Differential Tracking through Sampling and Linearizing the Local Appearance Manifold

Hua Yang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2008

Approved by:

Greg Welch, Advisor

Gary Bishop, Reader

Leonard McMillan, Reader

Marc Pollefeys, Reader

Leandra Vicci, Reader

Abstract

Hua Yang: Differential Tracking through Sampling and Linearizing the Local Appearance Manifold.
(Under the direction of Greg Welch.)

Recovering motion information from input camera image sequences is a classic problem of computer vision. Conventional approaches estimate motion from either dense optical flow or sparse feature correspondences identified across successive image frames. Among other things, performance depends on the accuracy of the feature detection, which can be problematic in scenes that exhibit view-dependent geometric or photometric behaviors such as occlusion, semi-transparancy, specularity and curved reflections. Beyond feature measurements, researchers have also developed approaches that directly utilize appearance (intensity) measurements. Such appearance-based approaches eliminate the need for feature extraction and avoid the difficulty of identifying correspondences. However the simplicity of on-line processing of image features is usually traded for complexity in off-line modeling of the appearance function. Because the appearance function is typically very nonlinear, learning it usually requires an impractically large number of training samples.

I will present a novel appearance-based framework that can be used to estimate rigid motion in a manner that is computationally simple and does not require global modeling of the appearance function. The basic idea is as follows. An n-pixel image can be considered as a point in an n-dimensional *appearance space*. When an object in the scene or the camera moves, the image point moves along a low-dimensional *appearance manifold*. While globally nonlinear, the appearance manifold can be locally linearized using a small number of nearby image samples. This linear approximation of the local appearance manifold defines a mapping between the images and the underlying motion parameters, allowing the motion estimation to be formulated as solving a linear system.

I will address three key issues related to motion estimation: how to acquire local appearance samples, how to derive a local linear approximation given appearance samples, and whether the linear approximation is sufficiently close to the real local appearance manifold. In addition I

will present a novel approach to motion segmentation that utilizes the same appearance-based framework to classify individual image pixels into groups associated with different underlying rigid motions.

Acknowledgments

I would like to express my sincere gratitude to my adviser Greg Welch. Throughout the years of my Ph.D. study, I have continuously benefit from his knowledge, his inspiration and his great efforts to explain things clearly and simply. His understanding, encouraging and personal guidance have made this long journey a pleasant one.

I would like to thank Marc Pollefeys for his insightful ideas that contribute to the basis of this dissertation. I am also very grateful to the other committee members: Gary Bishop, Leonard McMillan and Leandra Vicci. This dissertation could not have been finished without their invaluable suggestions and comments.

I am indebted to all the people who have helped my research from different perspectives. I would like to thank Jan-Michael Frahm for his valuable advice on several research projects. I wish to thank Adrian Ilie for his excellent camera calibration program that has been used in a number of experiments reported in this dissertation. I am grateful to Henry Fuchs and Andrei State for their guidance and support in my RA projects. I would like to express my appreciation to Herman Towles and John Thomas for their help in the hardware setup. I also thank Jingdan Zhang for the interesting discussion on a variety of research topics.

Lastly and most importantly, I would like to thank my wife Chen Jiang and my parents Jianhua Gu and Deben Yang. I would not have gone this far without their love and support. I dedicate this dissertation to my son, Michael Kai Yang, who has been cooperative during my disseratation writing.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DCC	Differential Camera Cluster
DLT	Direct Linear Transformation
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
FOV	Field of View
INM	Illumination Normal Map
KF	Kalman Filter
LEDs	Light-emitting Diodes
PCA	Principal Component Analysis
PnP	Perspective-n-Point
RBF	Radial Basis Functions
SFM	Structure from Motion
SSD	Sum-of-square Difference

Chapter 1

Introduction

1.1 Motion estimation

Images from cameras, or more generally measurements from visual sensors, carry a variety of information about the real world. The goal of computer vision is to develop theories and techniques to extract information from these visual inputs. This thesis concerns the recovery of motion information from image sequences, a classic computer vision problem known as *motion estimation* or *tracking*.

In computer vision, the kinematics of an object or a camera are usually represented as a state vector. As the object or the camera moves, the parameters of the kinematic vector change respectively. Tracking or motion estimation is then the process of analyzing the observed image sequence to compute the change of the target state vector from the reference frame to current frame. In practice, a motion estimation system usually employs a motion model to describe how the image should change with respect to the possible motions of the target. For instance, when the target object is planar or its motion is mostly restricted within a plane, a 2D motion model is usually used to compute an affine transformation or a homography. For a 3D rigid object or a camera, its kinematics can be represented as a 6D *pose* vector (3D orientation and 3D position) and the motion estimation is defined as the process of recovering the change of the

pose parameters. In more complicated cases, an articulated or deformable object can be represented as connected parts or meshes, and its motion is defined by the changes of the poses of the parts or the positions of the nodes. The discussion in this thesis will be focused on the recovery of 3D rigid motion. Throughout the thesis, the words *motion estimation* and *tracking* will refer to the process of estimating the change of pose parameters of cameras and objects.

1.2 Tracking as solving a reverse mapping

An image is a projection of a 3D scene onto a 2D plane, it is a function of the scene and its 3D pose with respect to the camera. As an object and/or the camera move (change their poses), the image changes over time. We can see that there exist two mappings: one relates the image to the pose, the other relates the change of the image to the motion (see Figure 1.1). If we consider the imaging process as a forward mapping from the pose to the image, tracking can be viewed as the process of solving one of the above two reverse mappings. Specifically, we can extract the difference between current image and the reference image, then map it to the target motion. Or if a global mapping between the image and the pose is feasible, we can estimate poses from images and subtract them to acquire the motion.

Figure 1.1 shows that motion estimation can be formulated as establishing a 2D-to-3D (images to poses or image differences to motions) mapping. Since the image is determined by both the scene and the pose, computing the reverse mapping requires decoupling the scene from the forward imaging function. The decoupling can be achieved by acquiring some invariant representation or model of the scene.

A scene model can be an explicit one that is given as a prior. In the context of model-based tracking, an offline model is usually used to provide a geometric and/or photometric representation of the scene. This offline model can be exact or generic.

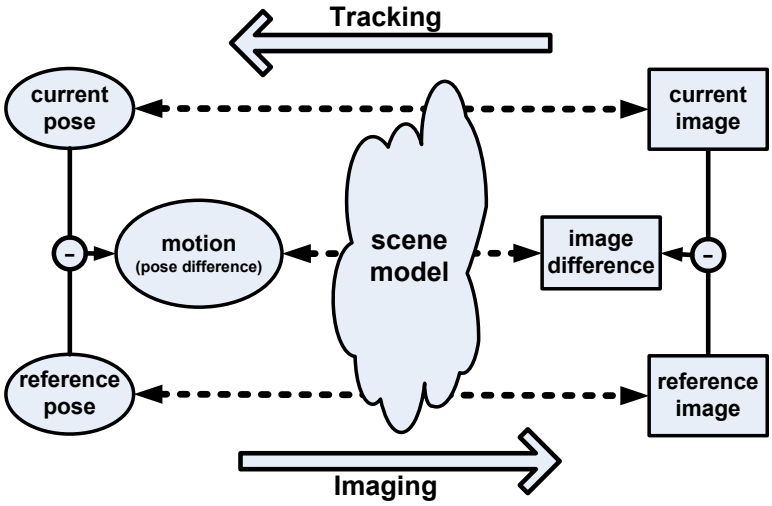


Figure 1.1: Motion estimation as solving a reverse mapping. The imaging process can be considered as a forward mapping from the pose to the image. Motion estimation can be solved by finding a reverse mapping. The top and bottom dashed lines represent the mapping between the image and pose. The middle dashed line indicates the mapping between the change of the image and the change of the pose. Since an image is a function of both the scene and the pose, the image and the pose or the image difference and the motion are related using a scene model.

It can be as simple as a collection of 3D artificial markers or, in a more complicated form, a graphics model with surface geometry and texture. While the models can come in different forms, they are usually target specific. So to track motions in a scene of moderate complexity, one usually needs to build a large number of object models, a task that is usually tedious if not infeasible. Thus model-based tracking methods are usually limited to handle individual objects in a constrained environment.

A scene model can also be implicit, computed online. For instance, the 3D structure of a rigid scene can be recovered simultaneously during tracking using the so-called Structure from Motion (SFM) approach. The basic idea is that, under certain camera projection model, the relative motion between the scene and the camera, and the positions of the 3D scene points, are constrained by the 2D correspondences of the projected scene points across multiple views. When correspondences are identified correctly and the rigidity of the scene holds, the motion and the 3D scene model can be recovered. However, the real world can be complicated. Some common phenomenons

like occlusion, semi-transparency and specularity challenge the point matching process. Moreover, SFM methods assumes rigidity of the 3D scene. This assumption can be violated in effect when the scene is observed through reflection or refraction of a curved surface.

The mapping between images and poses can also be learned from training data. Learning based tracking methods usually consist of two stages: the off line *learning* stage and the on line *tracking* stage. During the training stage, sample images taken at known poses are used to learn a parametric representation of the scene. Tracking is then formulated as the process of searching in this parametric space. The advantage of learning based approaches is that once a global scene model is learned the online tracking can be achieved very efficiently. However, the mapping between the image measurements and the pose parameters is usually highly nonlinear. A global learning process usually requires a large number of training samples, all taken at known poses. This labor intensive process greatly reduces the practical usage of learning based approaches.

In summary, motion estimation has been actively studied and various approaches have been developed. The state of the art algorithms have been successfully applied to real image sequences. However, motion estimation in a complex environment is still a challenging problem. The real world often exhibits complicated geometric or photometric behaviors such as occlusion, semitransparancy, specularity and curved reflections. Existing methods usually fail in such cases due to one or more of the following difficulties:

- modeling the sophisticated geometric and photometric behaviors,
- identify 2D correspondences when the scene appearance is view-dependent, and
- acquiring enough training samples to learn a global scene model.

In this thesis, I will introduce a motion estimation approach that addresses the challenging problem of tracking in scenes with complicated geometric and photometric behaviors. This novel framework is based on sampling and linearizing the local appearance manifold.

1.3 Tracking through sampling and linearizing the local appearance manifold

A 2D image can be considered as a point in a high dimensional *appearance space* (see Chapter 3). When an object in the scene or the camera moves, the image point moves along a certain manifold called the *appearance manifold*. We can see that there exists a mapping from the pose space to the appearance manifold. Moreover, for a non-degenerate scene, this mapping is invertible and tracking can be archived by learning the appearance manifold. An ideal solution is to learn it globally. However, this is usually impractical. The appearance manifold is typically highly nonlinear, and a global learning process usually requires numerous training samples. In fact, a global learning process is not necessary. Depending on the frame rate, the motion between two frames is usually restricted within a certain region. Therefore a local representation of the appearance manifold can be enough for tracking incremental motion. As discussed earlier, an appearance manifold is defined by its underlying motion. Since motion lies in a low dimensional space, the dimensionality of the appearance manifold is low. Therefore it should be possible to compute a linearization of the local appearance manifold with a small number of local image samples.

Based on the above observations, I present a framework that tracks incremental 3D rigid motion by sampling and linearizing the local appearance manifold. Local appearance samples are acquired using a camera cluster at run time. Using these samples, a linear approximation of the local appearance manifold is computed. Motion estimation is then achieved by solving a linear system. This method does not require any prior scene model or off line training process. It does not assume any 3D or 2D correspondence, thus can accommodate scenes with view-dependent appearance changes. As far as I know, this is the first motion estimation approach that addresses challenging scenes that exhibit complicated behaviors like semitransparancy, specularity and curved

reflections.

1.4 Thesis statement and main contributions

Thesis Statement

A 2D image can be considered as a point in a high dimensional appearance space. When the camera or a rigid object in the scene moves, the image point moves along a 6D appearance manifold. While globally nonlinear, the appearance manifold can be locally linearized using a set of 7 neighboring image samples. Such a local linearization of the appearance manifold can be used for 3D motion segmentation and tracking.

The main contributions of this thesis work can be summarized as follows:

I **A novel appearance based differential framework for tracking 3D rigid motion.** The local appearance manifold is linearized using a small number of image samples captured by a camera cluster. Tracking is effectively achieved by solving a linear system. This approach does not require any offline scene model or training images; nor does it assume any 3D or 2D correspondence. To my knowledge, it is the first method that accommodates scenes with view-dependent appearance changes. This framework can be integrated into a model-based framework. When a prior graphics model is provided, the local samples can be acquired through graphical rendering.

II **A spectral analysis of the linearization of a local appearance manifold.** Any linearization of a non-linear function can only be considered valid within a local region. To quantitatively determine the size of the local region, I formulate the linearization as the process of sampling and reconstructing the appearance signal. The Fourier analysis shows that to avoid aliasing, the image motion between appearance samples should be within one pixel. This analysis can be applied to

other approaches that assume local linearization, for instance, optical flow.

III **A pure appearance based approach to dense motion segmentation.** This approach is also based on a local linearization of the appearance function. The change of the intensity of a pixel can be considered as a linear function of the motion of the corresponding imaged surface. When a sequence of local image samples are captured, the intensity changes of a particular pixel over the sequence form a vector called *intensity trajectory*. The intensity trajectories of pixels corresponding to the same motion span a linear subspace. Thus the problem of motion segmentation can be cast as that of clustering local subspaces.

1.5 Thesis outline

The thesis is organized as follows. A review of various approaches to motion estimation is given in Chapter 2. Depending on their input measurements, motion estimation methods are categorized into three classes: feature-based, flow-based and appearance-based. Chapter 3 presents the novel framework of appearance-based differential tracking. The algorithm of tracking by sampling and linearizing the local appearance manifold will be introduced. Chapter 4 describes the techniques for acquiring local appearance samples. Chapter 5 presents a spectral analysis that justifies the linearization of the local appearance manifold from a signal process point of view. In Chapter 6, the locally linear appearance model is applied to solve the problem of motion segmentation. Experiment results on these new appearance-based approaches to tracking and motion segmentation are demonstrated in Chapter 7. Finally, Chapter 8 concludes this thesis and discusses the directions for future work.

Chapter 2

Related work

Motion estimation systems take an input image sequence and output motion parameters of some target objects, or the camera. As described in Chapter 1, this process can be formulated as solving a reverse mapping. An image I is a function F of the scene M and its pose S with respect to the camera.

$$I = F(S, M) \tag{2.1}$$

Motion can be recovered by computing one of the two reverse mappings F^{-1} : from the image I to the pose P (then subtract reference pose to current pose to get motion), or from the change of the image dI to the motion dS. [1].

$$S = F^{-1}(I, M) \tag{2.2}$$

$$dS = F^{-1}(dI, M) \tag{2.3}$$

In the above equations, I and dI are conceptually described as image and change in image. To quantitatively solve the problem, motion estimation systems extract a set of measurements from input images and describe I and dI using these measurements.

[1]Note that the two functions in (2.2) and (2.3) are usually different. Here I denote both of them as F^{-1} to indicate that motion estimation is a reverse process.

Image measurements can simply be pixel intensities or some high-level descriptors or features computed from basic intensity values. The above equations also show that to directly relate I to S or dI to dS, the scene M needs to be decoupled. Motion estimation algorithms achieve this by acquiring some invariant representation or model of the scene. A scene model can be an explicit one provided as a prior, or an implicit one acquired during tracking.

This chapter will start with a brief survey of image measurements. Based on their input image measurements, motion estimation algorithms will be categorized into three classes: feature-based, flow-based and appearance-based. Popular methods in each of these three classes will be reviewed. Different approaches to scene modeling will be introduced. Finally, to conclude this chapter, a general discussion on existing motion estimation techniques will be presented.

2.1 Image measurements

Vision-based motion estimation methods usually start by extracting measurements from input images. Different types of image measurements have been used. Some of them are low-level measurements that can be directly acquired from images. The most basic image measurements are pixel intensities and intensity derivatives.

Intensity or color. The intensity or color of an image pixel is determined by the illumination and the albedo of the corresponding surface patch. In the color representation, a variety of color spaces are used. The commonly used ones include RGB, HSV, Luv and Lab. Among all the image measurements, intensity and color are the simplest. However, they are sensitive to motion-dependent illumination variations hence their usage is usually prohibited in scenarios where the motion-dependent illumination effects are non-neglectable.

Intensity derivative. The derivative of a pixel intensity indicates the smoothness

of the observed image at that pixel. Compared with the original intensity measurements, intensity derivatives are more sensitive to imaging noise. This noise issue gets exaggerated for high order derivatives. In practice, only the first and the second order derivatives are used, and they are usually used as intermediate measurements for computing some high-level measurements like textures or optical flow.

The intensity and intensity derivative measurements have the advantage that they can be acquired with minimal computation power. However, being the simplest measurements, they are in general not very descriptive to the uniqueness of the scene. To achieve a better representation of the scene, motion estimation system usually employ further processing of the basic image measurements to form high-level descriptors or features. A list of commonly used high-level descriptors are presented as follows.

Intensity statistics. The probability density of the pixel intensities inside an image region provides a unique description of the appearance of the object corresponding to that region. The intensity statistics can be parametric such as a Gaussian Mixture (RMG98), or nonparametric such as color histogram (HVM98). Like the original intensity measurements, intensity statistics are usually sensitive to illumination changes.

Texture. As an image measurement, texture is usually defined as the variation pattern of pixel intensities within a processed image window. Texture measurement can be used to quantify the smoothness and regularity of the imaged surface. The difference between a texture and an intensity statistic is that the latter only represents color information while the former also records structure information. Some example texture descriptors are wavelets (LK89) and steerable pyramids (GBG$^+$94). Compared with pixel intensity or color, texture measurements are usually considered less sensitive to illumination changes.

Interest point. Interest point features are the most widely used image features in the area of tracking. Popular point feature detecting algorithms include the Harris corner detector (HS88), the KLT feature detector (TK91) (ST94), and the SIFT feature

detector (Low04). An overview of the state of the art in feature extractors is given by Mikolajczyk and Schmid (MS05). In general, point features extracted by these algorithms are invariant to smooth illumination changes. As a comparison, Harris features and KLT features are invariant to translation, rotation and uniform scaling in the spatial domain, but are not invariant to affine and projective projections. SIFT features are more robust to different transformations, but the SIFT detector is also relatively slow for real time tracking.

Edge. Edges correspond to discontinuities in image intensities. Possible causes of edges include discontinuities in depth, discontinuities in surface normals, and changes in material properties. An edge can be detected using an edge detector, which usually uses intensity derivatives. One of the most efficient and popular edge detectors is Canny (Can86). A review of edge detecting techniques can be found in (ZT98). Edge features are usually insensitive to illumination changes. However, edge features can cause difficulties for tracking in an image sequence where the viewpoint changes. This is because the occlusion relationship are view-dependent. Hence, edges caused by depth discontinuities may change substantially when the viewpoint changes.

Optical flow. Optical flow refers to the dense 2D displacement field that indicates the pixel motion on the image plane. For each pixel, its flow is the 2D projection of the 3D motion of its corresponding 3D scene point. The flow field is usually computed under the brightness constancy assumption. Popular optical flow techniques include differential methods by Lucas and Kanade (LK81) and Horn and Schunk (HS81), region-based matching by Anandan (Ana89) and phase-based correlation by Fleet and Jepson (FJ90). The readers are referred to the survey by Barron et al. (BFB94) for an evaluation of optical flow methods. Optical flow is a commonly used image measurement for recovering motion and scene geometry. However, like pixel intensities, optical flow measurements are sensitive to illumination changes. Moreover, since optical flow is usually computed across a local region, the flow estimates are noisy at occlusion boundaries.

So far, I have introduced a list of popular classes of image measurements. Among them, interest point and edge features represent the 2D projections of 3D geometric primitives. As they describe the structure and shape of the underlying 3D scene, I categorize them as **geometric** features. Similarly, I categorize intensity, intensity statistics and texture as **appearance** measurements, since they describe the appearance of the scene. The optical flow measurement is unique in that it directly encodes motion information. So I separate it from the other image measurements and classify it as **flow** measurements.

In accordance with the three classes of input measurements, I categorize motion estimation methods into: **feature-based**, **flow-based** and **appearance-based**. In the remainder of this chapter, I will discuss each of them in details.

2.2 Feature-based motion estimation

In a feature-based framework, the scene is represented as a collection of 3D geometric entities whose projections are observed in images as 2D features. Feature-based motion estimation systems use the positions of these 2D features to compute the change of the relative pose between the camera and the scene. To relate the 2D feature positions with the 3D pose parameters, they usually assume some knowledge about the scene structure, or more precisely, the 3D shapes and positions of the geometric entities. Such a scene structure can be provided explicitly as a prior model or be recovered simultaneously during tracking.

2.2.1 Model-based approaches

Most model-based approaches assume an explicit model that defines the 3D shapes and positions of a set of geometric entities of the scene in some global coordinate system. During tracking, the projections of these entities are identified and matched in the im-

age. Motion estimation is then formulated as the single-view 3D-to-2D pose estimation problem. The relative pose between the camera and the scene at each frame can be estimated by computing the projective transformation that best maps the model from its own 3D coordinate system to its current 2D image observations. Motion can then be acquired by subtracting pose estimates.

One way of acquiring a scene model is by adding *fiducials* or *markers* into the scene. Some commonly used fiducials include color-coded rings (SHC+96; CN98) and Light-emitting Diodes (LEDs) (WBV+01; BN95). In general, fiducials are some artificial markers whose 3D positions in the world coordinates are precisely measured by some offline process. These markers are designed such that they can be detected easily in the images and their image locations can be measured to a high accuracy. Therefore, fiducials provide easily detectable and accurate 3D-to-2D point correspondences. These correspondences can then be used to compute poses and motions in a relatively reliable and accurate manner. Due to such advantages, fiducial-based techniques have been widely used in Virtual Reality and Augmented Reality.

Fiducial-based techniques require instrumenting the scene. This task can be tedious or even impractical, especially for outdoor environments. Therefore, it is usually more convenient to rely on natural features present in the environment. While in theory any image feature that is detectable and measurable can be used, most practical motion estimation systems employ linear primitives such as interest points (Alt92; GRS+02; VLF04) or lines (Low87; Low92; NF93; Jur99; CMC03) or both (LHF90; HN96). There are two major reasons for the popularity of point or edge based methods. First, methods using simple linear primitives are computationally efficient and relatively easy to implement. Moreover, interest point and edge features are usually insensitive to illumination variances. To establish 3D-to-2D correspondences, the 3D positions of the points and lines need to be provided. This 3D information can be in the form of an off line CAD model built by commercial products, or can be acquired by triangulation of several key frames

with known poses during the initialization process. The fitting of higher order primitive models to their 2D projections have also been explored. For instance, quadratic primitives have been used in (Wei93; FPF99). In comparison with linear primitives, quadratic primitives provide more geometric constraints about the viewing parameters. However they are usually difficult to identify and extract from images in an accurate and reliable manner.

Fiducials and natural features can be used in a complementary way. An example was given in an early work of Bishop (Bis84). Noticing the task of motion estimation can be simplified by accelerating the imaging-processing-imaging circle [2], Bishop designed and fabricated a self-tracking system using multiple out-viewing VLSI chips that performed both imaging and processing. This integrated system used the natural features to estimate incremental motion at a very high framerate. Fiducials were used to address the drifting of the pose estimated by integrating the motion.

Estimating pose parameters

The viewing parameters can be computed from the correspondence between a set of geometric entities with known 3D positions and their 2D projections. This so-called 3D-to-2D pose estimation problem is one of the oldest problem of computer vision. Usually the pose parameters are estimated by maximizing an objective function that quantifies the 3D-to-2D alignment. Due to the large number of proposed algorithms, a complete survey of pose estimation is beyond the scope of this dissertation. Readers can refer to (LF05) and Chapter 20 of (Se07) for detailed reviews. Here, I will briefly introduce several of the most popular point-based approaches. The problem is formulated as: given correspondences between a set of n 3D points $M_i = [X_i \ Y_i \ Z_i]^T$ and their 2D projections $m_i = [x_i \ y_i]^T$, compute the projection matrix P that best maps M_i to m_i.

[2] Shorter imaging time results in a higher framerate, which results in a smaller motion, which results in simpler estimation problem, which results in shorter processing time, which in turn results in a possibly even higher framerate and shorter imaging time.

Direct Linear Transformation (DLT) . As shown in (2.4), each correspondence between M_i and m_i defines two equations on the elements of P. Based on this observation, DLT algorithms such as (Sut74; Gan84) solve the 11 entries of the camera projection matrix from at least six corresponding points.

$$\begin{aligned} x_i &= (P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14})/(P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}) \\ y_i &= (P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24})/(P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}) \end{aligned} \tag{2.4}$$

DLT is usually used as a camera calibration technique that determines both the pose and the camera intrinsic parameters up to a scale. This over-parameterization usually introduces instability and requires more correspondences for the purpose of pose estimation. In this case, it is usually preferable to estimate the camera intrinsic parameters separately.

Perspective-n-Point (PnP) . When camera intrinsic parameters are known, correspondences of 3 points can be used to estimate pose parameters with up to 4 solutions. Additional point correspondences are required to guarantee a unique solution. (FB87) shows that the solution is unique for 4 points on a plane or 6 points in general positions. Depending on the number of points used, the problem is known as P3P, P4P or in the general form PnP. Different approaches to PnP have been proposed (HLON91; QL99). They usually employ the constraints defined by the triangle equations shown in (2.5).

$$\begin{aligned} d_{ij}^2 &= d_i^2 + d_j^2 - 2d_i d_j \cos\theta_{ij} \\ \cos\theta_{ij} &= (m_i^T Q m_i)/((m_i^T Q m_i)^{\frac{1}{2}}(m_j^T Q m_j)^{\frac{1}{2}}) \end{aligned} \tag{2.5}$$

Here $d_{ij} = \|M_i - M_j\|$, $d_i = \|M_i - C\|$ and $d_j = \|M_j - C\|$ respectively represent the distances between the reference points M_i, M_j and the camera center C. θ_{ij} is the angle between the viewing lines CM_i and CM_j. For known camera intrinsic matrix K, the directions of CM_i and CM_j can be written as $K^{-1}m_i$ and $K^{-1}m_i$. $\cos\theta_{ij}$ can the be computed using the image projections m_i, m_j and matrix $Q = (KK^T)^{-1}$.

Gold Standard method. DLT and PnP algorithms are fast and can be solved in closed form. However, they achieve pose estimation by minimizing algebraic error that has no direct geometric meanings, thus are usually sensitive to measurement noise. When the error of image measurements m_i are independent and Gaussian, the optimal pose estimate can be computed by minimizing the sum of the reprojection errors in (2.6). This nonlinear optimization is typically solved in a iterative form. The initial pose estimate is usually provided by one of the DLT or PnP algorithms.

$$[R, T] = \arg\min_{(R,T)} \sum_i \|PM_i - m_i\|^2 \tag{2.6}$$

2.2.2 Structure from motion approaches

So far, we have assumed that the 3D structure of the scene is provided in terms of a prior model. For a rigid scene, its structure can also be acquired simultaneously during tracking. This problem is referred to as SFM in computer vision. Given a set of images of a rigid scene taken from different perspectives, SFM algorithms recover the 3D scene structure along with the camera motion across the image sequence. Depending on the input image measurements, SFM methods can be categorized into *feature-based* and *flow-based*. In this section, I will discuss the feature-based SFM. The flow-based approaches will be introduced in the next section.

Feature-based SFM considers a set of 3D features undergoing rigid motion with respect to the camera. Given the correspondences of 2D projections of the features across multiple perspectives, SFM systems estimate the positions of the features and the relative motion between the views, using the so-called *epipolar geometry constraint*. Commonly used features include interest points and line segments. Methods using the two classes of features are comparable in computational complexity. The former is the choice for most state of the art SFM systems as the real world usually consists of a cluttered background that causes difficulty when extracting and matching line features.

Our discussion here will focus on point-based methods. Some line-based examples can be found in (TK95; ZF92). For a comprehensive introduction to both classes, interested readers are referred to the multi-view geometry books by Hartley and Zisserman (HZ00) and Faugeras and Long (FL01).

The projections of a 3D point in a rigid scene into two views are related by the epipolar geometry constraint, which can be linearly formulated as:

$$m_1{}^T F m_2 = 0 \tag{2.7}$$

Here m_1 and m_2 are the image coordinates of the 2D projections of the same 3D point M observed in two views. The 3×3 singular matrix F is called the *fundamental* matrix. Given the camera intrinsic matrix K, F can be used to compute the *essential matrix* E:

$$E = K^T F K \tag{2.8}$$

The estimation of the essential matrix E is the key to SFM algorithms. Once estimated, E can be factorized to acquire the relative motion (the rotation and translation) between the two views. The 3D position M can then be easily acquired by triangulation. Note that the linear constraint in (2.7) is only defined up to a scale. The same scaling applies to the recovered scene structure and the translational component of the motion.

The recovery of the essential matrix using the epipolar geometry was first proposed by Lounget-Higgins. The so-called 8 point algorithm presented in (LH81) has the appealing property of being linear. However, the results obtained using this approach are usually not satisfactory in the presence of noise. From a numerical point of view, Hartley showed that the stability of the 8 point algorithm can be improved by a simple renormalization of the input measurements (Har97). The performance can be further improved by using more points and adopting least square techniques, or enforcing rank (Har97) and Degrees of Freedom (DOF) constraints (HN94; TBM94), or employing nonlinear optimization

(LF96). A review of these techniques is provided in (Zha98).

The performance of SFM can be improved by combining information from more than two views. For instance, the feature correspondences across a triplet of images is constrained by a *trifocal* tensor (SW95). When feature correspondences across a relatively large number of views are available, the optimal structure and motion can be estimated by solving a large nonlinear optimization problem known as *bundle adjustment* (TMHF00). The objective function is the sum of square of the reprojection error of all features from all views. Probably the most interesting technique in multi-view geometry is the simultaneously recovery of camera intrinsic parameters along with motion and structure, a technique called *auto-calibration*. Robust systems have been demonstrated in (PKG98; Nis03). A nice introduction to these techniques, as well as a comparison between the flow-based and the feature-based SFM approaches, can be found in (Zuc02).

2.3 Flow-based motion estimation

The term *optical flow* refers to the dense displacement field that indicates the pixel motion across frames. For each pixel, its flow is the 2D projection of the 3D motion of its corresponding scene point. Flow-based systems thus attempt to recover the underlying 3D motion from its 2D projection. This is usually achieved simultaneously with the recovery of the scene structure. While flow-based methods are akin to feature-based methods in the sense that both rely on (dense or sparse) image correspondences, these two approaches are quite different. Feature-based methods are based on epipolar geometry. They can, and are better at, recovering large motions. Flow-based methods, as we will discuss soon, use a locally linear model that is only valid for a small motion.

Under perspective projection, the flow of a pixel undergoing a small motion can be computed (Hor86) using (2.9) where (x, y) are the coordinates of the pixel, (u, v) are the image displacement or flow of the pixel, z is the depth of the pixel, T and R are the

translation and rotation of the relative motion between the scene and the camera. For a pinhole camera model with focal length f

$$[u, v]^T = AT/z + BR$$

$$A = \begin{vmatrix} -f & 0 & x \\ 0 & -f & y \end{vmatrix} \qquad B = \begin{vmatrix} \frac{xy}{f} & -(f + \frac{x^2}{f}) & y \\ (f + \frac{y^2}{f}) & -\frac{xy}{f} & -x \end{vmatrix} \tag{2.9}$$

Equation (2.9) shows that the flow field is related to two sets of parameters: the *global* parameters T and R that indicate the camera *motion*, and the *local* (per pixel) parameter z that indicates the 3D *structure* or *depth* of the scene. Based on this observation, researchers have developed many optical flow based motion estimation methods. These methods usually consist of two steps: first computing optical flow from input image sequences, then simultaneously recovering structure and motion from flow estimate. In the rest of this section, I will discuss the two steps accordingly, starting from flow estimation.

2.3.1 Flow estimation

The estimation of optical flow is based on the brightness constancy assumption (HS81). The brightness constancy assumption states that the brightness of a moving object remains constant and the change of the image intensity $I(x, y, t)$ is a result of a translation in the image plane.

$$I(x + u, y + v, t + \triangle t) = I(x, y, t) \tag{2.10}$$

where (x, y) are the coordinates of image pixel, (u, v) are the image displacement or flow of pixel (x, y) between the two images taken at time t and $t + \triangle t$ respectively. From the first order Taylor's expansion of (2.10), the well-known gradient constraint equation can be written as

$$I_x u + I_y v + I_t = 0 \tag{2.11}$$

where (I_x, I_y) are the spatial derivatives and I_t is the change of pixel intensity at (x, y).

Different techniques have been developed to estimate flow field (u, v). Based on the types of extracted image measurements, optical flow algorithms can be categorized into four classes: differential-based, energy-based, phase-based and correlation-based (BFB94). Here I will briefly introduce the more commonly used differential and correlation methods. Interested readers can refer to (BFB94) for a detailed review. Evaluations of optical flow methods can be found in (LHH+98; BRS+07).

Differential-based methods

Differential-based methods compute the flow field using (2.11). Since (2.11) only provides one constraint to two unknowns, we need to introduce other constraints to compute (u, v). Horn and Schunk (HS81) combined a global smoothness term with the gradient constraint equation to form an objective function for flow estimation.

$$\sum_D (I_x u + I_y v + I_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2) \tag{2.12}$$

The flow field can then be solved iteratively by minimizing the objective function. In (2.12), D represents a local region, and (u_x, u_y, v_x, v_y) are the flow derivatives that indicate the smoothness of the flow field inside D. Thus the second regulation terms can be expressed as the flow field is smooth within the local region.

Similarly, Lucas and Kanade (LK81; Luc84) added a global smoothness constraint by assuming constant flow within a local region. The objective function becomes

$$\sum_{(x,y)\in D} W^2(x, y)[I_x(x, y)u + I_y(x, y)v + I_t(x, y)]^2 \tag{2.13}$$

where $W(x, y)$ represents a window function that gives more weight to center pixels than the periphery ones. The flow field is then solved linearly in the least-square form.

The above methods are based on constancy or smoothness of the flow field within a lo-

cal region. The underlying assumption is that neighboring pixels are likely to correspond to the same surface. When the surface is smooth the flow field should change smoothly. Most differential-based methods employ this assumption. Clearly, this smoothness constraint is likely to be violated when the examined local region crosses surface boundaries.

Correlation-based methods

The most direct way to compute flow field is to match regions from the previous image to the current image. In this type of methods, the flow field is assumed to be constant within a local neighborhood. This formulation is similar to area based stereo correlation. The flow estimate can be computed by finding the best match that minimizes a distance function. For example, Anandan (Ana89) and Singh (Sin92) use Sum-of-square Difference (SSD) as the distance function shown in (2.14).

$$\sum_{(x,y)\in D} [I(x+u, y+v, t+\triangle t) - I(x,y,t)]^2 \tag{2.14}$$

The correlation-based methods again assume constant local flow field. The underlying assumption is that the intensity pattern of a small region is likely to remain constant over time, even when its position changes. This assumption can be violated, for instance, when the local region contains motion boundaries or when the change of illumination occurs.

