_1^*\mathbf{p}_1\mathbf{p}_1^T \tag{3.21}$$

As explained in Section 3.1.3, the projection of a line represented by the Plücker matrix $\mathbf{L}$ on $\mathbf{T}'$ is a conic given by the intersection of the image plane with the quadric $\mathbf{S}'^*_1\mathbf{L}\mathbf{S}'^*_2$. Therefore the projection of the ray through p on the image of camera $\mathbf{T}'$ is given by

$$\begin{aligned} 0 &= p'^T\mathbf{M}'^T\mathbf{S}'^*_1\mathbf{L}\mathbf{S}'^*_2\mathbf{M}'p' \\ &= p'^T\mathbf{M}'^T\mathbf{S}'^*_1(\mathbf{M}pp^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{S}_2 - \mathbf{S}_2\mathbf{S}_1^*\mathbf{M}pp^T\mathbf{M}^T)\mathbf{S}'^*_2\mathbf{M}'p' \end{aligned} \tag{3.22}$$

This equation defines a bi-quadratic relation between corresponding points p and p' in cameras $\mathbf{T}$ and $\mathbf{T}'$ respectively.

Using the Veronese map $\nu : \mathcal{P}^2 \rightarrow \mathcal{P}^5$, (3.22) can be rewritten as

$$\nu(p')^T\mathbf{F}\nu(p) = 0 \tag{3.23}$$

where $\mathbf{F}$ is a 6×6 matrix whose components depend on the values of the camera matrices $\mathbf{M}, \mathbf{S}_1^*, \mathbf{S}_2^*$ and $\mathbf{M}', \mathbf{S}_1'^*, \mathbf{S}_2'^*$. For each image point p in $\mathbf{T}$, $\mathbf{F}$ defines the conic on which the image point p' in $\mathbf{T}'$ must lie, and vice versa. I shall refer to such conics as *visibility curves*.

Definition 3.1 *The two conics on which two corresponding image points p, p' must lie, as determined by Eq. (3.23), are called* **visibility curves**.

As will be shown below, these curves play a role similar to the epipolar lines in the perspective model.

In analogy with the pinhole camera, I define the *fundamental matrix* of a pair of X-Slits cameras:

Definition 3.2 *Matrix F in Eq. (3.23) is called the* **fundamental matrix** *of a pair of X-Slits cameras $\mathbf{T}$ and $\mathbf{T}'$.*

Clearly, $\mathbf{F}$ always exists. It is similar to the conventional fundamental matrix in the sense that it captures the relative position of two X-Slits cameras, and that it makes it possible to get the visibility curves in one image from points in the other image.

Since $\mathbf{F}$ depends on cameras $\mathbf{T}$ and $\mathbf{T}'$, it is determined by 34 free parameters at most. The real number of free parameters is, however, much smaller. To see this, suppose $\mathbf{A}$ is a 4×4 matrix representing a projective transformation $\mathbf{p} \mapsto \mathbf{A}\mathbf{p}$. It can easily be verified that this transformation, when applied to a Plücker and a dual Plücker matrix, is given by $\mathbf{S} \mapsto \mathbf{A}\mathbf{S}\mathbf{A}^T$ and $\mathbf{S}^* \mapsto \mathbf{A}^{-T}\mathbf{S}^*\mathbf{A}^{-1}$, respectively. By substituting these mappings into (3.22) one obtains the same equation, and therefore (3.22) is invariant to a projective transformation of all its elements. Note, however, that due to the construction of matrix $\mathbf{M}$ with 2 points on the plane at infinity, we are only free to choose a 3D affine transformation to change the coordinate system; this removes 12 degrees of freedom, leading us to the conclusion that $\mathbf{F}$ is fully determined by 22 free parameters at most.

We can derive $\mathbf{F}$ directly from the camera parameters as follows. Let p and p' denote the corresponding projections of 3D point $\mathbf{p}$ in the two images. As discussed above, the projection ray of $\mathbf{p}$ in camera $\mathbf{T}$ is defined by the intersection of the two planes $\mathbf{S}_1^*\mathbf{M}p$ and $\mathbf{S}_2^*\mathbf{M}p$. It follows that $\mathbf{p}$ must lie on the two planes, namely, $\mathbf{p}^T\mathbf{S}_1^*\mathbf{M}p = 0$ and $\mathbf{p}^T\mathbf{S}_2^*\mathbf{M}p = 0$. A similar argument regarding camera $\mathbf{T}'$ allows us to conclude that $\mathbf{p}^T\mathbf{S}_1'^*\mathbf{M}'p' = 0$ and $\mathbf{p}^T\mathbf{S}_2'^*\mathbf{M}'p' = 0$.

Let us define the 4×4 matrix $\mathbf{B}$ whose columns are the vector representations of the 4 planes, namely

$$\mathbf{B} = \left[\begin{array}{cccc} \mathbf{S}_1^* \mathbf{M} p & \mathbf{S}_2^* \mathbf{M} p & \mathbf{S'}_1^* \mathbf{M'} p' & \mathbf{S'}_2^* \mathbf{M'} p' \end{array} \right] \tag{3.24}$$

Clearly $\mathbf{p}^T \cdot \mathbf{B} = 0$. This implies that the null space of $\mathbf{B}$ is not empty, and thus the determinant of $\mathbf{B}$ must be 0. The equation $\det(\mathbf{B}) = 0$ gives us another expression for the bi-quadratic relation between the image points described by the fundamental matrix $\mathbf{F}$. With some algebraic manipulations of $\det(\mathbf{B}) = 0$, we can arrive at the following form

$$0 = \det(\mathbf{B}) = \nu(p)^T \cdot \mathbf{H} \cdot \mathbf{G} \cdot \nu(p') \tag{3.25}$$

where $\mathbf{H}$ and $\mathbf{G}$ are two 6×6 matrices which depend each *only* on the cameras $\mathbf{T}$ and $\mathbf{T}'$ respectively.

Let us take a closer look at $\nu(p)^T \cdot \mathbf{H}$. For any matrix $\mathbf{X}^*$, I use the notation $\mathbf{X}^k$ to denote the k-th column of $\mathbf{X}^*$. By construction we have

$$\begin{aligned} \nu(p)^T \mathbf{H}^1 &= p^T \mathbf{M}^T [(\mathbf{S}_1^1)(\mathbf{S}_2^2)^T - (\mathbf{S}_2^1)(\mathbf{S}_1^2)^T] \mathbf{M} p \\ \nu(p)^T \mathbf{H}^2 &= p^T \mathbf{M}^T [(\mathbf{S}_1^1)(\mathbf{S}_2^3)^T - (\mathbf{S}_2^1)(\mathbf{S}_1^3)^T] \mathbf{M} p \\ \nu(p)^T \mathbf{H}^3 &= p^T \mathbf{M}^T [(\mathbf{S}_1^1)(\mathbf{S}_2^4)^T - (\mathbf{S}_2^1)(\mathbf{S}_1^4)^T] \mathbf{M} p \\ \nu(p)^T \mathbf{H}^4 &= p^T \mathbf{M}^T [(\mathbf{S}_1^2)(\mathbf{S}_2^3)^T - (\mathbf{S}_2^2)(\mathbf{S}_1^3)^T] \mathbf{M} p \\ \nu(p)^T \mathbf{H}^5 &= p^T \mathbf{M}^T [(\mathbf{S}_1^2)(\mathbf{S}_2^4)^T - (\mathbf{S}_2^2)(\mathbf{S}_1^4)^T] \mathbf{M} p \\ \nu(p)^T \mathbf{H}^6 &= p^T \mathbf{M}^T [(\mathbf{S}_1^3)(\mathbf{S}_2^4)^T - (\mathbf{S}_2^3)(\mathbf{S}_1^4)^T] \mathbf{M} p \end{aligned} \tag{3.26}$$

This defined matrix $\mathbf{H}$; moreover, it can be shown that the rank of $\mathbf{H}$ is at most 4 given that the 4×4 matrices $\mathbf{S}_1^*$ and $\mathbf{S}_2^*$ are anti-symmetric and of rank 2. $\mathbf{G}$ is defined in a similar way for camera $\mathbf{T}'$, and its rank is therefore also 4. Since the fundamental matrix $\mathbf{F} = \mathbf{H} \cdot \mathbf{G}$, we can conclude the following:

Proposition 3.1 *The rank of the fundamental matrix of the X-Slits projection is 4 at most.*

This proposition immediately gives us 4 independent constraints on the elements of $\mathbf{F}$. For example, we can choose four different 5×5 sub-matrices of $\mathbf{F}$, and require that the determinant of each equals 0.

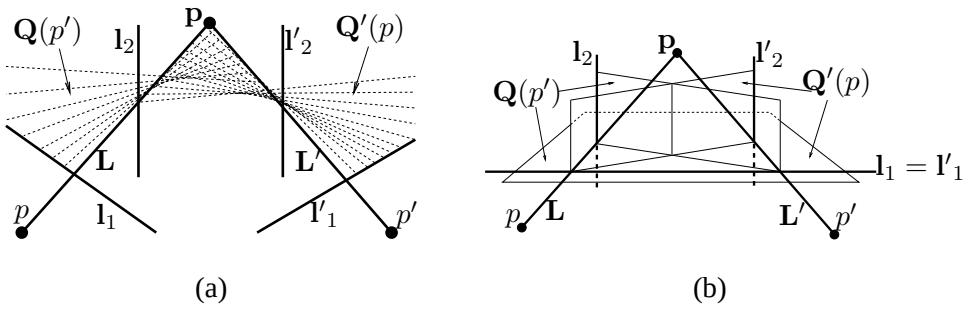


Figure 3.4: Visibility quadrics: (a) $\mathbf{Q}'(p)$ is the collection of projection rays of $\mathbf{T}'$ passing through line $\mathbf{L}$, while $\mathbf{Q}(p')$ is the collection of projection rays of $\mathbf{T}$ passing through $\mathbf{L}'$. (b) When the cameras have a shared slit ($\mathbf{l}_1 = \mathbf{l'}_1$), the visibility quadrics intersect in a plane.

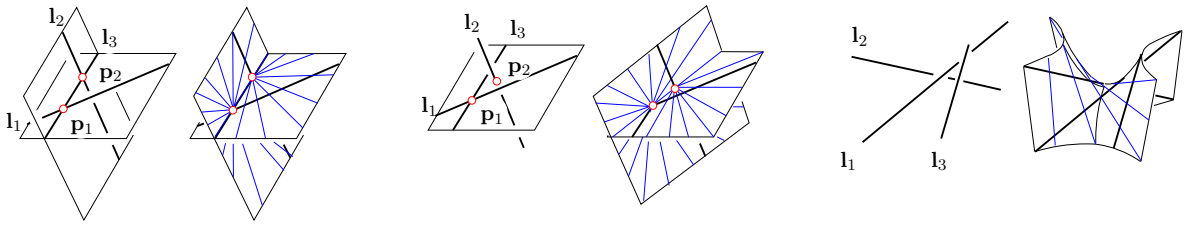
3.2.3 Visibility Quadrics

Let $\mathbf{L}$ denote the projection ray of camera $\mathbf{T}$ passing through image point p. Let $\mathbf{Q}'(p)$ denote the quadric $\mathbf{S'}_1^*\mathbf{L}\mathbf{S'}_2^*$ where $\mathbf{L}$ is defined in (3.21); thus $\mathbf{Q}'(p)$ is the projection of $\mathbf{L}$ in camera $\mathbf{T}$'. This quadric is a double-ruled surface that is ruled by the family of all rays of camera $\mathbf{T}'$ passing through the line $\mathbf{L}$ (see Fig. 3.4a). Similarly, let $\mathbf{L}'$ denote the projection ray of camera $\mathbf{T}'$ passing through image point p', and let the quadric $\mathbf{Q}(p') = \mathbf{S}_1^*\mathbf{L}'\mathbf{S}_2^*$ denote the projection of $\mathbf{L}'$ in camera $\mathbf{T}$.

Definition 3.3 *The quadric $\mathbf{Q}'(p)$ (resp. $\mathbf{Q}(p')$) for any image points p (resp. p') is called a* visibility quadric.

Visibility quadrics play a role similar to epipolar planes in the pinhole camera. However, unlike the perspective model, these quadrics are not necessarily symmetric with respect to the two cameras. For a given scene point $\mathbf{p}$ that is projected to p and p' in $\mathbf{T}$ and $\mathbf{T}'$ respectively, the corresponding surfaces of rays of $\mathbf{T}$ and $\mathbf{T}'$ are $\mathbf{Q}'(p)$ and $\mathbf{Q}(p')$ respectively. These quadrics do not usually coincide.

When $\mathbf{Q}'(p)$ and $\mathbf{Q}(p')$ do coincide, I refer to this quadric as an *epipolar quadric*. The property of an epipolar quadric is that all points on it are projected to a single conic in each camera, and the corresponding conics can be used for matching in the same way as epipolar lines are used in perspective images. I shall describe next the camera configurations in which this occurs for all scene points; in these cases the *visibility curves* in both cameras can be matched with each other, similarly to epipolar lines in the perspective camera. This notion will be made more precise next.



(a) (b) (c)

Figure 3.5: Illustrations for the proof of Lemma 3.2.

Epipolar Quadrics

In this section I assume that *slits are not visible by their camera*, because otherwise epipolar quadrics are not well defined for points on the slits To make the discussion precise, let us start with a few **definitions**: We say that two *lines intersect* if they have a common point. We say that lines are *disjoint* if they do not intersect. Let L be a set of lines. Denote by $\mathcal{T}(L)$ the set of lines that intersect every line in L and call $\mathcal{T}(L)$ *the transversal* of lines in L. Adopt that a set R of lines is called *regulus* if there are three pairwise disjoint lines $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ such that $R = \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3\})$.

Let $\mathbf{T}$ denote a X-Slits camera with slits $\mathbf{l}_1, \mathbf{l}_2$. We say that line $\mathbf{l}$ is a *projector* of a point $\mathbf{p}$ in camera $\mathbf{T}$ if $\mathbf{l}$ is transversal to slits $\mathbf{l}_1, \mathbf{l}_2$, and $\mathbf{p}$ is in $\mathbf{l} \setminus (\mathbf{l}_1 \cup \mathbf{l}_2)$. A nonempty quadric Q is said to be an *epipolar quadric* of two X-Slits cameras $\mathbf{T}, \mathbf{T}'$ if for every point $\mathbf{q} \in Q$ all projectors of $\mathbf{q}$ in $\mathbf{T}$ and $\mathbf{T}'$ are contained in Q.

Lemma 3.2 *Let $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ be three distinct lines, out of which at least two are disjoint. Then transversal $\mathcal{R} = \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3\})$ is either the union of two planar pencils of lines that have one line in common, or a regulus.*

Proof: Let w.l.o.g. $\mathbf{l}_1, \mathbf{l}_2$ be disjoint. Then one of the following 3 possibilities holds: (1) $\mathbf{l}_3$ intersects both $\mathbf{l}_1, \mathbf{l}_2$ (see Fig. 3.5a); (2) $\mathbf{l}_3$ intersects either $\mathbf{l}_1$ or $\mathbf{l}_2$, be it $\mathbf{l}_1$ w.l.o.g. (see Fig. 3.5b); (3) $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ are pairwise disjoint (see Fig. 3.5c).

case 1: Denote $\mathbf{l}_1 \cap \mathbf{l}_3$ by $\mathbf{p}_1$ and $\mathbf{l}_2 \cap \mathbf{l}_3$ by $\mathbf{p}_2$. The set of lines transversal to $\mathbf{l}_1, \mathbf{l}_3$ is the union of the set L_1 of all lines passing through $\mathbf{p}_1$ and the set L_2 of all lines in the plane spanned by $\mathbf{l}_1, \mathbf{l}_3$.

The lines in L_1 that intersect $\mathbf{l}_2$ form the pencil of lines with center $\mathbf{p}_1$ in the plane spanned by $\mathbf{l}_2, \mathbf{l}_3$. The lines in L_2 that intersect $\mathbf{l}_2$ form the pencil of lines with center $\mathbf{p}_2$ in the plane spanned by $\mathbf{l}_1, \mathbf{l}_3$. Clearly line $\mathbf{l}_3$ lies in both pencils.

case 2: Denote $\mathbf{l}_1 \cap \mathbf{l}_3$ by $\mathbf{p}_1$. Line $\mathbf{l}_2$ intersects the plane spanned by $\mathbf{l}_1, \mathbf{l}_3$ in a point $\mathbf{p}_2$. Now, the set of lines transversal to $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$ is the union of two planar pencils of lines that have one line in common by the same argument as in the previous case.

case 3: In this case $\mathcal{R}$ is a regulus, see [56, p. 42] for the proof. ■

Theorem 3.3 *Let* $\mathbf{T}$ *(resp.* $\mathbf{T}'$*) be a X-Slits camera with disjoint slits* $\mathbf{l}_1, \mathbf{l}_2$ *(resp. disjoint slits* $\mathbf{l}'_1, \mathbf{l}'_2$*). Then, every point in the set* $\mathcal{V}$ *of all points that have a projector in both cameras is contained in an epipolar quadric* $\iff$ *the cameras either share a slit, or slits* $\mathbf{l}_1, \mathbf{l}_2$ *intersect with slits* $\mathbf{l}'_1, \mathbf{l}'_2$ *in four pairwise distinct points.*

Proof: ($\Longrightarrow$) One of the following must be true: (1) all projectors of $\mathbf{T}$ intersect at least one of the slits $\mathbf{l}'_1$ or $\mathbf{l}'_2$; (2) there is a projector $\mathbf{l}$ of $\mathbf{T}$ so that $\mathbf{l}, \mathbf{l}'_1, \mathbf{l}'_2$ are pairwise disjoint.

case 1: The set of projectors of $\mathbf{T}$ is the union of two sets of lines

$$\begin{aligned} A &= \{\mathbf{m} \text{ is a projector of } \mathbf{T} | \mathbf{m} \cap \mathbf{l}'_1 \neq \emptyset\} = \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_1\}) \\ B &= \{\mathbf{m} \text{ is a projector of } \mathbf{T} | \mathbf{m} \cap \mathbf{l}'_2 \neq \emptyset\} = \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_2\}) \end{aligned} \tag{3.27}$$

It follows from Lemma 3.2 that transversal $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}\})$ for a line $\mathbf{l} \neq \mathbf{l}_1, \mathbf{l}_2$ is either a planar pencil of lines or a regulus. Therefore, from

$$\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2\}) = A \cup B = \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_1\}) \cup \mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_2\}) \tag{3.28}$$

it follows that $\mathbf{l}'_1 \in \{\mathbf{l}_1, \mathbf{l}_2\}$ or $\mathbf{l}'_2 \in \{\mathbf{l}_1, \mathbf{l}_2\}$. This is because $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2\})$ is not a surface (but rather a volume), since for every point $\mathbf{p}$ there is a line in $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2\})$ passing through $\mathbf{p}$.

case 2: Take $\mathbf{p} \in \mathbf{l} \cap \mathcal{V}$. There is an epipolar quadric Q containing $\mathbf{p}$. Thus $\mathbf{l}$ is in Q and also the regulus $\mathcal{R} = \mathcal{T}(\{\mathbf{l}'_1, \mathbf{l}'_2, \mathbf{l}\})$ is in Q by which Q is a regular double ruled quadric. Either (2.1) there is a line $\mathbf{s} \in \mathcal{R}$ which does *not* intersect both $\mathbf{l}_1, \mathbf{l}_2$, or (2.2) $\mathbf{l}_1, \mathbf{l}_2$ intersect all lines in $\mathcal{R}$.

- case 2.1: Regulus $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{s}\})$ is in Q and therefore lines $\mathbf{l}_1, \mathbf{l}_2$ are in Q. Lines $\mathbf{l}_1, \mathbf{l}_2$ are disjoint and consequently are in the same regulus. The same holds for $\mathbf{l}'_1, \mathbf{l}'_2$. Line s intersects $\mathbf{l}'_1, \mathbf{l}'_2$ but does not intersect $\mathbf{l}_1, \mathbf{l}_2$, and thus $\mathbf{l}_1, \mathbf{l}_2$ are in the opposite regulus to the regulus containing $\mathbf{l}'_1, \mathbf{l}'_2$ (in other words, they are in different rulings on the surface). Consequently, slits $\mathbf{l}_1, \mathbf{l}_2$ intersect with slits $\mathbf{l}'_1, \mathbf{l}'_2$ in four pairwise distinct points.

- case 2.2: Lines $\mathbf{l}_1, \mathbf{l}_2$ are in regulus $\mathcal{T}(\mathcal{R})$. Since $\mathbf{l}'_1, \mathbf{l}'_2 \in \mathcal{T}(\mathcal{R})$, all four $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_1, \mathbf{l}'_2$ are in the same regulus and are pairwise distinct because $\mathbf{l}$ intersects $\mathbf{l}_1, \mathbf{l}_2$ but does not intersect $\mathbf{l}'_1, \mathbf{l}'_2$. Then, however, no point $\mathbf{q} \in \mathcal{V} \setminus Q$ is contained in an epipolar quadric due to the following argument. Denote by $\mathbf{n}$ (resp. $\mathbf{n}'$) the line from $\mathbf{T}$ (resp. $\mathbf{T}'$) that passes through a point $\mathbf{q} \in \mathcal{V}$. Assume that there is an epipolar quadric Q' containing $\mathbf{q}$. Then both $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{n}'\})$ and $\mathcal{T}(\{\mathbf{l}'_1, \mathbf{l}'_2, \mathbf{n}\})$ are in Q', and thus all $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'_1, \mathbf{l}'_2$ are in Q'. However, now $Q = Q'$ since every four distinct lines from a regulus are exactly in one regulus. Therefore, $\mathbf{q} \in Q$.

($\Longleftarrow$) By the assumption one of the following holds: (1) the cameras share exactly one slit; (2) the cameras share both slits; (3) the cameras intersect in four distinct points.

case 1: Let w.l.o.g. $\mathbf{l}_1 = \mathbf{l}'_1$. A point $\mathbf{q} \in \mathcal{V}$ is not contained in $\mathbf{l}_1$ and therefore there is exactly one plane π through $\mathbf{l}_1$ and $\mathbf{q}$. Every projector from $\mathbf{T}$ or $\mathbf{T}'$, which contains $\mathbf{q}$, intersects $\mathbf{l}_1$ and is therefore in π; thus π is an epipolar quadric.

case 2: Let w.l.o.g. $\mathbf{l}_1 = \mathbf{l}'_1$ and $\mathbf{l}_2 = \mathbf{l}'_2$. Then every point $\mathbf{q} \in \mathcal{V}$ is projected in both $\mathbf{T}$ and $\mathbf{T}'$ by the same projector. Every $\mathbf{q} \in \mathcal{V}$ and its projector $\mathbf{l}$ are contained in, e.g., the regular epipolar quadric that contains regulus $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'\})$ for some line $\mathbf{l}'$ that contains $\mathbf{q}$ and does not intersect $\mathbf{l}_1$, $\mathbf{l}_2$.

case 3: Every point $\mathbf{q} \in \mathcal{V}$ is contained in exactly one projector $\mathbf{l}$ from $\mathbf{T}$ and in exactly one projector $\mathbf{l}'$ from $\mathbf{T}'$. We assumed that $\mathbf{l}_1, \mathbf{l}_2$ are transversal to $\mathbf{l}'_1, \mathbf{l}'_2$. Line $\mathbf{l}$ (resp. $\mathbf{l}'$) is transversal to $\mathbf{l}_1, \mathbf{l}_2$ (resp. $\mathbf{l}'_1, \mathbf{l}'_2$). Line $\mathbf{l}$ is transversal to $\mathbf{l}'$ since both contain $\mathbf{q}$. Lines $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'$ (resp. $\mathbf{l}'_1, \mathbf{l}'_2, \mathbf{l}$) are pairwise disjoint. Therefore, $\mathbf{q}$ is contained in a regular epipolar quadric that contains regulus $\mathcal{T}(\{\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}'\})$. ■

We can now conclude the following:

Corollary 3.4 *If two X-Slits cameras share a slit, then every point is contained in an epipolar plane, see Fig. 3.6b. Moreover, the epipolar planes form a pencil of planes.*

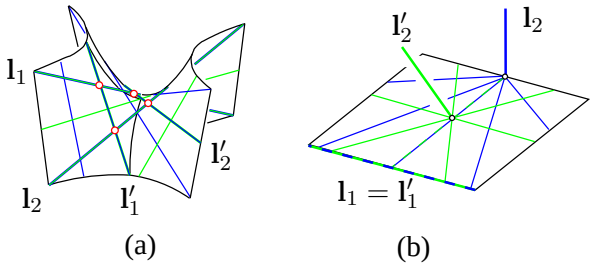


Figure 3.6: Epipolar quadrics of a pair of X-Slits cameras with nonintersecting slits. (a) The slits intersect in four pairwise disjoint points. (b) The cameras share a slit.

