MULTIPLE VIEW GEOMETRY APPLICATIONS TO ROBOTIC AND VIDEO MANIPULATION

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science

by

Visesh Chari
200507022
visesh@research.iiit.ac.in

International Institute of Information Technology
Hyderabad, India
June 2008

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Multiple View Geometry Applications to Robotic and Video Manipulation" by Visesh Chari, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Advisors: Dr. C. V. Jawahar / Dr. P. J. Narayanan

"Two roads diverged in a wood and I - I took the one less traveled by, and that has made all the difference. ."

- ROBERT FROST (1874 - 1963)

To Udaykumar and Chitra Chari, my parents.

Acknowledgements

I am infinitely indebted to Dr. C. V. Jawahar and Dr. P. J. Narayanan for their guidance and support through the past 5 years. My association with them started in the second year of my Bachelors, and I have immensely benefited from their wisdom and knowledge. It is some of their courses that initiated me into the field of computer vision, and consequentially into research. I am also grateful to Dr. Anoop Namboodiri, with whom I have had several discussions related to some of the work in this thesis.

To my lab mates through Bachelors and Masters, I owe a big thanks for the fun-centered atmosphere in CVIT. I have been fortunate to have interacted with a long list of people, including Somu, Sashi, Sesh, Pavan, Ravi Kiran, Vaikba, Karteek, Parry, Bansi (pseudo-cvitian), Shetty, Bapu, Majji, Fam, Fanta, Nir, Google, Jagga, Shekhar, Tintin, Don, J. Lo, Hafez, Santosh, Pati, Ben, Dada, Balu, Sachin, Sharma, Pulkit etc. Others in the CVIT lab, though not mentioned here, have been equally supportive. Financial Support from CVIT during the period of my Masters is acknowledged.

Finally, and most importantly, this thesis is dedicated to my parents, whose unconditional love and support I have enjoyed throughout my life.

Abstract

Computer Vision may be described as the process of scene understanding through analysis of images captured by a camera. Understanding of a scene has several aspects associated with it, and this makes the field of computer vision a very vibrant and active field of research. For example, a section of the computer vision research concentrates on the understanding of the inherent characteristics of an object (identifying clauses like "this is a face", "this is a car" etc.). Yet another branch focuses on answering questions like "find the given face in this image" or "find where cars occur in this image". Another, more primitive of the branches, concerns itself with the estimation of the *geometry* of the scene. It answers questions like "what is the shape of this face", or "how would this car look from that viewpoint". This branch, and its various derivatives come under the name "Multiple View Geometry". Technically, Multiple View Geometry concerns itself with the *geometric interaction of the 3D world with images captured by the camera, and the interpretation and manipulation of this information for various tasks.* Multiple View Geometry (MVG) is two decades old in its research, and borrows heavily from a related field called *Photogrammetry.* Over the course of these years, many algorithms have been proposed for the estimation of geometric quantities like the transformation between cameras viewing a scene, or the 3D structure of a particular object being viewed by multiple cameras etc. The field has matured recently, with focus shifting towards producing *globally optimal* estimates of geometric quantities like transformations and structure, analysis of cases where the problem of geometric inference or manipulation is NP-Complete, etc. Even before maturity, many of the algorithms in Multiple View Geometry have found applications. The simple mosaicing solutions available in digital cameras these days, owes its origin to one such algorithm. Applications have also been spawned in areas like animation for films, robot motion in automated surgery and industrial environments, security systems that employ hundreds of cameras, etc.

This thesis focuses on the application front of Multiple View Geometry, which has started gaining popularity. To this extent, we leverage some of the concepts of MVG, to develop new frameworks and algorithms for a variety of problems. For this reason, we choose to explore the use of MVG in various robotics and computer vision tasks in this thesis. We first propose a tracking framework that utilizes various cues like textures and edges to perform tracking of 2D and 3D objects in various views of a scene. Tracking refers to the task of estimating the location and orientation of an object with respect to a pre-defined world coordinate system. Traditionally, filters like the Kalman Filter and its variants have been used for tracking purposes. Problems like illumination change and occlusion have affected many of these algorithms that make assumptions like uniform intensity of objects across views, etc. We show that by embedding MVG into tracking algorithms, we can achieve efficient tracking of objects, that is robust to large changes in perspective, illumination and occlusion. A by-product is the estimation of the pose of the camera, which in itself is useful for tasks like localization in a mobile environment.