2.3.2 Motion estimation

Equation (2.9) shows that the flow field provides constraints to two sets of parameters: the *global* parameters that indicate the camera *motion* and the *local* (per pixel) parameters that indicate the 3D *structure* of the scene. It follows that both the structure and the motion can be recovered from the estimated flow field in an iterative form. In the absence of noise, the motion parameters can be recovered using the flow and depth

measurements at five or more pixels (Hor87). In practice, as the per-pixel flow estimate is usually noisy, more points are needed. Thus the structure from motion problem is usually solved using a least-square optimization scheme (ZT99; BH83; HJ92).

Bruss and Horn (BH83) proposed a method to estimate ego-motion by minimizing the sum-of-square of the optical flow residual $r = [u^*, v^*]^T - [u(T, \Omega, z),\ v(T, \Omega, z)]$. The least-square estimate of the depth z and the rotation Ω are computed as a function of translation T. These estimates are then substitute back to the residuals to form a function of the translation alone. A nonlinear solver is then applied to estimate T, z and R are then computed.

Heeger and Jepson (HJ92) proposed to estimate egomotion using subspace methods. Similar to (BH83), they used algebraic manipulation to separate the computation of translation from those of the depth and the rotation. However, they avoided the nonlinear optimization process required for computing the translation. Under the instantaneous-time assumption, for a given set of optical flow at n pixels they constructed $m - 6$ constraint vectors that are orthogonal to the camera translation. Thus the translation can be computed in a linear form.

2.3.3 Direct methods

The above methods recover structure and motion parameters using (2.9). The estimation accuracy depends on the quality of the input flow field. Unfortunately, as we have discussed, the computation of the optical flow is usually error prone. To address this issue, researchers have developed methods that bypass flow estimation (IA99; Ira02; Han91; HO93). The so-called *direct methods* solve two problems simultaneously: the estimation of structure and motion, and the estimation of optical flow (pixel correspondence).

Direct methods use the gradient constraint provided in (2.11). Using some parametric motion model, flow vectors u and v of a pixel are replaced with functions of the *global*

motion parameters and the *local* shape parameter. A set of n pixels provide n gradient constraints to $n+5$ unknowns: n local shape parameters plus 6 DOF rotation and translation minus 1 DOF scaling factor. To solve this ill-posed problem, more constraints need to be added. There are two general approaches to additional constraints: assuming smoothness of scene depth (Han91), or using multiple frames (IAC02). For instance, 3 frames (two motions) of n pixels provide $2n$ constraints for the $n + 11$ unknowns. Note that the latter approach also assumes local smoothness. This is determined by the use of (2.11), which is only applicable to small (sub-pixel) motion. To accommodate practical image motion that is almost always larger than a pixel, direct methods smooth the original image to acquire a filtered image with enlarged pixel size. The underlying assumption is that the 3D scene patch corresponding to an enlarged pixel is of constant depth and therefore uniform flow velocity. As we have discussed, the local smoothness assumption can be violated at occlusion boundaries. Therefore, the application of direct methods is usually constrained to close-to-planar scenes, where the motion can be described using simple parametric form such as affine or quadratic (TZ99).

2.4 Appearance-based motion estimation

The scene appearance in the 2D camera images changes with respect to the motion between the scene and the camera. The key to 3D appearance-based tracking is to acquire some appearance model that can parameterize this relationship. There are two general approaches to 3D appearance-based tracking. One approach utilizes a 3D renderable model that can be used to predict the scene appearance from different hypothesized views. Motion estimation is achieved by matching the real image appearance with the predicted ones. The other approach learns a parametric representation of the scene appearance using a set of training images. The motion estimate is computed by projecting current appearance observation into that parametric space.

Many appearance-based motion estimation algorithms aims at solving the problem of *visual tracking* in a 2D scenario. The goal is to consistently locate the image region occupied by a specific target in each frame of the input image sequence. Typically, the target object is represented using an appearance model that describes some invariant image property. Popular appearance models include global statistics such as mixture model (Fre00) or color histogram (CRM00), and texture such as wavelets (JFEM03). A major advantage of these models is that they can be acquired on line with minimum effort. The models are usually initialized using a small number of (usually one) frames and can often be refined during tracking. However, these appearance models are inherently 2D, and only encode information on object appearance changes with respect to an image or an affine motion.

2.4.1 Model-based approaches

For 3D tracking, some kind of 3D appearance representation is required to predict the appearances of the scene from different views. A popular class of approaches to tracking objects under changing views employs textured 3D graphical models of target objects. The appearance of the object at some predicted pose is generated through graphics rendering. Dellaert *et al.* (DTT98) proposed a Kalman filter based system that simultaneously estimates the pose and refines the texture of a planner patch. Cascia *et al.* (CSA00) used a textured cylindrical model for head pose estimation. Schodl (SHE98) and Chang (CC01) used a textured polygonal model for more accurate head pose estimation. An appealing property of a textured 3D model is that it can be rendered efficiently using graphics hardware. This advantage has made it the choice of most existing model-based methods. In theory, any model that can be used to synthesize appearance samples from different views can be used. For instance, a pre-acquired lightfield model is used in (ZFS02).

Usually, a 3D appearance model is built off line under an illumination that is usu-

ally different from the on line condition. Since the appearance of an object changes with respect to the illumination, model-based methods are prone to the illumination inconsistencies between the modeled and the real spaces. Different techniques have been developed to address the illumination issue. In (CC01), the change of illumination is addressed by using local correlation coefficients as the similarity measurements. Hager *et al.* (HB98) and Cascia *et al.* (CSA00) used a linear subspace to model the change of appearance under varying lighting conditions. Zhang *et al.* (ZMY06) proposed the use of belief propagation to estimate the intensity ratio between corresponding pixels of an image pair. They applied this illumination insensitive method to visual tracking. In (YWP06), we addressed model-based 3D tracking of Lambertian objects under directional light sources. A Kalman filter framework is applied to estimate the object motion, the illumination and refine the texture model simultaneously. This illumination insensitive model-based method is described in Section 4.2.

Acquiring an off line model can be tedious and sometimes impractical, for instance, consider camera tracking in an outdoor environment. Researchers thus have worked on developing methods that can recover the 3D shape (ZSM06) and appearance model (ZH01) from image data during tracking. While on line modeling is appealing, and can greatly boost the applicability of model-based approaches, the state of the art algorithms are still not robust enough in general . Tracking error usually causes the mapping of background data into the model, which in turn results in larger tracking error. The system therefore becomes unstable quickly. Practical model-based methods thus still rely on an off line modeling process to provide the full 3D model of the target object or at least an initial one to start with.

2.4.2 Image-based approaches

A parametric representation of the scene appearance can be acquired using training image samples taken from different known perspectives. The idea is that the set of

all possible images of the target object lie on a low-dimensional manifold that can be parameterized by the varying pose and illumination parameters.

Pioneered by the work of Murase and Nayar (MN95), a popular class of algorithms models scene appearance using linear subspaces. Usually, the appearance subspaces are computed by applying Principal Component Analysis (PCA) to a set of collected training images with known pose and illumination parameters.

Consider an n-pixel image I, which can be considered as a vector point in the high dimensional space R^n. At the training stage, a set of m images $\Psi = [I_1, I_2, ..., I_m]$ are taken from different poses $\Theta = [\theta_1, \theta_2, ..., \theta_m]$ under different illumination conditions $\Omega = [\omega_1, \omega_2, ..., \omega_m]$. We can compute the eigenvalues λ_i and eigenvectors V_i by applying eigen-decomposition of the $n \times n$ covariance matrix $Q = \Psi\Psi^T$. The eigenanalysis provides an efficient way of dimension reduction. As the image is a function of the pose and illumination parameters, the appearance manifold is low-dimensional. We can approximate the nonlinear appearance manifold using the linear subspace spanned by the first k eigenvectors [3]. More specifically, the least square approximation $\tilde{I}$ of the real image I can be computed as

$$\tilde{I} = \sum_{i=1}^{k} c_i V_i \tag{2.15}$$

where $c_i = I \cdot V_i$ represents the coordinate (projection) of the image in the appearance subspace spanned by the eigenvectors V_i. We can project all training images into the appearance subspace and acquire m discrete samples with known pose θ and illumination ω parameters. A parameterization of the appearance subspace $(\theta, \omega) = \Phi(c_1, c_2, ..., c_k)$ can then be computed from the discrete samples through, for instance, interpolation. During tracking, the input image is again projected into the appearance subspace. The pose and illumination parameters can be recovered using Φ.

The linear subspace model is first applied to object recognition by Murase and Na-

[3]Due to the highly nonlinear nature of the appearance manifold, k is usually chosen to be larger than the dimension of the underlying appearance manifold.

yar (MN95). They also demonstrated tracking 1D rotation of a rigid object. Later, Hager and Belhumeur (HB98), and Black and Jepson (BJ98) applied the appearance subspace model to visual tracking of rigid and articulated object respectively. The linear appearance model is simple and computationally efficient. However, globally approximating a nonlinear function using linear representation is fundamentally problematic. Systems adopting this linear model require higher dimensional parameters to model the underlying low dimensional nonlinear manifold. While projecting the problem to higher dimensions can reduce approximation errors, it also enlarges the search space and causes the optimization to be less stable. Moreover, as the dimensionality increases, the number of training images usually increases dramatically. The training process then becomes extremely tedious. For building a model for one specific target, it already involves tens if not hundreds of tuning and recording of the pose and illumination parameters.

To address the nonlinearity of the appearance manifold, a number of researchers have proposed to learn a nonlinear mapping between the image appearance and the pose parameters. For instance, Rahimi *et al.* (RRD05; RRD07) and Elgammal (Elg05) proposed to approximate the appearance function using a series of Radial Basis Functions (RBF). In theory, given enough input-output examples, any smooth nonlinear function can be learned using a family of functions such as RBF. However, that number of *enough* examples is usually high for a close approximation. For instance, 125 training images are used in (Elg05) to track an rigid object undergoing affine motion. To reduce the number of examples, a semi-supervised learning process that leverages the temporal coherence of video sequence is proposed in (RRD05). The authors reported tracking of a synthetic cube undergoing 3D rotation under a constant illumination. The training dataset includes 6 manually selected supervised images plus 1496 unsupervised images. Tested using the same training data, a mean estimation error of 4 degrees is reported.

In a closely related area, the problem of manifold learning has been actively studied recently. Manifold learning techniques recover the low-dimensional structure embedded

in the high-dimensional data. Usually, this is achieved by mapping the high-dimensional input data to low-dimensional global output coordinates, while preserving the local relationship between data points. For instance, Isomap (TdSL00) preserves the geodestic distances between local points, and LLE (RS00) preserves the linear interpolation weights between local points. The local neighborhood is defined by thresholding the pairwise distances between data samples. Clearly, the manifold needs to be densely sampled. Otherwise, neighboring points along the manifold are difficult to identify, the algorithms will then fail to recover the underlying structure. Experiment results show that, given enough training samples, these algorithms can almost always learn an accurate mapping from the high-dimensional data space to the low-dimensional parameter space. However, due to the requirement of dense sampling, it is impractical to apply learning methods to appearance-based 3D tracking. The study of manifold learning techniques, especially LLE, justify and emphasize that an age-old strategy also applies to appearance manifold: **approximating a globally nonlinear function can be done with a piecewise locally linear representation** .

2.5 Discussion

Feature-based approaches describe the scene as a collection of 3D geometric primitives such as points or lines. For model-based approaches, the 3D positions of these primitives in some global coordinate are given as priors. Motion is recovered by solving a 3D-to-2D pose estimation. By estimating the pose rather than the motion, model-based methods efficiently eliminate error accumulation. However, acquiring offline models can be tedious and sometimes infeasible, for instance, outdoor camera tracking.

In a different approach, feature-based SFM methods apply the epipolar constraint to corresponding features across multiple frames. The scene structure is recovered during tracking. As they estimate relative motion (essential matrix) rather than the pose,

feature-based SFM methods are prone to drifting. This issue, however, is well-balanced by the convenience of online modeling. Besides, with the development of multi-view techniques such as bundle adjustment, modern SFM systems have been able to track over a long period of time with little drift. The real challenge to feature-based SFM techniques comes from the task of corresponding features. Identifying and matching features in different views are hard in general, and can become really difficult in scenes presenting view-dependent appearance. For instance, scenes with occlusion boundaries, curved reflective surface, semitransparent surfaces, and specular reflections all change appearance dramatically as the viewpoint changes. Moreover, algorithms adopting the epipolar constraint are numerically unstable for small feature displacements. Therefore feature-based SFM methods usually generate less accurate results when estimating small scale motions.

Based on a local linear model, flow-based SFM systems handle small motions. Like feature-based approaches, they do not require a prior model of the scene. Motion and scene structure are both recovered using dense optical flow field. The performance of flow-based methods is usually restricted by the noisy input flow field. To compute flow estimates, flow-based methods usually make two assumptions: constant intensity and locally smooth depth. For simple scenes, [4] the constant intensity assumption can be considered valid under small motion. The assumption about local depth smoothness is usually violated at occlusion boundaries. Therefore, the application of flow-based SFM methods is usually restricted to scenes with smooth surfaces, or distant and close-to-planar scenes where an affine projection model can be used.

Appearance-based methods utilize the entire appearance observation. Compared with feature or flow measurements, appearance (intensity) measurements are easier to acquire. The computation time for feature extraction is eliminated, and the difficult

[4]Here normal indicate that the scene surface does not present complicated photometric properties, such as specularity or semi-transparency.

correspondence problem is avoided. Despite all of these advantages, appearance-based methods still receive less publicity among vision community. I believe **this unpopularity is, to a large extent, due to the difficulty of globally modeling scene appearance**. Existing 3D appearance-based methods assume an offline global appearance model of the scene. Such a global model is hard to acquire. Hundreds of training images need to be captured, all at known poses. Moreover, aside from poses, the scene appearance is also largely determined by illumination. Since illumination changes between the offline modeling stage and online tracking stage are usually inevitable, more training images need to be captured to accommodate this variation. Besides, adding illumination parameters projects the problem into a higher dimension. Estimation in this enlarged searching space becomes less stable. One day, these difficulties of global appearance modeling may eventually be solved, probably with a new generation of imaging hardwares. But what do we do now? My answer is to **go locally**.

Chapter 3

Differential Camera Tracking

The set of all possible n-pixel images of a rigid object lies on a low dimensional manifold embedded in R^n. Globally, this *appearance manifold* is highly nonlinear and requires numerous samples to be learned. But locally, due to its low-dimensional nature, it can be linearly approximated using a small number of neighboring image samples. In this chapter, I will introduce our novel differential tracking approach. The discussion will be focused on the problem formulation and the linearization of the local appearance manifold. The techniques for acquiring local appearance samples will be discussed in Chapter 4.

3.1 Tracking by surfing the appearance manifold

An image is a collection of pixel intensities. While an image is normally considered a 2D observation, it can also be formulated as a 1D vector that represents a point in a high dimensional space. The set of all n-pixel images forms an n-dimensional *appearance space*. The appearance space consists of image points captured at different places, under different lighting conditions. This dissertation addresses the set of all possible images of a rigid scene under a constant lighting captured from different perspectives. These images form a manifold of the appearance space called the *appearance manifold*.

Consider a rigid scene, a projective camera, and 3D rigid motions between them. At

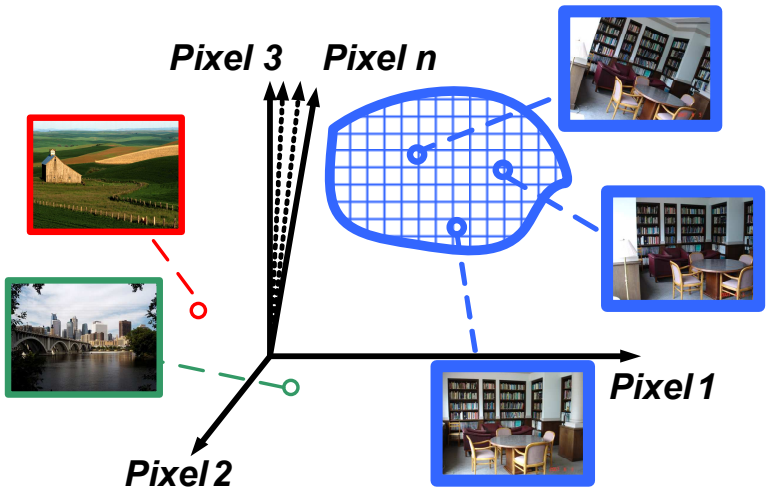


Figure 3.1: Appearance space and appearance manifold. Images of n pixels are represented as points in an n-dimensional appearance space. The right three images are taken in the scene under the same lighting. They lie on an appearance manifold (the blue meshed region).

each frame time, the camera captures from its current pose S an image I, a point in the appearance space. As the camera moves with respect to the scene and changes S, I also changes, moving along an appearance manifold. One can see that there exists a mapping from S to I, denoted as $I = F(S)$. Since the transformation space of the pose S has six DOF, the dimensionality of the appearance manifold is at most six. [1] Note that the appearance manifold is parameterized using pose parameters. Motion estimation can therefore be achieved through learning the appearance manifold.

Ideally, we would like to learn a global representation of the appearance manifold. However, due to its highly nonlinear nature, a global learning of the appearance manifold usually requires a large number of input-output examples, even for a static scene. The number can become extremely large for camera tracking when considering the possible movements of some scene objects. Moreover, as discussed in 2.5, to accommodate the almost inevitable changes of the lighting between the two stages of training and tracking, one usually needs to parameterize illumination. Modeling illumination increases the

[1] The dimension can be smaller than six in degenerate cases — for example, when a camera looks at a very distant scene.

dimensionality thus requires an even larger number of training samples. Due to the requirement of excessive sampling, global learning of the appearance manifold is usually infeasible in practice.

In fact, the task can be simplified. Most practical rigid motions are continuous. According to the speed of the motion and the frame rate of the camera, the relative displacement between the camera and the scene within a frame will be restricted within a certain region. Thus a local representation of the appearance manifold is sufficient for estimating the incremental motion. This observation motivates the development of our differential tracking approach, which is based on learning the local appearance manifold using a small number of neighboring samples. In particular, as the appearance manifold is 6D, a set of 7 samples in a general configuration is enough for a first order approximation. The differential tracking method works as follows. At each frame, we capture 7 samples and linearize the local appearance manifold around the current pose. At the next frame, when the camera moves to a different pose, we can compute the motion using the linearization from the previous frame. This pipeline continues as we capture another group of 7 samples and linearize around the new pose, then compute a new motion, and so on (see Figure 3.2). In this way, we generate a piecewise linear approximation of the appearance manifold along the path of the motion. As tracking is achieved by traversing a series of tangential planes of the manifold, we give this method a nickname *manifold surfing*.

3.2 Linearizing the local appearance manifold

Consider a rigid scene, a projective camera, and 3D rigid motions between them. Assume we simultaneously acquire a reference image I_0 at the current pose S_0, and m perturbed images I_k from nearby perspectives S_k $(k = 1, \ldots, m)$. We can compute m difference images $dI_k = I_k - I_t$ that correspond to m local motions $dS_k = S_k \dot{-} S_0$. Here $\dot{-}$ indicates

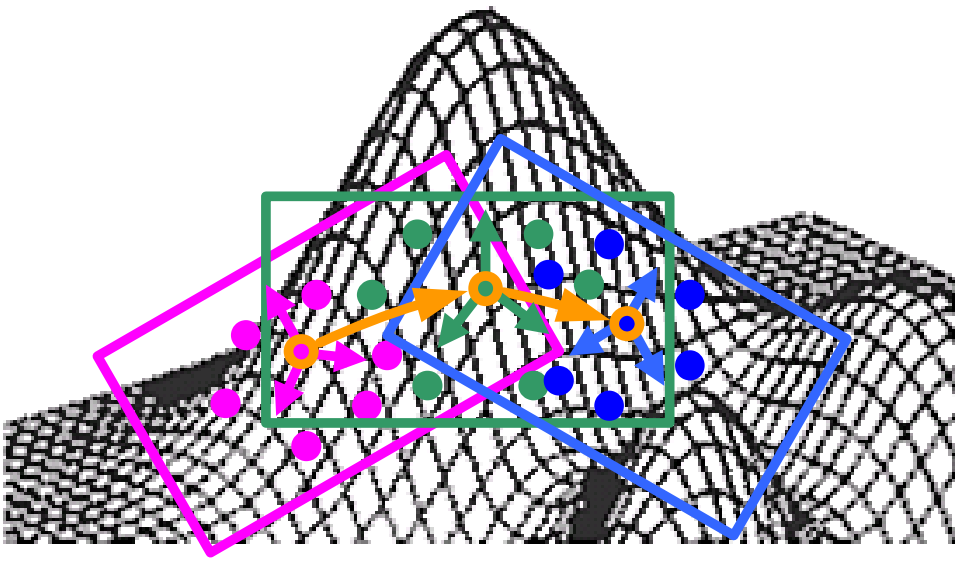


Figure 3.2: Manifold surfing. Tracking of three frames (two motions) is illustrated. The dots indicate local image samples captured at each of the three frames. The rectangles indicate the linear approximation of the local appearance manifold. The small coordinate frames indicate the relative poses between the camera and the scene at the three frames.

that dS_k is the rigid transformation from S_0 to S_k (see the discussion of dS in the next section for more details). We want to use these samples to linearize the appearance manifold around S_0:

$$I = I_0 + F\ (S \dot{-} S_0) \tag{3.1}$$

$$dI = F\ dS. \tag{3.2}$$

Here I and dI are n-pixel images represented as $n \times 1$ vectors, S and dS are 6×1 pose vectors, and F is the Jacobian (partial derivative) matrix $\partial I / \partial S$ of size $n \times 6$. Given m samples of dI_k and known dS_k, we can combine these sample vectors into matrices and write the linear equation as:

$$[dI_1, dI_2, \ldots, dI_m] = F\ \ [dS_1, dS_2, \ldots, dS_m] \tag{3.3}$$

If m is greater than 6 and the images and poses are not degenerate, the equation is of rank 6. Under the assumption of Gaussian noise, we can compute the least square

solution of F using the Moore-Penrose pseudo inverse as

$$F = [dI_1, dI_2, \ldots, dI_m] \ [dS_1, dS_2, \ldots, dS_m]^+ \tag{3.4}$$

The above discussion addresses the appearance manifold linearization problem in the general case where $m \geq 6$. For an efficient system, one would like to use minimum number of samples to expand the underlying 6D motion space. In this case, $m = 6$ and the Jacobian F becomes

$$F = [dI_1, dI_2, \ldots, dI_6] \ [dS_1, dS_2, \ldots, dS_6]^{-1} \tag{3.5}$$

3.3 Estimating incremental motion

Once a linear approximation is computed for the local appearance manifold, we can estimate the rigid motion using a linear solver. Assume at frame t, we capture a set of local image samples. We define one of them as the reference image I_0^t and its pose as the reference pose S_0^t. We then compute a Jacobian matrix F^t using the method described in the previous section. Then at frame $t+1$, we capture an updated reference image I_0^{t+1} at a new pose S_0^{t+1}. We can compute a temporal difference image $dI^t = I_0^{t+1} - I_0^t$ and estimate the motion $dS^t = S_0^{t+1} \dot{-} S_0^t$ as the least square solution of (3.6).

$$F^t \ dS^t = dI^t \tag{3.6}$$

Note that dS^t indicates the local rigid transformation from S_0^t to S_0^{t+1}. Specifically, dS^t can be represented as a 6D vector $[X^t, Y^t, Z^t, \alpha^t, \beta^t, \gamma^t]^T$, where $[X^t, Y^t, Z^t]$ represent the translation of the origin and $[\alpha^t, \beta^t, \gamma^t]$ are the Euler angles indicating the rotation of the frame axes. Using the homogeneous representation, the rigid transformation from

frame t to $t+1$ can be written as a 4×4 matrix P^t,

$$P^t = \begin{bmatrix} R^t & T^t \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

where $T^t = [X^t, Y^t, Z^t]^T$ and the 3×3 rotation matrix R^t can be written as the product of three matrices. Each of them represents the rotation around one of the three axes.

$$R^t = \begin{bmatrix} \cos\alpha^t & -\sin\alpha^t & 0 \\ \sin\alpha^t & \cos\alpha^t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta^t & 0 & \sin\beta^t \\ 0 & 1 & 0 \\ -\sin\beta^t & 0 & \cos\beta^t \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma^t & -\sin\gamma^t \\ 0 & \sin\gamma^t & \cos\gamma^t \end{bmatrix} \tag{3.8}$$

Note that the outer (left) matrix represents a rotation around one of the axis of the reference frame t, and the middle and the inner (right) matrices represent rotations around intermediate axes. In general, for an unique representation, one needs to define the order in which these rotations are performed. However, for small rotations, the results from all possible orderings of the 3 matrices can be considered the same. For instance, when all three Euler angles are less than 1°, the variation of the elements in R is less than 10^{-4}.

The advantage of using the homogeneous representation is that the incremental motion can be composited using matrix production. [2] Consider a sequence of l small motions between frame t and $t+l$. At each frame $t+i$ $(i = 1, \ldots, l)$, we sample and linearize the local appearance manifold around S_0^{t+i} and compute a rigid transformation matrix P^{t+i}. The transformation between S_0^t and S_0^{t+l} can be computed as

$$P = P^t P^{t+1} \cdots P^{t+l-1} \tag{3.9}$$

[2]This can be proved using the theory of Lie group and Lie algebra. The basic idea is that rigid motions form a Lie group $SE(3)$, and the tangent spaces at the origin of $SE(3)$ form a Lie algebra $se(3)$. Interested readers are referred to (SFP96).

The accumulated motion across these l frames can then be computed by decomposing P.

Chapter 4

Sampling the local appearance manifold

The local linearization of the appearance manifold requires input image samples. In this chapter, I describe techniques for sampling local appearance manifolds. As shown in Figure 4.1, there are two general approaches to acquiring appearance samples: through capturing or through rendering. The input image samples come from three sources: real images captured by physical cameras, synthetic images acquired by warping real images, and synthetic images acquired by rendering a 3D graphical model.

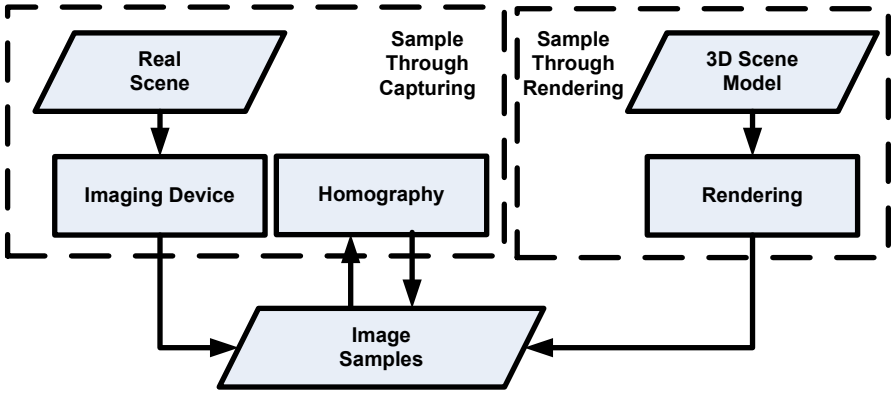


Figure 4.1: Techniques for acquiring appearance samples.

4.1 Sampling through capturing

As discussed in Section 2.4, almost all existing appearance-based methods address object tracking. To achieve that, they build an off-line appearance model of the target object

using a large number of training images, usually numbering in the hundreds. While such a training process is still feasible for modeling the appearance of an em object, it is normally impractical for camera tracking. A scene of moderate complexity usually consists of hundreds of individual objects, all can possibly move between the training stage and the tracking stage. Modeling the appearance of the whole scene thus requires building models for all the objects and initializing their poses before the tracking starts. On top of this, the model also needs to address the possible illumination changes between the training and tracking stages, as much time can pass between them.

When concerning the incremental motion, the impractical task of learning an off-line appearance model can be avoided. As I have described in Chapter 3, a linear approximation of the local appearance manifold can be applied to estimating an incremental motion. Since computing a local linearization only requires a small number of neighboring samples, we can now afford to directly sample the appearance manifold on line. Image samples from nearby viewpoints can be physically captured using closely placed cameras. In addition, using a homography transformation, we can warp a real image to acquire additional synthetic samples from the same viewpoint but different viewing directions.

4.1.1 Differential camera cluster

As shown in Figure 4.2 we have constructed a prototype Differential Camera Cluster (DCC) consisting of four synchronized and calibrated small baseline cameras: one *center* camera c_0 and three cameras c_1, c_2 and c_3 that are offset from the center. The coordinate frame of the cluster is defined to align with c_0. We call the other three cameras c_1, c_2 and c_3 *translational* cameras as they capture images from translated viewpoints.[1] At any point in time, the *center* camera and the *translational* cameras can be used to obtain four simultaneous appearance samples of the local appearance manifold. See the example

[1] In a general case, their axes do not need to be parallel with those of the *center* camera.

images indicated by the green (solid) arrows in Fig. 4.2. In addition we generate three warped images by rotating the image plane of c_0 around its three coordinate axes. See the example images indicated by the red (dashed) arrows in Fig. 4.2. One can consider these warped images as having been captured by three virtual *rotational* cameras c_4, c_5 and c_6, each with the same camera center as c_0 but with rotated axes. Thus at any frame the cluster effectively "captures" seven local appearance samples $I_0, \ldots, I_6$. In a non-degenerate case these images can be used to derive a first order approximation of the local appearance manifold as described in Chapter 3.

4.1.2 Inter-camera calibration

The camera cluster provides seven real-time appearance samples. Because the *rotational* images are warped versions of the *center* image, these four samples are from the same manifold. However, the *center* and the *translational* samples are captured using different cameras. To use these samples as if they are extracted from the same appearance manifold, we need to enforce geometric and photometric consistency across cameras.

Geometric calibration

To begin the geometric calibration, I use Bouguet's camera calibration toolkit (Bou08) to estimate and the projection matrix P_i for each of the four physical cameras ($i = 0, 1, 2, 3$ for the center and translational cameras c_0, c_1, c_2, c_3) [2]. I then decompose P_i as

$$P_i = K_i R_i [E| - C_i] \tag{4.1}$$

where E is the identity matrix, R_i is the rotation matrix that represents the orientation of the camera, C_i is the position of the camera center. Both R_i and C_i are defined in

[2]This calibration procedure also estimates lens distortion. The result parameters can be applied to warp the original images to generate radial-distortion-corrected images. The geometric calibration procedure described here assumes radial-distortion-corrected images

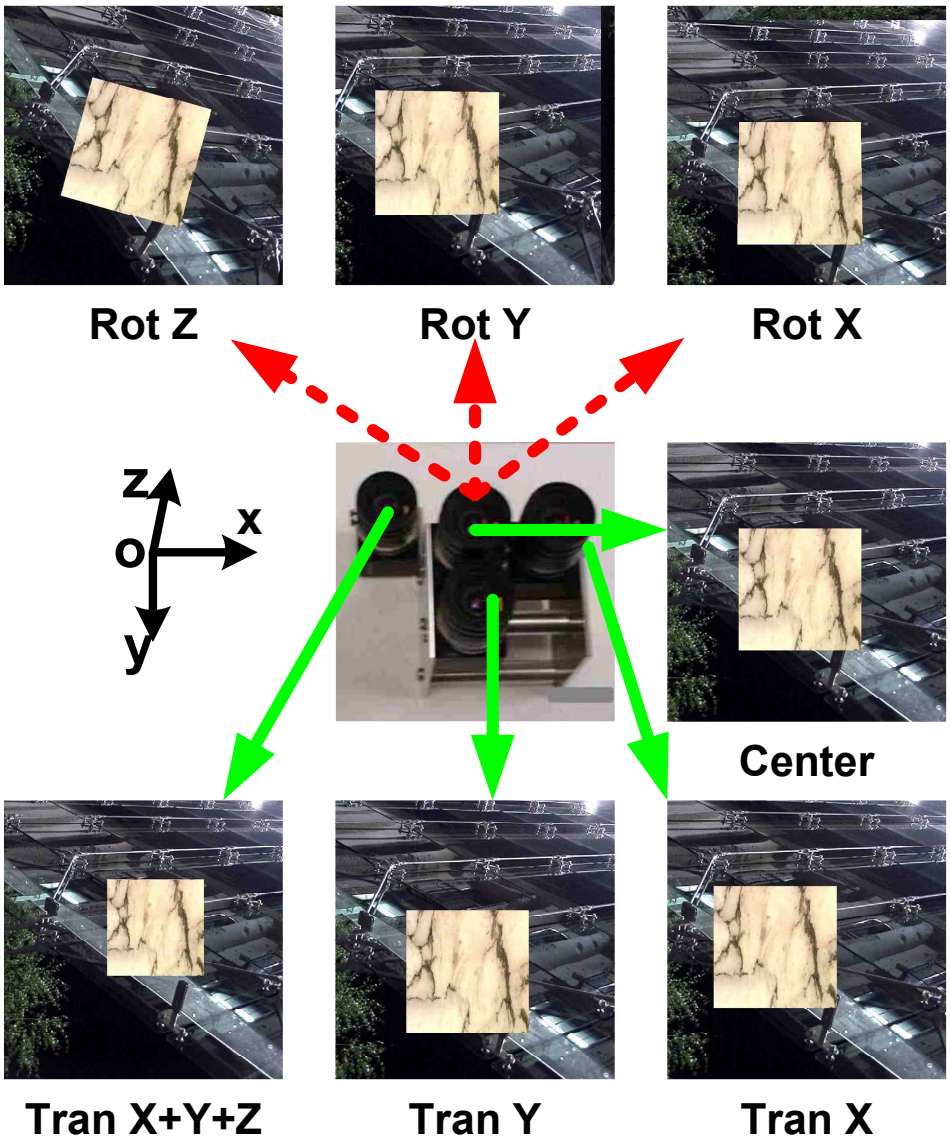


Figure 4.2: A prototype *differential camera cluster* (center) and illustrative images. Seven images were obtained: one center, three translated, and three rotated. Note that the images shown above were rendered with exaggerated baselines to make the differing perspectives more apparent. The center image shows a real DCC. The baselines between the cameras were about 34 mm in X and Y, and 66 mm in Z.

the same world coordinate system. K_i is called the *intrinsic matrix*. It represents the intrinsic parameters such as focal length, pixel size and principle point (see Appendix A for a detailed explanation). Due to the differences in these intrinsic parameters, different cameras capture different images from the same perspectives. Therefore images from

cameras of different intrinsic parameters do not lie on the same appearance manifold. To address this issue, I apply the intrinsic parameters of the center camera to all the translational cameras and create a new set of translational cameras $\tilde{c}_i$

$$\tilde{P}_i = K_0 R_i \left[E\right| - C_i] \tag{4.2}$$

The new images $\tilde{I}_i$ observed by $\tilde{c}_i$ can be computed by warping the original translational images I_i. As they share the same intrinsic parameters, they can be considered lying on the same appearance manifold. Note that the warping from I_i to $\tilde{I}_i$ only involves 2D transformations in the image plane. It does not change the position and orientation of the camera.