Vice versa, when the epipolar quadrics are planes in general, the cameras must have one common slit.

Corollary 3.5 *If the slits intersect in four pairwise distinct points, then every point is contained in a regular epipolar quadric, see Fig. 3.6a. Moreover, the epipolar quadrics form a pencil of quadrics.*

3.2.4 Visibility Curves

One property of visibility curves is that they must all intersect two specific points, which are the points where the slits intersect the image plane.

Proposition 3.6 *In camera* $\mathbf{T}$*, denote the image points where slits* $\mathbf{l}_1$ *and* $\mathbf{l}_2$ *intersect with the image plane as* c_1 *and* c_2 *respectively; then all scene lines are projected into conics that pass through* c_1 *and* c_2*.*

Proof: The line $\mathbf{L}$ is projected into the conic given by

$$p^T \mathbf{M}^T \mathbf{S}_1^* \mathbf{L} \mathbf{S}_2^* \mathbf{M} p = 0 \tag{3.29}$$

For $i = 1, 2$, the 3D point corresponding to c_i is $\mathbf{M}c_i$, and since this point lies on slit $\mathbf{l}_i$, it follows that $\mathbf{S}_i^* \mathbf{M} c_i = 0$. Therefore, (3.29) holds for $p = c_i$, which means c_1, c_2 both lie on the conic which is the projection of the line $\mathbf{L}$. ■

Corollary 3.7 c_1 *and* c_2 *lie on all visibility curves.*

The projection of a ray is the intersection of the image plane with a subset of a quadric double-ruled surface containing the slits of the camera and the ray of the other camera through the scene point, as discussed above and in Section 3.2.3. When this set is a plane, the visibility curve degenerates into a line. This gives us the following result:

Proposition 3.8 *When two X-Slits cameras share a slit, visibility curves are lines and can be matched, i.e., points on a visibility line of one camera can be matched to points on the corresponding visibility line of the other camera.*

This proposition shows that in the case of a shared slit, there is great similarity to the epipolar geometry of the perspective projection. The following lemma characterizes this similarity:

Lemma 3.9 *For two cameras $\mathbf{T}$, $\mathbf{T}'$, if the cameras have a common slit, then each visibility curve is composed of a pair of lines, one of which is the projection of a singular point; excluding singular points, the remaining family of lines is the family of lines induced by the perspective fundamental matrix.*

Proof: Since a slit is shared, let us assume w.l.o.g. that $\mathbf{S}_1^* = \mathbf{S}'^*_1$. It can easily be shown that $\mathbf{S}_1^*\mathbf{S}_2\mathbf{S}_1^* = \mu\mathbf{S}_1^*$, for some $\mu \in \mathcal{R}$. From (3.22) it follows that for each scene point projected to p, p' in $\mathbf{T}$, $\mathbf{T}'$ respectively,

$$\begin{aligned} 0 &= p'^T\mathbf{M}'^T\mathbf{S}'^*_1(\mathbf{M}pp^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{S}_2 - \mathbf{S}_2\mathbf{S}_1^*\mathbf{M}pp^T\mathbf{M}^T)\mathbf{S}'^*_2\mathbf{M}'p' \\ &= p'^T\mathbf{M}'^T\mathbf{S}_1^*(\mathbf{M}pp^T\mathbf{M}^T\mathbf{S}_1^*\mathbf{S}_2 - \mu\mathbf{M}pp^T\mathbf{M}^T)\mathbf{S}'^*_2\mathbf{M}'p' \\ &= p'^T\mathbf{M}'^T\mathbf{S}'^*_1\mathbf{M}pp^T\mathbf{M}^T(\mathbf{S}_1^*\mathbf{S}_2 - \mu I)\mathbf{S}'^*_2\mathbf{M}'p' \\ &\equiv p'^T\mathbf{A}p \cdot p^T\mathbf{B}p' \end{aligned} \tag{3.30}$$

Since $\mathbf{M}p$ and $\mathbf{M}'p'$ are 3D points on the image planes of $\mathbf{T}$ and $\mathbf{T}'$ corresponding to p and p' respectively, $p'^T\mathbf{A}p = 0$ if and only if p and p' are coplanar with the common slit.

Imagine that instead of the two X-Slits cameras $\mathbf{T}$ and $\mathbf{T}'$ we have two perspective cameras with focal centers on $\mathbf{l}_1$; clearly the same relationship would exist between corresponding image points.

This means that the first constraint, denoted as $p'^T\mathbf{A}p$ is equivalent to the constraint on matching points between two perspective cameras that lie on $\mathbf{l}_1$.

On the other hand, $(\mathbf{S}_1^*\mathbf{S}_2 - \mu I)\mathbf{S}_1^* = (\mu\mathbf{S}_1^* - \mu\mathbf{S}_1^*) = 0$. Therefore $p^T\mathbf{M}^T(\mathbf{S}_1^*\mathbf{S}_2 - \alpha I)\mathbf{S}_1^* = 0$, which means that $p^T\mathbf{M}^T(\mathbf{S}_1^*\mathbf{S}_2 - \alpha I)$ is a point on slit $\mathbf{l}_1$, and therefore the visibility line $\mathbf{B}^T p$ is a projection of a point on a slit. This projection is singular (a point is projected to a line), and therefore we exclude it from the set of points feasible for matching. ■

3.3 Summary

I presented a new non-perspective projection model, which is defined by two slits and a projection surface. This model can be physically realized, and has been built in the late 19th century. Algebraically, I showed that this model corresponds to a second-order transformation from three-dimensional space to two-dimensional space (while perspective projection is a linear, or first order, transformation).

The second-order nature of the X-Slits projection is further observed in its epipolar geometry. I studied the multiple view geometry and described the relation between two X-Slits images with the same slit configuration and different image planes, which is the equivalent of homographies in the perspective, as well as the relation between two arbitrary X-Slits cameras, which is the equivalent of the fundamental matrix. In both instances, corresponding image points have the same relation as in the perspective model, except that here the constraints are quadratic rather than linear.

Consequently, epipolar planes and epipolar lines become curved in the X-Slits projection. I defined visibility surfaces, which were shown to be double-ruled quadrics, and visibility curves, which were shown to be conics. Interestingly, unlike the perspective model, here visibility surfaces and curves need not correspond, except for certain special cases. In one of these cases, when the cameras share a slit, the epipolar geometry exhibits some similarity with the perspective epipolar geometry. This particular configuration turns out to be useful for image-based rendering applications, as presented next.

Chapter 4

Non-Perspective View Synthesis

In this chapter I describe how to synthesize X-Slits images from a sequence of perspective images acquired using a sideways-moving perspective camera. As shown in Section 4.1, a simple column-sampling method that does not require any knowledge about the scene can be used to generate X-Slits views with one slit coinciding with the camera path and the other at arbitrary locations. This method is also extended to non-linear slits.

In Section 4.2, this technique is used for creating an omni-directional virtual environment, where the point-of-view is moved by moving one of the slits. Since the X-Slits projection is not perspective, it appears distorted, and I present a method for reducing these distortions by perspective reprojection using a coarse approximation of the scene structure.

In Section 4.3 I consider the problem of optimal mosaicing based only on the available set of rays, without any information about the scene by approximating a perspective camera based on the set of rays alone.

4.1 Crossed-Slits Image Generation

I now consider the issue of synthesizing new X-Slits views from “regular” perspective images. The input sequence is assumed to be captured by a pinhole camera translating along a horizontal line in 3D space in roughly constant speed, and without changing its orientation or internal calibration. As I show

below, in the simplest case we can generate a new X-Slits image where the two slits of the underlying virtual X-Slits camera are defined as follows:

1. A horizontal slit that lies on the path of the optical center of the moving pinhole camera.

2. A vertical slit that is parallel to the image's vertical axis, and whose location is determined by the parameters of the mosaicing process.

In practice, new view synthesis is performed by non-stationary mosaicing. Basic non-stationary mosaicing is defined as follows:

- From each frame t, sample the vertical column (strip) centered on the horizontal coordinate $s(t)$.

- Paste the strips into a mosaic image, as in [55].

In the general case we may sample slanted strips rather than vertical columns (strips), and the orientation may also change as a function of t. In this case the "vertical" slit of the underlying virtual camera may not be parallel to the image's vertical axis. However, for clarity of presentation and without loss of generality, I will continue calling one slit of the new virtual camera "horizontal" and the other slit "vertical". Typically the "horizontal" slit is aligned with the path of the camera, while the second "vertical" slit is not constrained a priori and need not be orthogonal to the first slit.

The parameters of the strip sampling function $s(t)$ determine the location of the vertical slit of the virtual camera. A virtual walkthrough is obtained by generating a sequence of X-Slits images via non-stationary mosaicing, while moving the vertical slit along a planar path. Adjusting the image plane orientation is done by warping the mosaiced image, as described in Section 3.2.1.

In Section 4.1.1 I show how to sample vertical strips from the input images in the sequence in order to generate a valid X-Slits image. I also discuss the relation between the sampling function $s(t)$ and the parameters of the virtual X-Slits camera. In Section 4.1.2 I discuss implementation issues, including the treatment of deviation from constant speed and aspect ratio normalization.

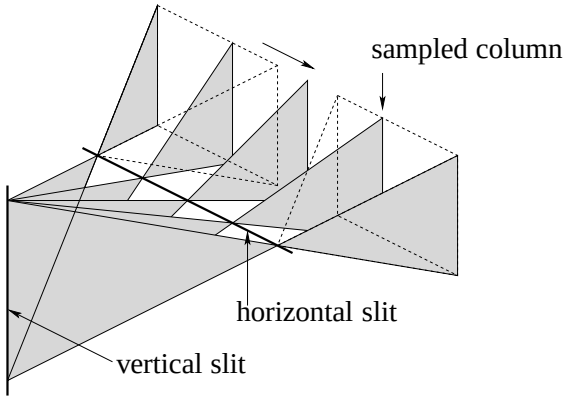


Figure 4.1: The non-stationary column sampling routine which is used to synthesize new images.

4.1.1 Non-Stationary Strip Sampling

I start my analysis with the simplest case where the input image sequence is generated by a camera moving sideways in a direction parallel to the X-axis of the image. The camera is also assumed to be internally calibrated. In this simple case the new synthesized image is an Orthogonal X-Slits image (see Section 3.1.2), and the non-stationary strip sampling is a linear function. I show below the exact relation between the parameters of the linear sampling function and the parameters of the corresponding virtual X-Slits camera. In Appendix A it is shown that even when the camera is not internally calibrated, any linear strip sampling function results in a X-Slits image (but not necessarily Orthogonal X-Slits). When the motion of the camera is not parallel to the image plane, the sampling function is not linear anymore.

When the basic assumptions of the analysis are violated, namely, the camera changes its orientation and internal calibration arbitrarily along the input sequence, we need to preprocess the sequence. One solution involves registering all the images with each other using the homography of the plane at infinity. This computation requires, however, either (partial) internal camera calibration or some domain knowledge (such as parallel lines in the scene) [26].

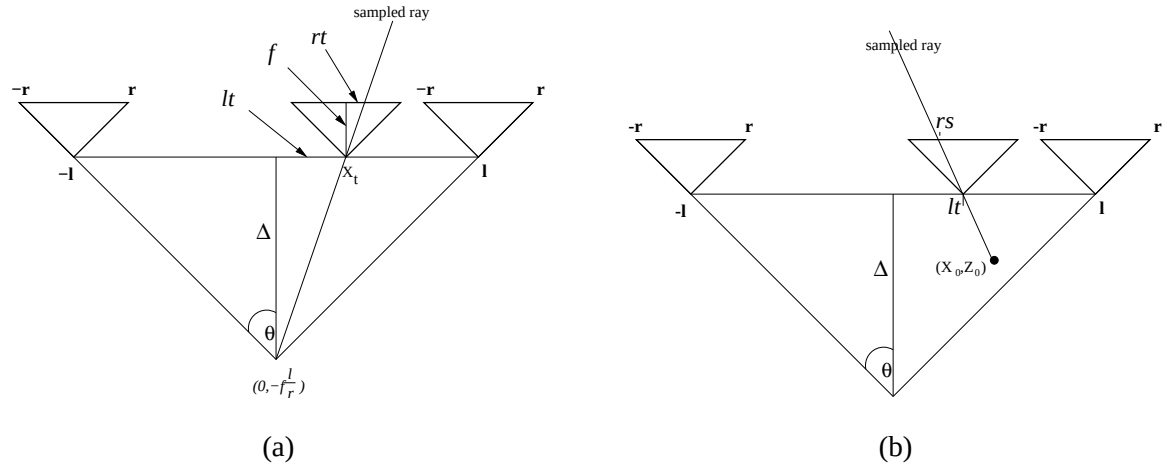


Figure 4.2: New image formation with two possible positions of the vertical slit (see text).

Mosaicing by Linear Strip Sampling

Let our input be a sequence of images captured by a pinhole camera translating in constant speed along the X axis from left to right. We generate a new panoramic image by pasting columns from the input images, as illustrated in Fig. 4.1. We start by sampling the left column of the first (leftmost) image, and conclude by sampling the right column of the last (rightmost) image. In between, intermediate columns are sampled from successive images using a linear sampling function.

A schematic illustration of this setup is given in Fig. 4.2a, in a top-down view. A sequence of positions of the real pinhole camera is shown, together with the corresponding field of view. The moving input camera, whose optical centers are located at positions $\mathbf{c}(t) = (X_t, 0, 0)$, generates images according to the following mapping:

$$\mathbf{p} = (X, Y, Z) \implies p = (x, y) = (f\frac{X - X_t}{Z}, f\frac{Y}{Z}) \tag{4.1}$$

Denote the range of columns (x) in each pinhole image as $[-r, r]$, and the range of camera pinhole positions (X_t) as $[-l, l]$ (see Fig. 4.2a). The new synthesized image is constructed by pasting columns from the input images. The range of columns in the synthesized image is $[-(r + l), r + l]$. For each $t \in [-1, 1]$, we assign to the $(l + r)t$ column of the new image the image values at the rt column of

the pinhole camera positioned at $(lt, 0, 0)$ (i.e., $X_t = lt$, see Fig. 4.2a). It now follows from Eq. (4.1) that $rt = f\frac{X-lt}{Z}$. In addition, for each column $x \in [-(r+l), (r+l)]$ in the new image, $t = \frac{x}{l+r}$ and therefore

$$X = \frac{rt}{f}Z + lt = x\left(\frac{r}{l+r} \cdot \frac{Z}{f} + \frac{l}{l+r}\right)$$

or

$$x = \frac{l+r}{r}f \cdot \frac{X}{Z + f\frac{l}{r}}$$

Observe that this defines a vertical slit at $Z = -f\frac{l}{r}$ (see Fig. 4.2a). The horizontal slit is at $Z = 0$ (all pinhole camera centers are at $Z = 0$). Eq. (4.1) can therefore be rewritten as

$$\mathbf{p} = (X, Y, Z) \implies p = (x, y) = (f_x \frac{X}{Z + \Delta}, f_y \frac{Y}{Z}) \tag{4.2}$$

where $f_x = \frac{l+r}{r}f$ is the horizontal focal length, $f_y = f$ is the vertical focal length, and $\Delta = f\frac{l}{r}$ is the distance between the two slits.

Suppose next that instead of taking the rt column from the camera at $(lt, 0, 0)$, we choose an arbitrary linear column sampling function. More specifically, for $t = \alpha s + \beta$, we take the rs column of the lt camera, see Fig. 4.2b. (Recall that r, l are fixed, while t, s are free parameters). Let the field of view of the original pinhole camera be 2θ. It can be shown as above that such a choice of columns defines the mapping

$$(x, y) = ((f + \frac{\alpha l}{\tan\theta})\frac{X - \beta l}{Z + \frac{\alpha l}{\tan\theta}}, f\frac{Y}{Z}) \tag{4.3}$$

This can be written simply as

$$(x, y) = (f_x \frac{X - X_0}{Z + \Delta}, f_y \frac{Y}{Z}) \tag{4.4}$$

where $X_0 = \beta l$, $\Delta = \frac{\alpha l}{\tan\theta}$, $f_y = f$ and $f_x = f + \Delta$.

The method described so far produces images which do not follow the perspective projection model. They do, however, follow the X-Slits projection model. To see this, observe that all the rays which participate in the generation of each new image, must intersect the following two lines:

1. The line of camera motion; this is because each projection ray must be collected by some camera whose optical center is on this line.

2. The vertical line located at (X_0, Z_0) (as in Eq. (4.4), where $Z_0 = Z + \Delta$).

The projection model is therefore defined by a family of rays intersecting a pair of lines (*"slits"*), projecting 3D points onto a plane. Moreover, the model is Orthogonal X-Slits (compare Eq. (4.4) with Eq. (3.8)).

In the derivation leading to (4.4) I effectively showed that any linear sampling function yields a valid new Orthogonal X-Slits image. Furthermore, we can set the location of the vertical slit to (X_0, Z_0) by fixing $\alpha = -\frac{Z_0}{l}\tan\theta$ and $\beta = \frac{X_0}{l}$. This result enables us to synthesize new views of the scene with any vertical slit of our choice, by sampling the columns of the original input sequence according to $t = \alpha s + \beta$, with α and β assigned the appropriate values.

4.1.2 Implementation Issues

In this section I address the case when the motion of the camera deviates from constant speed, and how the aspect ratio of the resulting mosaic is determined. I also present an alternative implementation of mosaicing, namely the slicing of space-time volume.

Variable Camera Speed

When the camera moves in a linear trajectory but varying orientation and speed, we compensate for this variability by estimating the camera motion (see [26]) and by derotating the image planes. I found that when the changes in camera orientation are small, a simple approximation is sufficient. Specifically, we compute the 2D rotation and translation between consecutive input frames using the method described in [14], and warp the images to cancel 2D rotation and vertical translation. The residual 2D translation is used as a rough approximation to the 3D velocity of the translating camera, and determines the thickness of the vertical strip. This approach is similar to the pushbroom mosaicing technique described in [55].

Aspect Ratio Normalization

The most apparent aspect of the distortion in X-Slits images is the variation of aspect ratio, as analyzed in Section 3.1.3. To reduce this distortion, we vertically scale the new images. This normalization is essential for achieving compelling results.

Specifically, the distortion on the *image plane* of objects at depth Z given in Eq. (3.12) can be written as $\frac{Z}{Z+\Delta} \cdot \frac{f_y}{f_x}$ in the notation of Eq. (4.4). In order to keep the horizontal field-of-view angle constant in the walk-through animation, we sample all the columns from left to right (from the appropriate frames, according to the column sampling function). Without any scaling, this process generates an image in which only the plane at infinity ($Z = \infty$) appears undistorted. Therefore, in order to cancel the distortion at depth Z_0, we scale the image vertically by the factor:

$$1 + \frac{\Delta}{Z_0} \tag{4.5}$$

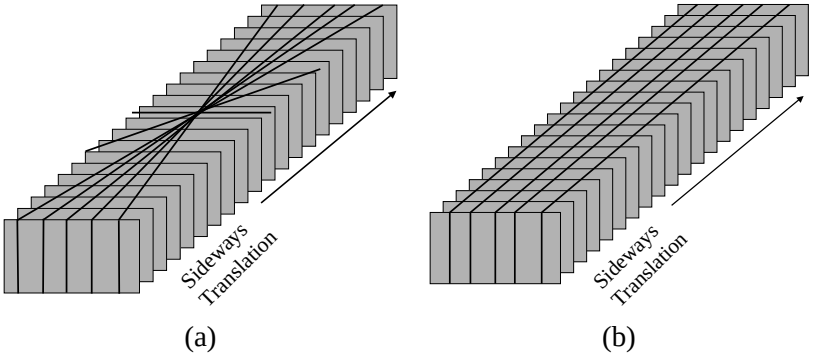
The Space-Time Volume

In Section 4.1.1 I described how to synthesize a X-Slits image by sampling columns from the input images using the following linear sampling formula:

$$t = \alpha s + \beta \tag{4.6}$$

where t denotes the camera translation. Recall that α, β are free parameters which control the location of the vertical slit.

A useful representation for the visualization of this process is the Space-Time Volume (or the *epipolar volume*), which is constructed by stacking all input images into a single volume. In case of constant sideways camera motion, any vertical planar slice in the volume according to (4.6) is a X-Slits image. This process is illustrated in Fig. 4.3; it assumes that the input sequence has high frame-rate and negligible spatial aliasing, so that simple interpolation (such as bilinear or bicubic) of the volume is sufficient. Thus rendering new X-Slits images is as simple as slicing a plane in the space-time volume.



(a) (b)

Figure 4.3: A schematic description of images generated as slices in the space-time volume. (a) Changing the orientation of the slice moves the vertical slit inside and outside the scene. (b) The central slice gives a pushbroom image (the "traditional" mosaic). Sliding parallel slices in the space-time volume results in different viewing directions of oblique pushbroom images.

4.1.3 Non-Linear Slits

One of the slits in the synthetic X-Slits images is the trajectory of the camera. When the camera's trajectory is not linear, a X-Slits view can be generated for which one of the slits is curved. An interesting family of such views has one circular slit and one linear slit. One way to generate such images is to use a camera rotating off-axis on a circle, as in concentric mosaics [64]. Concentric mosaics allow the generation of images in which the viewer can move continuously in a circular region. Each image generated from concentric mosaics is consistent with a circular X-Slits projection: One slit is the horizontal circular path of the camera center, and the second slit is a vertical linear slit placed at the location of the viewer. To generate an image from a different viewing position, the vertical slit is placed in the new location.

While images generated from the concentric mosaics point outward, and the viewer location is inside the circle, it is also interesting to generate inward looking images from locations outside the circle. This can be realized by moving the camera in a circle around an object, or by having a stationary camera viewing an object rotating on a turntable. Now the location of the viewer in the synthesized images can be as far or as close to the object as we wish, inside or outside the circular slit.

The column sampling function which generates circular X-Slits images is not linear. It is easier to

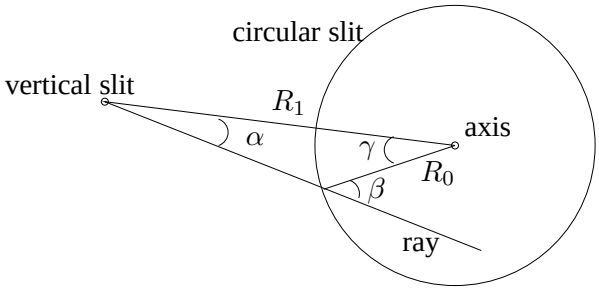


Figure 4.4: Synthesizing new views using the circular slit model

express it in angular terms, so I denote columns by their angle from the optical axis, and frames by the angular position of the camera. Assume that the images were taken by a pinhole camera rotating off-axis at radius R_0. In order to synthesize a X-Slits image with vertical slit at radius R_1, we take column β from the pinhole camera at γ and paste it as column α in the new image, as shown in Fig. 4.4. It can readily be seen that the following relation should hold: $\beta = \arcsin(\frac{R_1}{R_0}\sin\alpha)$ and $\gamma = \beta - \alpha$.

The distortions of such images are more complicated to analyze. However, in practice, since the field-of-view angles of cameras tend to be rather small, the circular slit in the relevant region is nearly linear, and therefore the distortions are approximately the same as with linear slits. Using the same aspect ratio normalization method as with linear slits, the results we achieve are quite convincing.

4.1.4 Results

In all my experiments I used a camera moving in the horizontal plane. As discussed above, new view generation in this case is done by sampling vertical strips from successive images and pasting them together into individual X-Slits images. The parameters of the strip sampling function determine the location of the vertical slit of the X-slit camera. In my experiments below, I manipulated the parameters of the sampling function so that the location of the vertical slit moves according to the desired egomotion. A very compelling impression of camera motion is obtained, even though the horizontal slit of the X-slits camera, which is the trajectory of the input camera, remains fixed.

In addition, I also simulated a change of camera orientation using the equations derived in Section 3.2.1. Note that the slits are left as are, changing only the orientation of the image plane. This

is because the full rotation of the slits would change the set of visible projection rays, and therefore cannot be performed by a 2D transformation of the image.

Next I discuss two applications: the generation of a virtual walkthrough from a sequence of perspective images, and 3D object visualization. The input video sequences used in these examples, as well as the synthesized walkthrough movies, are currently available on the web at
`http://www.cs.huji.ac.il/~daphna/demos.html#xslits .`

Virtual Walkthrough

In the first experiments (Figs. 4.5-4.8) I synthesized new sequences which correspond to a camera motion that has forward motion component, with visible parallax and lighting effects. In addition, the direction of the image plane was changed.

Another example used a sequence taken by a helicopter flying along a rocky coast in an unknown path and viewing direction (Fig. 4.9). Here I synthesized a new sequence which corresponds to a forward moving camera. This sequence was more challenging since the input sequence was taken in free motion with random disturbances (e.g., the effect of wind), and thus motion compensation was required (see discussion in Section 4.1.2).