Then we show an application of frequency domain based MVG to the task of robot positioning. Positioning (or *Visual Servoing*) is a task that enables a robot to assume a *desired pose* with respect to an *object of interest,* with the help of a camera. This object might be a heart, as in surgery, or an automobile part, as in industrial settings. We show that by using frequency domain techniques in MVG, we can achieve algorithms that require only rough correspondence between images, unlike earlier algorithms that needed specific point-to-point correspondences. This is further developed into a general framework for servoing that is capable of straight Cartesian paths and path following, which are recent problems in servoing.

Within computer vision, we explore the use of MVG for various image and video editing tasks. Tasks like removing a scene object from a video in a consistent manner would fall in this category (Predicting how the video would look like without the object). In this area, we propose an algorithm for video inpainting, where specific objects from a video are removed and resulting space-time holes are filled in a consistent manner. The algorithm is fully automatic unlike traditional image and video inpainting algorithms, and takes as input two functions; one function defines the object to be removed, and the other defines the background model that is used for hole-filling. We then extend this algorithim to obtain *Image Extrapolation*, which is concerned with prediction of the future of a scene using available content about it. This is different from Inference in the sense that no data is actually available to confirm our predictions and hence several alternatives remain equally viable. In this direction, we propose an inpainting based framework for Image Based Rendering (IBR). Image Based Rendering (IBR) concerns itself with algorithms for an image based representation of the 3D information of a scene. Novel views of the scene can then be rendered with this information. We extend IBR to include cases when 3D information about a particular scene is incomplete, by incorporating information about the type of scene being viewed (for eg. the face of a person). We then devise algorithms to transfer specific semantic characteristics to the current scene from similar scenes available to us.

Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

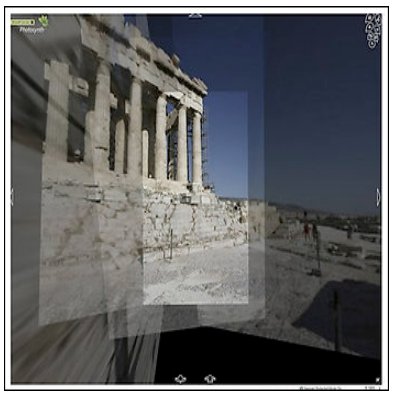
The understanding of visual information in the form of images has been a long-standing goal of Computer Vision. This understanding of visual content in a collection of images, sometimes time-sequenced (*i.e* a video) is desirable since many applications in Robotics, Graphics and even fields like Multimedia might benefit from it. Consequently, much effort has been devoted over the last few decades to this goal, labeled 'Scene-Understanding'. Although Scene-Understanding is a complex problem, considerable progress has been made at its various levels. One such level is the geometry of the scene, which relates its different images and is called Multiple View Geometry (MVG).

1.1 Multiple View Geometry: A Historical Perspective

The analysis of the geometry of the world using multiple views of it, captured by a camera came to computer science from its remote sensing counterpart: *photogrammetry.* Although photogrammetry research started as early as 1851 [14], the earliest of works on multiple view geometry can only be traced to the late-1960s [15]. In the past 3 decades however, computer vision research and especially research in multiple view geometry has progressed at an unimaginable rate, to result in softwares like the *Photosynth* [16]: a tool for visualization of photographs in 3 dimensional space. We look at some aspects of this research in the next few paragraphs.

Initial research: Although today's digital cameras are far more sophisticated than the *camera obscura,* that had its origin more than a 1000 years ago, the principle of a *pinhole* model for representing the image formation process still is a very practical approximation, and is widely used in computer vision. Some of the early works in multiple view geometry concentrated on extracting information about the cameras taking the photographs of a scene in a manner that was *independent of the scene being viewed* [17]. This was motivated by the fact that in order to infer the geometry of the world, it was first necessary to know the position of the camera(s) with respect to a previously chosen *world coordinate system.* This was when *projective geometry* found a use in computer vision, and has subsequently cemented a strong place in it [2]. Initial research also focused on robustness of the estimation of *projective* quantities like the *Fundamental Matrix* [2], which was necessary for accurate estimation of the camera *pose,* the geometric position of the camera with respect to a pre-defined world coordinate system.