In (4.1) and (4.2), R_i and C_i are defined in the world coordinate system. To simplify notation, I align the world coordinate system with the coordinate system of the center camera c_0 [3], thus $R_0 = I$, $C_0 = 0$ and $P_0 = [K_0|0]$. Note that R_i and C_i represent the rotation and translation from c_0 to c_i. The corresponding sampled motion $dS_i = [X_i, Y_i, Z_i, \alpha_i, \beta_i, \gamma_i]$ can then be computed from them and applied to (3.5) of Section 3.2.

After describing the process of geometrically calibrating the translational cameras and computing their corresponding sampled motion, I now discuss the generation of virtual *rotational* cameras c_j (j=4,5,6). Again, the world coordinate system is assumed to be aligned with the center camera c_0. The projection matrices of c_j can be written as (4.3). The rotation matrix R_j is in one of the three forms of (4.4). Here α, β and γ indicate the rotation angles around the Z, Y, and X axes of c_0 respectively. The rotational image samples I_j can be computed from I_0 using the homography transformation indicated by (4.3).

$$P_j = K_0 R_j \left[E|0\right] = P_0 R_j \tag{4.3}$$

[3]The alignment of the world coordinate system and the camera coordinate system of c_0 can be achieved by multiplying R_i (i=1,2,3) with the inverse of R_0 and change C_i as $C_i' = R_0(C_i - C_0)$.

$$R_j = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad or \quad \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad or \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \tag{4.4}$$

Photometric calibration

In the DCC setup, I used CCD cameras with linear intensity responses. The intensity value $I_i(x, y)$ of a pixel (x, y) observed by camera c_i can be written as

$$I_i(x, y) = b_i + g_i l_i(x, y) \tag{4.5}$$

where $l_i(x, y)$ is the number of electrons received by the CCD pixel, g_i is the gain, and b_i is the *black value* — the intensity value when the camera looks at a pure black scene. In theory, we can achieve photometric consistency by physically adjusting the cameras such that they have the same gains and black values. However, this task is hard in practice. The target parameters are largely determined by the chip circuits, which are usually not adjustable.

Instead of a pure hardware solution, I adopt the hardware-software-combined approach proposed by Ilie (IW05). This approach consists of two steps. In the first (hardware) step, we tune the register settings of the translational cameras c_i. The goal is to adjust their response functions such that the intensity values of a 24-sample color target are consistent, or as close as possible to the reference image observed by c_0. In the second (software) step, we use the same color target to compute, for each c_i, a linear transformation $\hat{I}_i = k_i I_i + d_i$ that minimizes the sum of square distances

$$\sum_{x_0, y_0, x_i, y_i} (I_0(x_0, y_0) - (k_i I_i(x_i, y_i) + d_i))^2 \tag{4.6}$$

where (x_0, y_0) and (x_i, y_i) are the pixels covered by the same color sample in I_0 and I_i.

We then use $\hat{I}_i$ as the appearance samples.

4.2 Sampling through rendering

While a complete model of the entire scene is usually not available, in some situations a graphical model of the target object can be acquired off-line. Such a model may range from textured 3D polygons to a light field. As long as the model is renderable, one can consider generating appearance samples by means of graphics rendering.

As an example, I will describe in this section an approach to tracking a Lambertian object under directional lighting using a 3D textured polygonal model. Specifically, I will present a Kalman filter framework that iteratively refines the motion estimate, the illumination, and the model texture. Again, I use appearance samples from nearby perspectives to linearly approximate the local appearance manifold. The local appearance samples consist of two sets of images: real images of the object captured by the camera at the current pose, and synthetic images rendered at nearby poses around the current estimated pose. To address the illumination inconsistency between the synthetic and the real samples, I estimate an Illumination Normal Map (INM)—a mapping from the surface normal of a 3D point to the intensity ratio of its projections in the real and synthetic images. The INM defines a transformation that can be used to generate illumination-corrected synthetic images as if they were taken under the same illumination as the real images. Figure 7.8 shows the illumination correction of a cube.

4.2.1 Illumination Normal Map

Illumination variation between the real and synthetic scenes can be modeled as the intensity ratios of corresponding pixels in two images. Consider a 3D point p on the surface of the target object. The projection of p on the image plane is (x, y). Under the assumption of a Lambertian surface and directional lighting, the pixel intensity $I(x, y)$

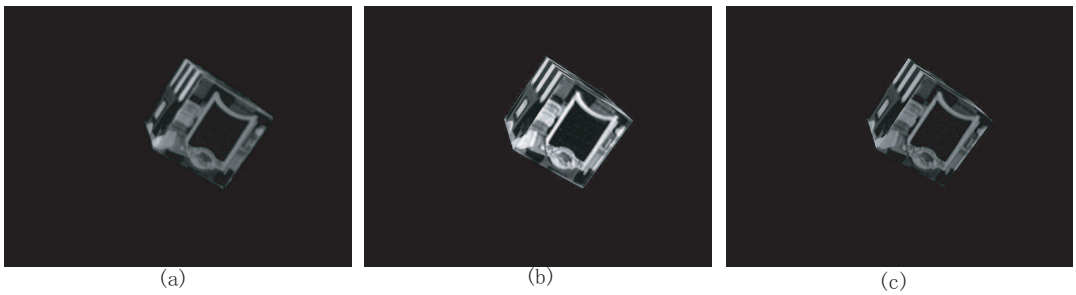


Figure 4.3: Illustration of illumination correction. (a) Real image. (b) Synthetic image without illumination correction. (c) Illumination corrected synthetic image.

can be written as

$$I(x,y) = \sum_k l_k \lambda(p) \max(L_k \cdot N(p), 0) \tag{4.7}$$

where $\lambda(p)$ is the nonnegative absorption coefficient (albedo) of the surface at p, L_k is the unit direction vector of light source k, l_k is the intensity of the light source k, $N(p)$ is the unit inward normal vector of the surface at p.

The intensity ratio r of corresponding pixels at (x, y) in the real and synthetic images can be computed as

$$r(x,y) = \frac{I(x,y)}{\hat{I}(x,y)} = \frac{\sum_k l_k \lambda(p) \max(L_k \cdot N(p), 0)}{\sum_k \hat{l}_k \lambda(p) \max(\hat{L}_k \cdot N(p), 0)} \tag{4.8}$$

where $\hat{I}(x,y)$ is the pixel intensity in the synthetic image, $\hat{l_k}$ and $\hat{L}_k$ represent the illumination of the modeled scene. Since the albedo $\lambda(p)$ is a constant, it can be moved outside the summation and canceled in the division. Thus (4.8) can be rewritten as

$$r(x,y) = \frac{\sum_k l_k \max(L_k \cdot N(p), 0)}{\sum_k \hat{l}_k \max(\hat{L}_k \cdot N(p), 0)} \tag{4.9}$$

One can see from (4.9) that the illumination ratio r of a point p is independent of the albedo a, and is a smooth function of the surface normal N. This key observation leads to the formulation of the INM, a lookup table that maps a normal to an illumination ratio. Each point in the INM corresponds to a unit normal vector N in the 3D space.

Its horizontal coordinate $\theta \in [0, 2\pi)$ and vertical coordinate $\phi \in [0, \pi)$ are defined by the projection of N in a fixed spherical coordinate system. Its value is the illumination ratio value ρ for the normal N (see Figure 4.4). An INM defines an illumination transformation between the real and the synthetic scene. A nice property of INM is that ρ is a smooth function of (θ, ϕ). Thus we can compute ρ at sparse grid-points and estimate the rest using bilinear interpolation.

Illumination correction

Consider a pixel (x, y) in the synthetic image. Given the geometry model and pose estimation of the object, the corresponding surface point p with normal $N(p)$ in the world space can be found through backprojection. Its coordinate (θ, ϕ) in the INM can then be derived by projecting $N(p)$ to a specified spherical space. Thus $\rho(\theta, \phi)$, which equals to $r(x, y)$, can be computed by comparing the corresponding pixel intensities in two images (see (4.8)).

Consider a Lambertian scene under directional lighting. In this case, points with the same surface normal are illuminated the same. Accordingly, their corresponding pixels should have a consistent illumination value within one frame. Moreover, when the scene lighting is static, this illumination ratio consistency constraint can be applied to pixels across multiple frames, as long as their associated scene points share the same surface normal. This means, with the presence of some measurement noise, we can acquire

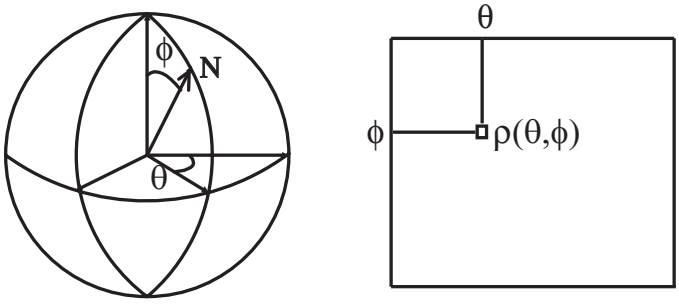


Figure 4.4: Spherical coordinate and INM.

multiple observations of the same illumination ratio variable. Based on this observation, I formulate the illumination estimation problem in a Kalman filter framework. The details will be explained later in the Section 4.2.2.

So far, we have made the assumption that the real and synthetic images are registered. However, due to the inevitable error in the pose estimate, the two images are almost never registered exactly in practice. Therefore the per-pixel-based illumination computation is subject to fail, for the ratio maybe computed by comparing the imaging of two different 3D surface points. However, since most object surfaces are piece-wise smooth and the illumination ratio is a smooth function of the surface normal, $r(x, y)$ should be spatially smooth. Discontinuity only happens at sharp edges where $N(p)$ changes abruptly. This observation leads to the area-based method for estimating illumination ratio. Specifically, we blur both the real and the synthetic images, and compute the illumination ratio from the corresponding pixels in the filtered images. Note that blurring across regions with normal discontinuities should be prohibited.

Once an INM is computed, we can use it to generate illumination-corrected synthetic images. For each pixel (x, y) in the synthetic image, we back-project it into the model space, find its normal and compute its INM coordinate (θ, ϕ). Since INM only defines ρ at some sparse grid points, we compute $\rho(\theta, \phi)$ by bilinear-interpolating the neighboring grid points. We then set the pixel intensity ratio $r(x, y) = \rho(\theta, \phi)$. In this way, we compute the intensity ratio for each pixel and form the intensity ratio map r. Finally we acquire the illumination-corrected image $\hat{I}(x, y) = I(x, y) * r(x, y)$.

4.2.2 Iterative estimation

We implement our model-based tracking system using the Kalman filter framework (see Appendix C for a introduction of the Kalman filter). The state of the system X is modeled as $[S, M, T]$, where S represents the object pose parameters, M indicates the INM and T denotes the texture of the object. Pixel intensities I are used as the system

measurements. The estimated measurement of a pixel (x, y) in the synthetic image is computed as:

$$I(x, y, S, M, T) = M(H_M(S, x, y)) \cdot T(H_T(S, x, y)) \tag{4.10}$$

where H_M and H_T are the back-projection functions. Given a pose estimate $\hat{S}$, they define the mapping from an image pixel at (x, y) to an INM pixel at (θ, ϕ) and a texture pixel at (u, v). In fact, the camera calibration and object model are also implicitly used by H_M and H_T. However, since they are constant throughout the image sequences, they are not included in the equation. One can see that (4.10) can be divided into two parts. The fist part $M(H_M(S, x, y))$ defines the illumination correction. The second part $T(H_T(S, x, y))$ represents the texture mapping.

H_M defines the mapping from the image coordinate system to the INM coordinate system. A pixel in the image may be mapped to a non-grid point in the INM. In this case, we compute the illumination ratio of the pixel by interpolating neighboring grid points, where the values are defined. A similar issue holds for the image-texture mapping H_T. Besides, the projection of a pixel in the texture space is an ellipsoid rather than a point (GH86). When a relatively high-resolution texture is provided, the ellipsoid usually covers multiple texture pixels. Therefore we compute the value of the image pixel as a weighted average of the covered texture pixels. (4.10) can then be rewritten as

$$I(x, y, S, M, T) = \sum_{\theta,\phi} M(\theta, \phi)\xi_M(H_M(S, x, y), \theta, \phi) \cdot \sum_{u,v} T(u, v)\xi_T(H_T(S, x, y), u, v) \tag{4.11}$$

where ξ_M and ξ_T are the filtering kernels centered at $H_M(S, x, y)$ and $H_T(S, x, y)$. Note that both H_M and H_T are nonlinear functions of the pose parameters S.

With the above definition of the system model, we use a Kalman filter to estimate the object pose, INM, and texture in a recursive fashion. The processing at each frame consist of four steps: prediction, pose update, illumination estimation and texture re-

finement.

Prediction

We predict the state $\hat{X}^-$ based on the previous state estimate $\hat{X}$ using (C-1). The system state consists of the object pose S, texture T and Illumination Normal Map M. T is fixed for a given object. S can be predicted using a simple motion model. In our system, we use a constant-speed motion model. M can be predicted using different dynamic models. For instance, if we assume static lighting, M would be constant, and an identity matrix should be used for the state-transition matrix A. However if for example the light sources' intensity and/or direction change, we can use a non-identity dynamic model matrix to predict M. The prediction step is important as it provides the initialization for the remaining steps.

Pose update

In this step, we fix the current estimate of INM $\hat{M}$ and texture $\hat{T}$, and update the pose estimate $\hat{S}$. Since $H_M(S)$ and $H_T(S)$ involve perspective projection, the intensity measurement I is a nonlinear function of the pose S (see (4.11)). This nonlinear estimation problem can be solved using an Extended Kalman Filter (EKF) (see Appendix C for a introduction of the EKF). We linearize $\hat{I}$ around the current pose estimate $\hat{S}^-$. Specifically, we use $\hat{M}$ and $\hat{T}$ to render illumination-corrected appearance samples at $\hat{S}^-$ and around it. We then use these synthetic images to compute the Jacobian matrix $F = \partial I/\partial S$ (see (3.5) in Section 3.2). The measurement function $\hat{I}$ can then be written as

$$\hat{I}(\hat{S}) = I_{\hat{S}^-} + F(\hat{S} - \hat{S}^-) = (I_{\hat{S}^-} - F\hat{S}^-) + F\hat{S} \tag{4.12}$$

where $I_{\hat{S}^-}$ is the image rendered at $\hat{S}^-$. After acquiring this local linearization, we apply the real image measurements (captured by the camera) to compute a new pose estimate $\hat{S}$ using an EKF.

Illumination estimation

In this step, we estimate the values of INM pixels by comparing the image intensities of corresponding pixels in the synthetic and the real image. This time, we fix the pose $\hat{S}$ and the texture $\hat{T}$. For a pixel at (x_0, y_0) whose normal is at $(\theta_0, \phi_0) = H_M(\hat{S}, x_0, y_0)$ with neighboring INM grid points (θ_k, ϕ_k), the measurement function and Jacobian matrix can be written as

$$\hat{I}(\hat{M}) = c(x_0, y_0) \sum_{k=1}^{n} \hat{M}(\theta_k, \phi_k) \xi_M(\theta_0, \phi_0, \theta_k, \phi_k) \tag{4.13}$$

$$H(\hat{M}) = c(x_0, y_0)[\xi_M(\theta_0, \phi_0, \theta_1, \phi_1), ..., \xi_M(\theta_0, \phi_0, \theta_n, \phi_n)] \tag{4.14}$$

where $c(x_0, y_0) = \sum_{u,v} \hat{T}(u, v) \xi_T(H_T(\hat{S}, x_0, y_0), u, v)$ is the intensity of pixel (x_0, y_0) computed directly from texture mapping, $\xi_M(\theta_0, \phi_0, \theta_k, \phi_k)$ is the interpolation kernel for computing $\hat{M}(\theta_0, \phi_0)$, and n is the size of the interpolation window. Currently, we compute $\hat{M}(\theta_0, \phi_0)$ using bi-linear interpolation of the surrounding $n = 4$ pixels.

The intensity measurement in (4.13) is a linear function of $\hat{M}$. So we apply the standard Kalman measurement update equations to estimate $\hat{M}$. Specifically, we send the intensity measurement of each pixel to the Kalman filter one at a time. Then H is an n-vector with elements $w_k = c(x_0, y_0)\ \xi_M(\theta_0, \phi_0, \theta_k, \phi_k)$, Kalman gain K is an n-vector and the noise covariance R becomes a scaler σ^2. In addition, we make the assumption that pixel values in the INM are independent of each other. Therefore we represent the process covariance P as an $n \times n$ diagonal matrix. Given the definition of these matrices, the measurement update equations (C-4)-(C-6) can be written as

$$K_k = P_{kk}^{-} w_k (\sum_{j=1}^{n} (P_{jj}^{-} w_j^2) + \sigma^2)^{-1} \tag{4.15}$$

$$\hat{M}_k = \hat{M}_k^{-} + K_k (I(x_0, y_0) - \hat{I}(x_0, y_0)) \tag{4.16}$$

$$P_{kk} = (1 - K_k w_k) P_{kk}^- \tag{4.17}$$

From the above equations one can see that the residual of intensity measurement of an image pixel is used to update the values of the INM pixels that contribute in computing its illumination ratio.

Texture refinement

We perform the texture refinement in a way similar to the illumination estimation. In this step, we keep the pose $\hat{S}$ and illumination $\hat{M}$ constant. For a synthetic pixel at (x_0, y_0) with texture coordinate $(u_0, v_0) = H_T(\hat{S}, x_0, y_0)$, we derive the following equations

$$\hat{I}(\hat{T}) = l(x_0, y_0) \sum_{k=1}^{n} \hat{T}(u_k, v_k)\ \xi_T(u_0, v_0, u_k, v_k) \tag{4.18}$$

$$H(\hat{T}) = l(x_0, y_0)[\xi_T(u_0, v_0, u_1, v_1), ..., \xi_T(u_0, v_0, u_n, v_n)] \tag{4.19}$$

where $l(x_0, y_0) = \sum_{\theta,\phi} \hat{M}(\theta, \phi) \xi_M(H_M(\hat{S}, x, y), \theta, \phi)$ is the illumination ratio at (x_0, y_0), and (u_k, v_k) are the n texture pixels inside the projection ellipsoid of pixel (x_0, y_0). Here $\hat{I}$ is a linear combination of texture intensities. Currently we use equal weights for all the texture pixels, i.e. $\xi_T(u_0, v_0, u_k, v_k)$ equals to $1/n$ if (u_k, v_k) is inside the projection ellipsoid around (u_0, v_0), or 0 otherwise.

The measurement estimate $\hat{I}(\hat{T})$ in (4.18) is a linear function of $\hat{T}$. Therefore, we can use a linear Kalman filter to estimate the texture $\hat{T}$ using (4.15)-(4.17). The elements in the H vector become $w_k = l(x_0, y_0)\ \xi_T(u_0, v_0, u_k, v_k)$.

Chapter 5

Analysis on sampling appearance manifolds

In the previous chapters, I have introduced the algorithms and techniques for sampling and linearizing the local appearance manifold. In theory, given seven samples one can always generate a linear approximation of the sampled appearance manifold, and such a linearization will be accurate within a *local* region. Intuitively, we know that for an accurate linear approximation, the displacement (between samples) needs to be small and the underlying appearance manifold (or usually a filtered version of it) needs to be smooth. But what exactly do *small* and *smooth* mean here?

From a signal processing point of view, the process of linearizing the local appearance manifold can be considered as reconstructing intensity signals using local samples (images). We can then apply the *sampling theorem* to determine the minimum sampling rate that avoids aliasing. Based on this observation, I have developed some Fourier spectral analysis on sampling the local appearance manifold. A similar analysis on sampling 4D light fields has been proposed in (CTCS00). Here I extend it to full 3D cases.

In the context of the appearance-based differential tracking, I apply the Fourier analysis to guide the design of a DCC and to determine the filter kernels for blurring images (smoothing appearance manifold). On a larger scope, I believe this analysis can also be used to explain the commonly used sub-pixel motion constraint in the optical flow field. Since the foundation for optical flow estimation, the brightness constancy

equation, is also based on a local linearization of the pixel intensities.

5.1 Sampling the appearance signals

The appearance of a static scene under constant lighting lies on a 6D manifold. Assume this appearance is measured by a camera and presented as an image consisting of pixel intensities/colors. For a specific pixel, its intensity can be represented as an 8D signal $I(x, y, T_x, T_y, T_z, R_x, R_y, R_z)$, where $[T_x, T_y, T_z, R_x, R_y, R_z]$ is the camera pose and $[x\ y]$ is the image coordinate of the pixel. In the differential tracking approach, we acquire discrete samples I_s of the continuous signal I.

The Fourier transform $\mathcal{I}$ of the continuous pixel intensity signal I is an 8D function (shown in (5.1)). Here $\mathcal{F}$ denotes the Fourier transform, $[\omega_u, \omega_v]$ are the angular frequencies of the spectral domain in the image dimensions, and $[\omega_{T_x}, \omega_{T_y}, \omega_{T_z}, \omega_{R_x}, \omega_{R_y}, \omega_{R_z}]$ are the angular frequencies of the spectral domain in the motion dimensions. The sampled signal I_s can be represented as the multiplication of I and an 8D comb function S (see (5.2)). $\triangle x, ..., \triangle R_z$ are the sampling intervals in each of the 8 dimensions. The Fourier transform of the sampled signal $\mathcal{I}_s$ is the convolution of two Fourier transforms $\mathcal{I}$ and $\mathcal{S}$ (see (5.3)). Since $\mathcal{S}$ is also an 8D comb function, $\mathcal{I}_s$ is a composition of replicas of $\mathcal{I}$ with intervals $\frac{2\pi}{\triangle x}, ..., \frac{2\pi}{\triangle R_z}$ in the 8 dimensions of the spectral domain.

$$\mathcal{I}(\omega_x, \omega_y, \omega_{T_x}, \omega_{T_y}, \omega_{T_z}, \omega_{R_x}, \omega_{R_y}, \omega_{R_z}) = \mathcal{F}\{I(x, y, T_x, T_y, T_z, R_x, R_y, R_z)\} \tag{5.1}$$

$$I_s = I * S = I(x, ..., R_z) * \sum_{m_x=-\infty}^{\infty} ... \sum_{m_{R_z}=-\infty}^{\infty} \delta(x - m_x \triangle x, ..., R_z - m_{R_z} \triangle R_z) \tag{5.2}$$

$$\begin{aligned} \mathcal{I}_s &= \mathcal{F}(I * S) = \mathcal{F}(I) \otimes \mathcal{F}(S) = \mathcal{I} \otimes \mathcal{S} \\ &= \mathcal{I}(\omega_x, ..., \omega_{R_z}) \otimes \textstyle\sum_{m_x=-\infty}^{\infty} ... \sum_{m_{R_z}=-\infty}^{\infty} \delta(\omega_x - m_x \frac{2\pi}{\triangle x}, ..., \omega_{R_z} - m_{R_z} \frac{2\pi}{\triangle R_z}) \end{aligned} \tag{5.3}$$

We can apply the well-known Nyquist–Shannon theorem to the sampling of the appearance signal. According to the theorem, aliasing occurs in the reconstructed signal

if the spectral replicas of the sampled signal overlap. Exact reconstruction is only possible when the sampling frequency is greater than twice the signal bandwidth. Since our tracking algorithm reconstructs (linearizes) the appearance (pixel intensity) signal from its local samples, it is clear that we need to sample at least twice as frequently as the bandwidth of the appearance signal in each of the 8 dimensions. The solution is then straight forward except that we do not know the bandwidth of the appearance signal. In the following sections, I will prove that the appearance signal, more specifically the camera-observed appearance signal, is bandlimited. I will show that its bandwidth is determined by the image resolution and the scene depth.

5.2 Fourier analysis of the local appearance manifold

In this section, I compute the 8D Fourier transform of the local appearance manifold. Without loss of generality, I will choose the reference pose to be [0 0 0 0 0 0]. For a more concise representation, I will first compute the Fourier transform of the reference image I_0 and six 1D sub-manifolds I_{T_x}, I_{T_y}, I_{T_z}, I_{R_x}, I_{R_y}, I_{R_z} that correspond to six basic motions: three translations along the axes and three rotations around the axes.

$$
\begin{aligned}
I_0(x,y) &= I(x,y,0,0,0,0,0,0) \\
I_{T_x}(x,y,T_x) &= I(x,y,T_x,0,0,0,0,0) \\
I_{T_y}(x,y,T_y) &= I(x,y,0,T_y,0,0,0,0) \\
I_{T_z}(x,y,T_z) &= I(x,y,0,0,T_z,0,0,0) \\
I_{R_x}(x,y,R_x) &= I(x,y,0,0,0,R_x,0,0) \\
I_{R_y}(x,y,R_y) &= I(x,y,0,0,0,0,R_y,0) \\
I_{R_z}(x,y,R_z) &= I(x,y,0,0,0,0,0,R_z)
\end{aligned}
\tag{5.4}
$$

I will then combine the analysis on the reference images and the six 3D appearance functions to compute the entire Fourier transform of the 8D appearance signal.

For simplicity of representation, I will begin the Fourier analysis on the local appearance manifold of a Lambertian scene where the brightness constancy assumption holds. I will extend the analysis on more general cases later in Section 5.3. When computing the Fourier transform of the Lambertian appearance manifold, I will use optical flow to relate current appearance samples to the reference appearance samples. I have computed the flow for the six basic motions. The result is shown in Table 5.1. Note that the rotational flow fields listed in Table 5.1 are computed using small angle approximation. While such a approximation can not be applied to large rotations, it is appropriate for our analysis of the local appearance manifold. Readers can read Appendix B for details.

Motion Type	Motion Magnitude	H-Flow(u)	V-Flow(v)
X Translation	T_x	$\frac{f}{Z}T_x$	0
Y Translation	T_y	0	$\frac{f}{Z}T_y$
Z Translation	T_z	$-\frac{x}{Z}T_z$	$-\frac{y}{Z}T_z$
X Rotation	R_x	$-\frac{xy}{f}R_x$	$-(f+\frac{y^2}{f})R_x$
Y Rotation	R_y	$(f+\frac{x^2}{f})R_y$	$\frac{xy}{f}R_y$
Z Rotation	R_z	$-yR_z$	xR_z

Table 5.1: Optical flow of six 1D motions. f denotes the camera focal length. x and y are the image coordinates. Z is the scene depth at pixel $[x, y]$

5.2.1 Fourier analysis of the reference image

The 2D Fourier transform of the reference image $I_0(x, y)$ acquired at pose [0 0 0 0 0 0] is written in (5.5).

$$\mathcal{I}_0(\omega_x, \omega_y) = \mathcal{F}\{I_0(x, y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I_0(x, y)e^{-j(\omega_x x+\omega_y y)}dxdy \tag{5.5}$$

Note that I_0 here is not the real appearance function $\hat{I}$ but a filtered and sampled version of it. First, the original image is captured by a camera at a certain resolution. Each intensity value is a weighted integral of the light rays arriving in the area of the

associated CCD pixel. Moreover, to further smooth the appearance function, we apply an additional software low-pass filtering process to the original camera image. Therefore even if the spectral support of the underlying scene appearance can be unbounded, the support of the observed appearance function I_0 is bounded by the bandwidth of the low-pass filter B_P. I_0 is sampled at a certain resolution. Assume I_0 has pixel aspect ratio of 1 and denote $\triangle x$ as the pixel size. $\triangle x$ and B_P are related. From sampling theorem, we know that the shortest frequency of the appearance signal that can be recovered from the image is of period of 2 pixels. Thus for a fixed $\triangle x$, we should choose $B_P \leq \pi/\triangle x$ to avoid aliasing. Based on this observation, we can compute the practical bandwidth of the reference image B_I using the bandwidth of the observed scene texture B_S and the resolution of the camera $\triangle x$ (shown in (5.6)).

$$|\omega_x| \ , \ |\omega_y| \leq \ B_I = min(B_S, \frac{\pi}{\triangle x}) \tag{5.6}$$

Note the integral in (5.5) is from $-\infty$ to ∞. The same lower and upper limits apply to all the integrals in this section. To simplify representation, I will not specify them again.

5.2.2 Fourier analysis of X or Y translation

I now compute the Fourier transform of $I_{T_x}(x, y, T_x)$. This 1D appearance manifold consists of the infinite set of images captured by a camera translating along the X axis. Under the brightness constancy assumption, we can relate I_{T_x} to I_0 using the X-translation flow listed in Table 5.1. Here Z is the scene depth at pixel (x, y). To begin, let us assume that the scene is a plane at a constant depth Z_0.

$$I_{T_x}(x, y, T_x) = I_{T_x}(x - \frac{f}{Z_0}T_x, y, 0) = I_0(x - \frac{f}{Z_0}T_x, y) \tag{5.7}$$

The Fourier transform of I_{T_x} can be computed as

$$\begin{aligned}\mathcal{I}_{T_x}(\omega_x,\omega_y,\omega_{T_x}) &= \int\int\int I_{T_x}(x,y,T_x)e^{-j(\omega_x x+\omega_y y+\omega_{T_x}T_x)}dxdydT_z \\ &= \int\int\int I_0(x-\tfrac{f}{Z_0}T_x,y)e^{-j(\omega_x x+\omega_y y+\omega_{T_x}T_x)}dxdydT_z\end{aligned} \tag{5.8}$$

Applying (5.9) to (5.8) and substituting ω_x with ω_u, we get (5.10)

$$u = x - \frac{f}{Z_0}T_x \qquad or \qquad x = u + \frac{f}{Z_0}T_x \tag{5.9}$$

$$\begin{aligned}\mathcal{I}_{T_x} &= \int\int\int I_0(u,y)e^{-j(\omega_u(u+\frac{f}{Z_0}T_x)+\omega_y y+\omega_{T_x}T_x)}dudydT_x \\ &= \int\int I_0(u,y)e^{-j(\omega_u u+\omega_y y)}dudy\int e^{-j(\omega_{T_x}+\frac{f}{Z_0}\omega_u)T_x}dT_x \\ &= \mathcal{I}_0(\omega_u,\omega_y)\delta(\omega_{T_x}+\tfrac{f}{Z_0}\omega_u) = \mathcal{I}_0(\omega_u,\omega_y)\mathcal{H}_{T_x}(\omega_u,\omega_v,\omega_{T_z})\end{aligned} \tag{5.10}$$

(5.10) shows that the spectral support $\mathcal{I}(\omega_u,\omega_y,\omega_{T_x})$ of a planar scene at constant depth Z_0 is a plane in the 3D Fourier domain that is defined by

$$\omega_{T_x} + \frac{f}{Z_0}\omega_u = 0 \tag{5.11}$$

Now, let us consider a scene with a varying depth. Let the minimum and maximum depth be Z_{min} and Z_{max}. The spectral support is then bounded by two planes: $\omega_{T_x} + \frac{f}{Z_{min}}\omega_u = 0$ and $\omega_{T_x} + \frac{f}{Z_{max}}\omega_u = 0$, as shown in Figure 5.3. The first row of Figure 5.3 shows a near plane a far plane and a tilted plane in between. The second row of Figure 5.3 shows the $X - T_x$ plane of the appearance function I_{T_x} [1]. The Fourier transforms of these 2D cross-sections of I_{T_x} are presented in the third row. We can see that the spectral support of the tilted plane is bounded by the spectral supports of the near and the far planes.

Applying (5.6) to (5.11), we can compute the bandwidth of $\mathcal{I}_{T_x}$ in the T_x dimension

[1] By fixing Y, the 2D cross-section of the appearance function I_{T_x} defines an epipolar image (EPI).

using (5.12). Here Z_{min} is the smallest scene depth.

$$|\omega_{T_x}| = |\frac{f}{Z}\omega_u| \leq \frac{f}{Z_{min}} \min(B_s, \frac{\pi}{\triangle x}) \tag{5.12}$$

The lowest sampling density along the X axis $\triangle T_x$ can then be computed as

$$\triangle T_{xmin} = \frac{\pi}{|\omega_{T_x}|} \leq \frac{Z_{min}}{f} \max(\frac{\pi}{B_s}, \triangle x) \tag{5.13}$$

The maximum horizontal and vertical image flow with respect to X translation can be computed as:

$$\begin{cases} l_x = \frac{f}{Z}\triangle T_x = \max(\frac{\pi}{B_s}, \triangle x) \\ l_y = 0 \end{cases} \tag{5.14}$$

For Y translation, the spectral support I_{T_y} can be analyzed using the same approach. The resulting minimum sampling density $\triangle T_y$ and the maximum image flow l_y have the same form as $\triangle T_x$ and l_x.