New Views of Extended X-Slits Images

In this example I show how to generate new views from a sequence of a rotating object, where the new sequence demonstrates forward motion with parallax (Fig. 4.10). The projection model of the new images correspond to non-linear slits, as discussed in Section 4.1.3.

Object Visualization

Here I demonstrate the use of the X-Slits projection for object visualization – an object can be "flattened", revealing several of its sides simultaneously, by positioning the vertical slit behind the object (Fig. 4.11). Since the image is a valid X-Slits image that can be characterized and analyzed, we need not worry about such issues as duplicate images, which usually require hand-crafted stitching.

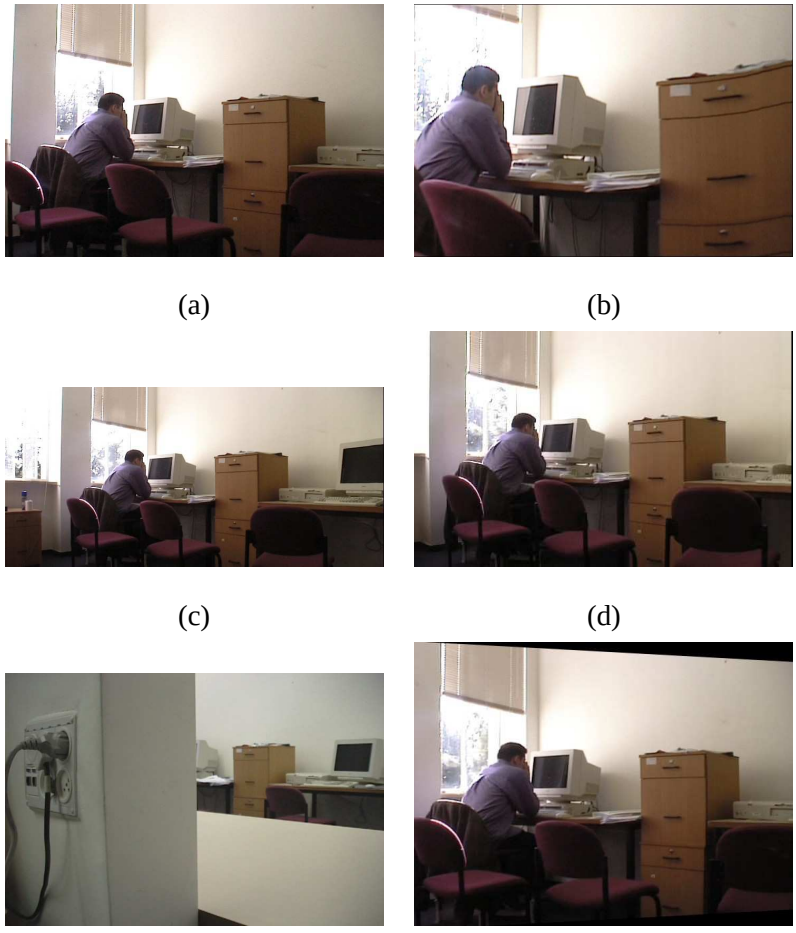
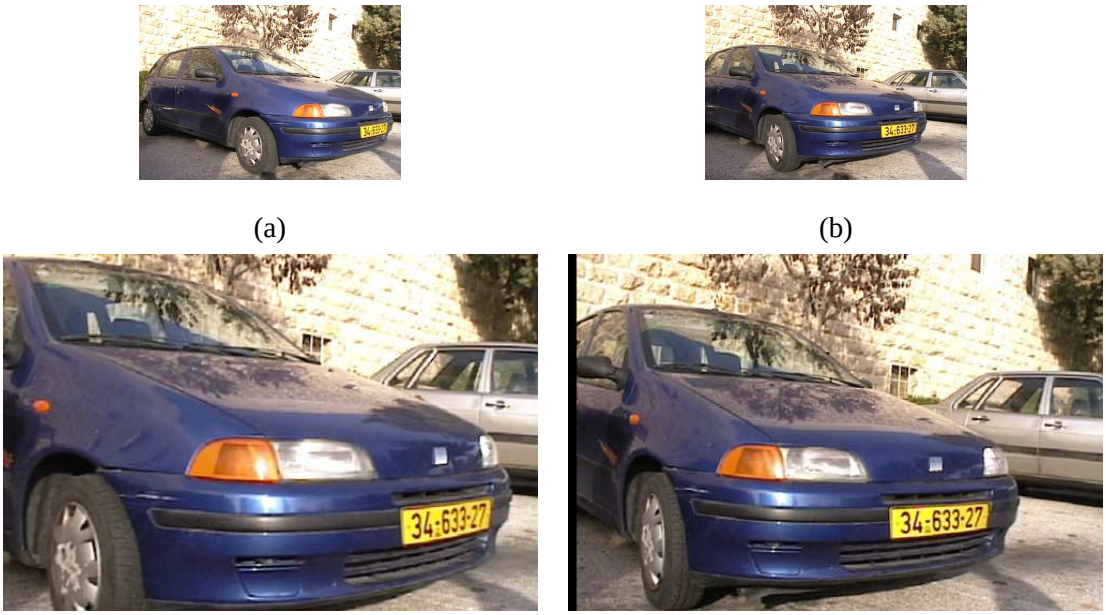


Figure 4.5: This scene is located in a small room where moving backward to capture the whole room is impossible. The scene was filmed by a sideways moving camera, total of 591 frames; one of the original frames is shown in (a). I show three new images: one where the vertical slit is located in front of the original track (b), and two where the vertical slit is located behind the original track (c-d). For comparison, I took a normal (pinhole) picture from the same location as (c), where part of the scene is obscured by the wall; this picture is shown in (e), and it demonstrates the ability to make images from impossible camera positions. Finally, (f) shows a simulated image where the camera was translated and rotated.

(a) (b) (c)

Figure 4.6: This scene was filmed by a sideways moving camera in our lab, total of 567 frames. I generated a X-Slits movie where the virtual camera rotated about an object in the scene (a-b), and then translated ahead in a diagonal (c).



(a) (b)

(c) (d)

Figure 4.7: Virtual walkthrough from a translating camera. (a), (b) Two frames from the input sequence. (c), (d) Two images rendered in forward motion. Note the apparently realistic changes in parallax and reflection.

Figure 4.8: This scene was filmed by a sideways moving camera, total of 585 frames. I generated a movie which included both rotation about the person and forward motion. Three frames of this movie are shown above; note the changes in the window reflections, which appear realistic.

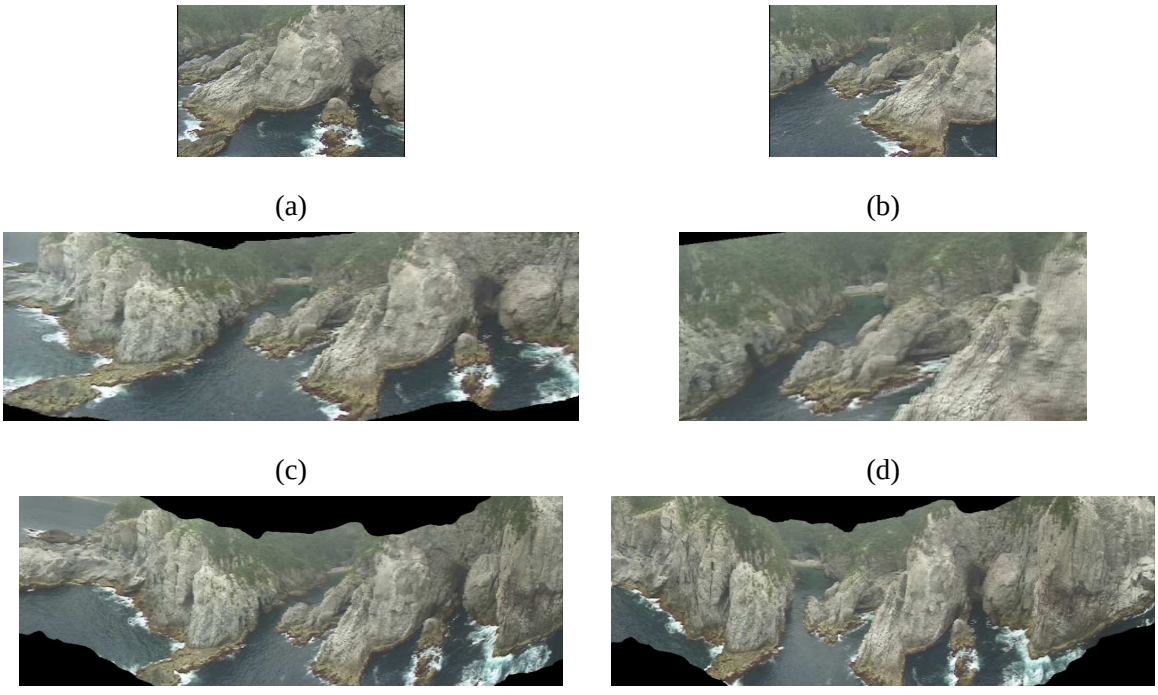
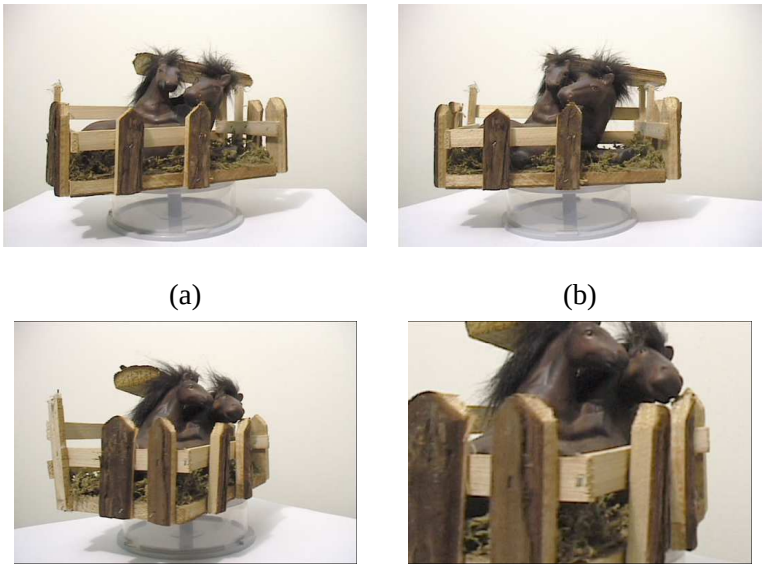


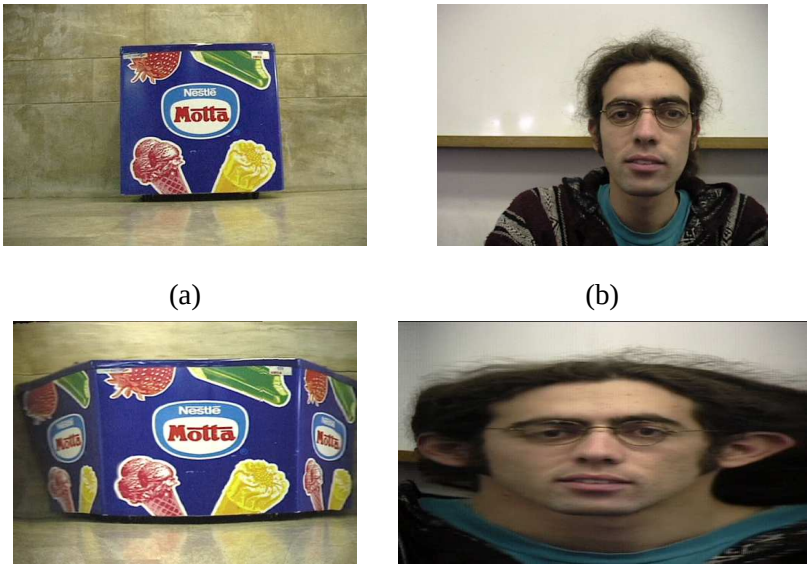
Figure 4.9: Virtual walkthrough generated from a sequence taken by a freely flying helicopter. (a), (b) Two frames from the input sequence. (c), (d) Two images rendered in forward motion (diagonal slices). (e), (f) Two images rendered in different viewing angles (parallel slices).



(a) (b)

(c) (d)

Figure 4.10: A rotating object: (a), (b) Two images from the original sequence of a rotating object. (c), (d) Two synthesized images from a forward moving viewpoint.



(a) (b)

(c) (d)

Figure 4.11: Object representation. (a), (b) The original input images. (c), (d) Visualization with the vertical slit located behind the object. The object is seen as if "opened" inside-out, giving a cubist effect: multiple sides are seen in a single picture.

4.2 Omnidirectional Crossed-Slits Mosaicing

I adopt here the circular X-Slits camera model, in which one of the slits is a circle in the $X - Z$ plane, and the second is a linear slit in the Y direction. In this case, each scene point defines a plane with the linear slit, which intersects the circular slit at *two* points, and thus each scene point has two rays. Of the two intersections with the circular slit, we choose the one that is closer to the scene point, and the corresponding ray is defined to be the unique ray through the scene point (this is the "outgoing" ray, emanating out of the circular slit). To complete the definition of the circular X-Slits camera model, the image surface is chosen to be the sphere at infinity, meaning that the correspondence between a ray and a point in the image is defined only by the azimuth and elevation angles. Such an image is *onmidirectional*, as rays in all directions are imaged.

We synthesize such images from a set of images taken by an input camera rotating off-axis. The *input camera* is a central camera, thus all the rays captured by the camera pass through a single point (its center of projection). Unlike regular perspective cameras which sample these rays on a planar rectangle (the image plane), the input camera samples the rays on a hemisphere at its center of projection. Generating a new omnidirectional X-Slits view of the scene consists of generating a spherical view from a chosen slit location, i.e., an image of the rays passing through a virtual slit.

The path of the input camera is assumed to be a circle of radius 1 in the $X - Z$ plane. The *virtual slit* is a vertical line passing through the point $\mathbf{e} = (x_e, 0, z_e)$, which is defined to be the location of the virtual "eye". The new view at each slit location is sampled on the *output sphere*, centered about the virtual eye $\mathbf{e}$. With this definition, moving around the scene is a matter of moving the virtual slit and generating the X-Slits image corresponding to each new position.

Omnidirectional X-Slits rendering is done, as in the linear case, by means of mosaicing. Strips are taken from each input image, and stitched together into a new image. The strips are selected according to the location of the virtual eye, and the result is a X-Slits image that looks as if it were taken from that location.

Specifically, given a virtual slit location, we need to determine which rays should be sampled from which input camera. Each input camera center $\mathbf{c}$ defines a plane Π_c with the virtual slit; all the rays on

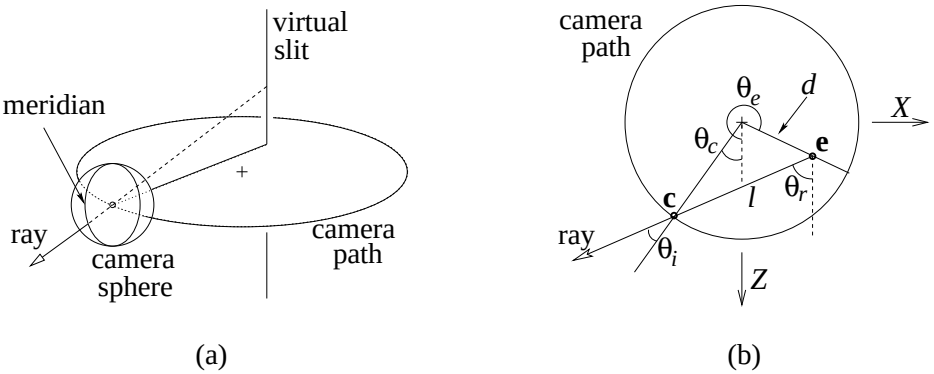


Figure 4.12: Omnidirectional X-Slits mosaicing. (a) Overview: the rays passing through the virtual slit and the input camera center form a plane of rays, which intersects the input hemisphere in a meridian. (b) Top view: determining which strip of pixels is sampled from which input image.

this plane pass through the virtual slit and a point on the circular slit (the camera center $\mathbf{c}$). Thus this plane includes all the rays that should be sampled from camera $\mathbf{c}$, given $\mathbf{e}$. The intersection of Π_c with the input camera hemisphere is a meridian (see Fig. 4.12a). Thus, each input camera contributes a strip of rays that lies on an input meridian.

The sampled strip is pasted as a strip on the output sphere. Since, by definition, Π_c passes through the center of the output sphere ($\mathbf{e}$), its intersection with the output sphere is also a meridian. Note, however, that the sampled *rays* generally do not pass through $\mathbf{e}$.

It follows from the discussion above that meridians from the input cameras are pasted as meridians in the output cameras. Hence, it would be beneficial to use an image representation that is based on latitude and longitude (Fig. 4.13b,c). In this representation, the coordinates of a pixel are its longitude and latitude on the sphere, so each meridian on the sphere is a column in the image. Generating omnidirectional X-Slits images becomes a matter of mosaicing vertical strips, as in linear X-Slits.

How do we determine which strip to sample from each input camera? Given a vertical slit passing through $\mathbf{e}$, let us define the polar coordinates $d = \sqrt{x_e^2 + z_e^2}$, $\theta_e = \arctan \frac{x_e}{z_e}$. As can be seen from Fig. 4.12b (and triangle geometry), for every θ_r, the strip to paste at the θ_r meridian of the output

(a) (b) (c)

Figure 4.13: Spherical images. (a) An image from the input sequence, as acquired with a panoramic lens. (b) The same image in latitude-longitude representation of a hemisphere. (c) An output spherical image.

sphere should be taken from the θ_i meridian of the input camera at θ_c, where

$$\begin{aligned} \theta_i &= \arcsin(d\sin(\theta_r - \theta_e)) \\ \theta_c &= \theta_r - \theta_i \end{aligned} \tag{4.7}$$

4.2.1 Distortion

In this section I analyze the nature of X-Slits distortions, caused by the absence of a single center of projection. Specifically, I compare between X-Slits images corresponding to the model described above, and the regular perspective image corresponding to an omnidirectional camera centered around the virtual "eye" $\mathbf{e}$.

Recall that the plane of rays Π_c determined by (4.7) is the same as if it were perspective projection, but the rays within the plane do not intersect in $\mathbf{e}$, but rather in the input camera centers $\mathbf{c}$, which are different for each plane Π_c. As a result, a scene point $\mathbf{p}$ that is seen at some elevation angle ϕ_r by the input camera, would be seen at a different elevation angle ϕ_r^* by the virtual eye $\mathbf{e}$ (see Fig. 4.14a). Without correcting the elevation angle of each ray, $\mathbf{p}$ would appear shifted vertically at a false location (Fig. 4.14a).

In order to cancel out this distortion, we need to determine the correct elevation angle for each input ray, estimating how the scene point would have been seen from $\mathbf{e}$. This would produce the correct perspective view of the scene, but it requires a dense and accurate knowledge of the 3D structure of the scene.

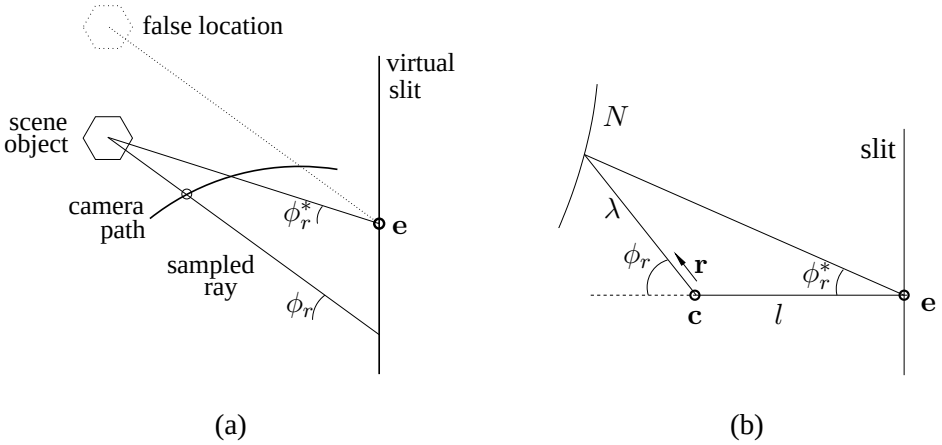


Figure 4.14: (a) Distortion: the object is seen at a different elevation angle from the input camera and from the virtual eye. (b) Normalization: cancelling out distortion by correcting the elevation angle of the input ray, using a normalization surface.

When accurate depth information is not available or hard to obtain, we can still produce appealing images by using a coarse estimation of depth. In general, we define a *normalization surface*, which crudely approximates the scene structure, and use it to reproject the rays before pasting them into the mosaic. This allows us to generate images that look compelling, without relying on an elusive depth map. The normalization procedure is described next.

4.2.2 Normalization

In general, normalization is done by intersecting each sampled ray with the normalization surface, and reprojecting this intersection through the virtual camera center $\mathbf{e}$. Given an input ray of azimuth θ_r and elevation ϕ_r, the input camera's position on the circle (denoted θ_c) is determined by (4.7), see Fig. 4.12. The sampled ray is defined as $\mathbf{c} + \lambda\mathbf{r}$, where $\mathbf{c} = (-\sin\theta_c, 0, \cos\theta_c)^T$ denotes the input camera location and $\mathbf{r} = (-\sin\theta_r\cos\phi_r, \sin\phi_r, \cos\theta_r\cos\phi_r)^T$ denotes the ray direction (see Fig. 4.14b). Given a normalization surface expressed implicitly as $N(\mathbf{p}) = 0$, the intersection is at $\min\{\lambda | N(\mathbf{c} + \lambda\mathbf{r}) = 0, \lambda > 0\}$.

To begin with, let us consider a *spherical* normalization surface. Thus, $N(x, y, z) = x^2 + y^2 + z^2 - R^2$, where R is the radius of the normalization sphere. Substituting $\mathbf{c} + \lambda\mathbf{r}$ into N gives

$$\lambda^2\mathbf{r}^T\mathbf{r} + 2\lambda\mathbf{r}^T\mathbf{c} + \mathbf{c}^T\mathbf{c} - R^2 = 0 \tag{4.8}$$

(a) (b) (c)

Figure 4.15: Normalization – a view of the synthesized scene without normalization (a), and with normalization using a sphere (b) and an ellipsoid (c). Note how the aspect ratio is different for objects at different depths in the spherical case.

Solving for $\lambda > 0$ yields the intersection of the ray with N at $\lambda = -k + \sqrt{k^2 + R^2 - 1}$, where $k = \cos\phi_r \cos\theta_i$. From Fig. 4.14b one can see that the ray should be reprojected through the virtual camera center at an elevation angle of

$$\phi_r^* = \arctan \frac{\lambda \sin\phi_r}{\lambda \cos\phi_r + l} \tag{4.9}$$

where $l = \cos\theta_i + d\cos(\theta_r - \theta_e - \pi)$ is the distance between the slit and the input camera (see Fig. 4.12b). Note that unnormalized omnidirectional X-Slits images ($\phi_r^* = \phi_r$) correspond to $R \to \infty$.

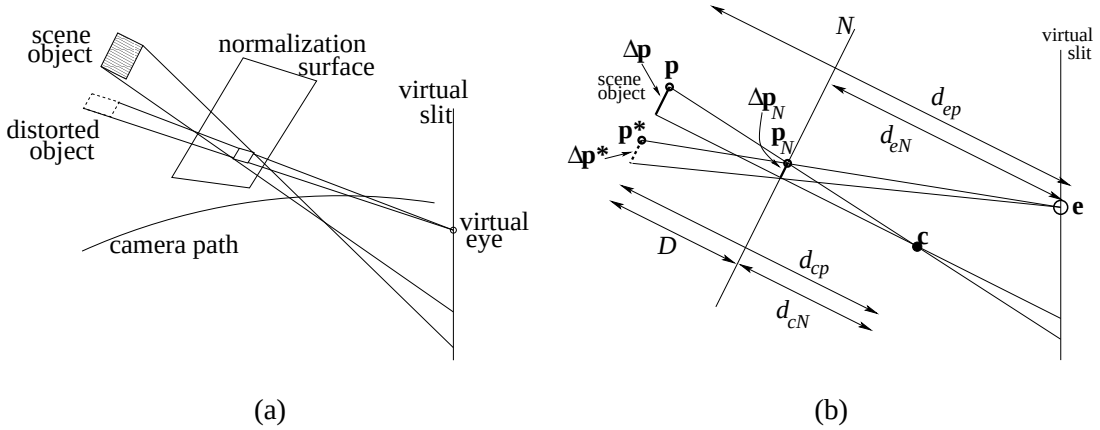
Normalization onto a sphere is appropriate for scenes that lie at a relatively constant distance from the viewer, e.g., a room viewed from its center. If the room is elongated, however, the sphere provides a poor approximation of the scene's structure (Fig. 4.15b). In this case, normalizing onto an ellipsoid may be more appropriate.