Intermediate years: The next few years saw research develop in several directions. In one thread, the so called *structure-from-motion* systems were developed which aimed at producing *completely* automatic solutions for producing *robust 3D reconstructions* of the world, from just a set of images



(a) Photosynth screenshot

(b) Automatic Photo Pop-up screenshot

Figure 1.1: Photosynth and Automatic Photo pop-up screenshots: Examples of immersive environments created with the aid of Multiple View Geometry (MVG). Where photosynth explicitly computes 3D structure in order to render photographs according to geometry, automatic photo-pop up simply "props" a *single* image to approximately represent a 3D scene. Both however, make extensive use of multiple view geometry quantities.

as input. Another interesting thread was the development of *singular-value-decomposition* based reconstruction algorithms that were first proposed for simple scene models, and then extended to include the standard perspective model. Alternatively, efficient and robust projective quantities and reconstruction algorithms were developed that assumed an underlying *model* for the *scene geometry*; a popular modeling assumption was assuming that the 3D world consists of several *scene planes*.

Current research: Multiple view geometry (MVG) research in recent years has had a distinctively different flavour as compared to earlier years. With fundamentals of projective geometry and its applications to MVG being firmly established [2], questions about the accuracy of 3D reconstruction and its applications are getting attention. Many optimization algorithms that produce *global optima* without initialization, are being proposed for estimating projective geometric quantities like the projective camera matrix and the *homography* [18, 19]. On the other end, applications like the *automatic Photo pop-up* [20] and *Photosynth* make extensive use of multiple view geometry algorithms, in order to give an immersive experience to the user. In the Robotics side, MVG has applications in problems like *Visual Simultaneous Localization and Mapping* (vSLAM), and *Visual Servoing*. Whereas vSLAM assists robots in navigating unknown areas while creating 3D maps of the same using a camera, visual servoing deals with manipulating a robotic arm with respect to the object of interest. This has use for example in positioning a camera with respect to a scene for film-making, or for positioning medical equipment with respect to the object of interest for surgery; in such cases precision is paramount and in vision researchers see potential for a highly accurate sensor.

1.2 Multiple View Geometry: Overview

Multiple View Geometry (MVG) concerns itself with the relationships between multiple images of a three dimensional scene. Most of the earlier development in this direction [2] assumes the scene to be rigid. Also, specific interest has been devoted in the past to study the inter-relationships between

images of a geometric entity (*for e.g.* a plane). With the assumption that the camera capturing images mimics the Pinhole model [2] a variety of theorems and equations have been developed that constrain multiple views of such entities. These constraints can then be used to successfully identify and recognize objects [21, 11, 22] and also retrieve information about the parameters of the camera(s) capturing the scene [2]. In Section (1.2.1), we explain the basics of the Pinhole camera model that governs the image formation process. Next, we proceed to elaborate on the definitions and relationships between multiple views of a rigid scene that are relevant to our work (Sections (1.3), (1.2.3), (1.6)). These constraints and relationships are further used in Chapters 3 and 6 for various purposes.

1.2.1 Pinhole Camera Model

As shown in Figure 1.2, the Pinhole Camera Model represents the basic concept of image formation through perspective projection. A scene point $\mathbf{X}$ forms an image point $\mathbf{x}$ which is defined as the intersection of the line joining $\mathbf{X}$ and the camera centre $\mathbf{C}$ and the image plane. Mathematically, this is represented by the perspective projection function, and, in the case where the scene point is described in a different coordinate system than the camera's, the function takes the following form

$$[\mathbf{x}\ 1]^\top = \frac{1}{\lambda}\mathbf{K}\ (\mathbf{R}\,\mathbf{X} + \mathbf{t})$$

where $(\mathbf{R}_{3\times3}, \mathbf{t}_{3\times1})$ represent the *pose* (or extrinsic or external parameters) of the camera with respect to the world coordinate system, and $\mathbf{K}_{3\times3}$ represents its internal parameters. While image and world points are 2D and 3D entities in the Figure 1.2, they are represented in homogeneous coordinates in subsequent equations. Thus $\mathbf{X}_{4\times1}$ now represents the 3D point and $\mathbf{x}_{3\times1}$ represents its 2D image. For simplicity, we do not alter the symbols.