5.2.3 Fourier analysis of Z translation

Now, let us compute the Fourier transform of $I_{T_z}(x, y, T_z)$. This 1D appearance manifold consists of the infinite set of images captured by a camera translating along the Z axis. Again, we begin by studying the Fourier transform of a plane at a constant depth Z_0. Using the Z-translation flow listed in Table 5.1, we can represent I_{T_z} as

$$I_{T_z}(x, y, T_z) = I_{T_z}(x + \frac{x}{Z_0}T_z, y + \frac{y}{Z_0}T_z, 0) = I_0(x + \frac{x}{Z_0}T_z, y + \frac{y}{Z_0}T_z) \tag{5.15}$$

The Fourier transform of I_{T_z} can be computed as

$$\begin{aligned} \mathcal{I}_{T_z}(\omega_u, \omega_v, \omega_{T_z}) &= \int\int\int I_{T_z}(x, y, T_z) e^{-j(\omega_x x + \omega_y y + \omega_{T_z} T_z)} dx dy dT_z \\ &= \int\int\int I_0(x + \tfrac{x}{Z_0}T_z, y + \tfrac{y}{Z_0}T_z) e^{-j(\omega_x x + \omega_y y + \omega_{T_z} T_z)} dx dy dT_z \end{aligned} \tag{5.16}$$

Let

$$\begin{cases} u = x + \frac{T_z}{Z_0}x \\ v = y + \frac{T_z}{Z_0}y \end{cases} \quad or \quad \begin{cases} x = u/(1+\frac{T_z}{Z_0}) \approx (1-\frac{T_z}{Z_0})u \\ y = v/(1+\frac{T_z}{Z_0}) \approx (1-\frac{T_z}{Z_0})v \end{cases} \tag{5.17}$$

Note that the approximation shown in (5.17) can be applied to our analysis of the local appearance manifold, where T_z is usually small compared to the scene depth. To give an idea, the approximation error when $T_z = Z_0/10$ is 1%. Applying (5.17) to (5.16) and substituting ω_x ω_y with ω_u ω_v, we have

$$\begin{aligned} \mathcal{I}_{T_z}(\omega_x, \omega_y, \omega_{T_z}) &= \int\int\int I_0(u,v) e^{-j(\omega_u(1-\frac{T_z}{Z_0})u + \omega_v(1-\frac{T_z}{Z_0})v + \omega_{T_z}T_z)}(1-\tfrac{T_z}{Z_0})^2 du dv dT_z \\ &= \int\int I_0(u,v) \int (1-\tfrac{T_z}{Z_0})^2 e^{-j(-\omega_u\frac{u}{Z_0} - \omega_v\frac{v}{Z_0} + \omega_{T_z})T_z} dT_z e^{-j(\omega_u u + \omega_v v)} du dv \end{aligned} \tag{5.18}$$

(5.18) can be considered as the Fourier transform of a product of two 2D signals: I_0 and an integral $\mathbb{I}_{T_z}$.

$$\mathbb{I}_{T_z}(u,v) = \int (1-\frac{T_z}{Z_0})^2 e^{-j(-\omega_u\frac{u}{Z_0} - \omega_v\frac{v}{Z_0} + \omega_{T_z})T_z} dT_z \tag{5.19}$$

Thus $\mathcal{I}_{T_z}$ can be computed as the convolution of the two Fourier transforms

$$\mathcal{I}_{T_z}(\omega_u, \omega_v, \omega_{T_z}) = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{F}\{\mathbb{I}_{T_z}(u,v)\} = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{H}_{T_z}(\omega_u, \omega_v, \omega_{T_z}) \tag{5.20}$$

Assuming $-\omega_u\frac{u}{Z_0} - \omega_v\frac{v}{Z_0} + \omega_{T_z} \neq 0$,

$$\begin{aligned} \mathcal{F}\{\mathbb{I}_{T_z}(u,v)\} &= \int\int\int (1-\tfrac{T_z}{Z_0})^2 e^{-j(-\omega_u\frac{u}{Z_0} - \omega_v\frac{v}{Z_0} + \omega_{T_z})T_z} e^{-j(\omega_u u + \omega_v v)} du dv dT_z \\ &= \int (1-\tfrac{T_z}{Z_0})^2 e^{-j\omega_{T_z}T_z} \int e^{-j(\omega_u(1-\frac{T_z}{Z_0})u)} du \int e^{-j(\omega_v(1-\frac{T_z}{Z_0})v)} dv dT_z \\ &= \int (1-\tfrac{T_z}{Z_0})^2 e^{-j\omega_{T_z}T_z} \delta(1-\tfrac{T_z}{Z_0}) \delta(1-\tfrac{T_z}{Z_0}) dT_z \\ &= (1-\tfrac{T_z}{Z_0})^2 e^{-j\omega_{T_z}T_z}|_{Z_0=T_z} \quad = 0 \end{aligned} \tag{5.21}$$

we know from (5.21) that $\mathcal{F}\{\mathbb{I}_{T_z}\} = 0$ implies $\mathcal{I}_{T_z} = 0$. Therefore (5.21) shows that the spectral support $\mathcal{I}_{T_z}$ is bounded within the region:

$$-\omega_u \frac{u}{Z_0} - \omega_v \frac{v}{Z_0} + \omega_{T_z} = 0 \tag{5.22}$$

For a scene with a varying depth. The spectral support is bounded by $-\omega_u \frac{u}{Z_{min}} - \omega_v \frac{v}{Z_{min}} + \omega_{T_z} = 0$, where Z_{min} is the minimum scene depth. The bounded effect of the spectral support is illustrated in Figure 5.3. The first row shows a near plane a far plane and a tilted plane in between. The fourth row shows the $Z - T_z$ plane of the appearance function I_{T_z}. The Fourier transform of the 2D cross-sections of I_{T_z} are presented in the fifth row. We can see that the spectral support of the tilted plane is bounded within the spectral supports of the near plane.

Note that the maximum value of the horizontal and vertical image coordinates u and v are determined by the camera Field of View (FOV). Let θ be the FOV of the camera (assuming equal horizontal and vertical FOV), we have

$$\begin{cases} |u| \leq f \tan(\theta/2) \\ |v| \leq f \tan(\theta/2) \end{cases} \tag{5.23}$$

Apply (5.6) and (5.23) to (5.22), we can compute the bandwidth of $\mathcal{I}_{T_z}$ in T_z dimension using (5.24).

$$|\omega_{T_z}| = |\frac{u}{Z}\omega_u + \frac{v}{Z}\omega_v| \leq 2\frac{f \tan(\theta/2)}{Z_{min}} \min(B_s, \frac{\pi}{\triangle x}) \tag{5.24}$$

The lowest sampling density along the Z axis $\triangle T_Z$ can then be computed as

$$\triangle T_z = \frac{\pi}{|\omega_{T_z}|} \leq \frac{Z_{min}}{f \tan(\theta/2)} \max(\frac{\pi}{2B_s}, \triangle x/2) \tag{5.25}$$

The maximum horizontal and vertical image flow with respect to Z translation $|l_x|$ and

$|l_y|$ can be written as:

$$\begin{cases} |l_x| = \frac{u}{Z}\triangle T_z \leq \max(\frac{\pi}{2B_s}, \triangle x/2) \\ |l_y| = \frac{v}{Z}\triangle T_z \leq \max(\frac{\pi}{2B_s}, \triangle x/2) \end{cases} \tag{5.26}$$

5.2.4 Fourier analysis of rotation around X or Y axis

The 1D appearance manifold $I_{R_x}(x, y, R_x)$ consists of the infinite set of images captured by a camera rotating around the X axis. Using the X-rotation flow listed in Table 5.1, we can relate I_{R_x} to I_0. Note that the warping is completely determined by the rotation angle R_x and is independent on the scene geometry.

$$I_{R_x}(x, y, R_x) = I_{R_x}(x + \frac{xy}{f}R_x, y + (f + \frac{y^2}{f})R_x, 0) = I_0(x + \frac{xy}{f}R_x, y + (f + \frac{y^2}{f})R_x) \tag{5.27}$$

The Fourier transform can be written as

$$\begin{aligned} \mathcal{I}_{R_x}(\omega_x, \omega_y, \omega_{R_x}) &= \int\int\int I_{R_x}(x, y, R_x) e^{-j(\omega_x x + \omega_y y + \omega_{R_x} R_x)} dx dy dR_x \\ &= \int\int\int I_0(x + \tfrac{xy}{f}R_x, y + (f + \tfrac{y^2}{f})R_x) e^{-j(\omega_x x + \omega_y y + \omega_{R_x} R_x)} dx dy dR_x \end{aligned} \tag{5.28}$$

Let

$$\begin{cases} u = x + \frac{xy}{f}R_x \\ v = y + (f + \frac{y^2}{f})R_x \end{cases} \quad or \quad \begin{cases} x = u/(1 + \frac{y}{f}R_x) \approx u - \frac{uv}{f}R_x \\ y = (v - fR_x)\frac{x}{u} \approx v - (f + \frac{v^2}{f})R_x \end{cases} \tag{5.29}$$

(5.29) assumes a small local rotation R_x. The error of this approximation depends on the rotation angle and the camera FOV. Usually it can be considered small enough for the analysis of local appearance manifolds. To give an example, for a camera with 90 ° FOV, the approximation error for a rotation of 6 ° is about 1%. The assumption on small local rotations has also been applied to rotation around Z axis, as will be shown

in (5.40). Applying (5.29) to (5.28) we have

$$\begin{aligned}
&\mathcal{I}_{R_x}(\omega_u, \omega_v, \omega_{R_x}) \\
&= \int\int\int I_0(u,v) e^{-j(\omega_u(u-\frac{uv}{f}R_x)+\omega_v(v-(f+\frac{v^2}{f})R_x)+\omega_{R_x}R_x)}(1-\frac{v}{f}R_x)(1-2\frac{v}{f}R_x)dudvdR_x \\
&= \int\int I_0(u,v)\int(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j(-\omega_u\frac{uv}{f}-\omega_v(f+\frac{v^2}{f})+\omega_{R_x})R_x}dR_x e^{-j(\omega_u u+\omega_v v)}dudv
\end{aligned} \tag{5.30}$$

(5.30) can be considered as the Fourier transform of a product of two 2D signals: I_0 and an integral denoted as $\mathbb{I}_{R_x}$.

$$\mathbb{I}_{R_x}(u,v) = \int(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j(-\omega_u\frac{uv}{f}-\omega_v(f+\frac{v^2}{f})+\omega_{R_x})R_x}dR_x \tag{5.31}$$

Thus $\mathcal{I}_{R_x}$ can be computed as the convolution of two Fourier transforms

$$\mathcal{I}_{R_x}(\omega_u, \omega_v, \omega_{R_x}) = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{F}\{\mathbb{I}_{R_x}(u,v)\} = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{H}_{R_x}(\omega_u, \omega_v, \omega_{R_x}) \tag{5.32}$$

Compute the second Fourier transform. Assume $-\omega_x\frac{xy}{f} - \omega_y(f+\frac{y^2}{f}) + \omega_{R_x} \neq 0$.

$$\begin{aligned}
&\mathcal{F}\{\mathbb{I}_{R_x}(u,v)\} = \int\int\int(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j(-\omega_u\frac{uv}{f}-\omega_v(f+\frac{v^2}{f})+\omega_{R_x})R_x}e^{-j(\omega_u u+\omega_v v)}dudvdR_x \\
&= \int e^{-j(\omega_{R_x}-f\omega_v)R_x}\int(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j\omega_v(1-\frac{vR_x}{f})v}\int e^{-j\omega_u(1-\frac{vR_x}{f})u}dudvdR_x \\
&= \int e^{-j(\omega_{R_x}-f\omega_v)R_x}\int(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j\omega_v(1-\frac{vR_x}{f})v}\delta(v-\frac{f}{R_x})dvdR_x \\
&= \int e^{-j(\omega_{R_x}-f\omega_v)R_x}[(1-3\frac{v}{f}R_x+2\frac{v^2}{f^2}R_x^2)e^{-j\omega_v(1-\frac{vR_x}{f})v}|_{v=\frac{f}{R_x}}]dR_x \\
&= \int e^{-j(\omega_{R_x}-f\omega_v)R_x} * \ 0 \ dR_x = 0
\end{aligned} \tag{5.33}$$

We know from (5.32) that $\mathcal{F}\{\mathbb{I}_{R_x}\} = 0$ implies $\mathcal{I}_{R_x} = 0$. Therefore (5.33) shows that the spectral support $\mathcal{I}_{R_x}$ is bounded within the region given by:

$$-\omega_u\frac{uv}{f} - \omega_v(f+\frac{v^2}{f}) + \omega_{R_x} = 0 \tag{5.34}$$

Note that u v ω_u and ω_v are bounded. Applying (5.6) and (5.23) to (5.34), we can compute the bandwidth of $\mathcal{I}_{R_x}$ in R_x dimension using (5.35).

$$|\omega_{R_x}| = |\omega_u \frac{uv}{f} + \omega_v(f + \frac{v^2}{f})| \leq f(1 + 2\tan^2(\theta/2)) \min(B_s, \frac{\pi}{\triangle x}) \tag{5.35}$$

The lowest sampling density in rotation around X axis $\triangle R_x$ can then be computed as

$$\triangle R_x = \frac{\pi}{|\omega_{R_x}|} \leq \frac{1}{f(1 + 2\tan^2(\theta/2))} \max(\frac{\pi}{B_s}, \triangle x) \tag{5.36}$$

The maximum horizontal and vertical image flow with respect to the rotation around X axis $|l_x|$ and $|l_y|$ can be written as

$$\begin{cases} |l_x| = \frac{uv}{f}\triangle R_x \leq \frac{\tan^2(\theta/2)}{1+2\tan^2(\theta/2)} \max(\frac{\pi}{B_s}, \triangle x) \\ |l_y| = (f + \frac{v^2}{f})\triangle R_x \leq \frac{1+\tan^2(\theta/2)}{1+2\tan^2(\theta/2)} \max(\frac{\pi}{B_s}, \triangle x) \end{cases} \tag{5.37}$$

For rotation around Y axis, the spectral support of the 1D appearance manifold I_{R_y} can be analyzed using the same approach. The resulting minimum sampling density $\triangle R_y$ has the same form as $\triangle R_x$.

5.2.5 Fourier analysis of rotation around Z axis

The 1D appearance manifold $I_{R_z}(x, y, R_z)$ consists of the infinite set of images captured by a camera rotating around the Z axis. Using the Z-rotation flow listed in Table 5.1, we can relate I_{R_z} to I_0 (shown in (5.38)). Note that the warping is completely determined by the rotation angle R_z and is independent on the scene geometry.

$$I_{R_z}(x, y, R_z) = I_{R_z}(x + yR_z, y - xR_z, 0) = I_0(x + yR_z, y - xR_z) \tag{5.38}$$

The Fourier transform can be written as

$$\begin{aligned}\mathcal{I}_{R_z}(\omega_x, \omega_y, \omega_{R_z}) &= \int\int\int I(x, y, R_z) e^{-j(\omega_x x + \omega_y y + \omega_{R_z} R_z)} dx dy dR_z \\ &= \int\int\int I_0(x + yR_z, y - xR_z) e^{-j(\omega_x x + \omega_y y + \omega_{R_z} R_z)} dx dy dR_z\end{aligned} \tag{5.39}$$

Let

$$\begin{cases} u = x + yR_z \\ v = y - xR_z \end{cases} \quad or \quad \begin{cases} x = (u - vR_z)/(1 + R_z^2) \approx u - vR_z \\ y = (v + uR_z)/(1 + R_z^2) \approx v + uR_z \end{cases} \tag{5.40}$$

Note that (5.40) assumes a small local rotation R_z. Applying (5.40) to (5.39) and substituting ω_x ω_y with ω_u ω_v, we have

$$\begin{aligned}\mathcal{I}_{R_z}(\omega_u, \omega_v, \omega_{R_z}) &= \int\int\int I_0(u, v) e^{-j(\omega_u(u - vR_z) + \omega_v(v + uR_z) + \omega_{R_z} R_z)} (1 + R_z^2) du dv dR_z \\ &= \int\int I_0(x, y) \int (1 + R_z^2) e^{-j(-\omega_u v + \omega_v u + \omega_{R_z}) R_z} dR_z e^{-j(\omega_u u + \omega_v v)} du dv\end{aligned} \tag{5.41}$$

(5.41) can be considered the Fourier transform of a product of two 2D signals: I_0 and an integral denoted as $\mathbb{I}_{R_z}$.

$$\mathbb{I}_{R_z}(u, v) = \int (1 + R_z^2) e^{-j(-\omega_u v + \omega_v u + \omega_{R_z}) R_z} dR_z \tag{5.42}$$

Thus $\mathcal{I}_{R_z}$ can be computed as the convolution of two Fourier transforms

$$\mathcal{I}_{R_z}(\omega_u, \omega_v, \omega_{R_z}) = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{F}\{\mathbb{I}_{R_z}(u, v)\} = \mathcal{I}_0(\omega_u, \omega_v) \otimes \mathcal{H}_{R_z}(\omega_u, \omega_v, \omega_{R_z}) \tag{5.43}$$

Compute the second Fourier transform. Assume $-\omega_u v + \omega_v u + \omega_{R_z} \neq 0$.

$$\begin{aligned}
\mathcal{F}\{\mathbb{I}_{R_z}(u,v)\} &= \int\int\int (1+R_z^2) e^{-j(-\omega_u v + \omega_v u + \omega_{R_z})R_z} e^{-j(\omega_u u + \omega_v v)} du dv dR_z \\
&= \int (1+R_z^2) e^{-j\omega_{R_z} R_z} \int e^{-j(\omega_u + \omega_v R_Z)u} du \int e^{-j(\omega_v - \omega_u R_Z)v} dv dR_z \\
&= \int (1+R_z^2) \delta(R_z + \tfrac{\omega_u}{\omega_v}) \delta(R_z - \tfrac{\omega_v}{\omega_u}) dR_z \\
&= (1+R_z^2)|_{R_z=\pm i} \ = 0
\end{aligned} \tag{5.44}$$

We can see from (5.43) that $\mathcal{F}\{\mathbb{I}_{R_z}\} = 0$ implies $\mathcal{I}_{R_z} = 0$. Therefore (5.44) shows that the spectral support $\mathcal{I}_{R_z}$ is bounded within the region given by

$$-\omega_u v + \omega_v u + \omega_{R_z} = 0 \tag{5.45}$$

Note that u v ω_u and ω_v are bounded. Applying (5.6) and (5.23) to (5.45), we can compute the bandwidth of $\mathcal{I}_{R_z}$ in R_z dimension using (5.46).

$$|\omega_{R_z}| = |\omega_u v - \omega_v u| \leq 2f \tan(\theta/2) \min(B_s, \frac{\pi}{\triangle x}) \tag{5.46}$$

The lowest sampling density in rotation around Z axis $\triangle R_z$ can then be computed as

$$\triangle R_z = \frac{\pi}{|\omega_{R_z}|} \leq \frac{1}{f \tan(\theta/2)} \max(\frac{\pi}{2B_s}, \triangle x/2) \tag{5.47}$$

The maximum horizontal and vertical image flow with respect to the rotation around Z axis $|l_x|$ and $|l_y|$ can be written as

$$\begin{cases} |l_x| = v\triangle R_z \leq \max(\frac{\pi}{2B_s}, \triangle x/2) \\ |l_y| = u\triangle R_z \leq \max(\frac{\pi}{2B_s}, \triangle x/2) \end{cases} \tag{5.48}$$

5.2.6 Discussion

So far, I have computed the spectral support for the six 1D appearance manifolds that correspond to the six basic motions. I have shown that their spectral supports are bounded along the motion dimensions. I have computed the bandwidths in the Fourier domain and the associated minimum sampling densities in the spatial domain. The result of the analysis is summarized in Table 5.2. Here $\triangle x$ represents the camera resolution, f denotes the camera focal length and Z_{min} represents the minimum depth of the observed scene. The first column shows the minimum sampling densities of the six appearance manifolds along their motion dimensions. The second and the third rows columns show the maximum horizontal and vertical flows associated with the minimum sampling rate. The combined maximum flows are shown in the last column. Note that for all six motions, the maximum allowed flow is one pixel. Therefore Table 5.2 reveals that for reconstructing the 1D appearance manifold, we should acquire samples densely enough such that the image flow between two adjacent samples is less than one pixel. Or, for a fixed sampling rate, we need to blur the original images such that the size of the blurred pixel is larger than the possible maximum flow.

Motion	Min sampling density	Max H-Flow(l_x)	Max V-Flow(l_y)	Max $l_x + l_y$
X Tran	$\frac{Z_{min}}{f}\triangle x$	$\triangle x$	0	$\triangle x$
Y Tran	$\frac{Z_{min}}{f}\triangle x$	0	$\triangle x$	$\triangle x$
Z Tran	$\frac{Z_{min}}{2f\tan(\theta/2)}\triangle x$	$\triangle x/2$	$\triangle x/2$	$\triangle x$
X Rot	$\frac{1}{f(1+2\tan^2(\theta/2))}\triangle x$	$\frac{\tan^2(\theta/2)}{1+2\tan^2(\theta/2)}\triangle x$	$\frac{1+\tan^2(\theta/2)}{1+2\tan^2(\theta/2)}\triangle x$	$\triangle x$
Y Ro	$\frac{1}{f(1+2\tan^2(\theta/2))}\triangle x$	$\frac{1+\tan^2(\theta/2)}{1+2\tan^2(\theta/2)}\triangle x$	$\frac{\tan^2(\theta/2)}{1+2\tan^2(\theta/2)}\triangle x$	$\triangle x$
Z Rot	$\frac{1}{2f\tan(\theta/2)}\triangle x$	$\triangle x/2$	$\triangle x/2$	$\triangle x$

Table 5.2: Minimum sampling density and maximum flow.

Figure 5.4 shows the spectral support of 1D appearance manifolds associated with four motions: X translation, Z translation, X rotation and Z rotation. The synthetic scene was a plane with random texture placed at $1m$ away. The camera resolution was 128×128. The camera FOV was 90 degrees. For each motion, 128 synthetic images

were generated at 128 poses with a uniform step size ($4mm$ and $2mm$ for X and Z translation, and 0.1 degree for both X and Z rotation). These images were stacked to form a 3D matrix. I then used $Matlab$ to compute the 3D Fourier transform. The resulting spectral supports are demonstrated using three cross-section images. We can see that the bounding regions in the resulting figures match the theoretical analysis from the previous sections.

5.2.7 Fourier analysis of the 8D appearance function

I now combine the analysis on the six 1D appearance sub-manifolds and compute the Fourier transform of the 8D appearance signal. We can decompose any camera motion from the reference pose $S_0 = [0, 0, 0, 0, 0, 0]$ to a specific new pose $S_6 = [t_x, t_y, t_z, r_x, r_y, r_z]$ into a sequence of six 1D motions: three translations along the coordinate axes and three rotations around the coordinate axes. Depending on the order of the motions, at the end of each step, the camera reaches at an intermediate reference pose. [2] Without loss of generality, we can assume the order to be translation in X, Y, Z directions and then rotations around X, Y, Z axes. We can apply the flow listed in Table 5.1 to relate the new image $I(x, y, t_x, t_y, t_z, r_x, r_y, r_z)$ with the reference image I_0 through six intermediate

[2]Note that Rx R_y R_z are Euler angles. To arrive at the correct target pose, the camera needs to perform the three 1D rotations in the order that is consistent with the definition of Euler angles.

steps.

$$
\begin{aligned}
S_0 &= [0,0,0,0,0,0] & I_0(x,y) &= I(x,y,0,0,0,0,0,0)\\
S_1 &= [t_x,0,0,0,0,0] & I_1(x,y,T_x) &= I_0(x-\tfrac{f}{Z}T_x,y)\\
S_2 &= [t_x,t_y,0,0,0,0] & I_2(x,y,T_y) &= I_1(x,y-\tfrac{f}{Z}T_y,t_x)\\
S_3 &= [t_x,t_y,t_z,0,0,0] & I_3(x,y,T_z) &= I_2(x+\tfrac{x}{Z}T_z,y+\tfrac{y}{Z}T_z,t_y)\\
S_4 &= [t_x,t_y,t_z,r_x,0,0] & I_4(x,y,R_x) &= I_3(x+\tfrac{xy}{f}R_x,y+(f+\tfrac{y^2}{f})R_x,t_z)\\
S_5 &= [t_x,t_y,t_z,r_x,r_y,0] & I_5(x,y,R_y) &= I_4(x-(f+\tfrac{x^2}{f})R_y,y-\tfrac{xy}{f}R_y,r_x)\\
S_6 &= [t_x,t_y,t_z,r_x,r_y,r_z] & I_6(x,y,R_z) &= I_5(x+yR_z,y-xRz,r_y)\\
& & I(x,y,t_x,t_y,t_z,r_x,r_y,r_z) &= I_6(x,y,r_z)
\end{aligned}
\tag{5.49}
$$

In (5.20) (5.32) (5.43), we have switched the order of integration and represented the Fourier transform $\mathcal{I}(\omega_x,\omega_y,\omega_{TR})$ of the target 3D appearance function as a convolution of two spectral functions. One is the 2D spectral $\mathcal{I}_0(\omega_x,\omega_y)$ of the reference image. The other is the 3D function $\mathcal{H}(\omega_x,\omega_y,\omega_{TR})$ that defines the spectral bounds of $\mathcal{I}$ in the motion dimension. Here ω_{TR} denotes the angular spectral frequency in the motion dimension. [3] Similarly, we can compute the Fourier transform of the 8D appearance function as

$$
\begin{aligned}
\mathcal{I} &= \mathcal{H}_{R_z}\otimes\mathcal{F}(I_6)=\mathcal{H}_{R_z}\otimes\mathcal{H}_{R_y}\otimes\mathcal{F}(I_5)=\cdots\\
&= \mathcal{H}_{R_z}\otimes\mathcal{H}_{R_y}\otimes\mathcal{H}_{R_x}\otimes\mathcal{H}_{T_z}\otimes\mathcal{H}_{T_y}\mathcal{H}_{T_x}\mathcal{I}_0
\end{aligned}
\tag{5.50}
$$

Note that each ω_{TR} is only defined in one $\mathcal{H}$. Its response in the other five $\mathcal{H}$ is defined as delta functions. Therefore the convolutions do not change the bounds of the spectral support in motion dimensions. The bandwidth of the 8D $\mathcal{I}$ in each of the six motion dimensions is the same as that of the associated 3D $\mathcal{I}$. The above convolutions appear to enlarge the bandwidth in ω_x and ω_y dimensions. This may be true for continuous

[3] Originally, we define the convolution in the 2D space of (ω_x,ω_y). To help understanding, we can add the motion dimension to $\mathcal{I}_0$ and represent it as $\mathcal{I}_0'(\omega_x,\omega_y,\omega_{TR})=\mathcal{I}_0(\omega_x,\omega_y)\delta(\omega_{TR})$. Then we can apply convolution in the 3D space. Similarly, we can add additional motion dimensions to $\mathcal{H}$ in (5.50) by multiplying it with 5 delta functions.

appearance functions. However, in practice, all the images are captured by a camera, and the spectral support of the measured appearance function is always bounded by the camera resolution.

In summary, I have shown that the spectral support of the 8D appearance function (6D appearance manifold) are bounded, and its bandwidths in the motion dimensions are the same as the ones of the six 1D appearance sub-manifolds. Therefore we can directly apply the analysis on sampling 1D motions to 6D cases. Specifically, we should either sample densely enough in each of the six motion dimensions or blur the image samples, such that the maximum flow between adjacent samples is less than one (blurred) pixel.

5.3 Fourier analysis on general scenes

In the previous sections, I assumed a Lambertian scene and analyzed the spectral support of the appearance function using the brightness constancy model. In this section, I will extend the analysis to scenes with more realistic geometric and photometric properties. I will address three cases: semitransparency, occlusion and specular highlight.

5.3.1 Semitransparancy

The 8D appearance function of a semitransparent scene can be modeled as the sum of the appearance functions of the two superimposed scene layers.

$$I = c_1 I_1 + c_2 I_2 \tag{5.51}$$

Here c_1 and c_2 denotes the transparent and reflective coefficient of the surface. The Fourier transformation of I can be written as

$$\mathcal{I} = \mathcal{F}(c_1 I_1 + c_2 I_2) = c_1 \mathcal{I}_1 + c_2 \mathcal{I}_2 \tag{5.52}$$

Let B_1 and B_2 be the bandwidth of $\mathcal{I}_1$ and $\mathcal{I}_2$. The bandwidth B of $\mathcal{I}$ is

$$B = max(B_1, B_2) \tag{5.53}$$

(5.53) shows that the spectral support of the appearance function of a semitransparent scene is again bounded. When both layers are Lambertian, the brightness constancy model can be applied to each of them. Which, according to the previous analysis, results in a sub-pixel motion constraint on each layer separately. In other word, we should choose a blurring filter such that the maximum pixel motions of both layers are within one blurred pixel.

5.3.2 Occlusion

In general, the appearance discontinuity at occlusion boundary can be really complicated thus difficult to model. In this subsection, I will study the Fourier spectrum of simple occluded scenes using an opacity-based model proposed by Zhang and Chen (ZC06). Under the assumption that the objects do not occlude themselves [4], we can formulate the appearance function of the occluded scene as

$$I(x, y) = g(x, y)I_1(x, y) + (1 - g(x, y))I_2(x, y) \tag{5.54}$$

where I_1 and I_2 are the appearance functions of the foreground and background objects without considering occlusion. g is the occlusion mask defined as

$$g(x, y) = \begin{cases} 1 & if(x, y) \in \Omega \\ 0 & otherwise \end{cases} \tag{5.55}$$

[4] As pointed out by Zhang and Chen, mutual occlusion is typically more significant than self-occlusion. The former often cause a sharp discontinuity in the appearance function, while the latter usually does not as the surface normal often changes slowly.

where Ω is the image support of the foreground. Let $\mathcal{G}$, $\mathcal{I}_1$ and $\mathcal{I}_2$ be the Fourier transforms of g, I_1 and I_2. The Fourier transform of I can be written as

$$\mathcal{I} = \mathcal{F}(gI_1 + (1-g)I_2) = \mathcal{G} \otimes \mathcal{I}_1 + \mathcal{I}_2 + \mathcal{G} \otimes \mathcal{I}_2 \tag{5.56}$$

The first part of (5.56) is the Fourier transform of the foreground image. Notice that g and I_1 are at the same depth. Their multiplication can still be considered a Lambertian surface [5]. Clearly, the second part I_2 is a Lambertian surface. Therefore, the previous analysis on Lambertian surfaces can be directly applied to the first two parts. In other words, the bandwidths of $\mathcal{G} \otimes \mathcal{I}_1$ and $\mathcal{I}_2$ in the image dimensions are determined by the blurring filter (see (5.6)). The bandwidths in the motion dimensions can be computed using the linear constraints defined in previous equations (see (5.13) (5.24) (5.35) (5.46)).

However the previous equations can not be applied to the third part, as their basic assumption of brightness constancy model does not hold for the multiplication of g and I_2. These two layers are at different depths therefore move at different speeds in the image plane. This relative motion between the background and the occlusion mask generates new frequency components in the motion dimensions, as demonstrated in the 1D examples that follows.

Consider two sinuous signals $\hat{g}(X)$ and $\hat{I}_2(X)$. Denote their images observed by a 1D projective camera placed at T as $g(x,T)$ and $I_2(x,T)$. Assume the images observed by the reference camera placed at the origin ($T = 0$) as $g(x,0) = \sin ax + \theta_1$ and $I_2(x,0) = \sin bx + \theta_2$. The observed combined reference image can then be written as:

$$I(x,0) = g(x,0)I_2(x,0) = \sin(ax + \theta_1)\sin(bx + \theta_2) \tag{5.57}$$

Assume $\hat{g}$ and $\hat{I}_2$ are at depths Z_1 and Z_2 and the camera focal length is f. The

[5] Readers are referred to (5.60) for a formal explanation. As g and I_1 are at the same depth, their image speeds $v_1 = v_2$. From the first delta function, we have $w_x = \pm a \pm b$. Substitute it into the second delta function, we have $\delta(\omega_T + w_x v_1)$. This constraint is the same as (5.11).

translational speed of g and I_2 are then $v_1 = \frac{f}{Z_1}$ and $v_2 = \frac{f}{Z_2}$. The image taken at position T can be written as

$$I(x,T) = g(x,T)I_2(x,T) = \sin\left(a(x - v_1T) + \theta_1\right)\sin\left(b(x - v_2T) + \theta_2\right) \tag{5.58}$$

Now let us compute the Fourier transform of the appearance signal. Notice that $g(x,T)$ and $I_2(x,T)$ can be represented as shifted version of $g(x,0)$ and $I_2(x,0)$. From previous analysis (see equations (5.7 – 5.11)), their Fourier transforms can be written as $\mathcal{G}(\omega_x,\omega_T) = \mathcal{G}_0(\omega_x)\delta(\omega_x v_1 + \omega_T)$ and $\mathcal{I}_2(\omega_x,\omega_T) = \mathcal{I}_{2_0}(\omega_x)\delta(\omega_x v_2 + \omega_T)$, where $\mathcal{G}_0$ and $\mathcal{I}_{2_0}$ are the spectral support of $g(x,0)$ and $I_2(x,0)$. Therefore we have

$$\mathcal{I}(\omega_x,\omega_T) = \mathcal{G}(\omega_x,\omega_T) \otimes \mathcal{I}(\omega_x,\omega_T) = \mathcal{G}_0(\omega_x)\delta(\omega_x v_1 + \omega_T) \otimes \mathcal{I}_{2_0}(\omega_x)\delta(\omega_x v_2 + \omega_T) \tag{5.59}$$

Substitute the Fourier transforms of sinuous signals $\mathcal{G}_0 = e^{-j\theta_1}\delta(\omega_x - a) + e^{j\theta_1}\delta(\omega_x + a)$ and $\mathcal{I}_{2_0} = e^{-j\theta_2}\delta(\omega_x - b) + e^{j\theta_1}\delta(\omega_x + b)$ into (5.59), and use the property of convolution with delta function $f(t) \otimes \delta(t - t_0) = f(t - t_0)$, we have:

$$\begin{aligned}
\mathcal{I}(\omega_x,\omega_T) \quad &= e^{-j(\theta_1+\theta_2)}\delta(\omega_x - a - b)\delta(\omega_T + av_1 + bv_2) \\
&+ e^{j(\theta_1+\theta_2)}\delta(\omega_x + a + b)\delta(\omega_T - av_1 - bv_2) \\
&+ e^{-j(\theta_1-\theta_2)}\delta(\omega_x - a + b)\delta(\omega_T + av_1 - bv_2) \\
&+ e^{j(\theta_1-\theta_2)}\delta(\omega_x + a - b)\delta(\omega_T - av_1 + bv_2)
\end{aligned} \tag{5.60}$$

Equation (5.60) shows that the Fourier transform consists of 4 points whose coordinates are $\omega_x = \pm a \pm b$ in the image dimension and $\omega_T = \pm av_1 \pm bv_2$ in the motion dimension respectively. Notice that the mask and background can have arbitrarily high but close frequencies. When $|a - b| < \frac{\pi}{\triangle x}$ ($\triangle x$ is the size of the blurred pixel), the image blurring filter will not filter out the difference frequency components $\omega_x = \pm(a - b)$, thus leaves two possible high frequency components of $\omega_T = \pm(av_1 - bv_2)$ in the motion dimension. When the background is at infinity, the frequency in the motion dimension

becomes $\omega_T = av_1$, clearly unbounded as a goes to infinity.

The previous example shows that the appearance signal can have high-frequency component in the motion dimensions, and this high-frequency component can not be filtered out by blurring in the image dimensions. Such un-filterable aliasing occurs when the spectral supports of both the occluder and background consist of significant yet close high-frequency components. In real world, this phenomenon happens, probably more often than we expected. Consider looking at a building with horizontal patterns through window blinds. However, in most cases the spectral support of the occluder is dominated by low-frequency component. The following example studies the effect of occlusion using a representative occluder: the gate function.