For an ellipsoid (or any quadric), let us redefine $\mathbf{c}$ and $\mathbf{r}$ as homogeneous coordinates in $\mathcal{P}^3$ with the forth coordinate set to 1 and 0, respectively, and let $N(\mathbf{p}) = \mathbf{p}^T\mathbf{Q}\mathbf{p}$ where $\mathbf{Q}$ is the 4×4 matrix that describes the quadric. The intersection is then at $\lambda = -k + \sqrt{k^2 - m}$ where

$$k = \frac{\mathbf{c}^T\mathbf{Q}\mathbf{r}}{\mathbf{r}^T\mathbf{Q}\mathbf{r}} \qquad m = \frac{\mathbf{c}^T\mathbf{Q}\mathbf{c}}{\mathbf{r}^T\mathbf{Q}\mathbf{r}} \tag{4.10}$$

and the ray is reprojected according to (4.9). Fig. 4.15c provides an illustrative example, and some comparisons between the different normalization methods.

In general, one can use an estimated sparse depth map to construct a general normalization surface, and use ray tracing techniques to reproject the rays.



(a) (b)

Figure 4.16: Distortion under normalization. (a) Overview: when the normalization surface is incorrect, reprojection makes the object appear shifted vertically. (b) Side view: the aspect ratio distortion as it is related to the distances between the input camera, the virtual eye, the normalization surface and the object.

4.2.3 Measuring Distortions

Since the normalization surface gives only a crude approximation of the scene structure, it is not likely to eliminate all distortions, and in some cases it may even introduce new distortions.

I propose to measure distortions in X-Slits images by the change in aspect ratio. In perspective projection, aspect ratio is preserved, and any rectangle in the scene that is parallel to the image plane would be projected into a rectangle with the same proportions between width and height. In X-Slits projection, this is usually not the case.

Normalization corrects this problem for objects that are on the normalization surface, since they are projected as if they were perspective. As I show below, the farther an object is from the normalization surface, the more distorted its aspect ratio would be.

Consider a point $\mathbf{p}$ on an object in the scene. We would like to estimate the aspect ratio distortion in a neighborhood around $\mathbf{p}$ in a normalized X-Slits image, when $\mathbf{p}$ is not on the normalization surface.

The point $\mathbf{p}$ and its neighborhood is captured by rays passing through the input cameras. These rays intersect the normalization surface N at $\mathbf{p}_N$, and are reprojected during normalization (Fig. 4.16a). This normalization is correct only for a point $\mathbf{p}$ on N; otherwise the object appears to be in a false location.

As we shall see, aspect ratio distortion is not necessarily constant, so we will estimate the *local* aspect ratio distortion at $\mathbf{p}$. Assume that the normalization surface around $\mathbf{p}_N$ is parallel to the object surface around $\mathbf{p}$. Denote the plane that contains the virtual slit and the ray through $\mathbf{p}$ by Π_p (Fig. 4.16b shows a view of Π_p). If the length on Π_p of the patch around $\mathbf{p}$ is $\Delta\mathbf{p}$, and the length of the patch as it is projected on N is $\Delta\mathbf{p}_N$, then it follows from triangle similarity that

$$\frac{\Delta\mathbf{p}}{\Delta\mathbf{p}_N} = \frac{d_{cp}}{d_{cN}} \tag{4.11}$$

where d_{cp} is the distance on Π_p between $\mathbf{c}$ and the plane tangent to the object at $\mathbf{p}$, and d_{cN} is the distance between $\mathbf{c}$ and the plane tangent to N at $\mathbf{p}_N$. For the same reason, if $\Delta\mathbf{p}^*$ is the length of the false object, then

$$\rho \equiv \frac{\Delta\mathbf{p}^*}{\Delta\mathbf{p}} = \frac{d_{ep}}{d_{cp}} \cdot \frac{d_{cN}}{d_{eN}} \tag{4.12}$$

Since the X-Slits projection only introduces distortions in the vertical direction, and since normalization also deals only with the vertical direction, there is no horizontal change in the way the patch around $\mathbf{p}$ is projected. Therefore, the ratio ρ is the aspect ratio distortion that point $\mathbf{p}$ undergoes when projected with a X-Slits projection normalized by surface N (compare to the result in Section 3.1.3).

Denoting the distance between the normalization surface and $\mathbf{p}$ as D (hence $d_{eN} = d_{ep} + D$ and $d_{cN} = d_{cp} + D$), when the virtual eye is *behind* the camera path (i.e., $d_{ep} > d_{cp}$), the aspect ratio grows with D: when D is positive – the object will appear taller, and vice versa (with correct aspect ratio when $D = 0$). This is reversed when the virtual eye is in front of the camera path.

When the normalization surface is not parallel to the scene object and there is a difference in elevation angle between them, it can be shown that the aspect ratio becomes

$$\rho = \frac{\Delta\mathbf{p}^*}{\Delta\mathbf{p}} = \frac{d_{ep}}{d_{cp}} \cdot \frac{d_{cN}}{d_{eN}} \cdot \frac{\sin\alpha_{pi}}{\sin\alpha_{po}} \cdot \frac{\sin(\alpha_{po}-\alpha_{pN})}{\sin(\alpha_{pi}-\alpha_{pN})} \tag{4.13}$$

where α_{pi}, α_{po} are the angles between the object plane and the input and reprojected rays, respectively, and α_{pN} is the elevation angle difference between the object plane and the normalization surface.

4.2.4 Discussion

As the distance between the scene and the normalization surface decreases, so does the aspect ratio distortion. Thus, in order to achieve correct aspect ratio, we need to approximate the scene as well as possible. However, this may lead to other, often worse distortions.

Specifically, Equation (4.12) states that the aspect ratio is a function of the distances between the normalization surface, the scene surface, the virtual eye and the input camera. Variation in these distances across the image causes variation in aspect ratio. Strong variations in aspect ratio may be caused by abrupt variations in the depth of the normalization surface, if these do not correspond to real variations in scene depth. It is usually hard to obtain a depth map that fits the scene structure accurately, especially where depth changes abruptly (e.g., at depth edges); in these areas in particular, abrupt changes in the normalization surface may cause strong noticeable distortions.

In contrast, depth discontinuities in the scene without corresponding variations in the normalization surface pose less of a problem, since the arising aspect ratio variation will occur over a few pixels spanning the edge in the image. It is therefore often preferable to simply use a smooth normalization surface, which provides only crude approximation of the depth structure of the scene.

4.2.5 Augmented Reality

Generating realistic views of a precaptured scene in realtime is useful for virtual reality. A user's head motion may be tracked and the appropriate views of the scene can be generated and displayed at a reasonable rate. However, the rendered scene is static, and it may be desirable to add virtual objects to the scene, which would be rendered and superimposed on the X-Slits image. I shall discuss only the geometric issues of augmented reality with the X-Slits projection.

Since the rendered scene is a X-Slits image, the added object must also be projected according to the same projection model in order to appear consistent. An object added in a certain place in the scene should appear in the same position when viewed from different viewpoints, and this can only be accomplished if the object is projected using the same projection model as the rest of the image.

Most computer graphic renderers generate perspective images. In order to use such engines for

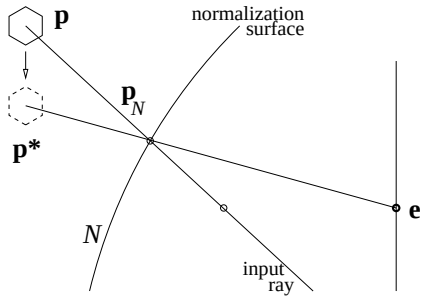


Figure 4.17: Augmented Reality. Scene point $\mathbf{p}$ is reprojected into the virtual eye through point $\mathbf{p}_N$ on the normalization surface, so if a virtual object is augmented at $\mathbf{p}$, it should be shifted vertically to appear correct.

the rendering of X-Slits images, we must first transform the augmented objects in a manner similar to the reprojection discussed above, so that when projected using the regular perspective projection, they would appear correct in the X-Slits image. As shown above, the distortions and normalization associated with the X-Slits projections are in the vertical direction alone, so only vertical shifting needs to be done when transforming the augmented object.

Specifically, suppose an object is augmented at point $\mathbf{p}$, and the ray through $\mathbf{p}$ intersects the normalization surface at $\mathbf{p}_N$. In order for the object to look as if it were at $\mathbf{p}$, it must be on the reprojected ray through $\mathbf{p}_N$ (see Fig. 4.17). Shifting the object's location vertically prior to imaging, so that it is on this ray, would give this effect.

Given a point (x, y, z) where we wish to add an object, and given a slit at $X = x_e, Z = z_e$, the azimuth of the object is $\theta_r = \arctan \frac{-\Delta x}{\Delta z}$ where $\Delta x = x - x_e$, $\Delta z = z - z_e$, and the elevation angle relative to the input camera is

$$\phi_r \quad = \quad \arctan \frac{y}{L - l} \tag{4.14}$$

where l is the distance between the slit and the input camera as in (4.9), and $L = \sqrt{\Delta x^2 + \Delta z^2}$ is the horizontal distance between the slit and the object. It now follows that the distorted location of the augmented object is $y^* = L \cdot \tan \phi_r^*$, where ϕ_r^* is given in (4.9). If we shift the object's position vertically to this height, it can be projected normally and appear as if it were in the X-Slits image of the scene at the desired location (Fig.4.18).

Since the distortion is variable, this transformation should be applied separately to every point on

(a) (b) (c)

Figure 4.18: Augmented Reality. (a) The image-based views of the scene with augmented objects. (b) The same scene from a different viewpoint. (c) Without vertical correction, the objects' locations are not correct.

the augmented object. In practice, it is usually sufficient to move the object according to just one point, e.g., the point where it is supposed to touch some real object in the scene.

4.2.6 Implementation Issues

Using the latitude-longitude representation of spherical images, the rendering of omnidirectional X-Slits images is just a matter of sampling columns from images and pasting them in the output image, like linear X-Slits rendering. Normalization requires vertical transformation of each pixel, which may be a costly calculation for a realtime application. However, when the input camera path is small in relation to the normalization sphere, ϕ_r^* is nearly linear in ϕ_r, and a (much faster) linear transformation is sufficient.

Specifically, the top and bottom pixels in each input column correspond to elevation angles $\phi_r =$

$\pm\pi$. Substituting these values in (4.9) gives the normalized elevation angles of $\phi_r^* = \pm \arctan \frac{\lambda}{l}$, so normalization can be done approximately by scaling each column linearly according to this formula.

Displaying the omnidirectional X-Slits images with a display device involves projecting the spherical image on a plane. Graphic engines that handle perspective projection are abundant, so this mapping can be done efficiently by mapping the rendered image on a sphere centered about the virtual eye. Furthermore, if the sphere is approximated as a mesh, the normalization can be done on each vertex of the mesh, instead of on each pixel of the X-Slits image.

The same graphic engine can be used for augmented objects. Provided their position is corrected according to (4.14), they can simply be rendered along with the mesh.

The scene shown in Fig. 4.15 and Fig. 4.18 is rendered from a set of 529 panoramic images (size 2500×1024). Rendering is done in realtime at a rate of 20 frames per second (size 1024×1024) on a Pentium IV 2.8GHz. Captures from my implementation can be viewed online at
`http://www.cs.huji.ac.il/~daphna/ibr` .

4.3 Optimal Mosaicing

Consider a perspective video camera moving continuously on a curved segment with its image plane orientation tangent to the curve. Assume w.l.o.g. that the segment length is 1, and let $t \in [0, 1]$ be a parameter describing the location of the camera in the segment. A multi-perspective mosaic is generated by selecting a vertical line in each frame $I(t)$ according to a *sampling function* $\phi(t)$, and pasting it into the mosaic. $\phi(t)$ denotes the location of the line sampled from frame $I(t)$. Let $\pi(t)$ be the plane joining the camera center of projection at location t to the sampled line $\phi(t)$. The pasting location in the mosaic is defined by the intersection of $\pi(t)$ with the mosaic manifold. In case the camera moves on a linear trajectory, this manifold is a plane. Otherwise, the manifold is determined by the camera trajectory. As in [55], the distance of the manifold is set to be equal to the camera's focal length, in order to maintain the vertical resolution of the image.

It is assumed that the camera motion and internal calibration are known, or were estimated from the video (for a review, see [26]), and that the horizontal field-of-view angle of the camera is θ.

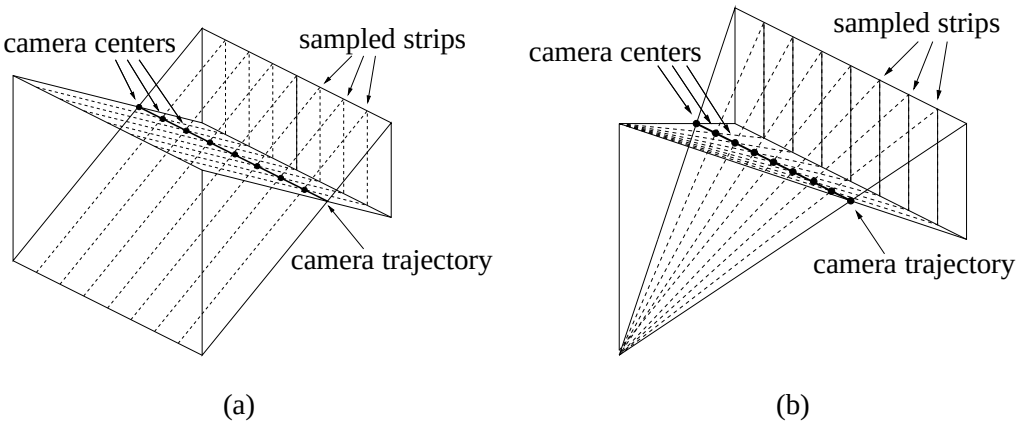


Figure 4.19: Mosaicing by (a) a constant sampling function and by (b) a linear sampling function.

I first analyze mosaics generated from linear camera trajectories, and find the optimal mosaic analytically. Two useful examples of sampling functions for linear camera trajectories, depicted in Figure 4.19, are the linear sampling function $\phi(t) = \alpha t + \beta$ and a special case of it, the constant sampling function $\phi(t) = \beta$ (where $\alpha = 0$). It was shown in Section 4.1.1 that in the former case, all rays pass through a vertical line in the plane

$$Z_2 = Z_1 + \frac{k}{\alpha} \tag{4.15}$$

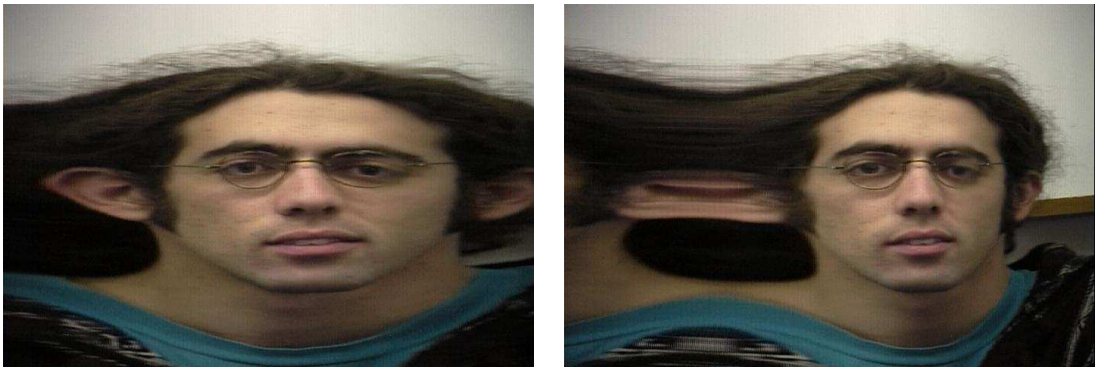
where Z_1 is the the plane of the camera trajectory and $k = \cot(\frac{\theta}{2})$; in the latter case, this plane is at infinity.

General smooth trajectories are analyzed in Section 4.3.6 using local linear approximations.

4.3.1 Necessary Conditions for a Good Mosaic

Let $V \equiv \{(X, Y, Z) | Z > 0\}$ be the set of viewed scene points, i.e. the points in front of the camera. I define the following necessary conditions for a good mosaic:

- *Unique Projection:* Every 3D point $P \in V$ is projected to a single point in the mosaic image.
- *Continuous Projection:* Connected sets of scene points are projected to connected sets of image points.
- *Data Utilization:* Strips are taken from all images.



(a) (b)

Figure 4.20: 3D object representation by mosaicing with a monotonic decreasing function. Mosaic (a) was generated by a linear sampling function, so every point on the object is associated with a single point on the image. Mosaic (b) was generated by a non-linear sampling function, and as can be seen, some scene points appear twice in the mosaic.

Unique projection is important in order to avoid duplicate images of an object in the mosaic image. In Section 4.3.2 I show that for linear camera trajectories, this condition holds if and only if the sampling function is monotonic non-decreasing.[1] In Section 4.3.3 I relax the unique projection condition by allowing a set G of points of measure 0 to violate the uniqueness condition. I show that in this case, G must be a line, and in case the camera moves on a linear trajectory, this corresponds to a linear sampling function. An almost-unique projection can be useful for constructing representations of convex objects. An example is shown in Figure 4.20.

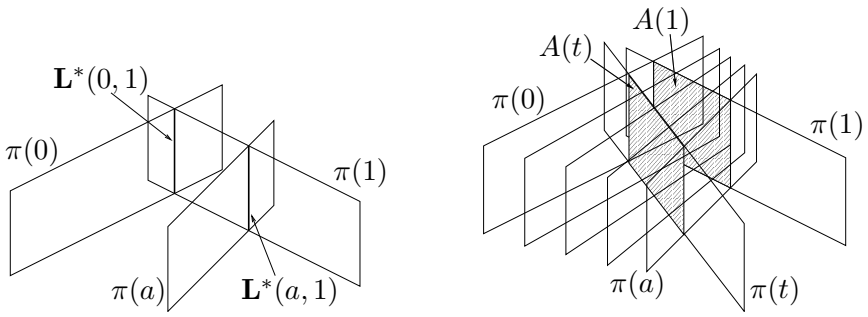
The requirement for a continuous projection is obvious – to avoid discontinuities in the mosaic image. It follows that the sampling function must also be continuous.

The data utilization requirement is important for ensuring maximal field of view when minimizing the geometric distortion.

4.3.2 Projection Uniqueness

The projection is unique if every scene point is projected to a single point in the mosaic image. A key observation is that the scene points V are *in front* of the camera. Hence the planes $\pi(t_1), \pi(t_2)$

[1] Without loss of generality, it is assumed that the camera is moving from left to right.



(a) (b)

Figure 4.21: Illustrations for the proof of Theorem 4.1

must not intersect in front of the cameras for any $0 \leq t_1 < t_2 \leq 1$. For a camera moving on a linear trajectory, this implies that the sampling must be *non-decreasing monotonic*.

4.3.3 Uniqueness Excluding a Set of Measure 0

Another useful criterion relaxes the requirement by allowing some points to violate the uniqueness condition; This set of points G is required to be of measure 0 (e.g. a point or a curve). If G does not include any scene point, no scene point would appear multiple times in the mosaic. As I show below, this criterion implies that G is a line.

Theorem 4.1 *For any continuous sampling function $\phi(t)$, if the set of points that are not uniquely sampled is of measure 0, then this set is a line.*

Proof: The planes $\pi(0), \pi(1)$ intersect in a line $\mathbf{l}$, and all of its points are sampled by both cameras $t = 0, 1$. I show that if there are points that are sampled by two cameras that are not on this line, then the set of all such points is of measure greater than 0: For every $s, t \in [0, 1]$, the intersection of plane $\pi(s)$ with plane $\pi(t)$ can be represented by the dual Plücker matrix:

$$\mathbf{L}^*(s,t) = \pi(s)\pi(t)^T - \pi(t)\pi(s)^T \tag{4.16}$$

(the planes are represented in homogeneous coordinates, see [26, p. 52]). Since $\pi(t)$ is continuous, it follows that $\mathbf{L}^*(s,t)$ is continuous in s, t. If there exists a point not on $\mathbf{l}$ that is sampled more than

once, then it lies on a plane $\pi(a)$ for some $a \in (0,1)$ such that $\mathbf{L}^*(0,1) \not\propto \mathbf{L}^*(a,1)$ (i.e., $\pi(0)$ and $\pi(a)$ intersect $\pi(1)$ in different lines). Refer to Figure 4.21 for illustrations. Consider the union of all lines of the form $\mathbf{L}^*(s,1)$ for $s \in [0,a)$ (which are intersections of the planes $\pi(s)$ in this range with $\pi(1)$). Since $\mathbf{L}^*(s,t)$ is continuous, it follows that this union is a set of area greater than 0 on the plane $\pi(1)$. Let $A(t)$ denote the set of all points on the lines associated with $\mathbf{L}^*(s,t)$ for all $s \in [0,a)$. Then the above can be written as $|A(1)| > 0$. Due to the continuity of $\pi(t)$, there exists an interval $(b,1]$ for which $|A(t)| > 0$ holds for *every* $t \in (b,1]$. Therefore, since all planes $\pi(t)$ are distinct, it follows that the union $\cup A(t)$ is a set of volume greater than 0. Since it is contained in the set of all points that are sampled more than once, this set cannot be of measure 0. ■

Result 4.2 *In the case of linear camera motion, the sampling functions satisfying the uniqueness criterion up to measure 0 are either monotonic non-decreasing or linear (see Figure 4.20).*

4.3.4 Perspectivity: a Measure for Geometric Quality

I consider perspective images to be non-distorted. Hence the distortions in a mosaic image are measured with respect to the closest perspective image. In [67], a distortion was measured with respect to the closest perspective image, with the distance defined as the sum of distances of matching image points. Such a measure, while visually compelling, required knowledge of the scene depth. Since in our case the scene depth is unknown, I compare the *3D to 2D projections* rather than the images. That is, we would like the 3D to 2D projection induced by the mosaicing method to be as close as possible to a perspective projection. In a perspective projection, all rays intersect in a point. Hence, for a multi-perspective mosaic, the set of sampled rays should be as closely bundled as possible. We find a center point that has a minimal distance to all sampled rays, and we measure how small this distance is.

First, I define the *local perspectivity distortion*, which implements the idea above locally, for a neighborhood around an image point. I then define a *global perspectivity distortion* by integrating the local perspectivity distortion on the entire image. I chose an additive measure, so that the perspectivity of one region in the image is not influenced by other regions in the image.

I first analyze the case of linear camera motion. For this case, the least distorted mosaic is derived

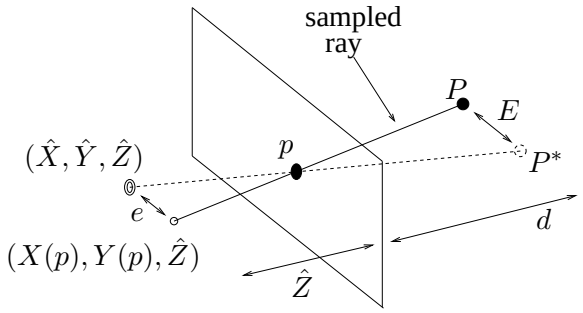


Figure 4.22: The relation between the local perspectivity distortion and the error in estimation of the 3D scene. See Section 4.3.5 for details.

analytically, and it turns out that the least distorted mosaic also has the widest field of view. Non-linear camera trajectories are analyzed in Section 4.3.6 using local linear approximations. The global perspectivity is minimized numerically using standard optimization techniques.

4.3.5 Perspectivity: Linear Camera Trajectory

I consider only monotonic non-decreasing sampling functions satisfying the necessary conditions defined in Section 4.3.1.

Given a sampling function ϕ, each image point is associated with a single ray. Let us denote the intersection of the ray of image point p with the plane $Z = \hat{Z}$ by $(X(p), Y(p), \hat{Z})$. I define the distortion of the sampling function ϕ with respect to a candidate center point $(\hat{X}, \hat{Y}, \hat{Z})$ in a neighborhood ω of image point p as -

$$n(\phi, p, \hat{X}, \hat{Y}, \hat{Z}) = \int_{\hat{p}\in\omega} \frac{(X(\hat{p})-\hat{X})^2 + (Y(\hat{p})-\hat{Y})^2}{\hat{Z}^2} d\hat{p} \tag{4.17}$$

and the *local perspective distortion* at p as -

$$n_L(\phi, p) = \min_{\hat{X},\hat{Y},\hat{Z}} n(\phi, p, \hat{X}, \hat{Y}, \hat{Z}) \tag{4.18}$$

The expression given in (4.17) measures the distance e of the ray from the candidate center point $(\hat{X}, \hat{Y}, \hat{Z})$, on a plane, relative to the depth $\hat{Z}$ of that plane (see Figure 4.22). The underlying idea is that an image is distorted if it is not consistent with a perspective image of a 3D scene. Consider a

scene point P at depth d, which is projected by a ray whose error is e. Were the image perspective, this image point would seem to be the projection of a scene point P^*, and the error in the 3D scene would be E, such that $E = e\frac{d}{\bar{Z}}$.

The global measure of distortion is obtained by integrating a local perspectivity distortion on the image. To cancel the effect of the proportions of the neighborhood ω, I define the *global perspectivity distortion* of a given sampling function ϕ as follows:

$$n_G(\phi) = \int_{p \in I} \frac{n_L(\phi, p)}{n_L(\hat{\phi}, p)} dp \tag{4.19}$$

where $\hat{\phi}(t)$ is a reference sampling function which can be chosen arbitrarily, and p is integrated over the image domain $I = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$. For simplicity, I choose the reference sampling function $\hat{\phi}(t) = 0$.

Theorem 4.3 *The global perspective distortion of a linear sampling function $\phi(t) = \alpha t + \beta$ is -*

$$n_G(\phi) = S\left(\frac{k}{k + \alpha Z_1}\right)^2 \tag{4.20}$$

where S is the image area (k and Z_1 are defined in (4.15)).

The proof of the theorem above is given in Appendix B. A direct result of the theorem above is the following:

Result 4.4 *The global perspectivity distortion of a linear sampling function $\phi(t) = \alpha t + \beta$ with $\alpha \geq 0$ is monotonic decreasing in α. The most distorted linear sampling is the constant sampling.*

Note that the distortion of a linear sampling function depends only on the slope of the function. Now let us study the general case of continuous non-decreasing sampling functions:

Theorem 4.5 *Given a continuous non-decreasing sampling function $\phi(t)$, let us denote the linear sampling function which agrees with $\phi(t)$ at $t = 0$ and $t = 1$ as $\phi'(t)$. If $\phi \neq \phi'$, then $n_G(\phi) > n_G(\phi')$.*

In order to prove the above, I first prove it for a polygonal sampling function, i.e., a function $\phi(t)$ for which the interval $[0, 1]$ can be divided into segments $[0, t_1, t_2, \ldots, 1]$ such that $\phi(t)$ is linear in each segment $[t_k, t_{k+1}]$.

Lemma 4.6 *Given a polygonal sampling function $\phi(t)$ and a linear sampling function $\phi'(t)$ such that $\phi(0) = \phi'(0)$ and $\phi(1) = \phi'(1)$, if $\phi \neq \phi'$ then $n_G(\phi) > n_G(\phi')$.*

Proof: The idea behind this proof is that by eliminating nodes in the polygon, the global perspectivity distortion does not increase. For any i, we eliminate the node i by defining a polygonal sampling function $\phi^*(t)$ which agrees with $\phi(t)$ everywhere except for the segment $[t_{i-1}, t_{i+1}]$, in which it is linear. As shown in (4.20), the distortion of a linear sampling function is proportional to the area of the image and to $\left(\frac{k}{\alpha Z_1 + k}\right)^2$. Denote the slopes of $\phi(t)$ in segments $[t_{i-1}, t_i]$ and $[t_i, t_{i+1}]$ by α_1 and α_2, respectively, and the slope of $\phi^*(t)$ in $[t_{i-1}, t_{i+1}]$ as α_3. The contribution of each segment to the global perspective distortion is proportional to its length, and therefore, $n_G(\phi) \geq n_G(\phi^*)$ if and only if

$$(t_i - t_{i-1})\left(\frac{k}{\alpha_1 Z_1 + k}\right)^2 + (t_{i+1} - t_i)\left(\frac{k}{\alpha_2 Z_1 + k}\right)^2 \geq (t_{i+1} - t_{i-1})\left(\frac{k}{\alpha_3 Z_1 + k}\right)^2 \tag{4.21}$$

It can be shown that this inequality always holds, and that it becomes an equality if and only if $\alpha_1 = \alpha_2 = \alpha_3$, i.e., if $\phi = \phi^*$.

By repeatedly applying the result above to ϕ we obtain $n_G(\phi) \geq n_G(\phi')$, and $n_G(\phi) = n_G(\phi')$ only if $\phi = \phi'$. ■

Now we can proceed and prove the theorem:

Proof of Theorem 4.5: For any $\varepsilon > 0$, we divide the interval $[0, 1]$ into segments $[0, \varepsilon, 2\varepsilon, \ldots, 1]$ and approximate $\phi(t)$ with a polygonal sampling function $\phi_\varepsilon(t)$ such that $\phi(k\varepsilon) = \phi_\varepsilon(k\varepsilon)$ for all k, and $\phi_\varepsilon(t)$ is linear in each segment $[k\varepsilon, (k+1)\varepsilon]$. From Lemma 4.6 it follows that $n_G(\phi_\varepsilon) > n_G(\phi')$. Since this is true for all $\varepsilon > 0$, and since $\phi(t)$ is continuous, it follows that $n_G(\phi) > n_G(\phi')$. ■

Combining Theorem 4.5 with Result 4.4, we obtain:

Result 4.7 *For a camera moving sideways on a straight line, the sampling function with the minimal perspectivity distortion is $\phi_{opt}(t) = x_{min} + t(x_{max} - x_{min})$. This linear sampling function starts with the leftmost column of the first image and finishes with the rightmost column of the last image.*

4.3.6 Perspectivity: Non-Linear Trajectory

In order to handle non-linear camera trajectories, I define the local perspectivity (Equation 4.18) based on a local linear approximation of the camera trajectory and a local planar approximation of the manifold. For each frame I_f, where $1 \leq f \leq N-K+1$, we compute a discrete version $d_L(\phi, f)$ of the local perspectivity (equation 4.18) over a set of K neighboring frames $I_f, \ldots I_{f+K-1}$ and minimize the sum of the discrete local perspectivities:

$$d_G(\phi) = \sum_f d_L(\phi, f) \tag{4.22}$$

To find the minimum of (4.22), we discretize the strip locations. Note that the local perspectivity $d_L(\phi, f)$ is defined by finding an optimal center of projection for each combination of rays. Computing these centers of projections for all possible sampling functions and for a large K is computationally intractable. This can be circumvented by selecting $K = 2$, in which case the local distortion $d_L(\phi, f)$ was derived analytically, as it corresponds to the linear perspectivity as defined in theorem 4.3. Once $d_L(\phi, f)$ is computed for all pairs of views, I use belief propagation [53] to find the optimum of equation 4.22. The complexity of this algorithm is linear in the number of frames, and quadratic in the number of possible strip locations in each frame.

4.3.7 Results

Figure 4.23 shows mosaicing results, using different sampling functions, from video sequences captured by a camera moving on a linear trajectory. I compare the optimal sampling function with the constant sampling function (linear pushbroom mosaicing), and with a non-linear monotonic sampling function $\tilde{\phi}(t)$ satisfying $\tilde{\phi}(0) = \phi_{opt}(0)$ and $\tilde{\phi}(1) = \phi_{opt}(1)$. This demonstrates two main results of this work. First, among all linear sampling functions $\phi(t) = \alpha t + \beta$, the least distorted results are achieved with the maximal α (compare Figure 4.23b vs. 4.23c). Second, among all monotonic functions aligned at the edge points $t = 0, 1$, the optimal sampling function is the linear one (compare Figure 4.23b vs. 4.23d).

Figure 4.24 compares a stereo mosaic generated by a constant sampling function (as done by [29,

(a1) (a2)

(b)

(c)

(d)

Figure 4.23: A comparison between different strip sampling methods. (a1) and (a2) are two rectified input images. Mosaic (b) was generated by the optimal sampling function, mosaic (c) by the constant sampling function (pushbroom mosaic) and mosaic (d) by a non-linear sampling function $\phi(t) = \sqrt{t}$.

54, 61, 77]) to one generated by the optimal linear sampling function. Note that in addition to the distortions in the image, there is a distortions in the disparity which is larger with the constant sampling.

As for non-linear camera trajectories, I computed the least-distorted mosaics for various camera trajectories, some examples of which are shown in Figure 4.25. In all cases I tested, the least distorted mosaic was obtained when the projection rays intersect in a line (i.e., a X-Slits image).

One practical case of a non-linear trajectory is when the camera moves on a circular arc. I examined visually the differences between the least-distorted mosaic and mosaics generated by constant sampling functions [54]. Various constant sampling functions were compared, each with a strip taken from a

(a1)

(a2)

(b)

(c)

Figure 4.24: A comparison between different strip sampling methods for stereo mosaics. The images should be viewed in full color using anaglyphic 3D glasses. (a1) and (a2) are two rectified input images. Mosaic (b) was generated by the optimal sampling function, mosaic (c) by the constant sampling function (pushbroom mosaic).

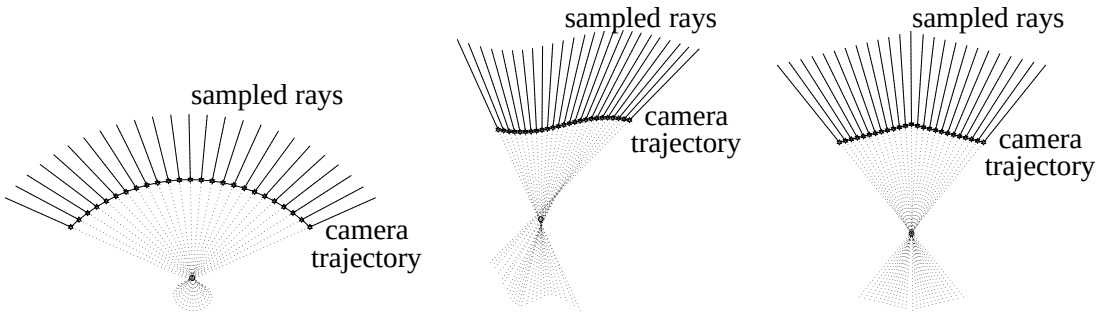


Figure 4.25: Generating least-distorted mosaics with non-linear camera trajectories. The illustrations show a top view of the camera trajectories and the planes of sampled rays, as computed by a numerical discrete optimization. In all cases, the optimal sampling is obtained when the sampled rays intersect in a line.

different offset from the center. The least distorted mosaic in this case is a Crossed-Slits mosaic, as shown in Figure 4.25a. I found that the differences in distortions with circular camera motion are not as significant as with linear camera motion, as the rays in this case are bundled together to begin with. Furthermore, in the case of non-linear trajectory, the distortion is also affected by the fact that the manifold is non-planar; this kind of distortion, which (unlike perspectivity distortion) can be treated with 2D warping, has not been discussed in this work.

4.4 Summary

I have presented techniques for efficiently rendering X-Slits images from image data acquired by a pinhole camera. The main application I pursued is view synthesis, or image-based rendering. View synthesis with the X-Slits camera is greatly simplified as compared with perspective view synthesis, since it is performed by non-stationary mosaicing, or by slicing the space-time volume. The X-Slits theory helps the user to "drive" the slicing process in order to get the desired effect. When compared with traditional mosaicing, X-Slits images can be shown to be closer to perspective images than linear pushbroom images.

Using my method one can also generate new images taken from "impossible" positions, like behind the back wall of a room or in front of a glass barrier. Movies with new egomotion can also be generated,

such as forward-moving movies from a side-moving input sequence. Although not perspective, the movies generated in this way appear compelling and realistic.

Given an input sequence of images taken by a panoramic camera rotating off-axis, one can generate omnidirectional views of the scene from different viewpoints by simple mosaicing under the circular X-Slits projection. I have shown how data should be represented in order to perform this task efficiently. I have analyzed the distortions present in such images and described a method for reducing them by using a coarse approximation of the scene structure. The result is a method that combines image-based rendering by ray sampling with approximating a perspective view using a coarse 3D model.

Augmenting objects into the image-based rendered scene requires that the objects obey the same geometric model as the background. I have shown how the location of the augmented objects can be shifted, so that they would appear veridical when rendered by a perspective engine, especially when viewed from different viewpoints.

Considering the general case of multi-perspective mosaicing of an unknown scene, I have developed a framework for quantifying distortion and a closed-form solution for the problem of generating the least distorted mosaic. When the camera moves on a linear trajectory, the least distorted mosaic is generated by the linear sampling function with the maximal slope. This mosaic also has the largest possible field of view. When the camera trajectory is not linear, the least-distorted mosaic can be derived numerically. I found that the distortions are especially significant when camera trajectory is close to linear.

Chapter 5

Motion Segmentation and Depth Ordering

In this chapter I present a method for motion segmentation and from a video sequence in general motion. This method is based on a spatio-temporal differential operator that responds to occlusion, introduced in Section 5.1. I develop a multi-scale method for extracting a motion boundary, yielding the segmentation. The behavior of this operator is demonstrated in Section 5.2 and analyzed in Section 5.3.

Based on the scale space behavior of this detector, I devise in Section 5.4 a novel algorithm that can determine depth ordering from just two frames. This algorithm assumes there is an average intensity difference between the layers (though not necessarily a local difference or a visible edge), and can be adjusted to work even without this assumption on three frames. In Section 5.5 I describe human experiments that show that people are also capable of determining depth ordering from just two frames under the same conditions as my algorithm.

5.1 Segmentation Algorithm

The motion segmentation algorithm I present is based on a differential operator defined in Section 5.1.1 that is applied to the video sequence and responds at motion boundaries. While this operator is shown

to detect motion boundaries in many cases, it is often unable to detect boundaries where certain degeneracies exist locally. This is solved by a cross-scale scheme presented in Section 5.1.2. Finally, closed contours are extracted using a saliency measure and a simple heuristic to overcome small gaps, presented in Section 5.1.3. See also Appendix C.2 for some implementation issues.

5.1.1 Occlusion Detector

Regarding the video sequence as a spatio-temporal intensity function, let $I(x, y, t)$ denote the intensity at pixel (x, y) in frame t. I refer to the average of the second moment matrix over a neighborhood ω around a pixel as the *Gradient Structure Tensor*

$$\mathbf{G}(x, y, t) \equiv \sum_{\omega} \nabla I \, (\nabla I)^T = \sum_{\omega} \begin{bmatrix} I_x^2 & I_x I_y & I_x I_t \\ I_x I_y & I_y^2 & I_y I_t \\ I_x I_t & I_y I_t & I_t^2 \end{bmatrix} \tag{5.1}$$

This matrix has been invoked before in the analysis of local structure properties. In [35], eigenvalues of $\mathbf{G}$ were used for detecting spatio-temporal interest points. In [42] it was suggested that the eigenvalues of $\mathbf{G}$ can indicate spatio-temporal properties of the video sequence and can be used for motion segmentation. The idea behind this is reminiscent of the Harris corner detector [24], as it detects 3D "corners" and "edges" in the spatio-temporal domain. Here I take a closer look and develop this idea into a motion segmentation algorithm.

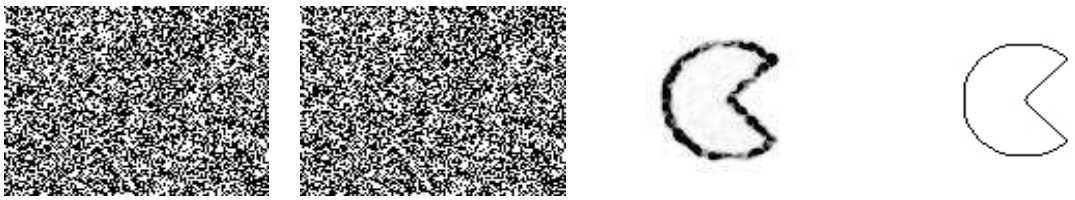
Specifically, if the optical flow in ω is (v_x, v_y) and the brightness constancy assumption [27] holds, then

$$\mathbf{G} \cdot (v_x, v_y, 1)^T = 0 \tag{5.2}$$

Hence, 0 is an eigenvalue of $\mathbf{G}$. Since $\mathbf{G}$ is positive-semidefinite, we can use the smallest eigenvalue of $\mathbf{G}$ as a measure of deviation from the assumptions above, which leads to the following definition:

Definition 5.1 *Let $\lambda(x, y, t)$ denote the smallest eigenvalue of the* Gradient Structure Tensor $\mathbf{G}(x, y, t)$. *The operator λ is the* occlusion detector.[1]

[1] Note that the values of λ at each pixel can be evaluated directly using Cardano's formula.



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

Figure 5.1: Random dots example. A shape is moving sideways, where both the shape and the background are covered by a random pattern of black and white dots. It is impossible to identify the moving object from each of the two frames (a) and (b) (a stereo pair) alone. The occlusion detector (c) (higher values of λ are darker) shows the outline of the object very clearly. Compare with the ground truth (d).

We do not normalize λ with respect to the other eigenvalues of $\mathbf{G}$ (as in [42]), since it may amplify noise.

In order to provide rotational symmetry and avoid aliasing due to the summation over the neighborhood ω, I define ω to denote a Gaussian window, and the operation $\sum_\omega$ in (5.1) stands for the convolution with a Gaussian. Since I do not assume temporal coherence of motion, the Gaussian window is restricted to the spatial domain.

Figure 5.1 demonstrates the detector results on a simple synthetic example. In this example there are no intensity or texture cues to indicate the boundaries of the moving object, and it can only be detected using motion cues. The value of λ, shown in Fig. 5.1c, is low in regions of smooth motion, and high values of λ describe the boundary of the moving object accurately.

The values of ∇I, and hence of λ, are invariant to translation transformations on I. Additionally, for any rotation matrix $\mathbf{R}$,

$$|\lambda\mathbf{I} - \mathbf{G}| = |\mathbf{R}(\lambda\mathbf{I} - \mathbf{G})\mathbf{R}^T| = \left|\lambda\mathbf{I} - \sum_\omega (\mathbf{R}\nabla I)(\mathbf{R}\nabla I)^T\right|$$

($\mathbf{I}$ is the identity matrix) and therefore the values of λ are also invariant to the rotation of I. The issue of scale invariance is discussed in Appendix C.1.

Velocity-Adapted Detector

While rotational invariance is desirable in the spatial domain, non-spatial rotations in the spatio-temporal domain have no physical meaning. It is preferable to have invariance to spatially-fixed shear transformations, which correspond to 2D relative translational motion between the camera and the scene. As suggested in [36] by the reference to *Galilean diagonalization*, one can use the velocity-adapted matrix $\tilde{\mathbf{G}}$ given by

$$\tilde{\mathbf{G}} = \begin{bmatrix} G_{11} & G_{12} & 0 \\ G_{21} & G_{22} & 0 \\ 0 & 0 & \lambda_T \end{bmatrix} \quad \text{where} \quad \lambda_T = \frac{\det(\mathbf{G})}{\det(\mathbf{G}^*)} \tag{5.3}$$

(G_{ij} denote the entries of $\mathbf{G}$, and $\mathbf{G}^*$ denotes the 2×2 upper-left submatrix of $\mathbf{G}$ containing only spatial information).

Definition 5.2 *The operator λ_T is the* velocity-adapted occlusion detector.

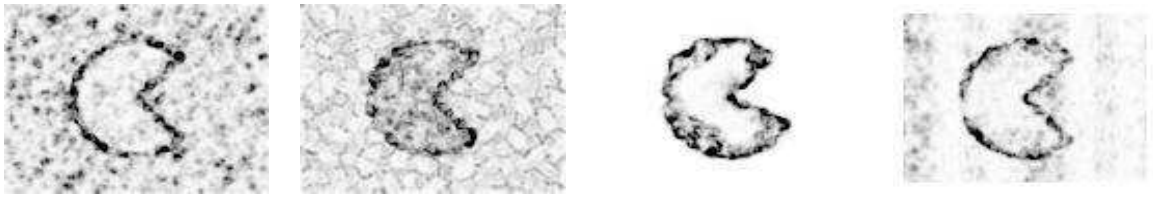
To justify this definition, observe that $\tilde{\mathbf{G}}$ is also invariant to translation and spatial rotation. The entry λ_T is an eigenvalue of $\tilde{\mathbf{G}}$, and it has been suggested that it encodes the temporal variation, being the "residue" unexplained by pure-spatial information.

In practice, λ_T gives results similar to λ, though it has certain advantages, as discussed in Section 5.3. Throughout this chapter I use λ to denote either operator, unless stated otherwise.

Detector Effectiveness

High values of λ indicate significant deviation from (5.2), which is often due to the existence of a motion boundary. Other sources of large deviations include changes in illumination (violation of the brightness constancy assumption), or when the motion varies spatially (motion is not constant in ω). However, often these events lead to smaller λ values as compared with motion boundaries (see Fig. 5.2), in which case the boundary response can be distinguished from a false response (e.g., by thresholding).

Low values of λ do not necessarily indicate that the motion in ω is uniform. The rank of $\mathbf{G}$ is affected by spatial structure as well as temporal structure, so λ may be low even at motion boundaries,



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

Figure 5.2: False λ response. The same example as in Fig. 5.1: (a) with 20% white noise; (b) with illumination change of 5%; (c) with the object rotating by 20°; (d) with both object and background patterns deformed smoothly.

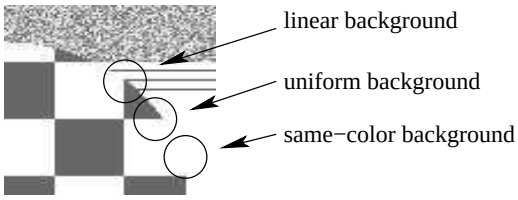


Figure 5.3: Areas where the λ detector is likely to give low values despite the existence of a local motion boundary.

when certain spatial degeneracies exist. Specifically, this occurs when there is local ambiguity, i.e., when the existence of a motion boundary cannot be determined locally. This includes areas where the occluding object and its background are of the same color, areas where the background is uniform in color, and areas where the background texture is uniform in the direction of the motion (Fig. 5.3). In the first case the rank of $\mathbf{G}$ is 0, and in the other cases the rank of $\mathbf{G}$ may be 1 or 2, depending on the appearance of the occluding object (recall that the λ detector is high when the rank of $\mathbf{G}$ is 3). In these cases, the background may be interpreted as part of the moving object, since no features in the background appear to vanish due to occlusion.

5.1.2 Extraction of Motion Boundaries and Scale Space Structure

The response of λ to occlusion occurs only where some background features become occluded. Clearly boundary location cannot always be inferred on the basis of local information alone. However, while there may be no cues to indicate the location of the boundary at a fine scale, there may be enough information at a coarser scale (i.e., in a larger neighborhood) and λ may respond. Thus a multi-scale element is incorporated in the algorithm, in order to detect motion boundaries that are not detectable at

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

Figure 5.4: Checkerboard example: (a) A frame from the sequence; (b) and (c) show the response of λ at fine ($s_{xy} = 1$) and coarse ($s_{xy} = 10$) scales respectively. At the fine scale, λ only responds at intensity edges (which appear as discrete "bursts"), while the entire contour is visible at the coarse scale, though with considerable distortion. (d) shows the final contour selected by integrating over scales.

fine scales.

Defining scale

In order to define the notion of scale in my algorithm, note that the evaluation of λ involves Gaussian convolutions in two different stages – during the estimation of the partial derivatives, and when taking the average over the neighborhood ω. In both cases, larger Gaussians lead to coarser structures, and I refer to the size of the Gaussian as the *scale*. In this work I only consider the spatial scale. As shown in Appendix C.1, these two scales are related, and I define a unified scale dimension, and a scaling-invariant operator $\lambda^{(s)}$ at any scale $s > 0$, using scale-normalization.

The notion of scale has been studied extensively for features such as edges and blobs. As with these features, different structures can be found at different scales. The response of λ to noise, which can occur in finer scales, is suppressed in coarser scales. On the other hand, localization is poor at coarse scales and motion boundaries may break and merge.

Figure 5.4 illustrates this idea – at fine scale (Fig. 5.4b), λ responds only at discrete locations, because the background consists of regions with constant color, and the occlusion can only be detected where there are color variations in the background. In the coarser scale (Fig. 5.4c), the neighborhood of every boundary point contains gradients in several directions and the boundary is detected continuously.

Image features, such as edges, typically shift and become distorted at coarse scales. The scale space structure of motion boundary edges (and in particular my occlusion detector) has its own par-

ticular biases in coarse scales. As discussed in Section 5.3, motion boundaries at coarse scales are shifted towards the occluded side, i.e., the occluding objects becomes "thicker". In addition, it can be shown that the bias is stronger when there is a large intensity difference between the object and the background, and it increases with scale.

Estimating derivatives in the temporal domain is prone to aliasing. See Appendix C.2 for implementation details, including elimination of aliasing and estimation from only two frames.