Consider a 1D gate function $\hat{g}(X)$ at depth Z_1, a general background $\hat{I}_2(X)$ at depth Z_2, and a 1D projective camera of focal length f. Denote their images captured at reference camera position ($T = 0$) as $g(x, 0)$ and $I_2(x, 0)$, where $g(x, 0)$ is of the form

$$g(x,0) = \begin{cases} 1 & |x| \leq \frac{d}{2} \\ 0 & otherwise \end{cases} \tag{5.61}$$

The images captured at camera position T can be represented as shifted version of the reference images: $g(x, T) = g(x - v_1 T, 0)$ and $I_2(x, T) = I_2(x - v_2 T, 0)$, where $v_1 = \frac{f}{Z_1}$ and $v_2 = \frac{f}{Z_2}$ are the image translational speeds. The Fourier transform of $g(x, 0)$ is $\mathcal{G}_0 = \frac{d}{\sqrt{2\pi}} sinc\frac{d\omega_x}{2\pi}$. The Fourier transform of the background $I_2(x, 0)$ can be represented as the sum of a series of frequency components $\mathcal{I}_{2_0} = \sum_m c_m \delta(\omega_x - \omega_m)$. Substituting $\mathcal{G}_0$ and $\mathcal{I}_{2_0}$ into (5.59) and using the convolution property of delta functions give:

$$\begin{aligned} \mathcal{I}(\omega_x, \omega_T) &= \frac{d}{\sqrt{2\pi}} sinc\frac{d\omega_x}{2\pi} \delta(\omega_x v_1 + \omega_T) \otimes \sum_m c_m \delta(\omega_x - \omega_m) \delta(\omega_x v_1 + \omega_T) \\ &= \frac{d}{\sqrt{2\pi}} \sum_m sinc\frac{d(\omega_x - \omega_m)}{2\pi} \delta(\omega_T + (\omega_x - \omega_m) v_1 - \omega_m v_2) \end{aligned} \tag{5.62}$$

Equation (5.62) shows that the Fourier transform in the image dimension is a sinc function. While the support of the sinc function goes to infinity, more than 95% of its

energy falls within the first envelop between $[-1,\ 1]$. Therefore it can be considered a low-pass filter with a cut-off frequency of 1. This leads to the bound $|\omega_x - \omega_m| \leq \frac{2\pi}{d}$. Notice that the image blurring filter provides another bound $|\omega_x| \leq \frac{\pi}{\triangle x}$. Substituting these two bounds to the delta function of (5.62) gives the upper bound in the motion dimension as $\omega_{T_{max}} = \max(\frac{2\pi}{d}, \frac{\pi}{\triangle x})v_1$. Notice that the second part corresponds to the sub-pixel motion constraint described earlier (see (5.11)). For an arbitrary d, the overall bandwidth in the motion dimension is determined by the first part $\omega_{T_{max}} = \frac{2\pi}{d}\frac{f}{Z_1}$ [6]. The corresponding step size for the lowest sampling rate is

$$\triangle T_{max} = \frac{\pi}{\omega_{T_{max}}} = \frac{1}{2}\frac{Z_1}{f}d. \tag{5.63}$$

Notice that $\frac{Z_1}{f}d$ is in fact the gate length of $\hat{g}(X)$ in the world space. (5.63) thus indicates that to avoid aliasing the sampling step size or the camera baseline should be smaller than half the size of the occluder.

For the differential camera cluster (DCC) used in the experiments of this thesis (see Chapter 7), the camera baseline is 34mm. This corresponds to a 68mm occluder. When the scene consists of occluders smaller than 68mm, aliasing will occur. However, notice that the Fourier transform in (5.62) has a coefficient of $\frac{d}{\sqrt{2\pi}}$. Which means its energy is weighted by $\frac{d^2}{2\pi}$. This weight drops fast as d decreases. Therefore, unless a majority part of the scene is made up of small occluders, aliasing caused by small occluders does not significantly affect the global Fourier transform of the appearance signal.

So far, our analysis has assumed the pinhole camera model where every point is in focus. In practice, the camera sees the scene through a lens that has a physical size. The back projection of an image point forms a cone in the scene space. The intensity of an image point is the mean intensity of the incident light rays within that cone. This

[6]This bandwidth can not be effectively reduced by blurring the images. Increasing the blurring magnitude may result in a smaller bound for ω_x. However, one can always choose a ω_x within that bound and a ω_m such that $|\omega_x - \omega_m| = \frac{2\pi}{d}$.

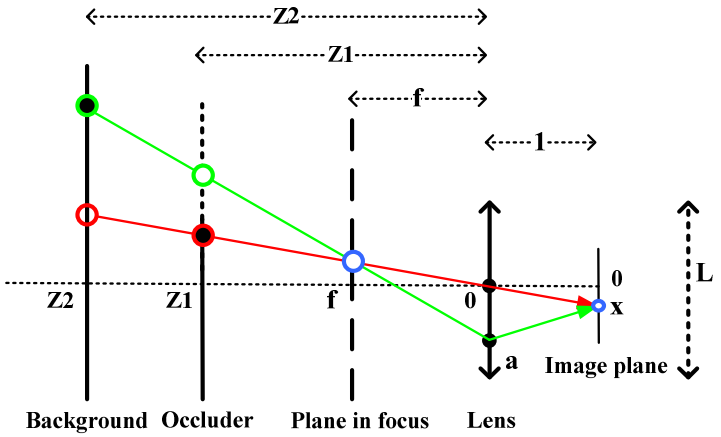


Figure 5.1: Illustration of occlusion observed by a thin-lens camera.

physical blurring of the lens can be considered as a low-pass filter, and can help alleviate the aliasing caused by occlusion.

Fig. 5.1 illustrates a thin-lens camera model. The size of the lens is L, the focal length is 1 and the image is focused at a plane of depth f. The scene consists of two planes, the occluder at depth Z_1 and the background at depth Z_2. Two incident light rays arriving at the same image point x are illustrated. The red ray passes through the center of the lens. The green ray shows the general case of a light ray hits the lens at a position a, gets refracted, then hits the image place at position x.

Now let us compute the Fourier transform of the appearance signal. From (5.56), we can see that the Fourier transformation of a scene with occlusion can be decomposed into three parts. Since the first two parts are merely the Fourier transform of Lambertian scenes which has been extensively studied in the previous sections, our following discussion will focus on the third part.

Using triangular equations, we can compute the coordinate of the two points where the red ray intersects the mask and the background (indicated by hollow and solid red circles in Fig. 5.1). Denoting the background intensity and the mask as I_2 and g, we

can represent the intensity of the light ray that passes through a and hits x as.

$$i(x,a) = g(-Z_1 x - \frac{Z_1 - f}{f}a) I_2(-Z_2 x - \frac{Z_2 - f}{f}a) \tag{5.64}$$

Let us now take into account the motion of the camera. Consider a reference image taken at camera position $X = 0$, and a new image taken at camera position $X = T$. Since the lens and the image plane move with the camera, the viewing ray passing through a and x in the reference image still passes them in the new image. However, the new ray has shifted by T in the world coordinate. Therefore, we can write

$$i(x,a,T) = g[-Z_1(x - \frac{a}{Z_1} + \frac{a}{f}) - T] I_2[-Z_2(x - \frac{a}{Z_2} + \frac{a}{f}) - T] \tag{5.65}$$

The overall intensity at $I(x,T)$ is the integral of $i(x,a,T)$ over a. Let L be the diameter of the lens, we can write the final intensity as

$$\begin{aligned} I(x,T) &= \tfrac{1}{L}\textstyle\int_{-L/2}^{L/2} i(x,a,T)da \\ &= \tfrac{1}{L}\textstyle\int_{-L/2}^{L/2} g[-Z_1(x - \frac{a}{Z_1} + \frac{a}{f}) - T] I_2[-Z_2(x - \frac{a}{Z_2} + \frac{a}{f}) - T]da \end{aligned} \tag{5.66}$$

Using (5.66), we can represent the Fourier transform as a multi-integral.

$$\mathcal{I}(\omega_x, \omega_T) = \int_a \int_T \int_x g[-Z_1(x - \frac{a}{Z_1} + \frac{a}{f}) - T] I_2[-Z_2(x - \frac{a}{Z_2} + \frac{a}{f}) - T] e^{-j(\omega_x x + \omega_T T)} dx dT da \tag{5.67}$$

Let us first compute the inner integral over x.

$$\begin{aligned} \dot{\mathcal{I}}(\omega_x) &= \textstyle\int g[-Z_1(x - \frac{a}{Z_1} + \frac{a}{f}) - T] I_2[-Z_2(x - \frac{a}{Z_2} + \frac{a}{f}) - T] e^{-j\omega_x x} dx \\ &= \textstyle\int g[-Z_1(x - \frac{a}{Z_1} + \frac{a}{f}) - T] e^{-j\omega_x x} dx \otimes \int I_2[-Z_2(x - \frac{a}{Z_2} + \frac{a}{f}) - T] e^{-j\omega_x x} dx \\ &= \tfrac{1}{Z_1} e^{-j(\frac{a}{Z_1} - \frac{a}{f} - \frac{T}{f})\omega_x} \mathcal{G}(\tfrac{\omega_x}{Z_1}) \otimes \tfrac{1}{Z_2} e^{-j(\frac{a}{Z_2} - \frac{a}{f} - \frac{T}{f})\omega_x} \mathcal{I}_2(\tfrac{\omega_x}{Z_2}) \end{aligned} \tag{5.68}$$

For general g and I_2, we can decompose them into series of sinuous signals and write their

Fourier transform as $\mathcal{G} = \sum_n c_m \delta(\omega_x - \omega_m)$ and $\mathcal{I}_{2_0} = \sum_n d_n \delta(\omega_x - \omega_n)$. Substituting this representation into (5.68) we have

$$\begin{aligned}\dot{\mathcal{I}}(\omega_x) &= \frac{1}{Z_1 Z_2} \sum_m c_m e^{-j(\frac{a}{Z_1} - \frac{a}{f} - \frac{T}{f})\omega_x} \delta(\omega_x - Z_1\omega_m) \otimes \sum_n d_n e^{-j(\frac{a}{Z_2} - \frac{a}{f} - \frac{T}{f})\omega_x} \delta(\omega_x - Z_2\omega_n) \\ &= \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) e^{-j(1-\frac{Z_2}{Z_1})(a-T)\omega_n} e^{-j(\frac{a}{Z_1} - \frac{a}{f} - \frac{T}{f})\omega_x}\end{aligned} \tag{5.69}$$

Substituting (5.69) into (5.67) and integrating over T, we have

$$\begin{aligned}&\dot{\mathcal{I}}(\omega_x, \omega_T) \\ &= \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) e^{-j(1-\frac{Z_2}{Z_1})(a-T)\omega_n} e^{-j(\frac{a}{Z_1} - \frac{a}{f} - \frac{T}{f})\omega_x} e^{-j\omega_T T} dT \\ &= \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) e^{-j(\omega_m + \omega_n - \frac{\omega_x}{f})} \int e^{j(\omega_m + \omega_n)T} e^{-j\omega_T T} dT \\ &= \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) \delta(\omega_T - \omega_m - \omega_n) e^{-j(\omega_m + \omega_n - \frac{\omega_x}{f})} da\end{aligned} \tag{5.70}$$

We now integrate over a and compute the complete Fourier transform.

$$\begin{aligned}&\mathcal{I}(\omega_x, \omega_T) = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} \frac{1}{Z_1 Z_2} \dot{\mathcal{I}}(\omega_x, \omega_T) \\ &= \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) \delta(\omega_T - \omega_m - \omega_n) e^{-j(\omega_m + \omega_n - \frac{\omega_x}{f})} da \\ &= \frac{1}{L} \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) \delta(\omega_T - \omega_m - \omega_n) \int_{-\frac{L}{2}}^{\frac{L}{2}} e^{-j(\omega_m + \omega_n - \frac{\omega_x}{f})a} da \\ &= \frac{1}{Z_1 Z_2} \sum_{(m,n)} c_m d_n \delta(\omega_x - \omega_m Z_1 - \omega_n Z_2) \delta(\omega_T - \omega_m - \omega_n) sinc \frac{(\omega_m + \omega_n - \frac{\omega_x}{f})L}{2\pi}\end{aligned} \tag{5.71}$$

We are now ready to discuss the optical blurring provided by the lens. (5.71) consists of a sinc function. As discussed earlier, sinc function is a low-pass filter and 95 % of its energy is contained within its first envelope of [-1,1]. This cut-off frequency provides one constraint $|\omega_m + \omega_n - \frac{\omega_x}{f}|L \leq 2\pi$. Notice that the second delta function of (5.71) provides another constraint $\omega_T = \omega_m + \omega_n$. Combing these two constraints gives a linear constraint that relates the image bandwidth to the motion bandwidth. $|\omega_T - \frac{\omega_x}{f}| \leq \frac{2\pi}{L}$. ω_x is bounded by the software blurring filter. Its bandwidth is $\frac{\pi}{\triangle x}$, where $\triangle x$ is the size of the blurred pixel. Substituting it to (5.71) gives the bandwidth in the motion dimension $\omega_T \leq \frac{\omega_x}{f} + \frac{2\pi}{L}$. Notice that f is the ratio between the focus depth and the

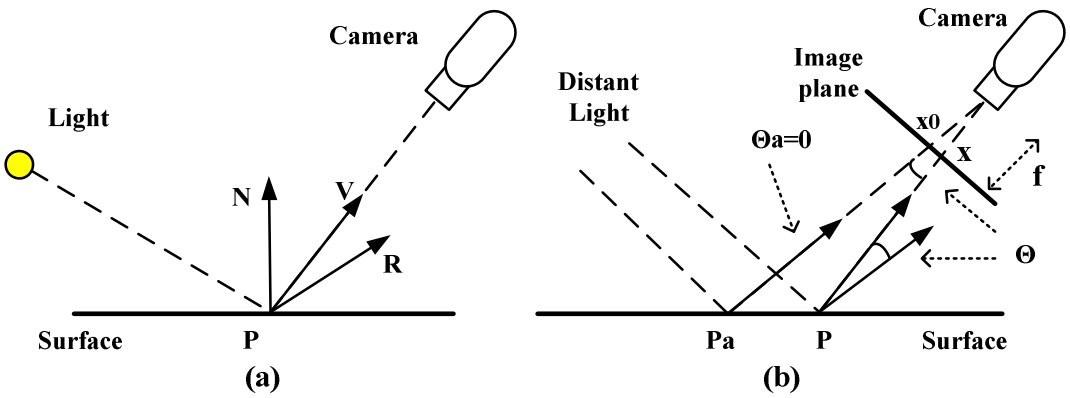


Figure 5.2: Illustration of specular reflection.

focal length (see Fig. 5.1). When the camera is adjusted to focus at infinity, f goes to infinity and the bandwidth in the motion dimension becomes

$$\omega_{T_{max}} = \frac{2\pi}{L}. \tag{5.72}$$

The corresponding sampling step size is

$$\triangle T_{max} = \frac{\pi}{\omega_{T_{max}}} = \frac{L}{2}. \tag{5.73}$$

The result shows that the physical blurring of the lens can eliminate any occlusion-based aliasing if the motion is less than half the size of the lens.

5.3.3 Specular hightlights

In this subsection, I will study the Fourier transform of specular highlights using Phong model [7]. Fig. 5.2(a) illustrates a camera observing a specular surface under a static lighting, N is the outward surface normal at surface point p, R is the direction of the

[7]In a perfect specular reflection, the reflection of the light is only visible when the surface normal is precisely halfway between the direction of the light and the direction of the viewer. However, due to the existence of microfacets, the reflected light is usually scattered around the direction of the mirrored reflection. A number of models exist to predict the distribution of the reflected light. Here we choose the most commonly used model, Phong model.

mirror-reflection of the incident ray at p, and V is the reverse viewing direction from p to the camera. Assume p is projected on the image plane at x. Using Phong model, the pixel intensity $I(x)$ can be represented using (5.74), where $\theta = \arccos(R \cdot V)$ is the angle between R and V, n is a constant scaler that indicates the surface specularly, and I_N is the maximum intensity acquired at $\theta = 0$.

$$I(x) = I_N(x)\cos^n(\theta(x)) \tag{5.74}$$

Consider a 1D camera placed at the reference pose $X = 0$. It captures an image I_0, where p is projected at x_0 with angle θ_0 between R and V and angle α_0 between R and the image plane. Let the camera translate T along the direction of the image plane, and captures another image $I(x, T)$, in which p is imaged at x_T with θ_T. When motion T is small (compared to the scene depth Z), the change of θ, $\cos\theta$ and the ratio of $\cos^n\theta$ can be linearized using small angle approximation, Taylor expansion, and the linear approximation of $(1+\epsilon)^n \approx 1 + n\epsilon$.

$$\theta_T = \theta_0 + \frac{\sin\alpha_0 T}{Z} \tag{5.75}$$

$$\cos\theta_T = \cos\theta_0 - \sin\theta_0 \frac{\sin\alpha_0 T}{Z} \tag{5.76}$$

$$\frac{\cos^n\theta_T}{\cos^n\theta_0} = (\frac{\cos\theta_0 - \sin\theta_0 \frac{\sin\alpha_0 T}{Z}}{\cos\theta_0})^n = 1 - n\tan(\theta_0)\sin\alpha_0 \frac{T}{Z} \tag{5.77}$$

Using equations (5.75–5.77) and the image flow of $x_T = x_0 - \frac{f}{Z}T$, we can represent $I(x, T)$ using the transform of reference image $I(x, 0)$ and compute its Fourier transform.

$$I(x,T) = I_0(x - \frac{f}{Z}T)(1 - n\tan(\theta_0(x))\frac{\sin(\alpha_0(x))T}{Z}) \tag{5.78}$$

$$\mathcal{I}(\omega_x, \omega_T) = \mathcal{I}_0(\omega_x)\delta(\omega_T + \frac{f}{Z}\omega_x) - \mathcal{I}_0(\omega_x)\delta(\omega_T + \frac{f}{Z}\omega_x) \otimes \frac{n}{Z}\mathcal{F}[\tan(\theta_0(x))\sin(\alpha_0(x))T]. \tag{5.79}$$

The Fourier transform $\mathcal{F}[\tan(\theta_0(x))\sin(\alpha_0(x))T]$ can be computed as

$$\begin{aligned}\mathcal{F}[\tan(\theta_0(x))T] &= \int\int \tan(\theta_0(x))\sin(\alpha_0(x))Te^{-j(\omega_x x+\omega_T T)}dxdT \\ &= \int \tan(\theta_0(x))\sin(\alpha_0(x))e^{-j\omega_x x}dx \int Te^{-j\omega_T T}dT \\ &= \Theta_0(\omega_x)j\sqrt{2\pi}\delta'(\omega_T)\end{aligned} \tag{5.80}$$

where $\Theta_0(\omega_x)$ is the Fourier transform of $\tan(\theta_0(x))\sin(\alpha_0(x))$. Substituting (5.80) into (5.79) gives

$$\mathcal{I}(\omega_x,\omega_T) = \mathcal{I}_0(\omega_x)\delta(\omega_T + \frac{f}{Z}\omega_x) - j\frac{n\sqrt{2\pi}}{Z}\mathcal{I}_0(\omega_x)\delta(\omega_T + \frac{f}{Z}\omega_x) \otimes \Theta_0(\omega_x)\delta'(\omega_T). \tag{5.81}$$

The first term of (5.81) is the Fourier transform of the stack of translated reference images. It provides the same constraint between the bandwidths of ω_x and ω_T as (5.11). Let us now study the second term that corresponds to the change of the intensities with respect to the change of the viewing directions. Using equations $\int f(x)\delta(x-a) = f(x-a)$ and $\int f(x)\delta'(x-a) = -f'(a)$, the convolution can be computed as

$$\begin{aligned}&\int\int \mathcal{I}_0(\omega_x - a)\delta[(\omega_T - b) + \tfrac{f}{Z}(\omega_x - a)]\Theta_0(a)\delta'(b)dadb \\ &= \int \mathcal{I}_0(\omega_x - a)\Theta_0(a)\delta'[\omega_T + \tfrac{f}{Z}(\omega_x - a)]da = \mathcal{I}_0'(-\tfrac{Z}{f}\omega_T)\Theta_0'(\omega_x + \tfrac{Z}{f}\omega_T)\end{aligned} \tag{5.82}$$

$\mathcal{I}_0'(-\frac{Z}{f}\omega_T)$ does not provide any constraint on the bandwidth of ω_T. However ω_T will be bounded when Θ_0' is bandlimited. Assume the cut-off frequency of Θ_0' is c, which gives the constraint $|\omega_x + \frac{Z}{f}\omega_T| \leq c$. Notice that the image blurring process provides another constraint $|\omega_x| \leq \frac{\pi}{\triangle x}$, where $\triangle x$ is the size of the blurred pixel. Combining these two constraints gives us the bound

$$|\omega_T| \leq \frac{f}{Z}(c + \frac{\pi}{\triangle x}). \tag{5.83}$$

The question then becomes whether Θ_0' is bounded. In the rest of this section, I will

first prove that Θ_0' is bounded for an planar specular surface. I will then conceptually discuss the bound for curved surfaces.

Consider a 1D camera observing the reflection of a distant light source through a planar specular surface (illustrated in Fig. 5.2(b)). Assume R and V overlaps ($\theta_a = 0$) at surface point p_a with projection x_a in the reference image . Without loosing generality, let us assume $x_a = 0$. The angle between R and V of a point p projected at x can then be written as $\theta_0(x) = \arctan(\frac{x}{f})$. Or, $\tan(\theta_0(x)) = \frac{x}{f}$. Since α_0 is the angle between R and the image plane, we have $\sin(\alpha_0(x)) = \frac{f}{\sqrt{x^2+f^2}}$. We can then compute $\Theta_0(\omega_x)$ as

$$\Theta_0(\omega_x) = \int \tan(\theta_0(x)) \sin(\alpha_0(x)) e^{-j\omega_x x} dx = \int \frac{x}{\sqrt{x^2+f^2}} e^{-j\omega_x x} dx = \frac{f}{\pi^{3/2}} K_1(f\omega_x) \tag{5.84}$$

where K_1 is the modified Bessel function of the second kind. From (5.84) we can compute that $\Theta_0'(\omega_x)$ is in the form of $K_2(f\omega_x)$. $K_2(f\omega_x)$ is an exponentially decaying function that goes to infinity at $\omega_x = 0$. Since most of the energy of $K_2(f\omega_x)$ is covered within an infinitely small band around $\omega_x = 0$, $\Theta_0'(\omega_x)$ has an infinitely small bandwidth or $c = 0$. Apply this result to (5.83), we can see that the angular frequency in the motion dimension is bounded by $\omega_{T_{max}} = \frac{f}{Z}\frac{\pi}{\triangle x}$. The bandwidths in the motion dimensions are not increased by specular highlights reflected from a planar surface. The corresponding maximum image flow is still $\triangle x$.

The calculation of the bandwidth of $\Theta_0'(\omega_x)$ for a curved surface is more complicated, and will not be covered in this thesis. In general, $\Theta_0'(\omega_x)$ of a scene consisting of curved surfaces has non-zero bandwidth. However, for a smooth surface with small curvature, $\tan(\theta_0(x))$ will change smoothly and $\Theta_0'(\omega_x)$ will have a small bandwidth. This will result in an increased but still bounded bandwidth in the motion dimension (see (5.83)). To accommodate the increased bandwidth in the motion dimension, we need to sample the appearance manifold at a higher spatial frequency. For scenes with high curvature surfaces, $\Theta_0'(\omega_x)$ will have large bandwidth in the motion dimension, and

aliasing will occur. However, a larger curvature also means a smaller radius. Therefore high curvature surface areas typically have small image supports, and specular highlights from these areas usually only contribute to an insignificant part of the global energy of the appearance signal.

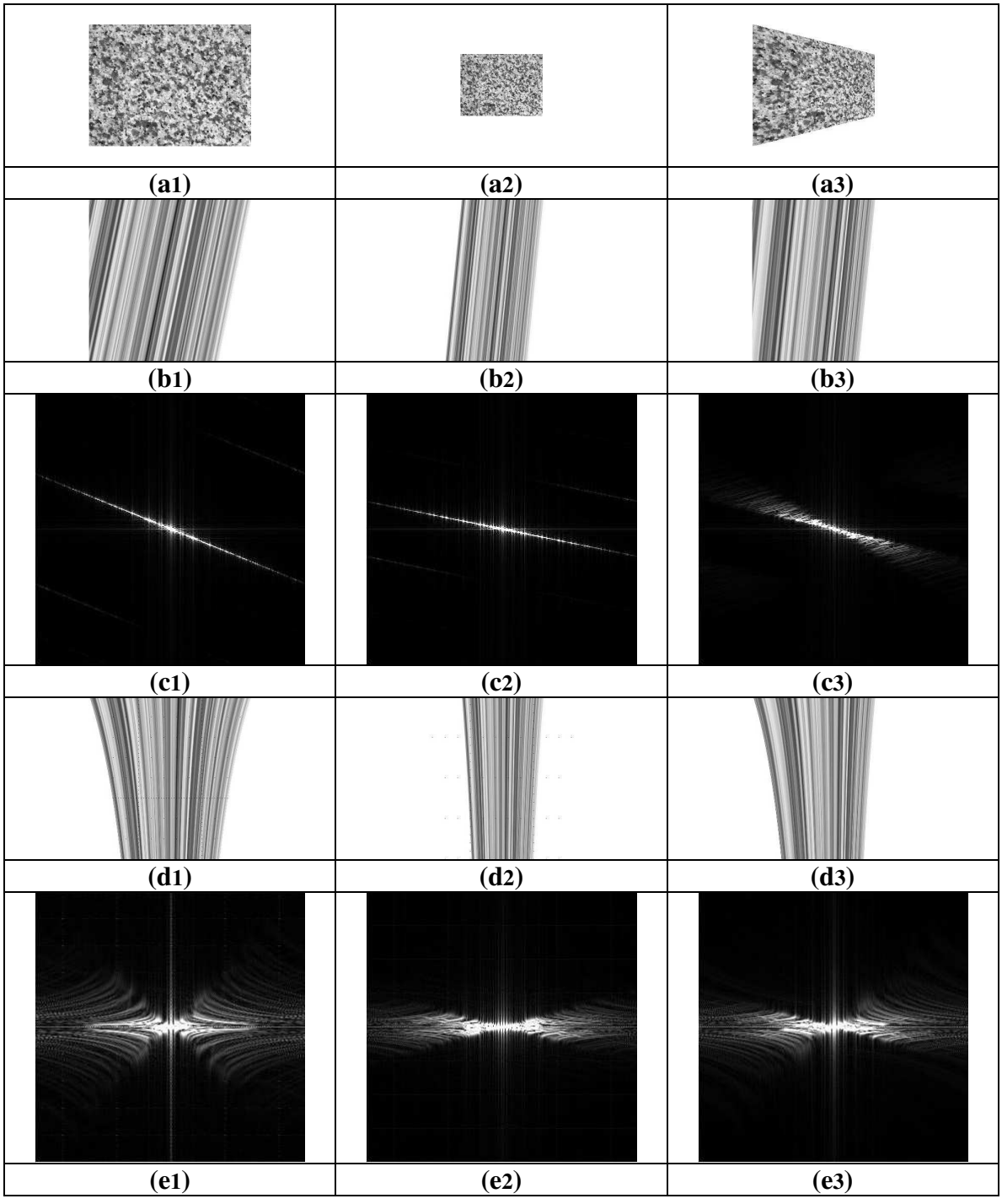


Figure 5.3: Fourier transform of the 2D cross-sections of the 3D appearance functions associated with X and Z translations. (a): planar scenes. (b): cross-section images of I_{T_x} on $X - T_x$ plane. (c): Fourier transform of (b). (d): cross-section images of I_{T_z} on $X - T_z$ plane. (e): Fourier transform of (d).

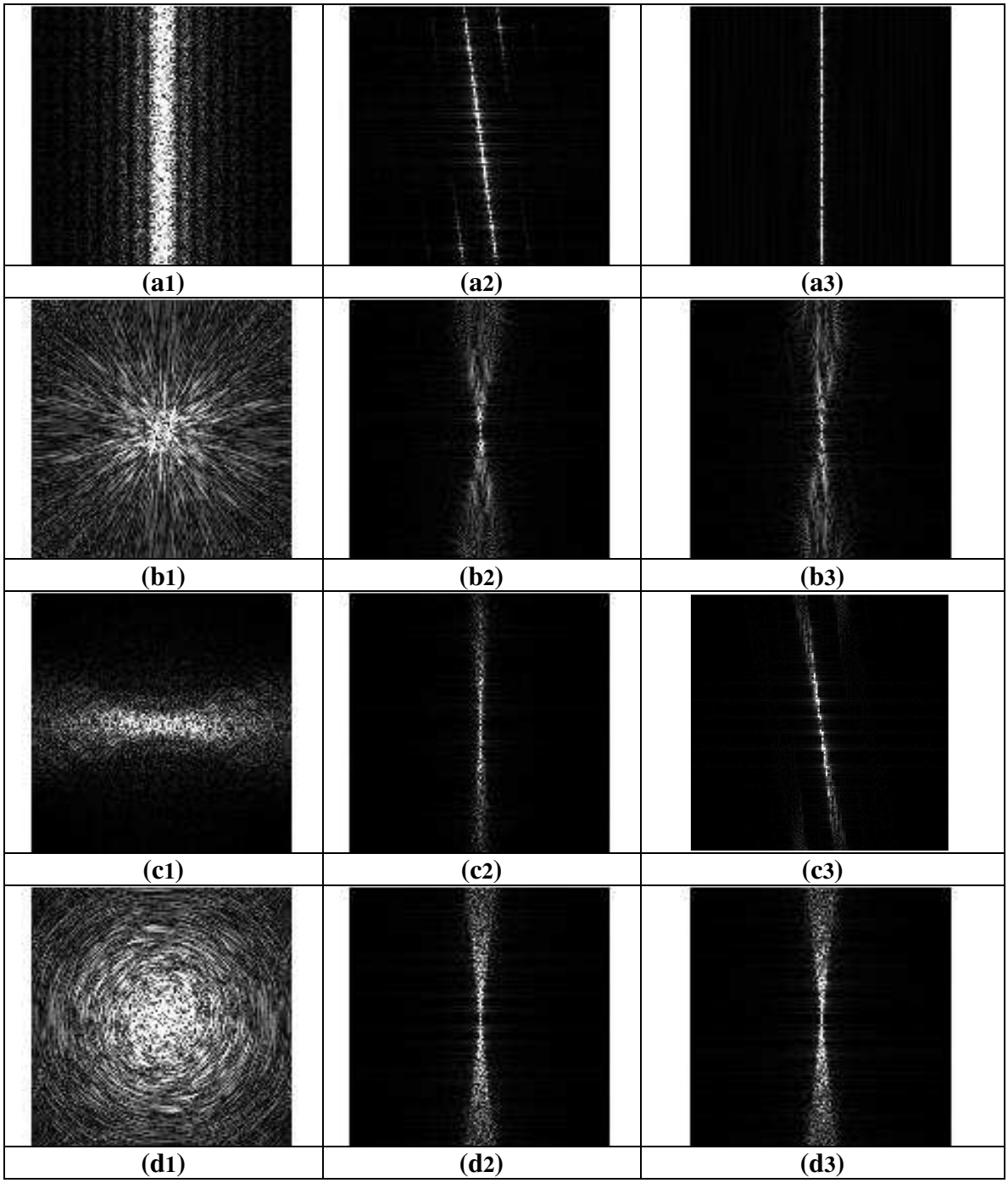


Figure 5.4: Fourier transform of four 3D appearance functions associated with four 1D motions. First row: X translation. Second row: Z translation. Third row: X rotation. Fourth row: Z rotation. First column: Spectral support on the $\omega_x - \omega_y$ plane (a1) $\omega_{T_x} = 0$, (b1) $\omega_{T_z} = 0$, (c1) $\omega_{R_x} = 0$, (d1) $\omega_{R_z} = 0$. Second column: Spectral support of the $\omega_y = 0$ plane. Third column: Spectral support on the $\omega_x = 0$ plane.

Chapter 6

Motion Segmentation

In the previous chapters, I have introduced the theory and techniques for sampling and linearizing the local appearance manifold. Based on this local linearization, I have developed a differential approach to tracking a rigid motion. In this chapter, I will discuss the usage of such a locally linear model for a different application: 3D motion segmentation. More specifically, I will present an approach to clustering individual image pixels into groups associated with different 3D rigid motions.

6.1 Related work

Motivated by the research in 2D motion estimation, in particular optical flow, most early approaches to motion segmentation address the problem of segmenting pixels using dense 2D flow fields. For instance, Black and Anandan use robust statistics to handle discontinuities in the flow fields (BA96). In layered approaches (XS05) (WA94), images are segmented into a set of layers. These methods work on image motion and can not be extended to accommodate 3D motion.

Common approaches to 3D motion analysis segmentation are feature-based. They usually aim at clustering feature points according to their underlying motion. Early work includes applying robust statistic methods like RANSAC (FB87). Ozden et al. present a feature based 3D reconstruction system that can simultaneously estimate the

camera motion and track independently moving objects (OSvG07). The system is based on several hypothesis tests for the segmentation of the different observed motions. Pioneered by Costeira and Kanade's work, multi-body factorization based methods have been proposed (CK95) (Gea98) for segmenting independent affine motions. These algorithms use as input a matrix of 2D feature trajectories (sequences of image coordinates of feature points across multiple frames), then use algebraic factorization techniques to cluster the feature trajectories into groups with different motions. One issue with the factorization method is that it assumes independent motion. Recently, to address more complicated scenes that exhibit partially dependent-motion, (VH04) (YP06) propose to solve motion segmentation by clustering the motion subspaces spanned by the feature trajectories.

Compared to the prosperous research in feature-based techniques, dense (per-pixel) 3D motion segmentation is to a large extend unexplored. To my knowledge, only one effort has been made to address this problem (ZMMI06). In this approach, image regions are segmented using optical flow, or more exactly, *covariance-weighted* optical flow approximated using spatial and temporal intensity derivative measurements [1], under the assumption of brightness constancy. A *covariance-weighted flow-field* matrix is formed by stacking row vectors of transformed horizontal and vertical flows of all image regions across multiple frames. Motion-based segmentation is achieved by factorizing the *covariance-weighted flow-field* matrix into a *motion* matrix and a *shape* matrix. Then regions with same motion are grouped by computing and sorting a reduced row echelon form of the *shape* matrix.

In this chapter, I will present an approach to clustering individual image pixels associated with different 3D rigid motions. Similar to (ZMMI06), my method is based on the observation that the image measurements captured from different perspectives

[1]Since the flow estimate is usually noisy, Irani proposed to compute a covariance-weighted flow-field using intensity measurement, and showed that the subspace constraints for the original flow-field can also be applied to the transformed flow-fields. Details can be found in (Ira02).

across multiple frames span a linear subspace. However, instead of 2D flow fields I use the less noisy 1D pixel intensities as the input measurements. Specifically, I introduce the notion of the pixel *intensity trajectory*, a vector that represents the intensity changes of a specific pixel over multiple frames. Like the 2D feature trajectories, the intensity trajectories of pixels associated with the same motion span a low-dimensional linear subspace. I therefore formulate the problem of motion segmentation as that of clustering local subspaces. Unlike the flow-based technique, this linear model of the intensity measurements does not require strict brightness constancy. As I will show later, it can be extended to accommodate more general cases, such as illumination changes on Lambertian surface under directional lighting. For segmenting motion subspaces, I apply spectral clustering to the intensity trajectories. This classification technique addresses some issues of direct matrix factorization, such as the noise-sensitivity (GW06) and the difficulty in handling partially dependent motion (Kan01).

6.2 Clustering motion subspaces

In this section, I describe the notion of the *intensity trajectory*, and show that the intensity trajectories of pixels associated with the same motion span a low-dimensional subspace.