Boundary Extraction in Scale Space

Since λ is computed by taking the average over a neighborhood, its response is diffuse. We want to extract a ridge curve where λ is strongest. This can be defined locally as points where λ is maximal in the direction of the maximal principal curvature, which can be expressed as

$$\begin{cases} \lambda_{xy}(\lambda_x^2 - \lambda_y^2) - \lambda_x\lambda_y(\lambda_{xx} - \lambda_{yy}) &= 0 \\ (\lambda_{xx} + \lambda_{yy}) \cdot \Big((\lambda_{xx} - \lambda_{yy})(\lambda_x^2 - \lambda_y^2) + 4\lambda_x\lambda_y\lambda_{xy}\Big) &< 0 \\ \lambda_x^2\lambda_{yy} - 2\lambda_x\lambda_y\lambda_{xy} + \lambda_y^2\lambda_{xx} &< 0 \end{cases} \tag{5.4}$$

Thus, at every scale s, the values of λ and its derivatives are computed, and the ridge can be extracted. For reasons of numerical stability, the derivatives of $\lambda^{(s)}$ are computed with the same Gaussian smoothing s used for computing $\lambda^{(s)}$, at each scale.

Different boundaries are extracted at different scales, as fine-scale boundaries may often split because of the absence of local information, and coarse-scale boundaries may disappear or merge. Since these may occur at different parts of the image at different scales, we need to construct a scale-adapted boundary, by selecting different scales for different localities (as in [38]). Considering the multi-scale boundary surface as the union of all ridges in $\lambda^{(s)}$ for $s \in (0, \infty)$, we want to find a cross-scale boundary where $\lambda^{(s)}$ is maximal. This can be expressed as

$$\begin{cases} \lambda_s &= 0 \\ \lambda_{ss} &< 0 \end{cases} \tag{5.5}$$

using the scale-derivatives of λ.

(a) (b)

Figure 5.5: Saliency measure. (a) All boundaries extracted from the random dots example with illumination changes (Fig. 5.2b); intensity codes λ response. (b) The most salient closed contour.

Combining (5.4) and (5.5) defines the final *cross-scale motion boundary*. It is a curve in the three-dimensional space $X-Y-S$, defined by the intersection of the two surfaces defined respectively by these 2 sets of equations.

5.1.3 Boundary Completion

As stated above, λ also has some false responses which lead to the selection of false boundary fragments. It is therefore necessary to define a saliency criterion, which is used to select the most interesting boundaries. Since we regard λ as a measure of local boundary strength, for each connected set of boundary points I define the *saliency measure* to be the sum of the value of λ along the boundary, as in [38]. This measure may be sensitive to fragmentation of the boundary, so in my implementation small gaps are tolerated.

Finally, segmentation is achieved by searching for closed contours with high saliency and small gaps. I employ a simple greedy heuristic to connect the motion boundary fragments into a continuous boundary with maximal saliency and minimal gaps. Since the extracted boundaries are usually almost complete, this heuristic gives good results (see Fig. 5.5).

5.2 Experimental Results

In my experiments I compared my algorithm with the most prominent motion segmentation approaches, wherever code was available. To begin with, I establish the baseline result by segmenting the optical flow. Such a segmentation lies at the heart of some more elaborate segmentation methods, such as [50].

I used a robust and reliable implementation of the Lucas-Kanade algorithm [40], and segmented it using a variety of edge operators, including Canny and various anisotropic diffusion methods and clustering methods (e.g., [73]), presenting the best results for each example.

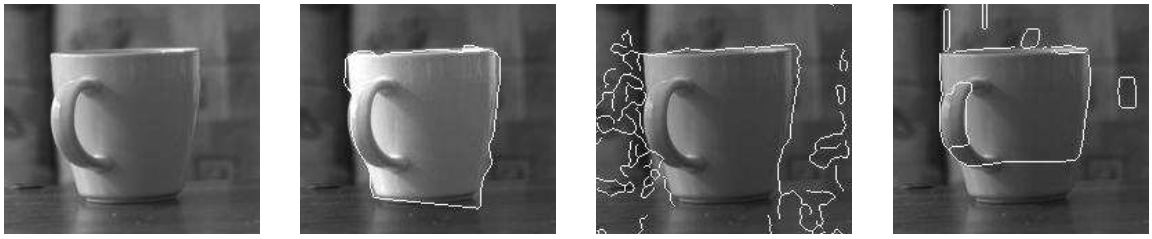
One influential motion segmentation approach relies on graph cuts [34] (and is therefore related to the more traditional regularization based approaches [43]). Code for two variants of this approach is available on the web by the respective authors [34, 69], and I could therefore use their code to establish credible comparisons. Note, however, that in both cases the publicly available code can only work with rectified images. Therefore, in order to obtain fair comparisons, I compared my results with the results of these algorithms only with rectified image pairs, when possible.

Figure 5.6 demonstrates my algorithm on a stereo pair. The most salient motion boundary is shown in Fig. 5.6b superimposed on the first input image. Fig. 5.6c illustrates the baseline result - the edges of the optical flow. Fig. 5.6d illustrates the best MRF-based segmentation using graph cuts [69]. More results are shown in Fig. 5.7.

Figure 5.8 shows my algorithm's performance on a video sequence with a dynamic scene, featuring non-rigid motion and illumination changes. The octopus and the reef below have similar color and texture, and thus spatial coherence is unreliable (note in particular the triangle-shaped projection near the octopus' head, which is in fact a background feature).

In Fig. 5.9, a large amount of noise was added to the synthetic checkerboard sequence, causing numerous optical flow estimation errors. The magnitude of the flow estimation error is often greater than the true flow (Fig. 5.9b), particularly around the centers of the squares, making segmentation based directly on the optical flow impossible. Results of my algorithm and MRF-based method are also shown.

The main weakness of many MRF-based methods is their reliance on spatial coherence, which leads to failure when no spatial edge coincides with the motion edge. This is demonstrated on the random dots example in Fig. 5.10a,b where such methods have no spatial support and therefore fail. Fig. 5.10c,d demonstrates my algorithm's advantage when no global motion model can be assumed. In this example, the texture of both the moving object and the background undergo smooth non-linear



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

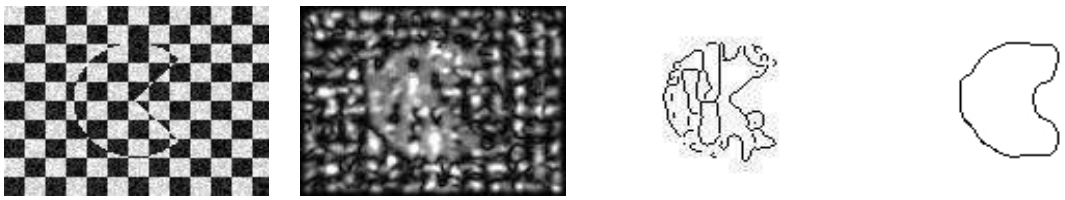
Figure 5.6: Cup example. (a) The left image of a stereo pair. (b) Most salient edge detected by my algorithm (with the area of the segment highlighted). (c) Edges in the horizontal component of the optical flow. (d) Edges from a graph cuts segmentation algorithm [34].

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

Figure 5.7: Flower example. (a) The left image of a stereo pair. (b) Most salient closed contour detected by my algorithm (with the area of the segment highlighted). (c) Edges in the optical flow. (d) Edges from a graph cuts segmentation algorithm [34].

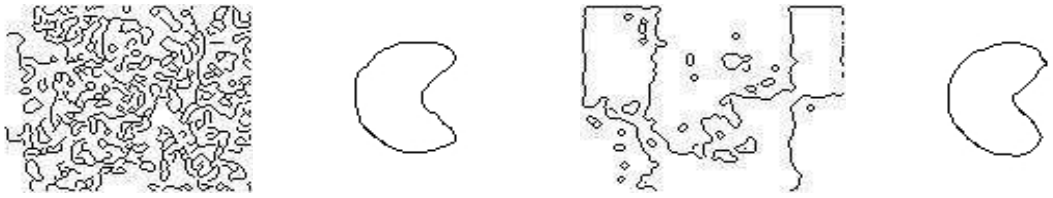
(a) (b) (c)

Figure 5.8: Octopus example. (a) A frame from the sequence. (b) The most salient closed contour detected by my algorithm (with the area of the segment highlighted). (c) Edges in the optical flow.



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

Figure 5.9: Checkerboard example with 25% white noise. (a) One of the frames; (b) Lucas-Kanade optical flow magnitude; (c) Segmentation using graph cuts; (d) The most salient contour found by my algorithm.



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

Figure 5.10: Random dots example (see Fig. 5.1). With 20% white noise: (a) Segmentation using graph cuts; (b) The most salient contour found by my algorithm. With smooth non-linear deformation: (c) Segmentation assuming affine motion using an implementation of [73]; (d) The most salient contour found by my algorithm.

deformation. The results of applying [73] show that when motion varies smoothly within an object, global model methods fail.

Figure 5.11 demonstrates how my algorithm works with very slow motion. As long as there are features in the background that become occluded, my algorithm can detect the motion boundary even at sub-pixel motion. Figure 5.11a shows results for a sequence where the foreground object moves by $1/2$ pixel. All MRF-based algorithms I applied failed to detect the foreground object altogether. Although the velocity in Fig. 5.11a is 8 times slower than in Fig. 5.11b, the values of λ in both cases are similar.

(a) (b)

Figure 5.11: Results on a random dots example with small motion of $1/2$ pixel per frame (a), and with larger motion of 4 pixels per frame (b).

5.3 Analysis

In order to analyze the performance of the proposed technique, consider a video of two moving layers l^1, l^2, where w.l.o.g. l^2 partially occludes l^1. A frame in the video sequence can be written as

$$I = l^1 \cdot (1 - \alpha) + l^2 \cdot \alpha \tag{5.6}$$

where α is the *matting map*. Assume w.l.o.g. that the occlusion edge is perpendicular to the X axis and that at frame $t = 0$ it is at $x = 0$. Further assume that the occlusion edge is a Gaussian-smoothed line, so α is of the form $\alpha_{s_0}(x) = \int_{-\infty}^{x} g_{s_0}(u)du$ (I denote the Gaussian function with variance s as g_s).

If the motions of l^1 and l^2 are (v_x^1, v_y^1) and (v_x^2, v_y^2) respectively, then the video volume is given by

$$I(x,y,t) \;=\; l^1(x-v_x^1 t, y-v_y^1 t)\cdot(1-\alpha(x-v_x^2 t)) + l^2(x-v_x^2 t, y-v_y^2 t)\cdot\alpha(x-v_x^2 t) \tag{5.7}$$

Note that the motion of α is the same as the motion of l^2, since it is the occluding layer.

Denoting the video volume of each layer as $I^k(x,y,t) = l^k(x - v_x^k t, y - v_y^k t)$, the gradient of the video volume is given by

$$\nabla I \;=\; (1-\alpha)\cdot\nabla I^1 + \alpha\cdot\nabla I^2 + (I^2 - I^1)\cdot g_{s_0}\cdot\mathbf{n} \tag{5.8}$$

where $\mathbf{n} = (1, 0, -v_x^2)^T$. Note that $\mathbf{n}$ is perpendicular in space-time to the occlusion edge $(0,1,0)^T$ and to the motion vector $\mathbf{v}^2 = (v_x^2, v_y^2, 1)^T$; i.e., $\mathbf{n}$ is the normal to the plane in the video space formed by the motion of the occlusion edge.

Therefore, ∇I is composed of the matting of ∇I^1, ∇I^2, and a component that depends on $I^2 - I^1$. Note that ∇I^1 is perpendicular to $\mathbf{v}^1$, while both ∇I^2 and $\mathbf{n}$ are perpendicular to $\mathbf{v}^2$. This means that ∇I is composed of two components that are related to the occluding layer and only one that is related to the occluded layer.

For scale space analysis I use the approximation

$$g * (f \cdot \alpha) \approx (g * f) \cdot (g * \alpha) \tag{5.9}$$

where g is a Gaussian function and α is an integral of a Gaussian as defined above. Eq. (5.9) is an equality when f is constant, and it provides a good approximation when f does not change rapidly near $x = 0$ (in each layer separately).

Applying (5.9), the gradient estimated at scale s, denoted by $\nabla I^{(s)} = \nabla(g_s * I)$, is

$$\nabla I^{(s)} \approx (1-\alpha_{s_0+s}) \cdot \nabla I^{1(s)} + \alpha_{s_0+s} \cdot \nabla I^{2(s)} + (I^{2(s)} - I^{1(s)}) \cdot g_{s_0+s} \cdot \mathbf{n} \tag{5.10}$$

5.3.1 Velocity-Adapted Occlusion Detector λ_T

I assume the 2D gradients in each layer are distributed isotropically, in the sense that the mean gradient is 0. Furthermore, I assume that they are uncorrelated. Thus, using (5.8) and (5.9), we can write the gradient structure tensor defined in (5.1) as

$$\begin{aligned} \mathbf{G}^{(s)} &\approx g_{s_\omega} * \left((1-\alpha_{s_0+s})^2 \nabla I^1 (\nabla I^1)^T + \alpha_{s_0+s}^2 \nabla I^2 (\nabla I^2)^T + (I^2 - I^1)^2 \cdot g_{s_0+s}^2 \cdot \mathbf{n}\mathbf{n}^T \right) \\ &\approx h_1 \cdot \mathbf{M}^1 + h_2 \cdot \mathbf{M}^2 + h_3 \cdot \mathbf{n}\mathbf{n}^T \end{aligned} \tag{5.11}$$

where

$$\mathbf{M}^k \equiv \begin{bmatrix} 1 & 0 & -v_x^k \\ 0 & 1 & -v_y^k \\ -v_x^k & -v_y^k & (v_x^k)^2 + (v_y^k)^2 \end{bmatrix} \quad \text{and} \quad \begin{aligned} h_1 &= c_1 \cdot (1 - \alpha_{s_0+s+s_\omega})^2 \\ h_2 &= c_2 \cdot \alpha_{s_0+s+s_\omega}^2 \\ h_3 &= c \cdot g_{s_\omega + (s_0+s)/2} \end{aligned} \tag{5.12}$$

The constants $c_k = \left\langle \|\nabla l^k\|^2 \right\rangle / 2$ and $c = \left\langle (l^2 - l^1)^2 \right\rangle / \sqrt{4\pi(s + s_0)}$ describe the distribution of intensities in the layers.

Then, the velocity-adapted occlusion detector from (5.3) can be shown to be

$$\lambda_T \quad = \quad \frac{(v_x^1 - v_x^2)^2}{1/h_1 + 1/(h_2 + h_3)} + \frac{(v_y^1 - v_y^2)^2}{1/h_1 + 1/h_2} \tag{5.13}$$

In the general case, the expression above is hard to analyze. Sampling shows that λ_T typically has a single local maximum. Although it may have two local maxima, this only happens when $c_2 > 9 \cdot c_1$ and $c > 180 \cdot c_1$ for $s \geq 1$, and the second local maximum is usually very subtle. Therefore, for all practical purposes, it can be assumed that λ_T has a single local maximum.

Furthermore, we can draw the following conclusions:

- In the special case where $c_1 = c_2$ (i.e., both layers have the same intensity variance) and $c \to 0$ (i.e., both layers have similar intensities), λ_T is maximal at $x = 0$.
- In the limit $c \to 0$, λ_T is maximal when $\alpha(x) = \sqrt[3]{c_1}/(\sqrt[3]{c_1} + \sqrt[3]{c_2})$, which means that the detected edge location is biased towards the layer with lower intensity variance. The magnitude of the bias is proportional to $\sqrt{s + s_0 + s_\omega}$.
- If only $c_1 = c_2$ is assumed, then $\frac{d\lambda_T}{dx}(x = 0) < 0$, therefore λ_T is maximal at a negative x, which means that the detected edge location is biased towards the occluded layer.

5.3.2 Occlusion Detector λ

Behavior analysis of the smallest eigenvalue λ is harder. Thus I make the further assumption that $l^1 = l^2$ along the edge. Then we can omit the last term in (5.11) and get

$$\mathbf{G} \quad = \quad c_1(1-\alpha)^2\mathbf{M}^1 + c_2\alpha^2\mathbf{M}^2 \tag{5.14}$$

Calculating the eigenvalue of (5.14), the following can be shown:

- The smallest eigenvalue of $\mathbf{G}$ is given by

$$\lambda = \frac{1}{2}\left(a - \sqrt{a^2 - 4b}\right) \quad \text{where} \quad \begin{aligned} a &= (1-\alpha)^2 c_1 \|\mathbf{v}^1\|^2 + \alpha^2 c_2 \|\mathbf{v}^2\|^2 \\ b &= (1-\alpha)^2 \alpha^2 c_1 c_2 \|\mathbf{v}^1 - \mathbf{v}^2\|^2 \end{aligned} \tag{5.15}$$

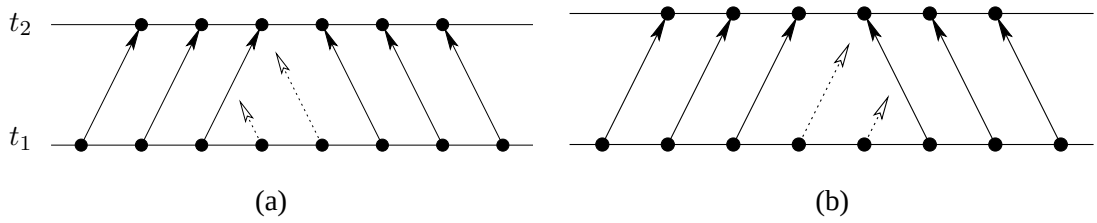


Figure 5.12: Two-frame occlusion problem. Two of the pixels in frame t_1 do not correspond to any pixel in t_2 due to occlusion, and they may belong either to the right (a) or the left (b) layer.

- λ has a single local maximum.
- If $c_1\|\mathbf{v}^1\|^2 = c_2\|\mathbf{v}^2\|^2$, then λ is maximal at $x = 0$ – where the edge is located.
- If $c_1\|\mathbf{v}^1\|^2 > c_2\|\mathbf{v}^2\|^2$, then λ is maximal at some $x > 0$, and vice-versa; in other words, the detected edge location is biased towards the layer with lower intensity variance and smaller absolute motion.

The biasing effect towards the occluded layer is not evident due to the particular assumptions I have made, although it was observed in my experiments. Note that λ is affected by absolute velocity, unlike the velocity-adapted operator λ_T.

5.4 Depth Ordering

I now present two algorithms for determining ordinal depth based on the occlusion detector defined in Section 5.1.1, using either two frames or three frames.

When only two frames are available, it is impossible to infer the order of depth from motion alone, without additional assumptions or prior knowledge. Consider a pair of images of a video sequence (or a stereo pair) that contain the motion of two layers where one partially occludes the other. As illustrated in Fig. 5.12, pixels that appear in one frame and become occluded in the other may belong to either of the layers. Whichever layer they belong to is the occluded layer, and since their interframe correspondence cannot be determined, both interpretations are equally valid. The two-frame algorithm, described in Section 5.4.1, is based on the assumption that there is a (possibly small) difference of

intensity between the layers on average.

The situation when more than two frames are available is considerably different. While there may be two interpretations to a two-frame sequence, additional frames can be used to rule out false interpretations. With a slight modification, my algorithm can be applied to three frames even when the two layers have the same intensity on average, and achieve better localization, see Section 5.4.2.

5.4.1 Two-Frame Algorithm

Given the scenario described above and generalizing (5.8), the space-time gradient of I is given by

$$\nabla I \quad = \quad \nabla I^1 \cdot (1-\alpha) + \nabla I^2 \cdot \alpha + (I^2 - I^1)\nabla\alpha \tag{5.16}$$

Observe that the expression above is a sum of three vectors – two of them proportional to the gradients of the two layers, and a third component that stems from the edge between the layers. Since the edge and the occluding layer have the same motion (or *common fate*), the gradient of I is more affected by the motion of the occluding layer than that of the occluded layer in areas of transition between layers. This asymmetry is manifested in a *bias* towards the occluded layer in the location of the detected motion boundary, as derived from (5.13).

First note that this bias typically grows with scale. This is because the components representing the gradients of each layer are smoothed across the motion boundary into the other layer, and the component that is due to the difference between the layers is smoothed in both directions. Therefore, the effect of the motion of the occluding layer expands farther into the occluded layer as I is further smoothed.

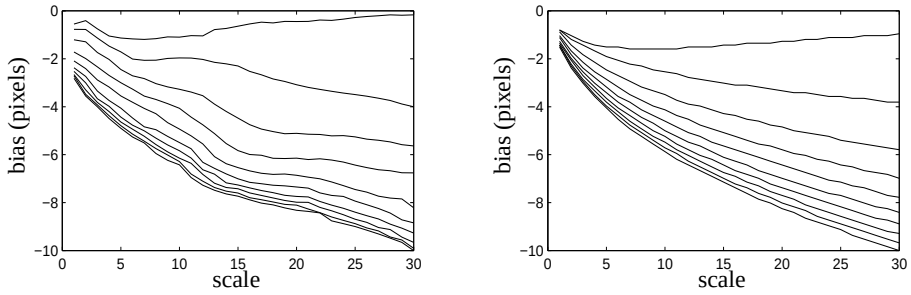
More specifically, consider the spatial scaling of a video I by σ, namely

$$J(x,y,t) = I(x/\sigma, y/\sigma, t) \tag{5.17}$$

Due to scaling invariance (Eq. C.4 in Appendix C),

$$\lambda_J^{(\sigma^2 s)}(x,y,t) = \lambda_I^{(s)}(x/\sigma, y/\sigma, t) \tag{5.18}$$

Thus, if at scale s_1 the maximum of $\lambda_I^{(s_1)}$ is obtained at some $x < 0$, then at scale s_2 the maximum of $\lambda_J^{(s_2)}$ would be obtained at $\sqrt{s_2/s_1} \cdot x$ when J is a scaling of I by s_2/s_1. If the values of c_1, c_2, c



(a) (b)

Figure 5.13: (a) The bias of the location of maximal $\lambda^{(s)}$ as a function of s (scale) on a synthetic random-dot pair. Each curve represents a different value of $\left\langle (I^2 - I^1)^2 \right\rangle$ ranging from 0 (top) to 0.2 (bottom). (b) The bias as predicted based on (5.13).

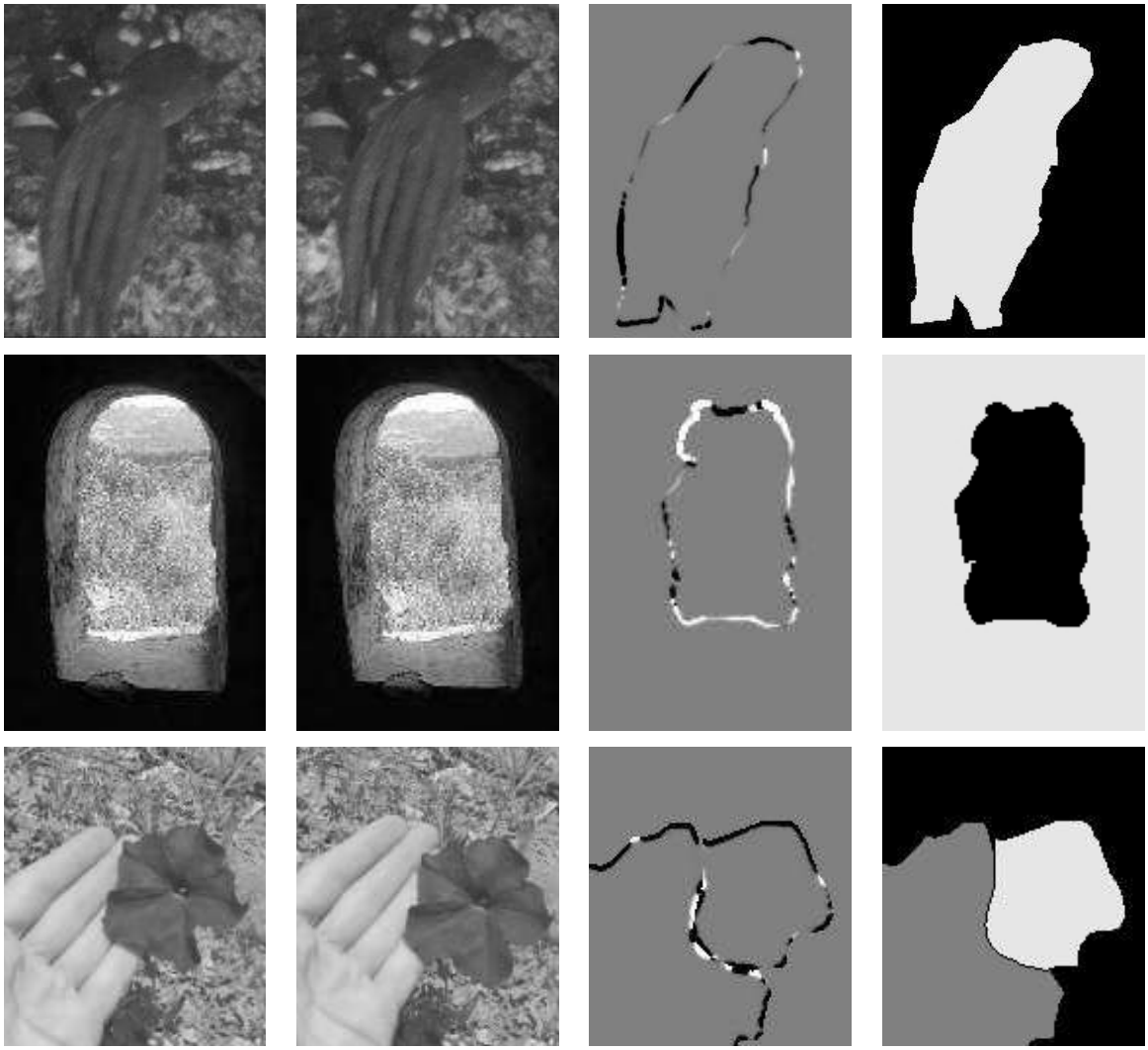
(defined in (5.12)) do not vary between the scales s_1 and s_2, then $\lambda_J^{(s_2)} \approx \lambda_I^{(s_2)}$ and the maximal λ for I at scale s_2 would also be at $\sqrt{s_2/s_1} \cdot x$. This means that the bias in the location of maximal $\lambda^{(s)}$ is proportional to $\sqrt{s}$, which means that not only is the location biased towards the occluded side, but this bias also grows with scale. This property of λ is demonstrated in Fig. 5.13 on a synthetic example of random dots. In real sequences, the assumption that the intensity distribution is similar in different scales is usually not satisfied. Nevertheless, the effect described above is still observed qualitatively, and can be used to determine depth ordering.

This observation can be used to design a depth-ordering algorithm. The algorithm starts by segmenting the two-frame sequence, to yield an estimate of the matting function $\hat{\alpha}$. For scale s, if the location of the maximum ridge in $\lambda^{(s)}$ is indeed biased towards the occluded side, then at points along the boundary of the segment, the direction of $\nabla\lambda = (\lambda_x, \lambda_y)$ should be towards the outside if the segment is the occluder, and towards the inside if it is occluded. Defining

$$d = \nabla\lambda \cdot \nabla\hat{\alpha} \tag{5.19}$$

we should expect that $d < 0$ if the segment is the occluder, and $d > 0$ if it is occluded. Thus, summing the value of d along a contour of the segment can determine which side of the contour is the occluder.

Since the bias effect grows with scale, it is preferable not to use small scales. On the other hand, higher scales distort the image data and other nearby image features may interfere with the value of d.



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

Figure 5.14: Results on real sequences of three dynamic scenes: (a),(b) The two frames. (c) Response of d (from Eq. (5.19)) coded as dark=negative, light=positive. (d) Final layers detected by the algorithm with relative depth coded as white=near, grey=middle, and black=far.

Therefore, we sum the value of d in several intermediate scales:

$$D = \sum_{s=s_1}^{s_2} \sum_{x \in \partial\hat{\alpha}} \nabla\lambda^{(s)} \cdot \nabla\hat{\alpha} \tag{5.20}$$

The response of d (from Eq. (5.19)) on boundary pixels in real sequences is shown in Fig. 5.14c. In the bottom row, points on the edge between the flower and the hand have positive values with respect

to the hand and negative values with respect to the flower. Relative depth is shown in Fig. 5.14d. The octopus in the top row and flower in the bottom row are correctly detected as the occluders, while the hand is detected as occluding the background and as occluded by the flower. The scene viewed through the window of the old ruin in the middle row is correctly detected as occluded. Note that the internal frame of this window is (correctly) not detected, since there is no depth discontinuity in this area.

5.4.2 Three-Frame Algorithm

Recall that high values of λ occur in areas where there is no smooth motion, i.e., at motion boundaries. At points with no correspondence (due to occlusion), the partial derivatives would have random values, leading to a high λ value, even if these points are not strictly boundary points. These areas are adjacent to the true motion boundary and the λ response would appear as a thick boundary region. Based on two frames alone, it is impossible to determine which side of the thick boundary is the true edge, which is equivalent to determining which side the occluded pixels belong to.

When three frames are available, I denote the response of λ on frames $(t, t-1)$ as λ_-, and frames $(t, t+1)$ as λ_+; t is the reference frame in both cases. I define

$$\lambda_{min} \equiv \min\{\lambda_-, \lambda_+\} \qquad \text{and} \qquad \lambda_{max} \equiv \max\{\lambda_-, \lambda_+\} \tag{5.21}$$

Points on the true motion boundary are detected by both λ_- and λ_+, thus $\lambda_{min} \gg 0$ at these points. Points that are not occluded in any of the frames are not detected by λ, thus $\lambda_{min} \approx 0$. There exist points that are occluded in $t-1$ and not in $t+1$ and vice versa, and in these points $\lambda_{min} \approx 0$ and $\lambda_{max} \gg 0$.

Therefore, the true motion boundary can be detected as the area where $\lambda_{min} \gg 0$. The regions where $\lambda_{min} \approx 0$ and $\lambda_{max} \gg 0$ belong to the occluded layer, and the relation between these regions and the boundary yields depth ordering. This is illustrated in Fig. 5.15.

This principle can be implemented by slightly modifying the two-frame algorithm as follows:

- Use λ_{min} for the segmentation to obtain $\hat{\alpha}$.
- Use λ_{max} in (5.19) to obtain the bias direction d.

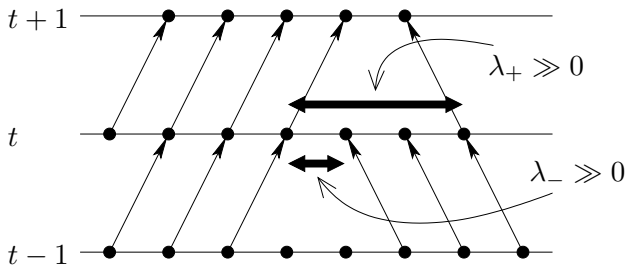


Figure 5.15: Three frames with pixel correspondence; pixels that have correspondences between t and $t-1$ and have no correspondence between t and $t+1$ are located to the right of motion boundary pixels, indicating that the right side is the occluded side.

(a) (b)

Figure 5.16: Results of a real three-frame sequence (octopus example from Fig. 5.14): (a) Edges based on λ_{min} (black) compared with the response of λ_{max} (gray) – the response is stronger outside the edge, indicating that the segment is the occluding layer; (b) Edges based on λ_{min} (black) compared with edges based on λ_+ (i.e., from two frames), showing that three frames give better localization.

Using λ_{min} for the segmentation gives better localization of the segment's edge, since it responds only to the true edge. Since λ_{max} responds also to occluded regions, its profile is biased towards the occluded side (as is the bias due to the intensity gap), and thus $d < 0$ if the segment is the occluder, and $d > 0$ if it is occluded.

Unlike the bias due to intensity difference, the bias that is due to occluded pixels is not affected by scale. Note that no intensity difference was assumed, so this bias can be detected even when there is no intensity difference between the layers. On the other hand, when there is an intensity difference, both effects contribute to the bias, boosting the correct assignment. An additional advantage of the three-frame algorithm is better localization of the segment boundary, as occluded pixels are distinguished

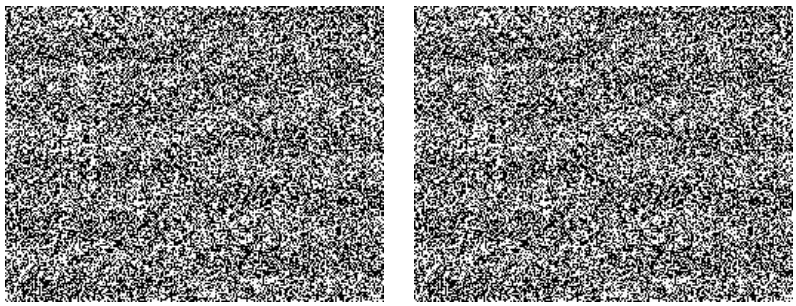


Figure 5.17: Two frames used in the experiment with density varying between 45% and 55%. The sequences used in the experiment are available on the web at `http://www.cs.huji.ac.il/~daphna/demos.html#motion` .

from boundary pixels.

Figure 5.16a shows the edges based on λ_{min} and λ_{max} from three frames of the octopus sequence. The λ_{max} edge is outside the λ_{min} edge, indicating that the segment is the occluder. The λ_{min}-based edge gives better localization of the motion boundary (compared with the two-frame result), as shown in Fig. 5.16b.

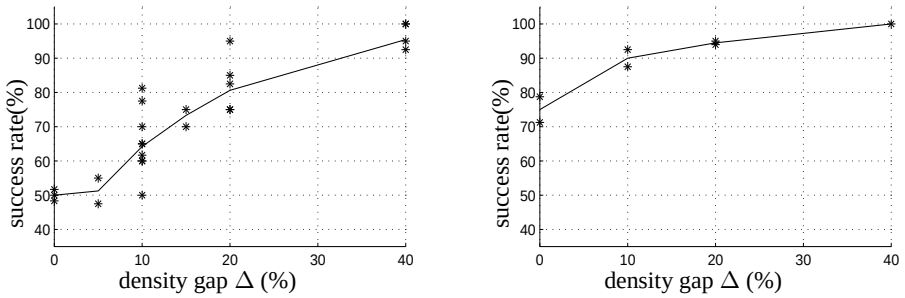
5.5 Human Experiments

The algorithms I have presented determine depth order from two or three frames based on motion alone. They perform well even when monocular segmentation is impossible. Below I show that human observers can also perform these tasks, with comparable success.

In Section 5.5.1 I describe the 2-alternative forced choice experiment, in which I presented subjects with random-dot sequences of two moving layers. In Section 5.5.2 and 5.5.3 I describe the results of experiments with two- and three-frame sequences, respectively.

5.5.1 Methods

In my experiments I presented subjects with sequences in which two layers with random-dot textures, one partially occluding the other, are moving horizontally in opposite directions. The boundary between the layers is the middle vertical line, and the density of the dots varies across each layer along the motion boundary. Figure 5.17 shows an example of such a sequence. Each side was the occluder



(a) (b)

Figure 5.18: Results of experiments on human subjects: (a) Two-frame sequences. (b) Three-frame sequences.

in half of the sequences, in random order (counter-balanced).

In each sequence, the density was characterized by some density gap Δ, so that the density varied between $(1 - \Delta)/2$ and $(1 + \Delta)/2$ across each layer. Participants were instructed to click on the side (left or right) where they thought the occluder was in each sequence. The experiments were conducted in sessions of 20 presentations, with 3-6 sessions per participant for each different value of density gap.

5.5.2 Two-Frame Sequences

Seven volunteers participated in this experiment. In each presentation, the two frames were displayed alternately at a rate of 3 frames/second. The density gap between the two frames was 0%, 5%, 10%, 15%, 20%, 40%.

For a density gap of 40%, subjects selected correctly in nearly 100% of the sequences. For a density gap of 0%, i.e., the density was uniform across the whole frame, subjects selected correctly in 50% of the sequences, i.e., no better than chance. This is consistent with the fact that both interpretations are equally valid in this case. The results are summarized in Fig. 5.18a.

For comparison, I applied the two-frame algorithm to the same sequences. For density gaps of more than 20%, the success rate was nearly 100%. As expected, when density was uniform, the success rate was 50% (in such sequences both interpretations are equally valid). The performance of the algorithm is summarized in Fig. 5.19a.

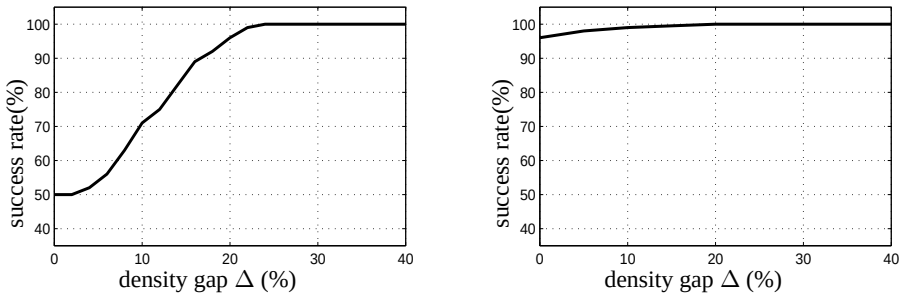


Figure 5.19: Performance of my algorithm on the experiment sequences: (a) Two-frame sequences. (b) Three-frame sequences.

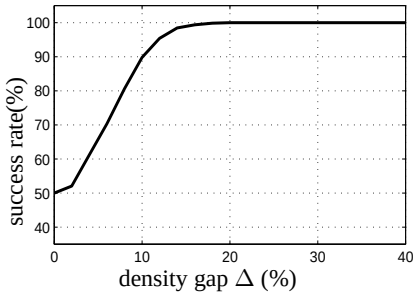


Figure 5.20: Performance of an ideal observer on two-frame experiment sequences.

5.5.3 Three-Frame Sequences

Two volunteers participated in this experiment. In each presentation, a sequence was played back and forth at a rate of 10 frames/second. The density gap between the two frames was 0%, 10%, 20%, 40%. Results for three frames were much better than those for two frames, as expected. In particular, for a density gap of 0% (uniform density), subjects selected correctly in 75% of the sequences, in contrast to the two-frame experiment in which subjects performed no better than chance. The results are summarized in Fig. 5.18b.

The three-frame algorithm, applied to the same sequences, gave the correct answer in nearly 100% of the sequences, and even with uniform density, its success rate was 96% (see Fig.5.19b).

5.5.4 Two-Frame Sequences: Ideal Observer Analysis

In order to evaluate the results of the two-frame experiments and algorithm, consider an ideal observer that "knows" the form of the distributions generating the sequences, but does not know which side is the occluder. Let H_1, H_2 denote the two possible choices: "left-front" and "right-front". For a given two-frame sequence I, the probability that it was generated as H_i is

$$\Pr(H_i|I) = \frac{\Pr(I|H_i) \cdot \Pr(H_i)}{\sum_k \Pr(I|H_k) \cdot \Pr(H_k)} \qquad \text{where} \qquad \Pr(I|H_i) = \prod_{x,y,t} \Pr(I(x,y,t)|H_i) \quad (5.22)$$

$\Pr(I(x,y,t)|H_i)$ and $\Pr(H_i)$ are known to the ideal observer. Thus, for any given I, the ideal observer can compute (5.22) for $i = 1, 2$, and then choose the most probable hypothesis. By sampling sequences, the probability of correct choice was estimated at 97.7% for $\Delta = 10\%$ and 100% for $\Delta = 20\%$. This provides a theoretical upper bound on the performance of an observer in this task.

A less informed observer, that does not know the exact form of the distribution used to generate the data, may consider all possible videos in which the density of dots in each layer remains constant within a small region. Such an observer can compare the density in neighborhoods of occluded pixels with nearby neighborhoods within either layer. For a neighborhood width of 16 pixels, such an *ad hoc* scheme chose correctly in 88% of the sequences for $\Delta = 10\%$, and 99.7% for $\Delta = 20\%$ (see Fig. 5.20).

5.6 Summary

The occlusion detector I have presented is useful for extracting motion boundaries. Since I do not make any assumptions regarding the color or texture properties of objects, or about the geometric properties of the motion, the algorithm works well on natural video sequences where such assumptions are often violated.

The algorithm relies mainly on background features which disappear and reappear as a result of occlusion. These features may be sparse and still indicate the location of motion boundaries, as the algorithm processes the data in multiple scales. As opposed to algorithms that rely on motion estimation, my algorithm usually does not require any texture on the occluding object.

Since occlusion is the main cue used by the algorithm, it works well when velocity differences between moving objects are small, since features will still disappear due to occlusion. Algorithms that rely on motion differences typically find it hard to distinguish between different objects in such cases.

I described a second algorithm, extending the occlusion detector to compute the depth ordering between the layers across the motion boundary. The algorithm was shown to give good results on real sequences with different occlusion settings. With only two frames, the algorithm relies on some (possibly small) difference in texture between the moving layers. Without this assumption, we face the well known inherent motion ambiguity, which states that depth ordering cannot be computed from two-frames and motion alone.

Can humans use a similar heuristic to get around this inherent ambiguity? I asked humans to rank the relative depth of two moving layers in two or three frames. In these experiments there was a difference in texture between the moving layers, but the difference was set to be local and small, so that it could not be detected in a single frame as a distinct boundary between the two layers. Nevertheless, when presented to human subjects in motion, this difference was sufficient for the detection of relative depth. I showed that my algorithm can also utilize this small difference to detect relative depth, giving qualitatively similar results (cf. Fig. 5.18 and Fig. 5.19).

Chapter 6

Summary

I presented two techniques which exploit information in the spatio-temporal structure of video data. We saw how a geometrically-constrained video sequence can be used to generate new views of the scene without scene reconstruction. For an unconstrained video containing general motion, we saw that the spatio-temporal structure of the video is highly regular and can be used for motion segmentation using a differential operator that detects occlusion.

View Synthesis: I presented a new non-perspective projection model, which is defined by two slits and a projection surface. Algebraically, this model corresponds to a second-order transformation from 3D to 2D, and it has an epipolar geometry that is also of second order. Given a video sequence acquired by a translating camera, we can generate new X-Slits views of the scene, making it possible to create a virtual environment using a simple and robust technique. Although not perspective, the movies generated in this way appear compelling and realistic.

Reducing the inherent distortions of the X-Slits projection can be done by reprojecting the image onto a coarse approximation of the scene structure. An alternative approach, without any assumptions about the scene structure, is to select rays so as to approximate the set of rays of a perspective camera.

Motion Segmentation: I have presented an occlusion detector that is used for extracting motion boundaries without any assumptions regarding color or texture properties, or about the geometric properties of the motion. The algorithm processes the detector's response in scale space, producing

a scale-adaptive segmentation which tolerates local ambiguities. The algorithm gives good results on real sequences.

Using the same detector, I described a depth ordering algorithm. The algorithm was shown to give good results on real sequences with different occlusion settings. With only two frames, the algorithm relies on an intensity difference between the moving layers, and does not depend on the existence of a visible intensity edge. The algorithm can be adjusted to work with three frames without this requirement.

Appendix A

Column Sampling Details

A.1 Linear Column Sampling: The Uncalibrated Case

Assume that the motion direction of the input camera is only parallel to the image plane, with the internal parameters of the camera being fixed but unknown. I now show that any linear sampling of the columns results in a valid X-Slits image.

Let **J** denote the inverse of the calibration matrix [26, p. 141], and assume that column $x = s(t)$ is sampled from the image captured at time t. The 3D physical location of this column on the camera's projection plane, in standard coordinates denoted p, q, is the line defined by:

$$\begin{pmatrix} p \\ q \\ 1 \end{pmatrix} = \mathbf{J} \begin{pmatrix} s(t) \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} J_{11}s(t) + J_{12}y + J_{13} \\ J_{22}y + J_{23} \\ 1 \end{pmatrix} \tag{A.1}$$

(**J** is upper triangular). Eliminating the free variable y, the line is parameterized by:

$$p = K_1 q + K_2 + K_3 s(t) \tag{A.2}$$

where

$$K_1 = \frac{J_{12}}{J_{22}},\ K_2 = J_{13} - \frac{J_{12}}{J_{22}}J_{23},\ K_3 = J_{11}$$

Assume as before that the camera center at time t is $\mathbf{c}(t) = (lt, 0, 0)$. Assume also that columns are sampled linearly, i.e., the column sampled from the image taken at time t is defined by $(s(t), y)\ \forall y$,

and $s(t) = \alpha t + \beta$ for some α, β. I will show that the result of pasting these columns together into a mosaic image is a X-Slits image.

First, from the definition it follows that the image taken at time t contributes a set of rays that lie on a 3D plane. This plane is defined by the translating camera center $\mathbf{c}(t)$ and the line on the image plane which corresponds to column $s(t)$ (see Fig. A.1, right inset). Denote this plane by $\pi(t)$. I first show that all the planes $\pi(t)$ intersect in a line; this line defines the vertical slit of our X-Slits camera. The horizontal slit is defined by the trajectory of the camera.

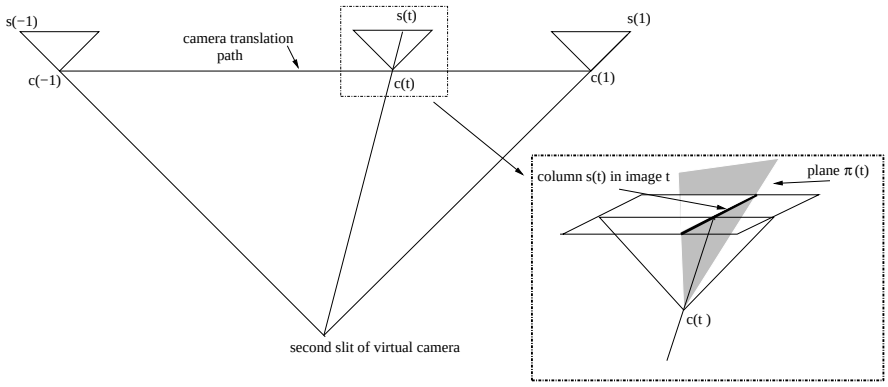


Figure A.1: Illustration of the relevant geometry, including the camera's path, the center of projection $\mathbf{c}(t)$, the column sampling function $s(t)$, and the plane $\pi(t)$.

Specifically, the plane $\pi(t)$ is defined by the camera center $\mathbf{c}(t) = (lt, 0, 0)$ and two points on the line given in (A.2); we choose two such points by setting $q = 0$ and $q = 1$:

$$\begin{pmatrix} lt \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} K_2 + K_3(\alpha t + \beta) \\ 0 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} lt \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} K_1 + K_2 + K_3(\alpha t + \beta) \\ 1 \\ 1 \end{pmatrix} \tag{A.3}$$

As can be readily verified, the plane incident with all three points is defined by

$$\begin{aligned} \pi(t) &= (-1, K_1, K_2 + K_3(\alpha t + \beta), lt) = \\ &= (-1, K_1, K_2 + K_3\beta, 0) \; + \; t(0, 0, K_3\alpha, l) \end{aligned} \tag{A.4}$$

It follows from Eq. (A.4) that the family of planes $\pi(t)$ is a pencil of planes which intersect in a line, and I define this line to be the vertical slit of the camera. The slit direction is determined by the

cross product:

$$(-1, K_1, K_2 + K_3\beta) \times (0, 0, K_3\alpha) \propto (K_1, 1, 0) \tag{A.5}$$

From the requirement that the slit coincides with $\pi(t)$ from Eq. (A.4) for every t, we arrive at the following slit equation:

$$\begin{pmatrix} \frac{-l}{K_3\alpha}(K_2 + K_3\beta) \\ 0 \\ \frac{-l}{K_3\alpha} \end{pmatrix} + \lambda \begin{pmatrix} K_1 \\ 1 \\ 0 \end{pmatrix} \qquad \forall\, \lambda \in \mathcal{R}$$

Thus by collecting vertical strips linearly and placing them in a mosaic, the resulting image is equivalent to a X-Slits image whose slits are parallel to the image plane. In case of zero skew, $J_{12} = 0$ and so $K_1 = 0$; now the two slits are orthogonal as in the Orthogonal X-Slits camera discussed in Section 3.1.2.

A.2 Input Camera Moving on a General Straight Line

I shall now analyze the sampling function when the motion of the input camera is not parallel to the image plane. Let the motion direction be an arbitrary line in 3D. Let $\mathbf{c}(t) = \mathbf{c}_0 + \Delta\mathbf{c} \cdot t$ denote the camera position at time t (in $\mathcal{R}^3$) and let the pinhole camera's projection matrix be $[\mathbf{A}; -\mathbf{A}\mathbf{c}(t)]$, where $\mathbf{A} = [A_1, A_2, A_3]^T$ denotes the 3×3 matrix determined by the camera's internal calibration and its orientation. It follows that when the camera is located at $\mathbf{c}(t)$, a scene point $\mathbf{p} \in \mathcal{R}^3$ is projected to $(\frac{x}{w}, \frac{y}{w})$ such that

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} A_1^T(\mathbf{p} - \mathbf{c}(t)) \\ A_2^T(\mathbf{p} - \mathbf{c}(t)) \\ A_3^T(\mathbf{p} - \mathbf{c}(t)) \end{pmatrix} \tag{A.6}$$

Suppose we sample strips from these images, at varying positions and varying orientations. Let $s(t)$ and $\alpha(t)$ denote the sampling functions, giving the position of the strip on the image x-axis and the strip's orientation, respectively, so that we sample in image t the points on the oriented line

$\frac{x}{w} = s(t) + \alpha(t)\frac{y}{w}$. Using (A.6) we obtain

$$x - \alpha(t)y - s(t)w = A_1^T(\mathbf{p} - \mathbf{c}(t)) - \alpha(t)A_2^T(\mathbf{p} - \mathbf{c}(t)) - s(t)A_3^T(\mathbf{p} - \mathbf{c}(t)) = 0 \qquad \text{(A.7)}$$

This defines a plane (on which $\mathbf{p}$ lies) – the plane coincident with the sampled oriented line in image t and the center of projection of the camera $\mathbf{c}(t)$. Denote this plane by $\pi(t)$ (as in Fig. A.1). This plane in homogeneous coordinates is given by

$$\pi(t) = \begin{pmatrix} A_1 - \alpha(t)A_2 - s(t)A_3 \\ -(A_1^T - \alpha(t)A_2^T - s(t)A_3^T) \cdot \mathbf{c}(t) \end{pmatrix} \qquad \text{(A.8)}$$

For our sample to give a X-Slits image, it is necessary and sufficient that all these planes $\forall t$ intersect in a line. There are three cases to note in (A.8):

1. If $s(t) = constant$ and $\alpha(t) = constant$, then $\pi(t)$ is linear in t and therefore describes a line. When $\alpha(t) = 0$, we get the linear (oblique) pushbroom camera model [23, 54], see Fig. 4.3.