6.2.1 Intensity trajectory matrix

To begin, let us consider a projective camera and a scene with constant uniform illumination. The image intensity of a pixel (x, y) is a function of the pose $P = [X, Y, Z, \alpha, \beta, \gamma]$ of the corresponding imaged surface point in the camera viewing space. Let $I(x, y, P)$ be the image intensity, or a filtered version of the image intensity. Using the brightness constancy equation, we can compute a local linearization of the intensity function using

a Taylor expansion

$$I(x, y, P + dP) = I(x, y, P) + \frac{\partial I(x, y)}{\partial P} dP \tag{6.1}$$

where dP represents a 3D motion in viewing space. If dP is small, namely image motion caused by dP is sub-pixel, the change of intensity dI can also be locally linearized as

$$dI(x, y) = I(x, y, P + dP) - I(x, y, P) = \frac{\partial I(x, y)}{\partial P} dP \tag{6.2}$$

Now consider a reference image I_0 captured at pose P_0, and a sequence of f images I_i at nearby poses P_i. We can compute f difference images $dI_i = I_i - I_0$. Thus each pixel (x, y) is associated with an f-vector of intensity changes $[dI_1(x, y), dI_2(x, y), ..., dI_f(x, y)]^T$ that corresponds to a chain of motions $[dP_1, dP_2, ..., dP_f]^T$. We call this f-vector of intensity changes a pixel *intensity trajectory* (as oppose to the 2D *feature* trajectories). Combining the intensity trajectories of all n image pixels, we can construct an intensity trajectory matrix W. The rows of W represent difference images, and its columns represent pixel intensity trajectories.

$$W = \begin{vmatrix} dI_{1,1} & ... & dI_{1,n} \\ \vdots & \ddots & \vdots \\ dI_{f,1} & ... & dI_{f,n} \end{vmatrix}$$

Readers may have found out that the intensity trajectory matrix W is similar to dI used in (3.3) of Section 3.2. Like dI, W is constructed using changes of pixel intensities. Besides, as I will show in the next section, W also provides a linear subspace representation of the local appearance manifold. The difference is, for motion estimation, we use subspaces spanned by the difference images (multiple pixels single frame) or rows of W. While for motion segmentation, we are interested in subspaces spanned by the intensity trajectories (single pixel multiple frames) or columns of W.

6.2.2 Motion subspaces

Consider a scene with a single 3D rigid motion. Using equation (6.2), W can be decomposed into two matrices: a *motion matrix* M of size $f \times 6$ and an *intensity Jacobian matrix* F of size $n \times 6$ as follows.

$$W = MF^T \tag{6.3}$$

$$M = \begin{vmatrix} dP_{1,1} & \dots & dP_{1,6} \\ \vdots & \ddots & \vdots \\ dP_{f,1} & \dots & dP_{f,6} \end{vmatrix}, F = \begin{vmatrix} \frac{\partial I}{\partial P}_{1,1} & \dots & \frac{\partial I}{\partial P}_{1,6} \\ \vdots & \ddots & \vdots \\ \frac{\partial I}{\partial P}_{n,1} & \dots & \frac{\partial I}{\partial P}_{n,6} \end{vmatrix}$$

If the scene texture and the motion are non-degenerate, M and F are of rank 6. Thus the intensity trajectory matrix W is at most rank 6 (less for degenerate cases). In other words, the intensity trajectories of pixels associated with a single 3D rigid motion span a linear subspace of rank less than or equal to 6.

Now consider a scene with k different rigid motions. In this case, the image pixels belong to k different groups that are associated with the k underlying motions. To show the structure of the W matrix, we assume a certain permutation matrix Λ to sort the pixels with respect to their associated motions, such that

$$\begin{aligned} W\Lambda &= |W_1, W_2, ..., W_k| \\ &= |M_1, M_2, ..., M_k| \begin{vmatrix} F_1^T & & & \\ & F_2^T & & \\ & & \ddots & \\ & & & F_k^T \end{vmatrix} \end{aligned} \tag{6.4}$$

For each groups of pixels, we have

$$W_i = M_i F_i^T \qquad (i = 1, 2, ...k) \tag{6.5}$$

where M_i and F_i are the motion matrix and intensity Jacobian matrix for the i-th group, and W_i is the concatenation of intensity trajectories of pixels in that group. Again $rank(W_i) \leq 6$. Equations (6.4) and (6.5) show that the pixel intensity trajectories extracted from an image sequence captured in a scene with k rigid motions can be clustered into k groups. Each of them span a linear subspace of rank less than or equal to 6. Therefore we can achieve motion segmentation by clustering subspaces expanded by the intensity trajectories.

Although I will only discuss 3D rigid motion in this thesis, I believe the above analysis can be extended to more general motion such as articulated, non-rigid motion. Just like parameterizing dI into a 6D space for rigid motion, for a general motion of rank m, we can map dI into an mD space represented by its motion parameters. In this case the motion vector dP becomes an mD vector. We can still decompose the intensity trajectory matrix W into the motion matrix M and the intensity Jacobian matrix F. All three matrices are of rank m. Thus the intensity trajectories of pixels corresponding to a general motion of rank m span an mD subspace.

6.2.3 Motion subspaces under directional illumination

The previous analysis assumes brightness constancy. In this section, I will show that such a constraint can be relaxed to accommodate scenes with Lambertian objects and constant directional light sources.

Consider a scene that consists of m light sources with directions L_i and magnitudes l_i $(i = 1...m)$, and a 3D point p on a convex object with surface normal N and albedo λ. If we denote the incidence angle, the angle between the ray from light source i and the surface normal at p, to be θ_i, the intensity of p can be written as

$$I = \sum_{i=1}^{m} l_i \ \lambda \ max(L_i \cdot N, 0) = \sum_{i=1}^{m} l_i \ \lambda \ max(\cos\theta_i, 0) \tag{6.6}$$

Denote the half cosine function as $k_i = max(cos\theta_i, 0)$.

Its derivative can be written as [2]

$$\frac{\partial k_i}{\partial \theta_i} = \begin{cases} -\sin\theta_i & -\frac{\pi}{2} < \theta_i < \frac{\pi}{2} \\ 0 & otherwise \end{cases} \tag{6.7}$$

Now let us consider the change of the intensity caused by the motions of the object and the camera. We denote object motion as dP_o. Unlike the dP used in previous sections, dP_o is defined in the world space. We begin our discussion by assuming a fixed camera. When the object motion consists of nonzero rotational components, the surface normal and the incidence angles will change accordingly. Denote the change of the incidence angle of light source i as $d\theta_i$. For a small dP_o and thus a small $d\theta_i$, we can apply Taylor extension and represent the change of the pixel intensity dI as a linear function of $d\theta_i$.

$$dI = \lambda \sum_{i=1}^{m} l_i \frac{\partial k_i}{\partial \theta_i}\, d\theta_i \tag{6.8}$$

Equation (6.8) shows that the change of intensity dI of point p is a linear function of $d\theta_i$. For fixed distant light sources, the incidence angle θ_i (i=1...m) is determined by the surface normal N. Therefore, θ_i is function of N. Under small motion, we can approximate the change of incidence angle $d\theta_i$ as a linear function of the change of the surface normal dN.

$$\begin{aligned} d\theta_i &= \arccos(L_i \cdot (N + dN)) - \arccos(L_i \cdot N) \\ &\approx -\frac{1}{\sqrt[2]{1-(L_i \cdot N)^2}} L_i \cdot dN = -\frac{1}{\sin\theta_i} L_i \cdot dN \end{aligned} \tag{6.9}$$

If θ_i is not zero, it is clear that $d\theta_i$ is a linear function of dN. Notice that θ_i is zero only when N and L_i align with each other. In this case, dN is perpendicular to L_i. Using a

[2]The partial derivative $\frac{\partial k_i}{\partial \theta_i}$ is unbounded at $\theta_i = 0$. This indiscontinuity only affects pixels lying exactly on the illumination silhouette of the light sources. In practice, its effect is usually blurred out by the low-pass filtering process.

small angle approximation of $\sin\theta = \theta$, we have $d\theta_i = \sin d\theta_i = dN$. Equation (6.8) and (6.9) show that dI is a linear function of $d\theta_i$, which is a linear function of dN. Since dN is clearly a linear function of dP_o, we show that the change of intensity dI is a linear function of the object motion dP_o.

Under the small motion assumption, dN has only two degrees of freedom (on the plane perpendicular to N). Thus the change of intensity dI caused by the change of illumination lies in a 2D subspace. For a fixed camera, the relative motion between the object and the camera dP is the same as dP_o. Thus the 2D illumination subspace is embedded in the 6D motion subspace. In more general cases, where both the object and the camera move independently, dP is independent of dP_o. The 2D illumination subspace and the 6D motion subspace are orthogonal. Therefore, for a scene with convex Lambertian objects and constant directional light sources, the intensity trajectories of pixels corresponding to the same underlying motion generally span a 8D subspace.

6.3 Motion segmentation by clustering local subspaces

I have shown that given a number of local image samples captured at nearby poses (sub-pixel motion), one can construct an intensity trajectory matrix W. The 3D motion segmentation can be formulated as clustering columns of W with respect to their different underlying motion subspaces.

The clustering of columns can be achieved by factorizing the measurement matrix (CK95) (Gea98) (ZMMI06). However, matrix factorization requires the underlying motions to be independent (Kan01), an assumption that is often violated in real environments. Recently, researchers have attempted to address partially-dependent motion. Most notably are Vidal and Hartley's algebraic-based approach (VH04) and Yan and Pollefeys's spectral-based approach (YP06). A good review of these methods can be

found in (TV07).

In this thesis, I employ the so called Local Subspace Affinity (LSA) method for clustering motion subspaces (YP06). The LSA algorithm is based on the local linear projection and spectral clustering. Instead of working directly on the trajectory matrix W, LSA fits a local subspace for each point and constructs a similarity matrix A using the pairwise distances between the local subspaces. Motion segmentation is achieved by spectral clustering of the similarity matrix. The algorithm can be described in four steps:

Step 1. Dimension reduction and data normalization: Remove redundant dimensions (usually contributed by noise) by projecting the trajectories from R^f onto a lower dimensional space R^l using SVD. Then normalize these l-vectors onto a unit hyper-sphere.

Step 2. Local subspace estimation: For each projected point p_i, find its nearest neighbors on the hyper-sphere (not from the image space) and compute a local linear subspace S_i of dimension m.

Step 3. Similarity matrix construction: Compute the distances (principle angles) between local subspaces, and construct a similarity matrix A, using Equation (6.10), where θ_{ijh} is the h-th component of the principle angle vector between two local subspaces S_i and S_j.

$$A_{ij} = \exp(-\sum_{h=1}^{m} \sin^2 \theta_{ijh}) \tag{6.10}$$

Step 4. *Spectral clustering*: Apply the spectral clustering (NJW01) to the similarity matrix A and segment data into k clusters, where k is the number of different rigid motions in the scene.

There are two potential causes of segmentation error in the above algorithm. First, the neighbors selected in step 2 can be pixels of different subspaces. Second, the se-

lected neighbors may not fully span the underlying motion subspace. In both cases, the local subspace tend to have similar distances to several motion subspaces, and misclassification may occur. To address this issue we have developed a refinement procedure (Step 5). In this procedure, we identify ambiguous pixels by comparing their distances to different motion subspaces, then reclassify them using the spatial continuity of the moving objects.

Step 5a.1: For each cluster, compute a global motion subspace spanned by all the pixels belonging to it, using the result from step 4.

Step 5a.2: For each pixel, compute the pixel-to-cluster distance as the distance between its local subspace and its classified global subspace. Then for each cluster, compute the median of the in-cluster pixel-to-cluster distance.

Step 5a.3: For each pixel compute the distances between its local subspace and all k global subspaces, normalized by the median in-cluster distance. Compute the ratio of the smallest and the second smallest normalized distances. Classify a pixel as an ambiguous-pixel if its ratio is bigger than a threshold (in all the experiments we set it to be 0.7).

Step 5b: For each ambiguous pixel, search for its neighbors in the *image space* and classify it to the majority class.

Step 5 exploits the fact that an object covers a continuous region in the image. Based on the same spatial continuity assumption, we can further refine the segmentation with a simple outlier-correction process. We consider an isolated pixel, a single pixel labeled differently from all its neighbors, to be an outlier. Therefore, we search for these single-pixel holes and group them to be consistent with their neighbors.

Note that in the original work (YP06), the dimensions of the projected space l and the local subspace m are automatically determined using a rank detection algorithm to accommodate general unknown motion such as articulated or non-rigid motion. Since

this thesis only addresses 3D rigid motion, I choose l and m to be $6k$ and 6 for scenes with uniform lighting or $8k$ and 8 for directional lighting.

6.4 Acquiring local appearance samples

So far, we have formulated the problem of motion segmentation as clustering linear subspaces spanned by columns of the intensity trajectory matrix. To construct the matrix, we need to capture a sequence of local appearance samples. In theory, to span a motion subspace of rank k, we need at least $k+1$ image samples. This number is usually bigger in practice due to the noise issue. Since our subspace formulation is based on linearizing the local appearance manifold (see Equation (6.1)), the motion of the imaged surface across the sequence needs to be small (within the linear region).

It is feasible to acquire sufficient local samples for normally moving objects using commodity imaging devices. First, we can use the common technique of blurring the original image to smooth the appearance manifold. The enlarged linear region can then accommodate a larger motion. Secondly, the speed of commodity camera has become high enough to densely sample motion in most practical scenes. For instance, the Point-Gray Flea2 camera can capture at 80 frames-per-second at VGA resolution. Moreover, for sampling 3D rigid motions in an ambient-lit environment, we can reduce the number of temporal frames by using a small-baseline camera cluster such as the DCC prototype described in Section 4.1 [3]. At each frame, a DCC provides seven appearance samples, all captured from slightly different perspectives. Commercial products are also available.

[3]This is based on the dual relationship between the camera motion and the object motion. The basic idea is that images captured by cameras from different perspectives can be considered as appearance samples of the target object at different poses. Note that for a non-ambient lighting environment, an object can be illuminated differently at different poses. However, the appearance samples from the DCC are captured simultaneously when the object is at a certain pose. They do not record the possible illumination changes associated with the real object motion. Therefore, strictly speaking, this technique can only be applied in scenes where the brightness constancy assumption holds. However, in common cases the target object moves slowly with respect to the camera frame rate, the appearance difference caused by the illumination changes is usually small, and the illumination consistency requirement is usually satisfied in a relaxed form.

For instance, using its 5x5 camera array (12mm spacing), the Point-Gray's ProFUSION captures 25 local appearance samples at a time.

Chapter 7

Experiment results

In this chapter, I will evaluate the appearance-based differential tracking and motion estimation algorithms that I have introduced in Chapter 3 and 6. I will first present results on differential camera tracking where I captured local appearance samples on line using a DCC. Then in the sample-through-rendering setup, I will demonstrate tracking a rigid object with an off-line appearance model. I will show results on tracking, illumination correction and texture refinement. Finally, I will present experiment results on appearance-based motion segmentation.

7.1 Differential camera tracking using online samples

I begin the demonstration of differential camera tracking with a synthetic experiment. I simulated a scene that consists of a textured planar patch and a curved mirror, both contained inside a cube. The inner six surfaces of the (surrounding) cube were textured. I placed the simulated DCC in front of the curved mirror such that it viewed some of the scene beyond the edges of the mirror, and some of the reflection. Traditional tracking techniques would not perform well on this data, since the epipolar constraint does not hold for the distorted scene as reflected in the curved mirror.

In Figure 7.1, I present some tracking results on the synthetic scene over 40 frames. Figure 7.1(a) shows an original image (640×640). The border of the curved mirror is marked green. We can see that the reflection of the planar patch is distorted by the curved mirror. When generating the image sequences, I restricted the maximum extent of the camera motion so that the pixel motion for frontal scene points would be less than 4 pixels. To smooth the appearance manifold, I blurred the images (shown in Fig. 7.1(e)) using a Gaussian kernel of 160×160 with $\sigma = 24$ and sub-sampled them at 32 cycles (20 pixels). I chose a σ that was larger than the analysis result shown in Chapter 5 to accommodate the reflection of the rear scene, which could move faster in the image plane than the front scene. I show the translation and rotation estimates in Fig. 7.1 (b-d) and (f-h). The horizontal axes represent frame numbers and the vertical axes represent the accumulated motion across the previous frames. The red (solid) curves represent the true value, and blue (dashed) curves represent the estimated value. The translation and rotation of the camera are defined in the coordinate system of the center camera at the first frame.

In the second experiment, I demonstrate tracking a real DCC. The DCC was built using four PointGrey Flea digital color cameras (see Figure 4.2). The color images were converted to grayscale for scene appearance samples. The baselines between the cameras were about 34 mm in the X and Y direction, and 66 mm in the Z direction. As described in section 4.1.2, geometric and photometric consistencies were enforced across the four cameras.

To obtain some form of a ground-truth reference motion path, I moved the camera cluster along a pre-determined grid points marked on a table (the X-Z plane) while imaging the scene. The results are shown in Fig. 7.2. An original image and its blurred version is shown in Fig. 7.2(a) and (e). The resolution of the original image is 1024×768. The image was blurred using a Gaussian kernel of 160×160 with $\sigma = 24$, and then sub-sampled at a ratio of 20 to 1. The accumulated translations and rotations are shown

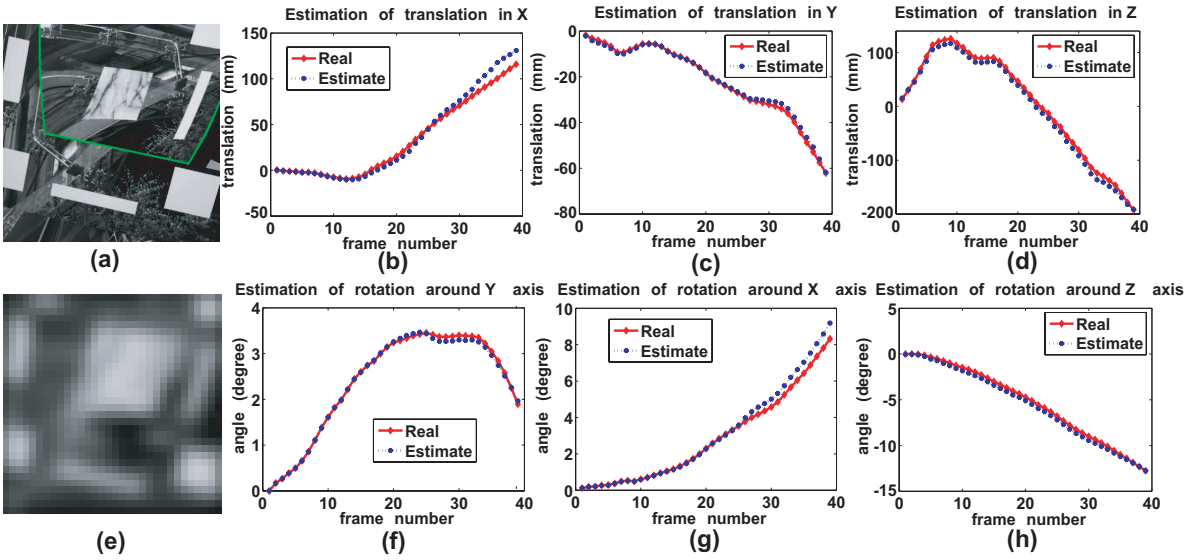


Figure 7.1: Tracking in a synthetic scene with a curved mirror over 40 frames. (a) An image of the synthetic scene. The border of the curved mirror is marked in green. (e) A blurred image. (b)-(d) Estimation of camera translations (in mm) in X, Y and Z directions. (f)-(h) Estimation of camera rotation angles (in degree) around Y, X and Z axes. Red (solid) curves represent the true values of the accumulated motion, blue (dashed) curves represent the estimated values of the accumulated motion.

in Fig. 7.2(b-d) and (f-h). Again, the translations and rotations are defined in the coordinate of the center camera at the first frame. We can see that the algorithm achieves good estimates of X and Y translations, and the estimated rotations and Z translations are small. I believe the exhibited error is due primarily to the inherent drift in any (incremental) approach, and to registration error introduced by our manual alignment process.

The third experiment shows the tracking of a hand-held camera cluster over 200 frames. As the ground truth motion was unknown, I illustrate the tracking accuracy using projection error. I chose seven reference points in the scene. Using a standard OpenCV KLT tracker, I extracted and matched their 2D coordinates in two camera images of the first frame at sub-pixel accuracy. I then back-projected them and computed their 3D positions in the coordinates of the center camera at the first frame. As the

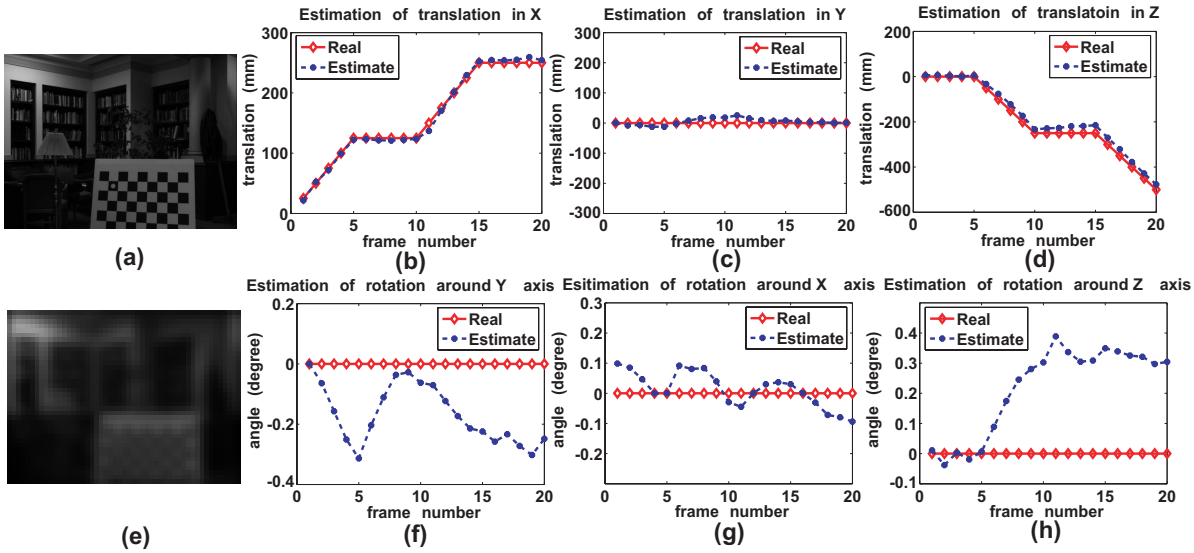


Figure 7.2: Tracking a controlled camera motion over 20 frames. (a) An image of the real scene. (e) A blurred image. (b)-(d) Estimation of camera translations (in mm). (f)-(h) Estimation of camera rotation angles (in degree). Red (solid) curves represent the true values of the accumulated motion, blue (dashed) curves represent the estimated values of the accumulated motion.

cluster moved, I estimated its incremental motion at every frame and compute the accumulated motion between the current frame and the first frame (see section 3.3). These accumulated motion parameters can be used to compute a projection matrix of the center camera at its current pose. I then used the estimated projection matrix to project the reference 3D points onto the current center image, and indicated their projections with white patches surrounded by red circles (see Figure 7.3). Figure 7.3(a) shows the reference points in the center image of the initial frame. Their projections using the estimated motion parameters are shown in Figure 7.3(b-f). Since the reference points are initially back-projected using small baseline stereo, the computed 3D positions may be considerably off from their true values. Therefore the final projection result contains error from both the incremental tracking and the initial triangulation process.

In the fourth experiment, I tracked the DCC moving in a scene with semi-transparency. Specifically, I pointed the cameras toward a large window and captured an image sequence at dawn. The indoor lighting environment was adjusted such that the cameras

saw both the outside and the reflection of the inside (see Fig. 7.4). This was a challenging environment for most feature-based motion estimation systems. Since the image was a superimposition of two layers undergoing different motion, typical feature-based algorithms would encounter difficulties extracting and matching features. As I did in the third experiment, I computed the projection matrices from the camera motion estimate and projected the reference scene points onto the center image (Fig. 7.4(b-f)). However, since the standard KLT method does not handle semi-transparency well, I manually selected the reference points in the initial frame (Fig. 7.4(a)) at an integer pixel accuracy.

In the previous experiments, I tested the differential tracking algorithm on some unknown camera motion. The estimation result was illustrated using backprojection error. For a direct evaluation, I have set up another experiment where I used an external tracker to acquire the ground-truth of the motion. Specifically, I used the NDI Optotrak 3020, a high-end commercial optical tracking system. The key component of the system is a position sensor that consists of three cameras mounted permanently in a bar (see Figure 7.5(b)). These cameras are calibrated by the manufacturer and are designed to see only the wire-coupled (synchronized) LED markers inside its working volume. To relate the DCC to the Optotrak, I mounted a rigid panel with four markers on the box of the cluster (see Figure 7.5(c)). I measured the positions of the markers using the Optotrak system software and stored them as a rigid object model. At runtime, the Optotrak captured the 3D position for each marker and output the pose of the rigid model. Since the DCC and the markers are fixed to each other, I then applied a coordinate transformation to acquiring the pose of the camera cluster. In this experiment, I used a DCC built with four black-and-white PointGrey flea2 cameras.

I present results in Figure 7.5. The red (solid) curves represent the accumulated motion reported by the Optotrak system, and blue (dashed) curves represent the estimate from our differential tracker. We can see that the differential tracker worked reason-

ably well. The maximum accumulated translation and rotation error in one dimension over 120 frames was about 10mm and 0.5°. During that period of time, the maximum translation and rotation of the camera in one dimension was about 370mm and 6°.

7.2 Differential object tracking using an off-line appearance model

In the previous section, I have tested tracking camera motion using local appearance samples captured online. While a complete model of the entire scene is hard to acquire, we can usually afford to model the appearance of a specific object. In this case, local appearance samples can be acquired using the sample-through-rendering technique (see section 4.2). In this section, I will apply the differential approach to tracking a rigid object with an appearance model. I will show results on tracking, illumination correction and texture refinement.

First I present results on synthetic data. The synthetic scene consisted of a textured sphere (an earth model), two cameras, and two distant light sources. One light source remained static, and the other changed its direction. To start the experiment, I randomly generated a motion sequence [1]. I then applied this motion sequence to the textured sphere and rendered a sequence of images from the camera views with OpenGL lighting enabled. The resulting synthetic image sequence was used as the test data.

During tracking, for each frame I rendered synthetic images of the sphere around its current estimated pose. But this time, the OpenGL lighting was disabled. To acquire illumination-corrected local appearance samples, I estimated an *illumination normal map* (INM) and applied it to the images rendered without lighting (see section 4.2.1). The process of illumination correction is illustrated in Figure 7.6. I computed the two

[1] I generated a sequence of random numbers. I then used these random numbers as the acceleration parameters and integrated them to form a motion sequence.

INMs shown in (c) by matching pixel intensities in image pairs (a,d) and (b,e). Although (a) and (b) were textured differently, they were rendered under the same illumination. As shown in (c), the INMs computed from (a) and (b) are very similar. This shows that the INM estimation was insensitive to the surface texture. After illumination estimation, I applied the upper INM to (d) and generated the illumination-corrected image (f). I then used (f) as one of the local appearance samples. The results on tracking and INM estimation of the synthetic sphere under varying lighting conditions are shown in Figure 7.7. We can see from the estimated INMs (middle row) that the direction of the light source had changed within the sequence. Watch the dark spot in the right side of the first INM moves to left side in the third INM. The resolution of the INMs shown in 7.6 and 7.7 is 72×36, or a spacing of 5×5 degrees between pixels. The tracking result is illustrated in the bottom row. Wire-frame sphere rendered using the estimated pose parameter are superimposed onto the original images.

I also tested the differential tracking method on real data. I used two calibrated and synchronized Point Grey Dragonfly cameras to capture image sequences of an ambient cube undergoing 6D free motion. The cameras were set to capture at 15 frames per second. Photo-consistency was not enforced, thus an INM is estimated for each of the two cameras. The scene environment consisted of an ambient lighting and two strong distant light sources. The pose of the cube was manually initialized. Figure 7.8 illustrates the effect of illumination correction. Figure 7.9 shows the results of tracking with and without illumination correction. Wire-frame cubes were rendered using the estimated pose parameters and superimposed onto the original images. With illumination correction, the system tracked the cube well in the entire sequence of 200 frames (see the top row). However it lost tracking in only a few frames without illumination correction (see the bottom row). In addition to estimating the motion and illumination, the system also refined the texture of the input object model as it included texture in the state parameters of the Kalman filter. The result of the texture refinement is presented in Figure

7.10. Note that the texture extracted from real image (d) is significantly darker than (a), which indicates an illumination difference between the modeled and the real scene. However the refined texture (c) was estimated using illumination-corrected images thus preserved the inherent luminance from the initial texture input (b).

7.3 3D dense motion segmentation

As described in Chapter 6, the linearization of the local appearance manifold also provides a means for (per-pixel) dense 3D motion segmentation. In this section, I will present results on clustering pixels associated with different underlying rigid motions. I tested my appearance-based motion segmentation algorithm on three real data sets. All of them contained two rigid motions: the camera and one moving object.

I will first show the result of classifying intensity trajectories captured using a camera cluster. Specifically, I built a DCC using four Point-Gray Flea2 black-and-white cameras. The DCC captures seven spatial samples at a time thus helps to reduce the number of temporal frames. In addition, this cluster setting can ensure the capturing of the full 6D motion subspace for any object in the scene [2], even if the real object motion within the sequence is degenerate. Note that while I used a DCC in the first experiment, for the above reasons, the motion segmentation algorithm is general and is not restricted to a cluster setup. In the second and the third experiments, I tested it using a single camera setup.

The cameras that I used in the experiments captured images at VGA resolution. However, to accommodate larger motion, I blurred the images to smooth the appearance manifold. In all the experiments, I blurred the original images and sub-sample the blurred images at a 20-to-1 rate. I then ran the motion segmentation algorithm on the sub-sampled images. As a result of the sub-sampling, one pixel in the resulting

[2]It only guarantees the motion subspace for each object to be 6D. The motion subspaces of different objects can still be partially dependent.

segmented images corresponds to a 20×20 block in the original images. Note that because the cameras are packed closely in the cluster, some cameras see the lenses of other cameras in the border area. In addition, the blurring process introduces some additional border effects. For these reasons, in the experiments I only processed the inner regions of the images.

The first experiment demonstrates motion segmentation in a scene with two controlled rigid motions. To control the motion, I mounted the camera cluster on a 1D translational platform, and a checkerboard on a rail. Between each frame, I shifted the camera and the checkerboard (4mm for the camera, 5mm for the checkerboard) along the directions of their rails. I captured six frames for a total of 42 real and synthetic images, and extracted intensity trajectories from these images. The classification results are presented in Figure 7.11. Figure 7.11(a) shows the segmentation without the refinement step (see Step 5 of the classification pipeline in 6.3). The pixel classification result is super-imposed on the original image. Boundary pixels are not processed (they are marked as black). Dark gray and light gray are used to indicate foreground and background pixels, respectively. Ambiguous pixels identified during the refinement process are marked white in Figure 7.11(b). The refined motion segmentation results are shown in Fig. 7.11(c). The misclassification error (number of mis-classified pixels divided by the number of all processed pixels) was 2.5%. Figure 7.11(d) shows the similarity matrices permuted using the initial (top) and refined segmentations (bottom).

I ran the differential tracker on this sequence. The output estimate of the controlled motion (restricted to the X-Z plane of the camera coordinate frame) is shown in Figure 7.11(e,f). There are five lines in both figures. The *Estimated Whole* line (black, dashed) indicates the motion computed using *all* of the pixels under the assumption of a single rigid scene. Using the segmentation results, I also tested estimating two motions by running the tracker on two groups of pixels separately. In Figure 7.11(e,f), the upper pairs of lines represent the real (true) and estimated motion of the checkerboard with

respect to the camera. The lower pairs of lines represent the real (true) and estimated motion of the background. The estimate using all of the pixels (unsegmented) appears to be a weighted average of the two underlying motions, as one would expect, and is clearly wrong for either motion. The result using the segmented pixels appears to be very accurate for the background motion, and reasonably accurate for the foreground motion [3]. Note that neither the segmentation nor the tracking algorithms assumes any prior information about the scene geometry.

In the second experiment, I tested the algorithm on segmenting two free-form rigid motions—both the camera cluster and the checkerboard were moved by hand. The motion segmentation results are shown in Fig. 7.12. Pixels corresponding to the moving checkerboard are marked white. The remaining pixels are classified as background. For a clearer representation, the boundary pixels are excluded.

To explore the use of the algorithm in a single camera setting, I ran it again on the above sequence. But this time I only used images captured by the center camera plus the three synthetic rotational cameras. The results are shown in Fig. 7.13. Although only one physical camera was used, the segmentation result is comparable to the cluster setup.

The last experiment demonstrates 3D motion segmentation in a scene with directional lighting using a single camera setup. The scene was illuminated with multiple ceiling lights and a strong light source from the side. A person was sitting on a chair and rotating. All light sources were static and constant. Again, I used images captured by one physical camera and three synthetic rotational cameras. Fig. 7.14(a) presents the illumination effect of the side light. The segmentation result is shown in (b)-(f). Pixels corresponding to the person and the chair are marked gray. Notice that most of the segmentation error are from pixels on the back of the chair. This is due to the plain texture in that area.

[3]It is less accurate since fewer pixels are used for estimating the foreground motion

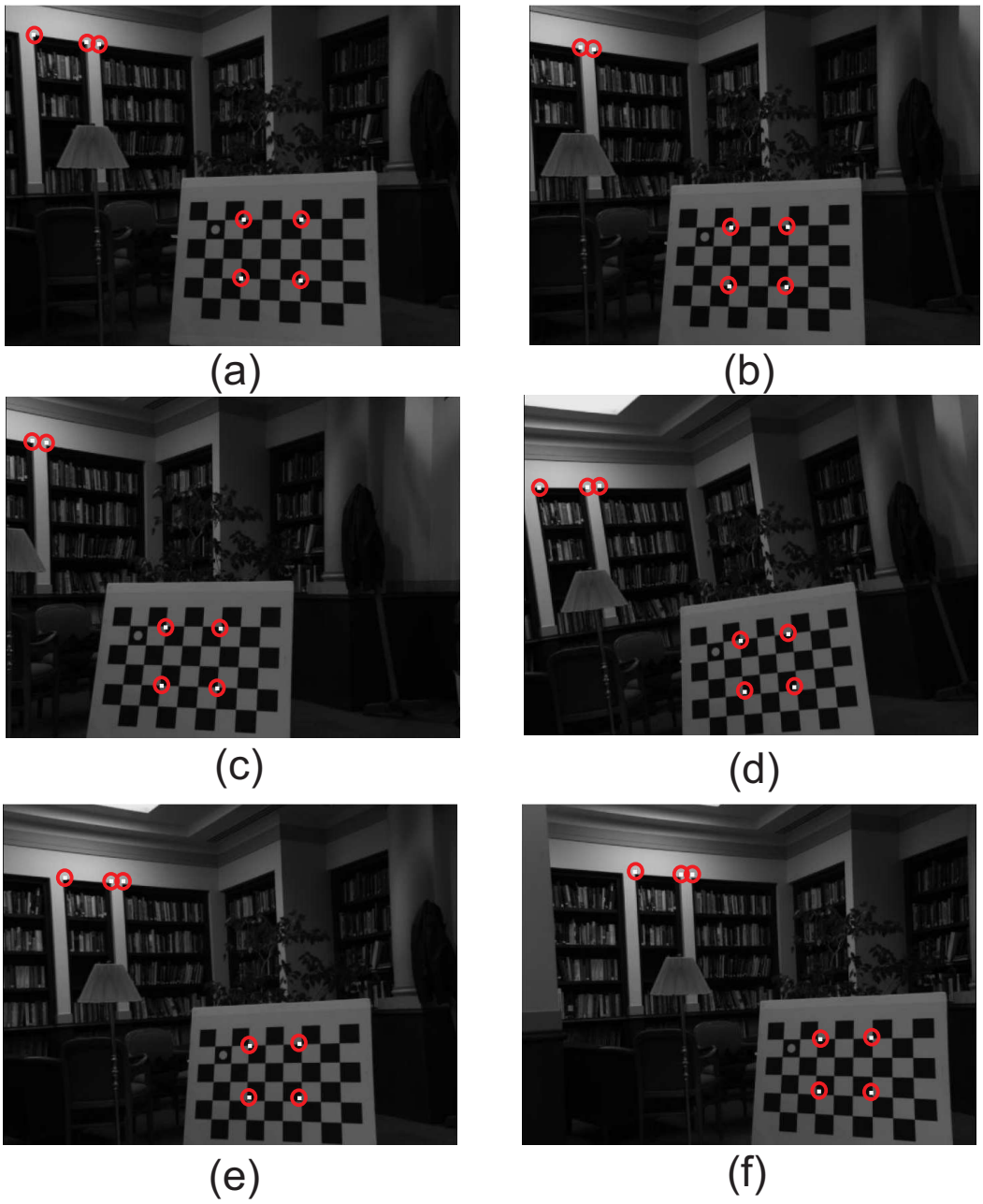


Figure 7.3: Tracking a hand-held camera motion. The tracking results are illustrated through projecting seven 3D scene points onto the *center* image. The projection matrix is computed using the motion estimate. The reference scene points are marked as white patches surrounded by red circles. (a) The extracted reference points in the *center* image at the initial frame. (b)-(f) The projected points in the *center* image at frame 20, 60, 160, 180 and 200.