2. If $A_3^T\Delta\mathbf{c} = 0$, $s(t)$ is linear in t and $\alpha(t) = constant$, then once again $\pi(t)$ is linear in t. This is the case when the camera motion is parallel to the projection plane - the Orthogonal X-Slits projection discussed in Section 3.1.2 (when $\alpha(t) \neq 0$, we get the tilted slit variant discussed in 3.1.3).

3. In the general case, if $\alpha(t)$ and $s(t)$ are of the form

$$\alpha(t) = -\frac{a_2 + b_2 t}{a_1 + b_1 t}, \quad s(t) = -\frac{a_3 + b_3 t}{a_1 + b_1 t} \qquad \text{(A.9)}$$

Substituting these expressions into (A.8) gives

$$\begin{aligned} \pi(t) \quad &\propto \quad \pi(t) \cdot (a_1 + b_1 t) = \\ &= \quad \begin{pmatrix} (a_1 + b_1 t)A_1 + (a_2 + b_2 t)A_2 + (a_3 + b_3 t)A_3 \\ -((a_1 + b_1 t)A_1 + (a_2 + b_2 t)A_2 + (a_3 + b_3 t)A_3)^T(\mathbf{c}_0 + \Delta\mathbf{c} \cdot t) \end{pmatrix} \end{aligned} \qquad \text{(A.10)}$$

which is linear in t if and only if

$$(b_1 A_1 + b_2 A_2 + b_3 A_3)^T \Delta\mathbf{c} = 0 \qquad \text{(A.11)}$$

It is easy to see that the first two cases are special cases of (A.9).

There are 4 degrees of freedom (up to scale) in choosing a pair $\langle \alpha(t), s(t) \rangle$ of sampling functions which satisfy constraint (A.11). Each pair of sampling functions of the form (A.9) for which (A.11) holds implies that all $\pi(t)$ intersect in a single line (the virtual slit). And vice versa, it can also be shown that every family of planes $\pi(t)$ which intersect in a single line uniquely defines such a pair of sampling functions.

We can therefore conclude the following: using the sampling method described above with sampling functions as in (A.9), and given a linear camera path, we can produce the image of any X-Slits camera with one slit overlapping the camera path.

Appendix B

The Distortion of Linear Sampling Functions

In Section 4.1.1 it was shown that any linear sampling function $\phi(t) = \alpha t + \beta$ corresponds to an Orthogonal X-Slits projection of the form given in (3.8). This can be rewritten using (4.15) as:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} x(1 - \frac{Z}{Z_1}) \\ y(1 - \frac{\alpha Z}{\alpha Z_1 + k}) \end{pmatrix} \tag{B.1}$$

(it is assumed that $Z_1, \alpha \geq 0$).

Assuming $\omega = [x_1, x_2] \times [y_1, y_2]$, substituting (B.1) into (4.17), $n(\phi, p, \hat{X}, \hat{Y}, \hat{Z})$ equals:

$$\int_{x_1}^{x_2} \int_{y_1}^{y_2} \frac{(x(1 - \frac{\hat{Z}}{Z_1}) - X_0)^2 + (y(1 - \frac{\alpha \hat{Z}}{\alpha Z_1 + k})) - Y_0)^2}{\hat{Z}^2} dy dx \tag{B.2}$$

Integrating, and then differentiating with respect to $\hat{X}, \hat{Y}, \hat{Z}$, the minimum is achieved in (X_0, Y_0, Z_0):

$$\begin{aligned} X_0 &= (x_1 + \tfrac{1}{2}\Delta x)(1 - \tfrac{Z_0}{Z_1}) \\ Y_0 &= (y_1 + \tfrac{1}{2}\Delta y)(1 - \alpha \tfrac{Z_0}{\alpha Z_1 + k}) \\ Z_0 &= \frac{\Delta x^2 + \Delta y^2}{\frac{\Delta x^2}{Z_1} + \frac{\alpha \Delta y^2}{\alpha Z_1 + k}} \end{aligned} \tag{B.3}$$

where $\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$. Substituting the above into (B.2) we obtain the local perspectivity

distortion n_L of $\phi(t) = \alpha t + \beta$

$$n_L(\phi) = \frac{1}{12} \frac{\Delta x^3 \Delta y^3}{\Delta x^2 + \Delta y^2} \left(\frac{1}{Z_1} - \frac{\alpha}{\alpha Z_1 + k} \right)^2 \tag{B.4}$$

The global perspectivity for a linear sampling function $\phi(t) = \alpha t + \beta$ is obtained by substituting (B.4) into (4.19), and integrating over the image domain:

$$n_G(\phi) = \int_{p \in I} \frac{n_L(\phi, p)}{n_L(\hat{\phi}, p)} dp = S \left(\frac{k}{\alpha Z_1 + k} \right)^2 \tag{B.5}$$

where $S = (x_{max} - x_{min})(y_{max} - y_{min})$ is the image area.

Appendix C

Motion Segmentation Details

C.1 Scale Normalization

One problem with multi-scale analysis is that derivatives decrease with scale. Indeed, if $0 \leq I \leq 1$, then

$$|I_x|, |I_y| \leq \frac{1}{\sqrt{2\pi s_{xy}}} \tag{C.1}$$

when smoothing with a spatial Gaussian of variance s_{xy}. This well-known problem can be handled by scale normalization, as proposed in [38]. Scale normalization is done by defining the *scale-normalized* partial derivatives

$$I_x^{(s_{xy})} = \sqrt{s_{xy}} \cdot \frac{\partial}{\partial x}(g_{s_{xy}} * I) \quad \text{and} \quad I_y^{(s_{xy})} = \sqrt{s_{xy}} \cdot \frac{\partial}{\partial y}(g_{s_{xy}} * I) \tag{C.2}$$

where $g_{s_{xy}}*$ stands for convolution with a Gaussian with variance s_{xy}. Thus $I_x^{(s_{xy})}$ and $I_y^{(s_{xy})}$ are used in the evaluation of λ instead of I_x and I_y. Note that scale normalization does not violate the assumptions leading to the definition of λ in Section 5.1.1.

One important property of scale normalization is that λ becomes invariant to spatial scaling of I. This means that λ gives comparable values for a video sequence in different resolutions.

To see this, let us scale I by σ, and define

$$J(x, y, t) = I(x/\sigma, y/\sigma, t) \tag{C.3}$$

Substituting (C.3) into (C.2) yields

$$\nabla J^{(\sigma^2 s_{xy})}(\sigma x, \sigma y, t) \quad = \quad \nabla I^{(s_{xy})}(x, y, t) \tag{C.4}$$

Let s_ω denote the variance of the Gaussian window ω, and let $\mathbf{G}^{(s_{xy}, s_\omega)}[I]$ denote the second moment matrix defined in (5.1), with the scale of differentiation s_{xy} and scale of averaging s_ω. From (C.4) it follows that

$$\left(\mathbf{G}^{(s_{xy}, s_\omega)}[I]\right)(x, y, t) \quad = \quad \left(\mathbf{G}^{(\sigma^2 s_{xy}, \sigma^2 s_\omega)}[J]\right)(\sigma x, \sigma y, t) \tag{C.5}$$

That is to say, if J is a scaling by σ of I, then the value of λ at (x, y, t) in I at scales s_{xy}, s_ω will be the same as at the corresponding point in J at scales $\sigma^2 s_{xy}, \sigma^2 s_\omega$.

For the purpose of computing a good *occlusion detector*, it follows from (C.5) that as long as the computation scans all scales in scale space, the result does not depend on the image resolution. Note that in order for λ to be scale-invariant, it follows from (C.5) that s_ω must be proportional to s_{xy}, as in [35]. In my implementation I use $s \equiv s_{xy} = s_\omega$, which defines a single scale s. I denote the λ evaluated at scale s as $\lambda^{(s)}$.

C.2 Implementation Issues

C.2.1 Temporal Aliasing

Since real video data is discrete, the partial derivatives in the definition of λ must be estimated. This is done by convolving I with the partial derivatives of a 3-dimensional Gaussian. Rotational invariance implies that the spatial variance in the X and Y directions should be the same, and the kernel is therefore an ellipsoidal Gaussian with spatial variance s_{xy} and temporal variance s_t. Due to the distortion introduced by the convolution, it is desirable that these values be small.

Estimating the temporal partial derivative from video presents a severe aliasing problem. Since video frames represent data accumulated during short and sparse exposure periods, and since a feature may move several pixels between two consecutive frames, data is aliased in the temporal domain

significantly more than in the spatial domain. I overcome this problem by taking advantage of the spatio-temporal structure of video, as described next.

Suppose that the velocity in a certain region is $v = (v_x, v_y)$, and therefore

$$I(x, y, t) = I(x - v_x t, y - v_y t, 0) \tag{C.6}$$

The temporal derivative in $t = 0$ is given by

$$I_t = -v_x I_x - v_y I_y \tag{C.7}$$

In discrete video, I_t can be estimated by convolution in the T direction, which, due to (C.6), is the same as convolution in the v direction of a subsample of $I(x, y, 0)$ at intervals of size $|v|$. In order to avoid aliasing due to undersampling while estimating I_t, the Sampling Theorem requires I to be band-limited, so that its Fourier transform vanishes beyond $\pm\frac{1}{2|v|}$. This can be achieved by smoothing with a spatial Gaussian. However, smoothing poses a notable drawback, as it distorts the image data, causing features to disappear, merge and blur.

An alternative approach, closely related to the concept of "warping" (e.g., [40]), would be to take advantage of prior estimates of the optical flow. If a point is estimated to move at velocity $u = (u_x, u_y)$, we can use the convolution of I in the direction of $(u_x, u_y, 1)$ to estimate the directional derivative I_u and apply

$$I_t = I_u - u_x I_x - u_y I_y \tag{C.8}$$

The convolution that yields I_u is equivalent to subsampling in the direction of $(v - u)$, and thus the estimate of I_t is unaliased if the Fourier transform vanishes beyond $\pm\frac{1}{2|v-u|}$. This occurs when either the estimated velocity u is close to the real velocity v, or the region is smooth. This is particularly important, as the estimation of optical flow in smooth regions is often inaccurate. In other words, this estimation of I_t is tolerant to inaccuracy in motion estimation exactly when it is least reliable. The figures in Section 5.2 demonstrate the algorithm's tolerance to poor motion estimation.

Note that the spatial smoothness of u is not required. Also note that temporal smoothing has no effect on the aliasing problem, and it is desirable to have as little temporal smoothing as possible.

C.2.2 Differentiation with Two Frames

Differentiation, as described earlier, is done by convolution with derivatives of a spatio-temporal Gaussian, which requires several frames to achieve a good estimation. When only two frames are available, special care should be taken to provide a consistent estimation of spatial and temporal derivatives. Given two frames $I(x, y, 0)$ and $I(x, y, 1)$, let us define

$$I^*(x, y, t) = \begin{cases} I(x, y, 0) & t \leq 0 \\ I(x, y, 1) & t > 0 \end{cases} \tag{C.9}$$

Then, for any temporal variance s_t, the partial derivative estimates are

$$\begin{aligned} I_x^* &= \frac{1}{2}(I(x, y, 0) + I(x, y, 1)) * g_x \\ I_y^* &= \frac{1}{2}(I(x, y, 0) + I(x, y, 1)) * g_y \\ I_t^* &= (I(x, y, 1) - I(x, y, 0)) * g \end{aligned} \tag{C.10}$$

(where $*g$, $*g_x$ and $*g_y$ denote convolutions with the spatial Gaussian and with its X and Y derivative respectively).

C.2.3 Application to Optical Flow

It is well known that the computation of optical flow in textureless regions and along straight lines (aperture problem) is ill-posed. When these situations occur, the rank of $\mathbf{G}$ is 0 and 1, respectively. These situations arise from spatial structure alone, and can therefore be detected by the spatial 2D second moment matrix (used, for example, in the Lucas-Kanade algorithm [40]), in order to mark these regions as unreliable (as done in many implementations). Optical flow is also unreliable at motion boundaries, which may be treated by the joint estimation of motion and segmentation [60, 72].

These two cases can be treated jointly using the rank of $\mathbf{G}$. Optical flow in regions where $rank(\mathbf{G}) \neq \mathbf{2}$ can be estimated by filling from adjacent regions where $rank(\mathbf{G}) = \mathbf{2}$. In a coarse-to-fine algorithm, this should be done at each scale.

Bibliography

[1] D. Weinshall, M.Lee, T. Brodsky, M. Trajkovic, D. Feldman. New view generation with a bi-centric camera. In *Proc. of 7th European Conference on Computer Vision*, I:614–628, Copenhagen, May 2002.

[2] D. Feldman, A. Zomet, S. Peleg D. Weinshall. Video Synthesis made simple with the X-Slits Projection. In *Proc. of IEEE Workshop on Motion and Vision Computing*, Orlando, December 2002.

[3] A. Zomet, D. Feldman, S. Peleg, D. Weinshall. Mosaicing New Views: The Crossed-Slits Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):741–754, June 2003.

[4] D. Feldman, T. Pajdla, D. Weinshall. On the Epipolar Geometry of the Crossed-Slits Projection. In *Proc. of 9th IEEE International Conference on Computer Vision*, 2:988–995, Nice, October 2003.

[5] D. Feldman, A. Zomet. Generating Mosaics with Minimum Distortions. In *Proc. of 2nd IEEE Workshop on Image and Video Registration*, Washington DC, July 2004.

[6] D. Feldman, D. Weinshall. Realtime IBR with Omnidirectional Crossed-Slits Projection. In *Proc. of 10th IEEE International Conference on Computer Vision*, 1:839–845, Beijing, October 2005.

[7] D. Feldman, D. Weinshall. Motion Segmentation Using an Occlusion Detector. In *Proc. of Workshop on Dynamical Vision, 9th European Conference on Computer Vision*, Graz, May 2006.

[8] D. Feldman, D. Weinshall. Motion Segmentation and Depth Ordering Using an Occlusion Detector. Submitted to *TPAMI*.

[9] E. H. Adelson, J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, 3–20. MIT Press, 1991.

[10] D. Aliaga, I. Carlbom. Plenoptic stitching: A scalable method for reconstructing interactive walkthroughs. In *Proc. of SIGGRAPH'01*, 443–450, 2001.

[11] A. Aner, J. R. Kender. Video Summaries through Mosaic-Based Shot and Scene Clustering. In *Proc. of ECCV'02*, IV:388–402, Copenhagen, May 2002.

[12] N. Apostoloff, A. Fitzgibbon. Learning spatiotemporal T-junctions for occlusion detection. *CVPR'05*, 553–559.

[13] H. Bakstein, T. Pajdla. Rendering Novel Views from a Set of Omnidirectional Mosaic Images. In *Workshop on Omnidirectional Vision and Camera Networks 2003*, IEEE June 2003.

[14] J. Bergen, P. Anandan, K. Hanna, R. Hingorani. Hierarchical model-based motion estimation. In *Proc. of ECCV'92*, 237–252, 1992.

[15] L. Bergen, F. Meyer. A novel approach to depth ordering in monocular image sequences. *CVPR'00*, II:536–541.

[16] M.J. Black, D.J. Fleet. Probabilistic Detection and Tracking of Motion Boundaries. *IJCV* 38(3):231–245, 2000.

[17] C. Buehler, M. Bosse, S. Gortler, M. Cohen, L. McMillan. Unstructured lumigraph rendering. In *Proc. of SIGGRAPH'01*, 425–432, 2001.

[18] G. T. Chou. A Model of Figure-Ground Segregation from Kinetic Occlusion. *ICCV'95*, 1050–1057.

[19] A. W. Cunningham, T. F. Shipley, P. J. Kellman. Interactions between spatial and spatiotemporal information in spatiotemporal boundary formation. *Percept. Psychophys.* 60(5):839-851, 1998.

[20] T. Darrell, D.Fleet. Second-order method for occlusion relationships in motion layers. MIT Media Lab Technical Report 314, 1995.

[21] E. Gamble, T. Poggio. Visual integration and detection of discontinuities: The key role of intensity edges. AI Lab, MIT, A.I. Memo No. 970, 1987.

[22] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen. The lumigraph. In *Proc. of SIGGRAPH'96*, 43–54, 1996.

[23] R. Gupta, R. I. Hartley Linear Pushbroom Cameras. *TPAMI*, 9(19):963–975, 1997.

[24] C. Harris, M. Stephens. A combined corner and edge detector. In *Proc. of Alvey Vision Conference*, 147–151, 1988.

[25] R. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proc. of ECCV'92* 579–587, 1992.

[26] R. I. Hartley, A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[27] B. Horn, B.Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[28] M. Irani, P. Anandan. Video indexing based on mosaic representations. In *Proc. of IEEE*, 86(5):905–921, May 1998.

[29] H. Ishiguro, M. Yamamoto, S. Tsuji. Omni-directional stereo. *TPAMI*, 14(2):257–262, February 1992.

[30] S. Kang. A survey of image-based rendering techniques. In *Videometric VI*, 3641:2–16, Jan 1999.

[31] G. A. Kaplan. Kinetic disruption of optical texture: the perception of depth at an edge. *Percept. Psychophys.* 6(4):193–198, 1969.

[32] Q. Ke, T. Kanade. A subspace approach to layer extraction. In *Proc. of CVPR'01*, 255–262.

[33] R. Kingslake. *Optics in Photography*. SPIE Optical Engineering Press, 1992.

[34] V. Kolmogorov, R. Zabih. Computing Visual Correspondence with Occlusions using Graph Cuts. In *Proc. of ICCV'01*, 2:508–515.

[35] I. Laptev, T. Lindeberg. Space-time Interest Points. In *Proc. of ICCV'03*, 432–439, 2003.

[36] I. Laptev, T. Lindeberg. Velocity adaption of space-time interest points *ICPR'04*, 1:52-56.

[37] M. Levoy, P. Hanrahan. Light field rendering. In *Proc. of SIGGRAPH'96*, 31–42, 1996.

[38] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2):117–154, 1998.

[39] T. Lindeberg, B. M. H. Romeny. Linear scale-space: (I) Basic theory and (II) Early visual operations. In: Romeny (ed.) *Geometry-Driven Diffusion*, 1–77, Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.

[40] B. D. Lucas, T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. of IJCAI'81*, 674–679, 1981.

[41] L. McMillan, G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proc. of SIGGRAPH'95*, 39–46, Los Angeles, August 1995.

[42] M. Middendorf, H.-H. Nagel. Estimation and Interpretation of Discontinuities in Optical Flow Fields. In *Proc. of ICCV'01*, 1:178-183.

[43] D. W. Murray, B. F. Buxton. Scene segmentation from visual motion using global optimization. *TPAMI*, 9(2):220–228, March 1987.

[44] K. M. Mutch, W. B. Thompson. Analysis of Accretion and Deletion at Boundaries in Dynamic Scenes. *TPAMI* 7(2):133–138, 1985.

[45] S. K. Nayar, A. Karmarkar. 360 x 360 mosaics. In *Proc. of CVPR'00*, volume 2, pages 388–395, 2000.

[46] B. Newhall. *The History of Photography, from 1839 to the present day*, 162. The Museum of Modern Art, 1964.

[47] M. Nicolescu, G. Medioni. A Voting-Based Computational Framework for Visual Motion Analysis and Interpretation. *TPAMI*, 27(5):739–752, May 2005.

[48] S. A. Niyogi. Detecting kinetic occlusion. *ICCV'95*, 1044–1049.

[49] J.M. Odobez, P. Bouthemy. MRF-based motion segmentation exploiting a 2D motion model robust estimation. In *Proc. of ICIP'95*, 3:628–631, 1995.

[50] A. S. Ogale, C. Fermüller, Y. Aloimonos. Motion Segmentation Using Occlusions. *TPAMI* 27(6):988–992, June 2005.

[51] T. Pajdla. Stereo with Oblique Cameras. *IJCV*, 47(1/2/3):161–170, 2002.

[52] H. Pao, D. Geiger, N. Rubin. Measuring Convexity for Figure/Ground Separation. *ICCV'99*, 948–955.

[53] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.

[54] S. Peleg, M. Ben-Ezra, Y. Pritch. Omnistereo: Panoramic stereo imaging. *TPAMI*, 23(3):279–290, March 2001.

[55] S. Peleg, B. Rousso, A. Rav-Acha, A. Zomet. Mosaicing on Adaptive Manifolds *TPAMI*, 10(22):1144–1154, 2000.

[56] H. Pottmann, J. Wallner. Computational Line Geometry. Springer Verlag, Berlin, Germany, 2001.

[57] P. Rademacher, G. Bishop. Multiple-center-of-projection images. In *SIGGRAPH'98*, 32:199–206, 1998.

[58] X. Ren, C. C. Fowlkes, J. Malik. Figure/Ground Assignment in Natural Images. *ECCV'06*, II:614–627.

[59] E. Saund. Perceptual Organization of Occluding Contours of Opaque Surfaces *CVIU*, 76(1):70–82, October 1999.

[60] H. S. Sawhney, S. Ayer. Compact Representations of Videos Through Dominant and Multiple Motion Estimation. *TPAMI*, 18(8):814–830, August 1996.

[61] S.M. Seitz. The space of all stereo images. In *Proc. of ICCV'01*, I:26–33, 2001.

[62] J. Semple, G. Kneebone. Algebraic Projective Geometry. Clarendon Press, Oxford. 1998.

[63] J. Shi, J. Malik. Motion Segmentation and Tracking Using Normalized Cuts. In *Proc. of ICCV'98*, 1154–1160, 1998.

[64] H. Shum, L. He. Rendering with concentric mosaics. In *Proc. of SIGGRAPH'99*, 299–306, August 1999.

[65] H. Y. Shum, R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *Proc. of ICCV'99*, 1:14–21, 1999.

[66] P. J. Sloan, M. F. Cohen, S. J. Gortler. Time Critical Lumigraph Rendering. In *Symposium on Interactive 3D Graphics*, 17–24, April 1997.

[67] R. Swaminathan, M.D. Grossberg, S.K. Nayar. A perspective on distortions. In *Proc. of CVPR'03*, II:594–601, 2003.

[68] T. Takahashi, H. Kawasaki, K. Ikeuchi, M. Sakauchi. Arbitrary view position and direction rendering for large-scale scenes. In *Proc. of CVPR'00*, 296–303, June 2000.

[69] M. Tappen, W. T. Freeman. Comparison of Graph Cuts with Belief Propagation for Stereo, using Identical MRF Parameters. In *Proc. of ICCV'03*, 900–907, 2003.

[70] W. B. Thompson, K. M. Mutch, V. A. Berzins. Dynamic Occlusion Analysis in Optical Flow Fields. *TPAMI* 7(4):374–383, 1985.

[71] D. Tweed, A. Calway. Integrated Segmentation and Depth Ordering of Motion Layers in Image Sequences. *BMVC'00*.

[72] Y. Weiss. Smoothness in layers: motion segmentation estimation using nonparametric mixture. In *Proc. of CVPR'97*, 520–526, 1997.

[73] Y. Weiss, E. H. Adelson A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. In *Proc. CVPR'96*, 321–326.

[74] D.N. Wood, A. Finkelstein, J.F. Hughes, C.E. Thayer, D.H. Salesin. Multiperspective panoramas for cel animation. In *Proc. of SIGGRAPH'97*, 31:243–250, 1997.

[75] A. Yonas, L. G. Craton, W. B. Thompson. Relative motion: Kinetic information for the order of depth at an edge. *Percept. Psychophys.* 41(1):53–59, 1987.

[76] J.Y. Zheng, S. Tsuji. Panoramic Representation for Route Recognition by a Mobile Robot *IJCV*, 9(1): 55–76, October 1992.

[77] Z. Zhu, E.M. Riseman, A.R. Hanson. Parallel-perspective stereo mosaics. In *Proc. of ICCV'01*, II: 345–352, 2001.