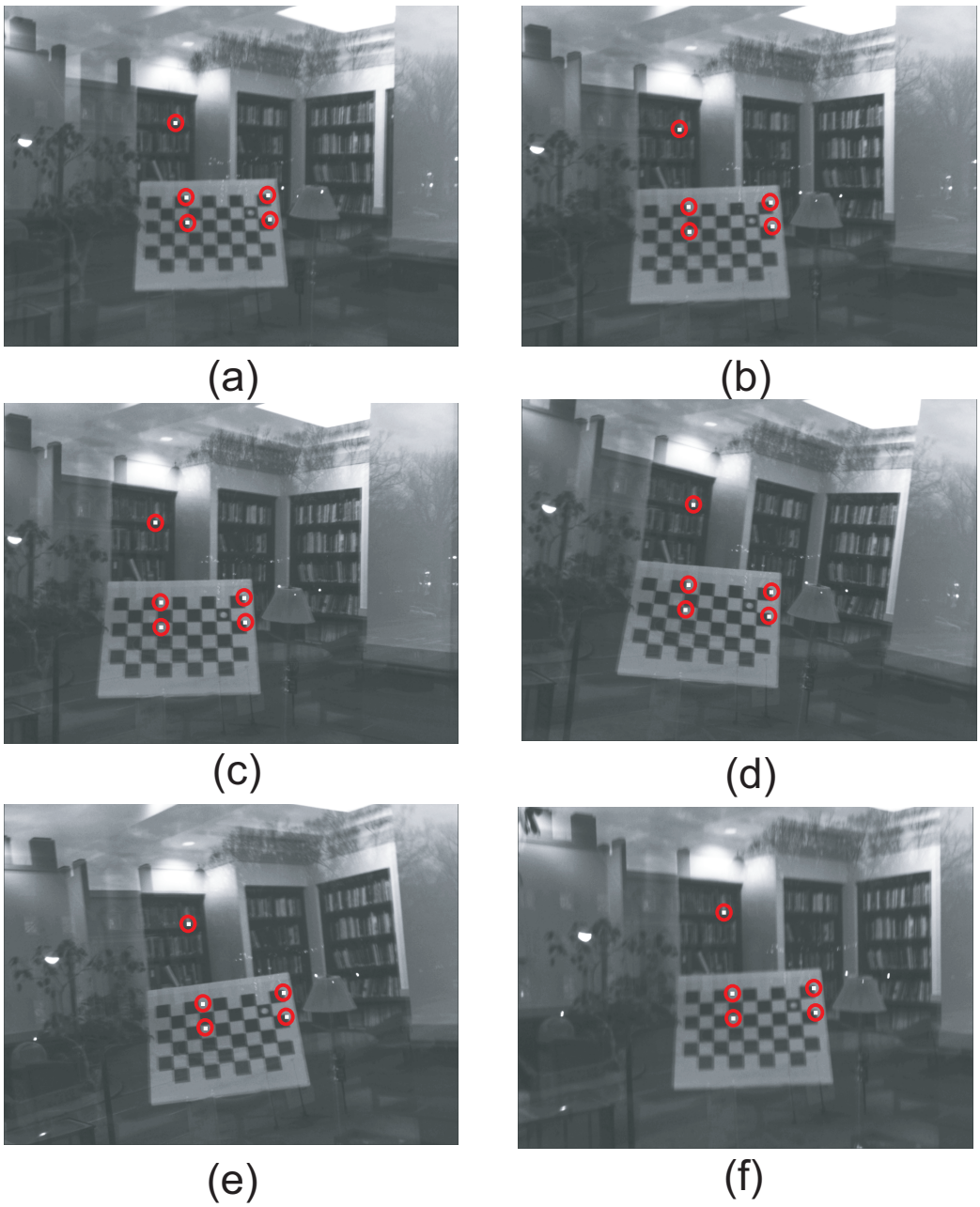


Figure 7.4: Tracking a hand-held camera cluster moving in a scene with semi-transparency. Watch the *ghosting* of the outside trees and chimneys on top of the inside bookshelves. The tracking results are illustrated through projecting five 3D scene points onto the *center* image. The projection matrix is computed using the motion estimate. The reference points are marked as white patches surrounded by red circles. (a) The manually selected points in the *center* image at the initial frame. (b)-(f) The projected points in the *center* image at frame 10, 20, 30, 50 and 60.

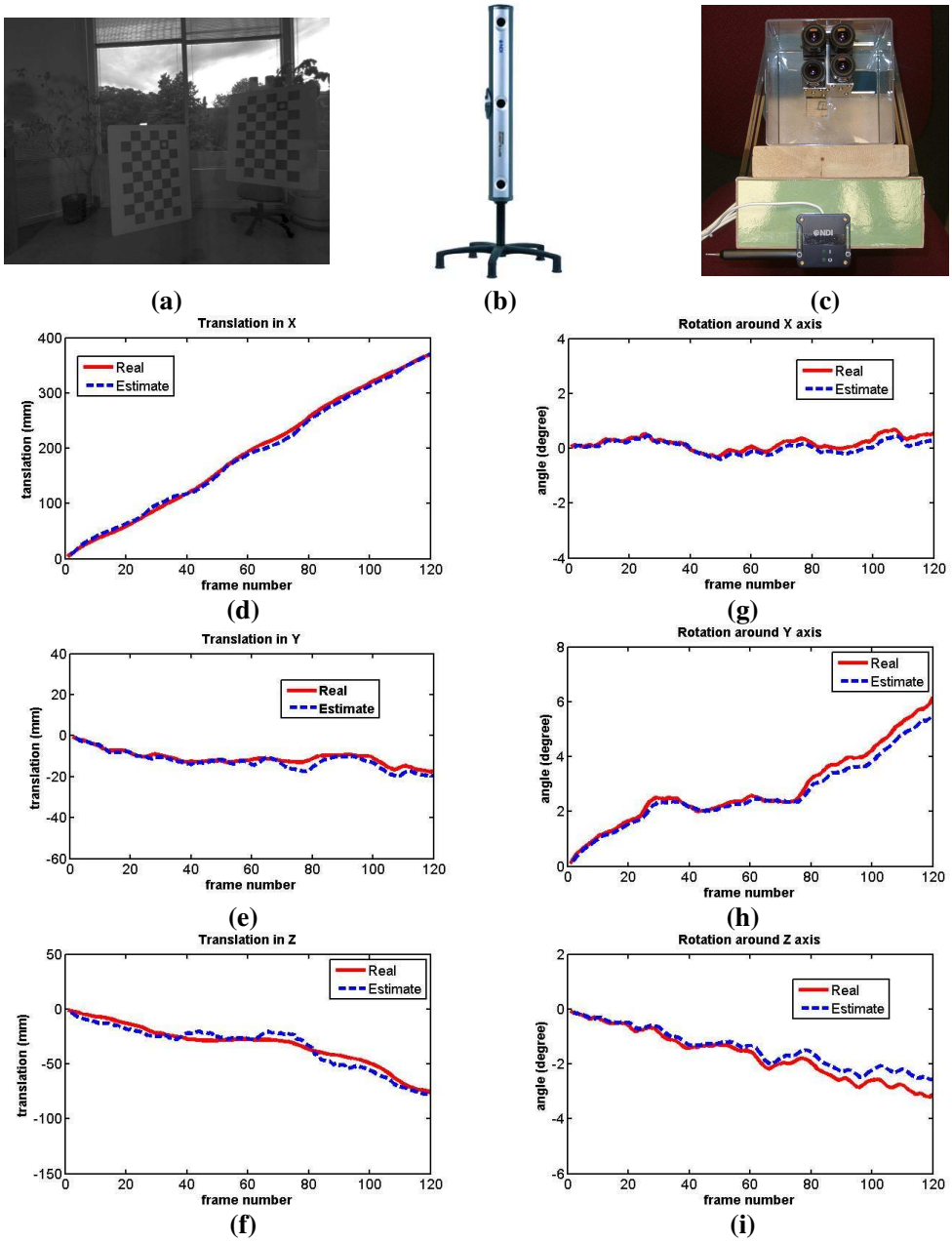


Figure 7.5: Tracking a hand-held camera with known ground-truth motion. (a) An image of the scene. (b) An Optotrak position sensor with three lenses mounted on the stabilized bar. (c) A rigid panel with four LEDs are mounted on the box containing the DCC. (d)-(f) Estimation of camera translations (in mm) in X, Y and Z directions. (g)-(i) Estimation of camera rotation angles (in degree) around Y, X and Z axes. Red (solid) curves represent the accumulated motion reported by the Optotrak system, blue (dashed) curves represent the values estimated by the differential tracker.

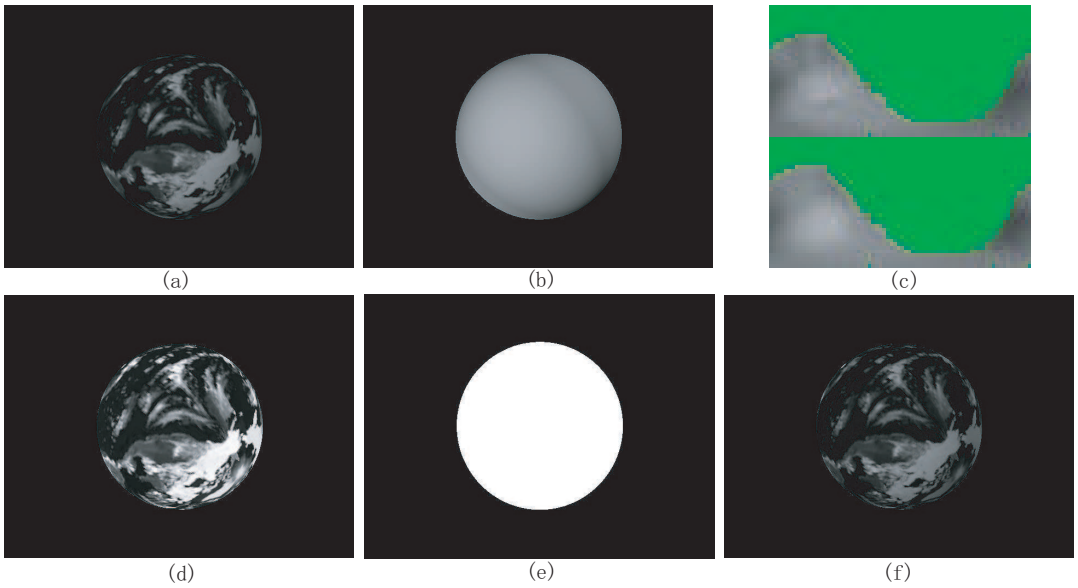


Figure 7.6: Illustration of the illumination correction process. (a) An illuminated textured sphere. (b) An illuminated texture-less white sphere. (c) Upper: an INM computed by comparing (a) and (d). Lower: an INM computed by comparing (b) and (e). (d) The textured sphere rendered without lighting. (e) The solid white sphere rendered without lighting. (f) The illumination-corrected textured sphere transformed from (d) using the upper INM in (c). Note that (a) and (b) are rendered under the same lighting condition.

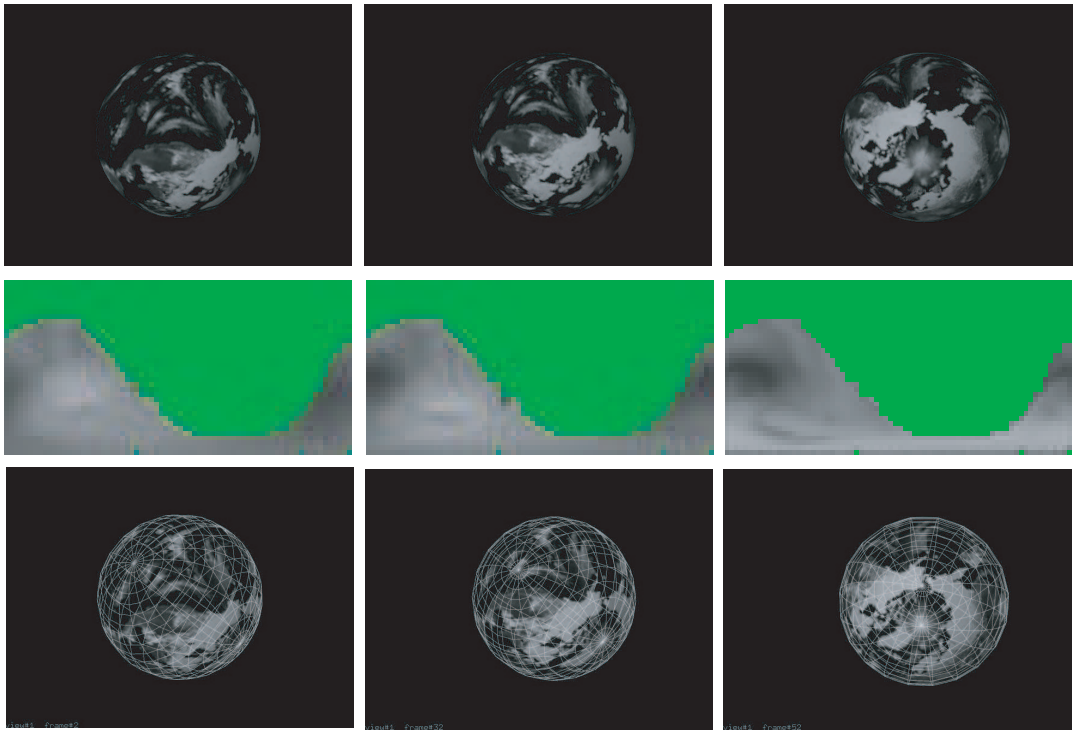
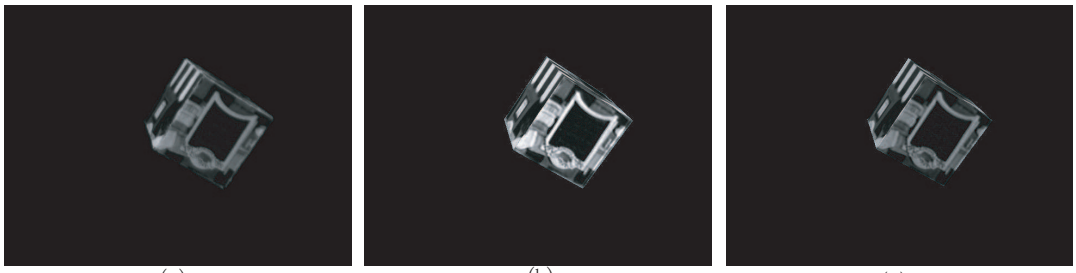


Figure 7.7: Tracking and INM estimation of a synthetic image sequence with varying illuminations. The first row shows three frames of the original synthetic sequence with lighting. The second row represents the estimated INMs for these frames. One can see that the lighting direction changes over frames. The last row represents the tracking results. Wire-frame sphere rendered using the estimated pose parameter are superimposed onto the original images.



(a) (b) (c)

Figure 7.8: Illumination correction on real data. (a) Real image. (b) Synthetic image without illumination correction. (c) Illumination corrected synthetic image.

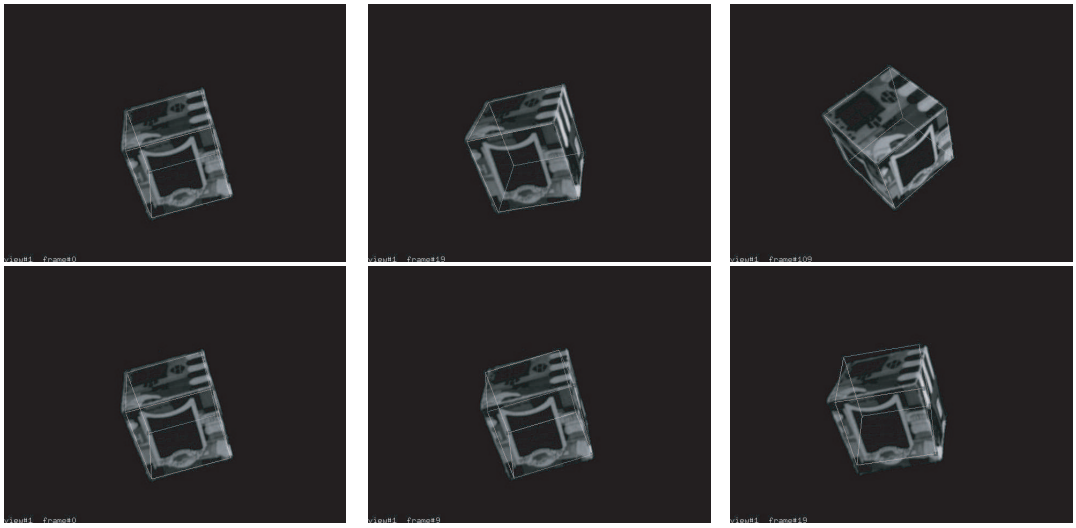


Figure 7.9: Tracking of a real cubic object under static directional lighting. Wire-frame cube rendered using the estimated pose parameter are superimposed onto the original images. The first row shows the tracking results (frames 1, 20 and 110) with illumination correction. The second row shows the tracking results (frames 1, 10 and 20) without illumination correction.

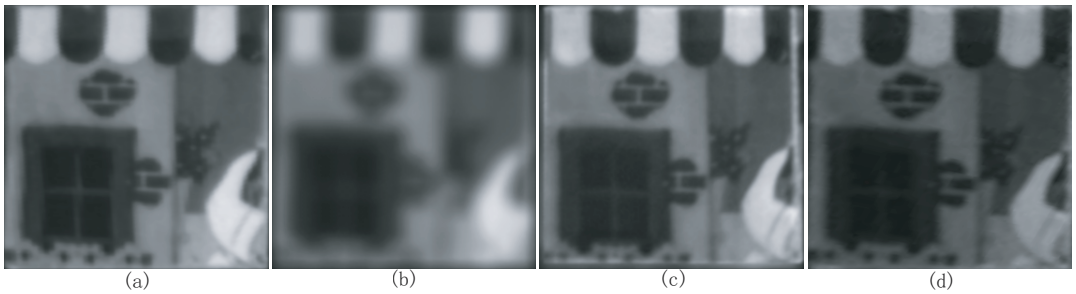


Figure 7.10: Texture refinement results on real data. (a) Original high-resolution texture. (b) Input texture of the initial off-line model. (c) Refined texture after 100 frames. (d) Texture extracted directly from a real image.

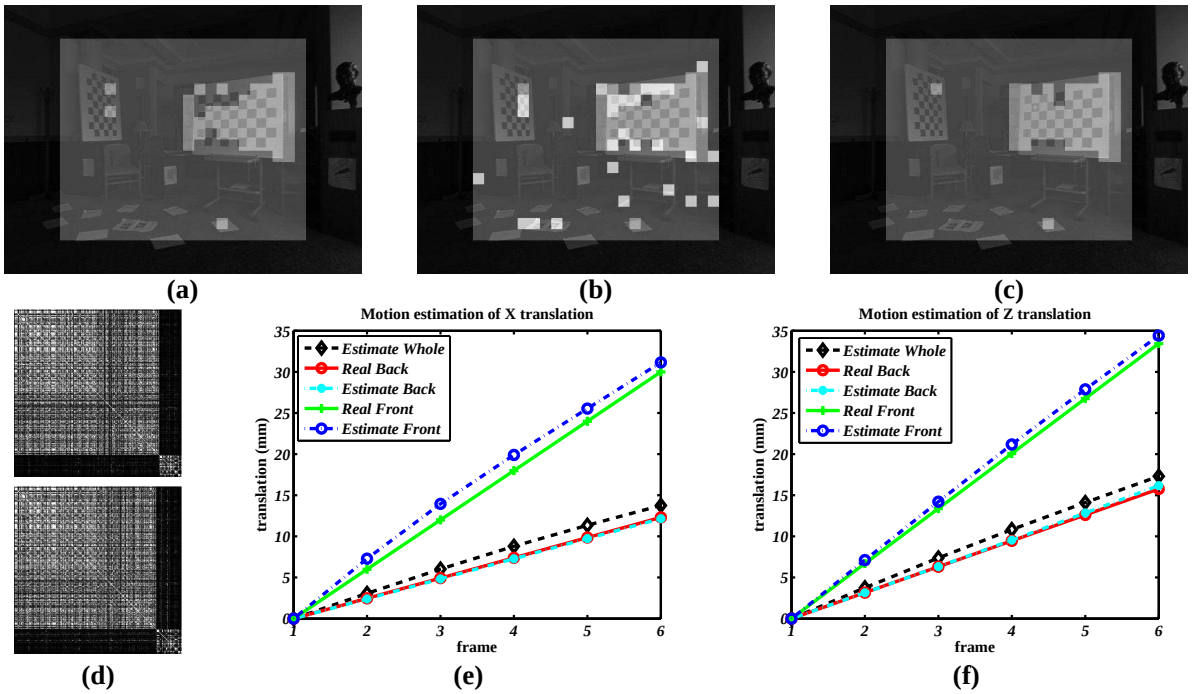


Figure 7.11: Motion segmentation and tracking results for a controlled sequence. (a) Segmentation results before refinement. (b) Segmentation results with ambiguous-pixels. (c) Segmentation results after refinement. (d) Similarity matrices before (top) and after (bottom) refinement. (e) Motion estimation of X translation. (f) Motion estimation of Z translation.

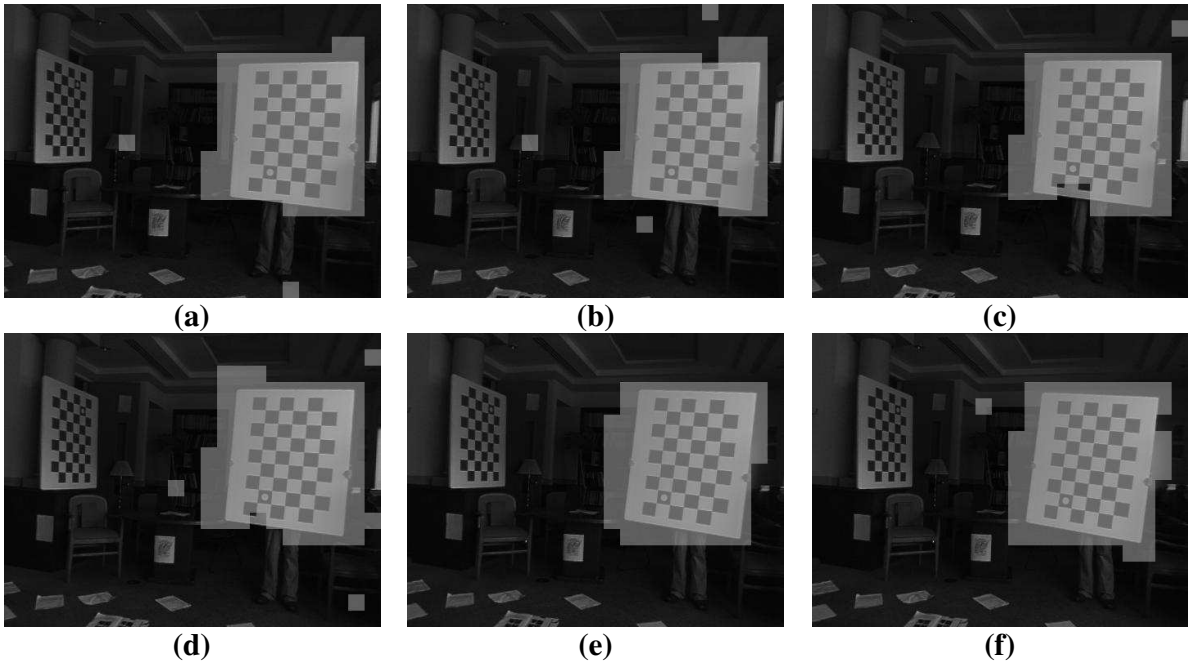


Figure 7.12: Segmenting free-form rigid motions using a DCC. The checkerboard and the camera were moved by hand. A sequence of 45 frames were captured. Images (a)-(f) show the segmentation results of 6 frames.

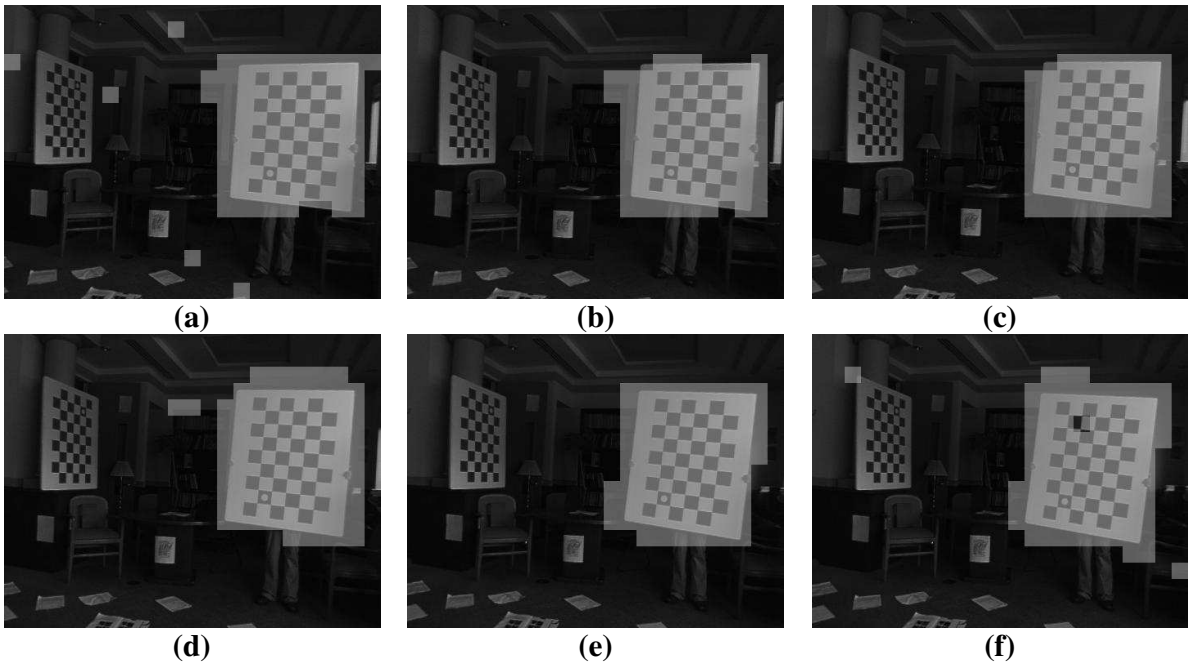


Figure 7.13: Segmenting free-form rigid motions using a single camera. The scene and the motion were the same as the one shown in Fig. 7.12. Images (a)-(f) show the segmentation results of the same 6 frames that are shown in Fig. 7.12.

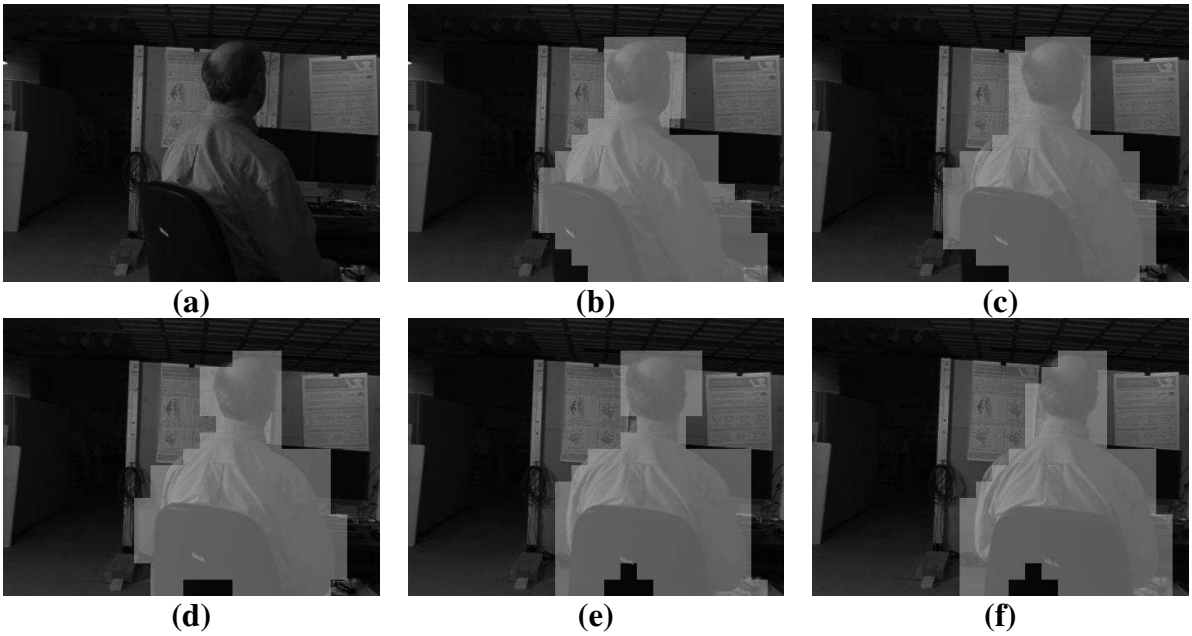


Figure 7.14: Motion segmentation in a scene with directional lighting using a single camera. A person was sitting on a chair rotating; the camera was moved by hand. A sequence of 40 frames were captured. (a) An image from the original sequence showing the person was illuminated by a directional light source from the left side. (b)-(f): Segmentation results of 5 frames.

Chapter 8

Conclusion and future work

In the previous chapters, I have presented the framework of differential tracking. I have described the algorithm of tracking through sampling and linearizing the local appearance manifold. I have developed techniques for acquiring appearance samples. I have analyzed the manifold linearization process from a signal processing point of view. In addition to tracking, I have also applied the locally linear appearance model to motion segmentation. I have tested these novel approaches on both real and synthetical image sequences. In this chapter, I will summarize the contributions and discuss some future research directions.

8.1 Summary of the thesis

The entire thesis is based on the local linearization of the appearance manifold. An image is considered as a point in a high dimensional space. As an object in the scene or the camera moves, the image point moves along a low-dimensional appearance manifold. The key observation is that while the appearance manifold is *globally* nonlinear thus is hard to learn, it can be *locally* linearized using a small number of nearby samples.

Motivated by the above observation, I have developed a novel differential approach to track incremental rigid motion. At each frame, a set of seven on-line local image samples are captured to linearize the appearance manifold around the current pose. At the next

frame, when the camera moves to a new pose, the incremental motion is estimated using the linearization from the previous frame. This pipeline continues as a new set of local samples are captured to compute a new linearization around the new pose, which is used to compute the new motion. By generating a piecewise linear approximation of the underlying manifold, tracking is achieved by traversing a series of tangential planes along the path of the motion.

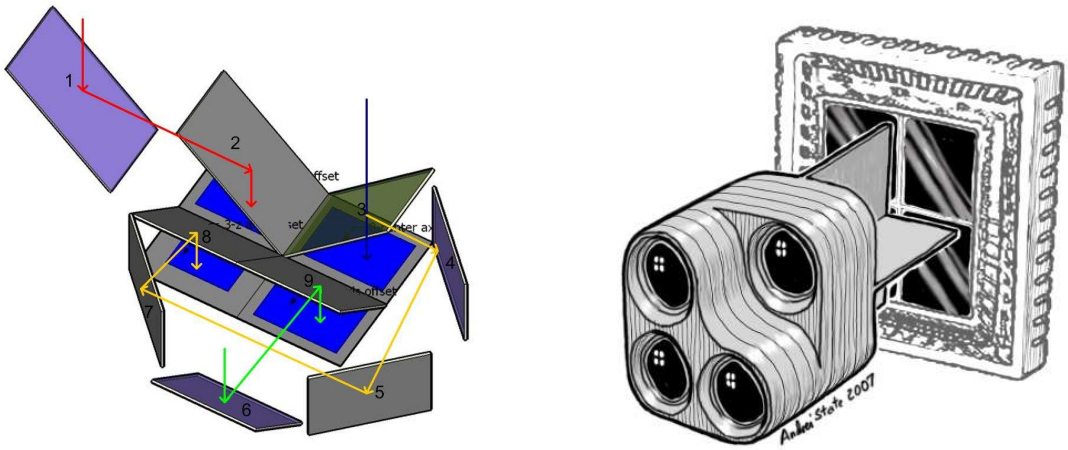
This differential tracking approach has several appealing properties. First, compared with feature-based methods, it directly employs intensity measurements thus avoids the difficulty of feature detection and matching. Therefore it can accommodate scenes with view-dependent appearance variances such as occlusion, semitransparency and curved reflection. Secondly, in contrast to the conventional appearance-based methods, it does not assume a learned off-line global model. The small number of local image samples can be captured on-line using a cluster setup. Thirdly, it is computationally simple. Once image samples are acquired, the remaining processes of manifold linearization and motion estimation only involve linear solver. Therefore, it can be integrated into a compact system with limited processing power.

The local linearization of the appearance manifold requires input image samples. I have presented two techniques for sampling the appearance manifold. For camera tracking in a general environment with no prior model, I have developed a DCC prototype to capture simultaneous offset samples of the scene. One issue with this *sampling through capturing* technique is that images captured from different cameras lie on different appearance manifolds. To address this issue, I applied inter-camera calibration to enforce geometric and photometric consistency across cameras. For model-based object tracking where a graphical model of the target object is available, appearance samples can be generated through graphic rendering. In this so-called *sampling through rendering* approach, local appearance samples consist of both real and synthetic images. To address the illumination inconsistency between the real and the modeled environments, I have

developed a Kalman filter framework to correct the illumination of the synthetic images.

The differential tracking is based on a linearization of the appearance manifold. In theory, given seven samples one can always generate a linear approximation of the sampled appearance manifold. However this linearization is only accurate within a local region. Intuitively, an appropriate linearization requires the displacement between samples to be small and the appearance manifold (or usually a filtered version of it) to be smooth. To quantitatively determine the locally linear region, I have studied the process of locally sampling and linearizing the appearance manifold from a signal processing point of view. Using the Fourier analysis I have shown that to avoid aliasing the maximum image shift (sum of the horizontal and vertical shifts) between adjacent image samples should be smaller than one (blurred) pixel. I have applied this spectral analysis to guide the design of a DCC and to determine the filter kernels for blurring images (smoothing appearance manifold). I believe the analysis can also be used to explain the commonly used sub-pixel motion constraint in the optical flow field, as the computation of dense optical flow is based on a local linearization of the brightness constancy assumption.

In addition to tracking, the local linear appearance model can also be applied to 3D motion segmentation. I have presented a novel appearance-based approach to cluster individual pixels into groups associated with different underlying rigid motions. Based on a local linear mapping between the changes of the pixel intensities and the underlying motion, I have introduced the notion of pixel intensity trajectories — a vector that records the intensity changes of a specific pixel over a sequence. I have shown that, similar to 2D feature trajectories, the intensity trajectories of pixels corresponding to the same motion span a linear subspace. Therefore I have formulated motion segmentation as clustering local linear subspaces. Note that I have applied a locally linear appearance subspace representation to both tracking and motion segmentation. However, the two subspaces are different. For tracking, I use subspaces spanned by difference images



(a) **(b)**

Figure 8.1: Design of a single chip DCC.

(multiple pixels single frame). While for motion segmentation, I employ subspaces spanned by intensity trajectories (single pixel multiple frames).

8.2 Future work

In the future, I would like to investigate several research directions to enhance the performance and extend the applicability of the approaches described in this thesis.

Sensor design

One lesson I have learned from the development of the DCC prototype is the difficulty of repeatedly performing photometric calibration. Due to fabrication variations and electronic noise, different cameras, even of the same model, can exhibit different intensity responses. To address this issue, I have adopted a semi-automatic inter-camera photometric calibration process to align the camera response curves (see Section 4.1.2 for details). However, the resulting alignment is only optimized for the lighting condition under which the cameras are calibrated. When the lighting condition changes, or when the camera settings such as shutter speed or gain values have been changed, the previous calibration becomes invalid. While it may be possible to develop a fully automatic

photometric calibration procedure, I believe the real solution comes from the designing of more suitable imaging hardwares.

Figure 8.1(a) illustrates a solution using a special optical design. By adding a mirror array in front of a conventional camera, we can acquire four offset images using a single imaging sensor. The four blue rectangles indicate that the lens of the original camera is divided into four quadrants that correspond to the four offset cameras. [1] The other nine indexed rectangles indicate eight mirrors and a beam splitter (rectangle 3). The arrows indicate the optical paths of the offset cameras. Blue, red, green and yellow represent the *center*, X, Y and Z translational cameras respectively. Note that the Z camera is accomplished through beam splitting.

Compared with the multiple camera setup, this single-sensor-multiple-camera design has several advantages. Most importantly, photometric consistency is guaranteed across the four offset cameras, as they all share the same chip circuit. Besides, the system is more compact. Moreover, the bandwidth for transferring images is reduced. Instead of four images, we only need to take one full-resolution image and divide it to acquire four quarter-resolution image samples. Note that the reduced resolution does not affect the performance of our differential tracker, as it only requires low-resolution inputs.

The above optical design converts a conventional camera into a differential sensor. Looking into the future, we can imagine a sensor that is specifically designed for differential tracking. A conceptual illustration of such a sensor is shown in Figure 8.1(b). The differential sensor consists of four lenses and four separate optical paths, but only one CCD panel. Note that since we only need low-resolution input from the sensor, we can use a CCD with large pixels. As more photons are received by each pixel within an unit time interval, a larger pixel size means higher signal-to-noise ratio for a certain shutter speed, or a higher shutter speed for a certain signal-to-noise ratio. Note

[1] The separation of the lens is achieved by adding a cross baffle between the lens and the CCD chip, as illustrated in Figure 8.1(b).

that the signal-to-noise ratio of the sensor determines the maximum number of gradient level or the intensity resolution it can provide, and its shutter speed determines its highest framerate. Therefore by using larger CCD pixels, we can capture images with higher intensity resolution at a faster speed. Both may improve the performance of the differential tracker.

Finally, we can imagine a compact system consists of small optical sensors and specialized embedded hardware such as FPGA. We can use such a stand-alone system as a probe for sensing incremental motion.

Hybrid system

As an incremental method, the differential tracker is prone to the issue of drifting. [2] To address this issue, we can consider implementing a hybrid system. At the lower-level, the differential tracker will perform incremental motion estimation at a high framerate. At the upper-level, a host program will take motion estimate stream with periodic key frames from the differential tracker to provide key frame correction. For instance, we can consider combining the appearance-based differential tracker with a feature-based SFM system that usually generates less accurate results when estimating small scale motions. In such a hybrid system, the low-level differential tracker will provide efficient small motion estimation and the high-level SFM program will perform the computation-demanding keyframe bundle adjustment.

A generalized framework

Another interesting research direction is to accommodate non-standard visual sensors. The appearance-based framework described in this thesis is based on a general manifold representation. Potentially, it can be applied to any vision sensor whose measurements can be vectorized and represented as a point on a differentiable manifold. For example,

[2] Note that the accumulated error from the previous frames does not affect the accuracy of the motion estimate of the current frame.

two parallel 1D cameras with a position offset can be used to track 1D motion. It may also be possible to apply the framework to heat sensors.

Finally, I would like to extend the framework to segment and track non-rigid motion that is usually represented as a linear combination of a set of basis motions. Again, I will represent images as points on an appearance manifold. I will then apply the same algorithm to linearize the local appearance manifold and estimate the motion. The only difference is that the manifold and motion are now parameterized by the coefficients of the motion basis.

Filter design and adaptive framerate

In Chapter5, I studied differential tracking from a signal process point-of-view. To address sampling issue, I applied Fourier analysis to the 8D appearance function and computed its bandwidth in motion dimensions. The analysis was based on the assumption that the images was blurred and its spectral support was bounded. I then derived the sub-pixel motion constraint that can be used to determine the cut-off frequency of the filter.

In the experiments of this dissertation work, I empirically chose the commonly used Gaussian filter and acquired reasonably good results. However, this choice is not fully justified. One potential issue with Gaussian blurring is its high pass-attenuation. In fact, both low and high frequency components are useful to motion estimation. The former plays a critical role in estimating relatively large motions, and the latter is more sensitive to small ones. Therefore a Gaussian filter with a cut-off frequency determined by the baseline of the camera cluster may not be the best choice for estimating camera motion that are much smaller than the baseline.

In the future, I would like to test the differential tracking algorithm using different filters. An immediate candidate would be a *sinc* filter that has equal weight for all the frequency components below its threshold. Another idea is to develop a differential

tracker with adaptive framerates. Instead of estimating motion for each frame, the *smart* tracker can choose to hold and only perform motion estimation when it detects enough motion. The final solution might be a combination of these two methods.

Appendix A: Projective camera model

This appendix describes the decomposition of the projection matrix P for a projective camera model. Detailed discussion on various camera models can be found in Chapter 5 of (HZ00).

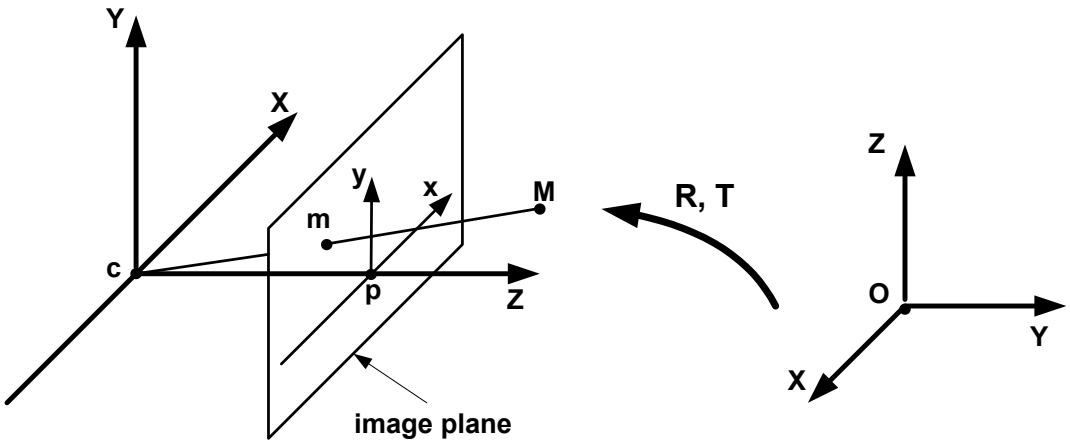


Figure 8.2: Rigid transformation between the world and camera coordinate systems. The left side shows the coordinate system of a pinhole camera. The right side shows the world coordinate system. R and T indicate the rotation and translation from the world coordinate to the camera coordinate.

The left side of Figure 8.2 shows a pinhole camera model. c is called the *camera center*. It is the center of projection and the origin of the camera coordinate system. Z axis is perpendicular to the image plane and is called the *principle axis*. p is called *principle point*. It is the intersection of the principle axis and the image plane. A 3D point M is projected to a 2D point m in the image plane. Representing M and m in the homogeneous coordinate as $[X, Y, Z, 1]^T$ and $[x, y, w]^T$, the center projection can be written in a matrix form as

$$m = PM \tag{A-1}$$

The 3×4 matrix P is called the *camera projection matrix*. If we define the camera

center to be the origin and the three camera axes to be the axes of the world coordinate system, P can be written as

$$P = diag(f, f, 1)\,[E|0] \tag{A-2}$$

where E represents the identity matrix.

Now, let us extend the discussion to the projective camera model. In this case, the projection matrix P can be decomposed as

$$P = K\,[R|T] \tag{A-3}$$

The 3×3 matrix R and the 3×1 vector T represent *extrinsic parameters*. They defines the rigid transformation (rotation and translation) from the world to the camera coordinate.

$$M_{cam} = RM_{world} + T \tag{A-4}$$

The translation $T = -RC$, where C is the world coordinate of the camera center. K is the *intrinsic matrix* representing the camera intrinsic parameters. It is of the form

$$K = \begin{bmatrix} f/p_x & s & x_0 \\ 0 & f/p_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-5}$$

where f is the focal length, p_x and p_y are the pixel size in x and y directions of the image plane, $[x_0, y_0]$ are the image coordinate of the principle point, and s is the skew parameter. For normal CCD cameras, $s = 0$.

Appendix B: Motion projection and image flow

In this appendix, the image flow is computed using the pinhole camera model. The result is used by the Fourier analysis of Chapter 5.

Let M be a 3D point in the scene. M is observed twice by a pinhole camera from a reference pose C_0 and an offset pose C_1 (after certain motion). Representing M in the homogeneous coordinate as $[X, Y, Z, 1]$. Its 3D cartesian coordinates with respect to both views, $[\dot{X}_0, \dot{Y}_0, \dot{Z}_0]$ and $[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]$, can be computed using (A-1). Its image coordinates $[x_0, y_0]$ and $[x_1, y_1]$ can be computed as

$$[x, y] = [\dot{X}/\dot{Z}, \dot{Y}/\dot{Z}] \tag{B-1}$$

Without loosing generality, we can define the world coordinates system to be same as the coordinate system of the pinhole camera at C_0. Therefore C_0 and C_1 become $[0, 0, 0, 0, 0, 0]$ and $[T_x, T_y, T_z, R_x, R_y, R_z]$, where $[T_x, T_y, T_z]$ and $[R_x, R_y, R_z]$ represent the translation and rotation from C_0 to C_1. We can then write the projection matrices P_0 and P_1 as (B-2) and (B-3). $R_1 = |R_{ij}|(i, j \in [1, 2, 3])$ is the orthonormal rotational matrix computed from $[R_x, R_y, R_z]$.

$$P_0 = \begin{vmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} = \begin{vmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \tag{B-2}$$

$$P_1 = \left| \begin{array}{ccc} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{array} \right| \left| \begin{array}{ccc} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{array} \right| \left| \begin{array}{cccc} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \end{array} \right| \tag{B-3}$$

For any camera motion $[T_x, T_y, T_z, R_x, R_y, R_z]$, we can compute the corresponding image flow $[u, v] = [x_1 - x_0, y_1 - y_0]$ using (A-1) and (B-1—B-3). For the analysis of local appearance manifold, we are particularly interested in 6 simple camera motion, namely 3 pure translations along the coordinate axes plus 3 rotations around the coordinate axes. Each of the 6 motions corresponds to a change of pose parameter C in only one dimension. The flows are listed in Table (8.1). The analysis in Chapter 5 only uses the result from Table 8.1. Readers that are short of time can skip the rest of the section, which mainly consists of the equations for computing flows.

Substitute (B-2) to (A-1) and (B-1). We can compute the 3D cartesian coordinates $[\dot{X}_0, \dot{Y}_0, \dot{Z}_0]$ and the 2D image coordinate $[x_0, y_0]$ of point M in the reference view as:

$$[\dot{X}_0, \dot{Y}_0, \dot{Z}_0]' = P_0[X, Y, Z, 1]' = [fX, fY, Z]' \tag{B-4}$$

$$[x_0, y_0] = [\dot{X}_0/\dot{Z}_0, \dot{Y}_0/\dot{Z}_0] = [fX/Z, fY/Z] \tag{B-5}$$

For camera translation in X or $C_1 = [T_x, 0, 0, 0, 0, 0]$. We can compute P_1, $[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]$ and $[x_1, y_1]$ using (B-6–B-8).

$$P_1 = KR_1|E, [T_x, 0, 0]'| = \left| \begin{array}{cccc} f & 0 & 0 & T_x \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right| \tag{B-6}$$

$$[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]' = P_1[X, Y, Z, 1]' = [fX + fT_x, fY, Z]' \tag{B-7}$$

$$[x_1, y_1] = [\dot{X}_1/\dot{Z}_1, \dot{Y}_1/\dot{Z}_1] = [(fX + fT_x)/Z, fY/Z] \tag{B-8}$$

We can then compute the flow $[u, v]$ as

$$[u, v] = [x_1 - x_0, y_1 - y_0] = [fT_x/Z, 0] \tag{B-9}$$

In a similar form we can compute the flow field for Y translation ($C_1 = [0, T_y, 0, 0, 0, 0]$) as $[u, v] = [0, fT_y/Z]$.

For camera translation in Z or $C_1 = [0, 0, T_z, 0, 0, 0]$. We can compute P_1, $[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]$ and $[x_1, y_1]$ using (B-10–B-12). The flow can be computed using (B-13).

$$P_1 = KR_1|E, [0, 0, T_z]'| = \begin{vmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & T_z \end{vmatrix} \tag{B-10}$$

$$[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]' = P_1[X, Y, Z, 1]' = [fX, fY, Z + T_z]' \tag{B-11}$$

$$[x_1, y_1] = [\dot{X}_1/\dot{Z}_1, \dot{Y}_1/\dot{Z}_1] = [fX/(Z + T_z), fy/(Z + T_z)] \tag{B-12}$$

$$[u, v] = [x_1 - x_0, y_1 - y_0] = [\frac{fX}{Z}\frac{T_z}{Z + T_z}, \frac{fY}{Z}\frac{T_z}{Z + T_z}] = [x_0\frac{T_z}{Z + T_z}, y_0\frac{T_z}{Z + T_z}] \tag{B-13}$$

When T_z is small compared to Z the flow can be approximated as $[u, v] = [\frac{x_0}{Z}T_z, \frac{y_0}{Z}T_z]$.

For camera rotation around X axis or $C_1 = [0, 0, 0, R_x, 0, 0]$. We can compute P_1, $[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]$ and $[x_1, y_1]$ using (B-14–B-16).

$$P_1 = KR_1|E, \mathbf{0}| = \begin{vmatrix} f & 0 & 0 & 0 \\ 0 & f\cos R_x & -f\sin R_x & 0 \\ 0 & \sin R_x & \cos R_x & 0 \end{vmatrix} \tag{B-14}$$

$$[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]' = P_1[X, Y, Z, 1]' = [fX, fY\cos R_x - fZ\sin R_x, Y\sin R_x + Z\cos R_x]' \tag{B-15}$$

$$[x_1, y_1] = [\dot{X}_1/\dot{Z}_1, \dot{Y}_1/\dot{Z}_1] = [\frac{fX}{Y\sin R_x + Z\cos R_x}, \frac{fY\cos R_x - fZ\sin R_x}{Y\sin R_x + Z\cos R_x}] \tag{B-16}$$

When the rotation angle R_x is small, we can use the following approximations that apply

to a small angle $\triangle$.

$$1/(1+\triangle) = 1 - \triangle + O(\triangle^2) \tag{B-17}$$

$$\begin{aligned} \cos\triangle &= 1 \\ \tan\triangle &= \sin\triangle = \triangle \end{aligned} \tag{B-18}$$

(B-16) can then be simplified as (B-19) and the flow can be computed using (B-20).

$$\begin{aligned} x_1 &= \tfrac{fX}{Y\sin R_x + Z\cos R_x} = \tfrac{fX}{Z}\tfrac{1}{\cos R_x + \frac{Y}{Z}\sin R_x} \approx \tfrac{fX}{Z}\tfrac{1}{1+\frac{Y}{Z}R_x} \approx \tfrac{fX}{Z}(1-\tfrac{Y}{Z}R_x) \\ y_1 &= \tfrac{fY\cos R_x - fZ\sin R_x}{Y\sin R_x + Z\cos R_x} = f\tfrac{\frac{Y}{Z}-\tan R_x}{\frac{Y}{Z}\tan R_x + 1} \approx f(\tfrac{Y}{Z} - R_x)(1-\tfrac{Y}{Z}R_x) \approx f(\tfrac{Y}{Z} - (1+\tfrac{Y^2}{Z^2})R_x) \end{aligned} \tag{B-19}$$

$$\begin{aligned} u &= x_1 - x_0 = \tfrac{fX}{Z}(1-\tfrac{Y}{Z}R_x) - \tfrac{fX}{Z} = -\tfrac{fX}{Z}\tfrac{Y}{Z}R_x = -\tfrac{fX}{Z}\tfrac{fY}{Z}\tfrac{R_x}{f} = -\tfrac{x_0 y_0}{f}R_x \\ v &= y_1 - y_0 = f(\tfrac{Y}{Z} - (1+\tfrac{Y^2}{Z^2})R_x) - \tfrac{fY}{Z} = -f(1+\tfrac{Y^2}{Z^2})R_x = -(f + \tfrac{y_0^2}{f})R_x \end{aligned} \tag{B-20}$$

Similarly for camera rotation around Y axis or $C_1 = [0,0,0,0,R_y,0]$, we can compute the flow as $[u,v] = [(f+\frac{x_0^2}{f})R_y, \frac{x_0 y_0}{f}R_y]$.

For camera rotation around Z axis or $C_1 = [0,0,0,0,0,R_z]$. We can compute P_1, $[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]$ and $[x_1, y_1]$ using (B-21–B-23).

$$P_1 = KR_1|E,\mathbf{0}| = \begin{vmatrix} f\cos R_z & -f\sin R_z & 0 & 0 \\ f\sin R_z & f\cos R_z & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \tag{B-21}$$

$$[\dot{X}_1, \dot{Y}_1, \dot{Z}_1]' = P_1[X,Y,Z,1]' = [fX\cos R_z - fY\sin R_z, fX\sin R_z + fY\cos R_z, Z]' \tag{B-22}$$

$$[x_1, y_1] = [\dot{X}_1/\dot{Z}_1, \dot{Y}_1/\dot{Z}_1] = [\frac{fX\cos R_z - fY\sin R_z}{Z}, \frac{fX\sin R_z + fY\cos R_z}{Z}] \tag{B-23}$$

Apply (B-18) to simplify (B-23), the flow can be computed as:

$$\begin{aligned} u &= x_1 - x_0 = \tfrac{fX\cos R_z - fY\sin R_z}{Z} - \tfrac{fX}{Z} = -\tfrac{fY}{Z}R_z = -y_0 R_z \\ v &= y_1 - y_0 = \tfrac{fX\sin R_z + fY\cos R_z}{Z} - \tfrac{fY}{Z} = \tfrac{fX}{Z}R_z = x_0 R_z \end{aligned} \tag{B-24}$$

I have now shown the equations for computing the optical flows of the six 1D motions. In particular, I have derived some simplified forms for small camera motions along the local appearance manifold. The result are summarized in Table 8.1.

Motion Type	Motion Magnitude	H-Flow(u)	V-Flow(v)
X Translation	T_x	$\frac{f}{Z}T_x$	0
Y Translation	T_y	0	$\frac{f}{Z}T_y$
Z Translation	T_z	$-\frac{x}{Z}T_z$	$-\frac{y}{Z}T_z$
X Rotation	R_x	$-\frac{xy}{f}R_x$	$-(f+\frac{y^2}{f})R_x$
Y Rotation	R_y	$(f+\frac{x^2}{f})R_y$	$\frac{xy}{f}R_y$
Z Rotation	R_z	$-yR_z$	xR_z

Table 8.1: Optical flow of six 1D motion. f denotes the camera focal length. x and y are the image coordinates. Z is the scene depth at pixel $[x, y]$.

Appendix C: Kalman Filtering

The Kalman filter (KF) is a popular and time-tested Bayesian estimation tool (Kal60). In the classical KF framework, an uncontrolled linear system is modeled as

$$\hat{X}_t^- = A\hat{X}_{t-1} + W_{t-1} \tag{C-1}$$

$$\hat{Z}_t = H_t\hat{X}_t^- + V_t \tag{C-2}$$

(C-1) is called the state-transition or process equation, where t is the time step, $\hat{X}$ is the estimate of the real system state X, $\hat{X}^-$ is the *a priori* (predicted) state, A is the deterministic state transition between steps, and W is the process noise. (C-2) is known as the measurement equation, where $\hat{Z}$ is the estimate of the real measurement Z, H represents the linear observation function that relates the state of the system to its measurements, and V is the measurement noise. W and V are assumed to be normally distributed with covariances Q and R respectively (used below), spectrally white, and independent of each other. In addition to estimating the system state $\hat{X}$, the filter also estimates the error covariance P. Similar to (C-1) it is assumed the *a priori* error covariance P^- can be modeled (predicted) using P from the previous step as

$$P_t^- = AP_{t-1}A^T + Q \tag{C-3}$$

A Kalman filter performs system state estimation using a prediction-correction mechanism. At each frame, the filter predicts $\hat{X}^-$ and $\hat{Z}$ using the deterministic portions of (C-1) and (C-2), and P^- using (C-6). It then corrects the state by subtracting the estimated from the real observations, and factoring the residual back into the estimated state. The *a posteriori* state and error covariance then serves as the basis for the next

step. The *measurement update* equations can be written as

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R)^{-1} \tag{C-4}$$

$$\hat{X}_t = \hat{X}_t^- + K_t(Z_t - H_t \hat{X}_t^-) \tag{C-5}$$

$$P_t = (I - K_t H_t) P_t^- \tag{C-6}$$

The *Kalman gain* K_t computed in (C-4) provides the optimal weighting of the observation residual in (C-5), minimizing the mean of the diagonal of P.

While (C-1)–(C-6) model linear systems, the filter can be reformulated to accommodate a nonlinear process function $a(X)$ and/or observation function $h(X)$ using linear approximations around $\hat{X}$. In this *extended Kalman filter* (EKF) formulation the matrices A and H in (C-1) and (C-2) are replaced by the Jacobian matrices $A = \partial a / \partial \hat{X}$ and $H = \partial h / \partial \hat{X}$ respectively. Note that our model-based object tracking system uses an EKF, where the Jacobian matrix H is computed from local appearance samples. More complete description about KF/EKF can be found in (WB95).

Bibliography

[Alt92] T. Alter. 3d pose from three corresponding points under weak-perspective projection. Technical Report AIM-1378, Massachusetts Institute of Technology, 1992. 13

[Ana89] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989. 11, 21

[BA96] M. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996. 85

[BFB94] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994. 11, 20

[BH83] A. Bruss and B. Horn. Passive navigation. *Computer Vision, Graphics and Image Process*, 21, 1983. 22

[Bis84] G. Bishop. *The Self-Tracker: A Smart Optical Sensor on Silicon.* PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1984. 14

[BJ98] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998. 27

[BN95] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5), 1995. 13

[Bou08] J. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguet/calib_doc/`, 2008. 40

[BRS+07] S. Baker, S. Roth, D. Scharstein, M. Black, J. Lewis, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2007. 20

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8:679–714, 1986. 11

[CC01] Y. Chang and Y. Chen. Robust head pose estimation using textured polygonal model with local correlation measure. In *PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, pages 245–252, 2001. 24, 25

[CK95] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, page 1071, 1995. 86, 92

[CMC03] A. Comport, É. Marchand, and F. Chaumette. A real-time tracker for markerless augmented reality. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 36, 2003. 13

[CN98] Y. Cho and U. Neumann. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the Virtual Reality Annual International Symposium*, page 212, 1998. 13

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–151, 2000. 24

[CSA00] M. Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An apporach based on registration of texture-mapped 3d models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(4), 2000. 24, 25

[CTCS00] J. Chai, X. Tong, S. Chan, and H. Shum. Plenoptic sampling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, 2000. 52

[DTT98] F. Dellaert, S. Thrun, and C. Thorpe. Jacobian images of superresolved texture maps for model-based motion estimation and tracking. In *Proceedings of the Fourth Workshop on Applications of Computer Vision*, 1998. 24

[Elg05] A. Elgammal. Learning to track: Conceptual manifold map for closed-form tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–730, 2005. 27

[FB87] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740, 1987. 15, 85

[FJ90] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990. 11

[FL01] O. Faugeras and Q. Luong. *Multiple View Geometry in Computer Vision: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. MIT Press, ISBN: 0262062208, 2001. 17

[FPF99] A. Fitzgibbon, M. Pilu, and R. Fisher. Direct least square fitting of ellipses. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(5):476–480, 1999. 14

[Fre00] B. Frey. Filling in scenes by propagating probabilities through layers and into appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–192, 2000. 24

[Gan84] S. Ganapathy. Decomposition of transformation matrices for robot vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 130–139, 1984. 15

[GBG+94] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. RakshitLucas, and C. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 220–228, 1994. 10

[Gea98] C. Gear. Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150, 1998. 86, 92

[GH86] N. Greene and P. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6), June 1986. 48

[GRS+02] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab. Marker-less tracking for ar: A learning-based approach. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, page 295, 2002. 13

[GW06] A. Gruber1 and Y. Weiss. Incorporating constraints and prior knowledge into factorization algorithms - an application to 3d recovery. *Lecture Notes in Computer Science*, 3940:151–162, 2006. 87

[Han91] J. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 156–162, 1991. 22, 23

[Har97] R. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593, 1997. 17

[HB98] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998. 25, 27

[HJ92] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion i: algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992. 22

[HLON91] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 592–598, 1991. 15

[HN94] T. Huang and A. Netravali. Motion and structure from feature correspondences: A review. *Proceeding of IEEE*, 82(2):252–268, 1994. 17

[HN96] R. Holt and A. Netravali. Uniqueness of solutions to structure and motion from combinations of point and line correspondences. *Journal of Visual Communication and Image Representation*, 7:126–136, 1996. 13

[HO93] K. Hanna and N. Okamoto. Combining stereo and motion for direct estimation of scene structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 357–365, 1993. 22

[Hor86] B. Horn. *Robot Vision*. McGraw-Hill, New York, 1986. 18

[Hor87] B. Horn. Motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 1(3):239–258, 1987. 22

[HS81] B. Horn and B. Schunk. Determing optical flow. *Artificial Intelligence*, 17:185–204, 1981. 11, 19, 20

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988. 10

[HVM98] C. Hashizume, V. Vinod, and H. Murase. Robust object extraction with illumination-insensitive color description. In *Proceedings of the International Conference on Image Processing*, pages 50–54, 1998. 10

[HZ00] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000. 17, 124

[IA99] M. Irani and P. Anandan. About direct methods. In *Proceedings of the ICCV Workshop on Vision Algorithms*, pages 267–277, 1999. 22

[IAC02] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(11):1528–1534, 2002. 23

[Ira02] M. Irani. Multi-frame correspondence estimation using subspace constraints. *International Journal of Computer Vision*, 48(3):173–194, 2002. 22, 86

[IW05] A. Ilie and G. Welch. Ensuring color consistency across multiple cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1268–1275, 2005. 43

[JFEM03] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003. 24

[Jur99] Frederic Jurie. Solution of the simultaneous pose and correspondence problem using Gaussian error model. *Computer Vision and Image Understanding: CVIU*, 73(3):357–373, 1999. 13

[Kal60] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 131

[Kan01] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 586–591, 2001. 87, 92

[LF96] Q. Luong and O. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75, 1996. 18

[LF05] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. *Found. Trends. Comput. Graph. Vis.*, 1(1):1–89, 2005. 14

[LH81] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981. 17

[LHF90] Y. Liu, T. Huang, and O. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(1), 1990. 13

[LHH+98] H. Liu, T. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding: CVIU*, 72(3):271–286, 1998. 20

[LK81] B. Lucas and T. Kanade. An interactive image registration technique with an application to stereo vision. *DARPA IU Workshop*, pages 121–130, 1981. 11, 20

[LK89] B. Lucas and T. Kanade. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. 10

[Low87] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987. 13

[Low92] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992. 13

[Low04] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 11

[Luc84] B. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1984. 20

[MN95] H. Murase and S. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995. 26, 27

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. 11

[NF93] N. Navab and O. Faugeras. Monocular pose determination from lines: Critical sets and maximum number of solutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 254–260, 1993. 13

[Nis03] D. Nister. An efficient solution to the five-point relative pose problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 02, page 195, 2003. 18

[NJW01] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2001. 93

[OSvG07] K. Ozden, K. Schindler, , and L. van Gool. Simultaneous segmentation and 3d reconstruction of monocular image sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007. 86

[PKG98] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 90–95, 1998. 18

[QL99] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, 1999. 15

[RMG98] Y. Raja, S. McKenna, and S. Gong. Segmentation and tracking using color mixture models. In *Proceedings of the Asian Conference on Computer vision*, pages 607–614, 1998. 10

[RRD05] A. Rahimi, B. Recht, and T. Darrell. Learning appearance manifolds from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 868–875, 2005. 27

[RRD07] A Rahimi, B. Recht, and T. Darrell. Learning to transform time series with a few examples. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(10):1759–1775, 2007. 27

[RS00] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. 28

[Se07] R. Stolkin and etal. *Scene Reconstruction, Pose Estimation and Tracking*. I-Tech Education and Publishing, Vienna, Austria, 2007. 14

[SFP96] S. Soatto, R. Frezza, and P. Perona. Motion estimation on the essential manifold. *IEEE Transactions on Automatic Control*, 41(3):393–413, 1996. 36

[SHC+96] A. State, G. Hirota, D. Chen, W. Garrett, and M. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438, 1996. 13

[SHE98] A. Schodl, A. Haro, and I. Essa. Head tracking using a textured polygonal model. In *Proceedings of Workshop on Perceptual User Interfaces*, 1998. 24

[Sin92] A. Singh. Optic flow computation: A unified perspective. *IEEE Computer Society Press*, 92, 1992. 21

[ST94] J. Shi and T. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 10

[Sut74] I. E. Sutherland. Three-dimensional data input by tablet. *ACM SIGGRAPH Computer Graphics*, 8(3):86–86, 1974. 15

[SW95] A. Shashua and M. Werman. Trilinearity of three perspective views and its associated tensor. In *Proceedings of the IEEE International Conference on Computer Vision*, page 920, 1995. 18

[TBM94] P. Torr, P. Beardsley, and D. Murray. Robust vision. In *Proceedings of the British Machine Vision Conference*, pages 145–154, 1994. 17

[TdSL00] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000. 28

[TK91] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991. 10

[TK95] C. Taylor and D. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, 1995. 17

[TMHF00] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000. 18

[TV07] R. Tron and R. Vidal. A benchmark for the comparison of 3d motion segmentation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 93

[TZ99] P. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Proceedings of the ICCV Workshop on Vision Algorithms*, pages 278–294, 1999. 23

[VH04] R. Vidal and R. Hartley. Motion segmentation with missing data by power factorization and by generalized pca. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–316, 2004. 86, 92

[VLF04] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(10), 2004. 13

[WA94] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994. 85

[WB95] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995. 132

[WBV+01] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. High-performance wide-area optical tracking: The hiball tracking system. *Presence: Teleoperators and Virtual Environments*, 10(1), 2001. 13

[Wei93] I. Weiss. Noise-resistant invariants of curves. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(9):943–948, 1993. 14

[XS05] J. Xiao and M. Shah. Accurate motion layer segmentation and matting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 698–703, 2005. 85

[YP06] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. pages 865–877, 2006. 86, 92, 93, 94

[YWP06] H. Yang, G. Welch, and M. Pollefeys. Illumination insensitive model-based 3d object tracking and texture refinement. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 869–876, 2006. 25

[ZC06] C. Zhang and T. Chen. Light field sampling. In *Synthesis Lectures on Image, Video, and Multimedia Processing*, volume 2, pages 1–102. 2006. 70

[ZF92] Z. Zhang and O. Faugeras. Estimation of displacements from two 3-d frames obtained from stereo. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(12):1141–1156, 1992. 17

[ZFS02] M. Zobel, M. Fritz, and I. Scholz. Object tracking and pose estimation using light-field object models. In *Proceedings of the Vision, Modeling, and Visualization Conference*, pages 371–378, 2002. 24

[ZH01] Z. Zivkovic and F. Heijden. A stabilized adaptive appearance changes model for 3d head tracking. In *RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, page 175, 2001. 25

[Zha98] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998. 18

[ZMMI06] L. Zelnik-Manor, M. Machline, and M. Irani. Multi-body factorization with uncertainty: Revisiting motion consistency. *International Journal of Computer Vision*, 68(1):27–41, 2006. 86, 92

[ZMY06] J. Zhang, L. McMillan, and J. Yu. Robust tracking and stereo matching under variable illumination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 871–878, 2006. 25

[ZSM06] K. Zimmermann, T. Svoboda, and J. Matas. Multiview 3d tracking with an incrementally constructed 3d model. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 488–495, 2006. 25

[ZT98] D. Ziou and S. Tabbone. Edge detection techniques: An overview. *International Journal of Pattern Recognition and Image Analysis*, 8(4):537–559, 1998. 11

[ZT99] T. Zhang and C. Tomasi. Fast robust and consistent camera motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 164–170, 1999. 22

[Zuc02] M. Zucchelli. *Optical Flow Based Structure from Motion*. PhD thesis, Royal Institute of Technology KTH, Stockholm, Sweden, 2002. 18