The parameter λ ensures that the last coordinate of the homogeneous image point $\mathbf{x}$ is 1, thus ensuring projection. Finally, an alternate way of representing the same equation in homogeneous coordinates after dropping the term λ for convenience and assuming the equality is only upto scale is

$$\mathbf{x} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]\,\mathbf{X}$$

where $\mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]$ represents the parameters of the camera and is known as the camera matrix. For the purpose of most derivations, the internal parameters $\mathbf{K}$ are either assumed equal to the identity matrix $\mathbf{I}_{3\times3}$.

In the context of pose, it is useful to describe the parametrization that we use for representation in many of the following chapters, since many different methods of representation for rotation exist. We use the parametrization based on quaternions, that represent rotation as a set of four parameters. Quaternions have been proved to be a very useful representation due to several advantages over other existing methods of representation like Euler angles. They are a compact representation of rotation that is numerically stable, and vary continuously in the $\mathfrak{R}^4$ space with rotation unlike Euler angles which have discontinuities. Their relationship to the actual rotation matrix does not involve trigonometric functions, and they make coupling of rotations an easy and simple task at the parameter level.

A quaternion representation of rotation is written as a normalized 4 dimensional vector $\mathbf{q} = [q_1\ q_2\ q_3\ q_4]^\top$. The rotation matrix $\mathbf{R}$ corresponding to a quaternion $\mathbf{q}$ is then given by

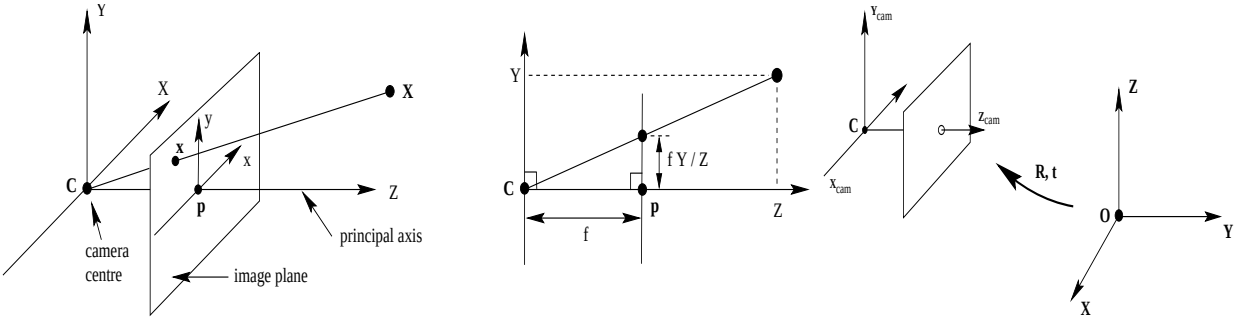


Figure 1.2: The image formation process. The left figure shows the case where a point in the camera coordinate system $\mathbf{X}$ is *projected* onto the image plane at the point $\mathbf{x}$. Point $\mathbf{C}$ is the camera centre or the origin of the camera coordinate system and the $\mathbf{Z}$ axis pierces through the image at $\mathbf{P}$. The side view in the left figure shows the actual projection process. The focal length f is part of the camera's internal parameters $\mathbf{K}$. The figure on the right shows perspective projection for the general case where the camera and world coordinate system, in which $\mathbf{X}$ is described, do not coincide. Image are courtesy [2].

$$\mathbf{R} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 - q_1q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_3 + q_2q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 - q_1q_2) \\ 2(q_2q_4 - q_1q_3) & 2(q_1q_2 + q_3q_4) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix}$$

It can be shown that the quaternion parametrization obeys the following constraint:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

During the course of a process like tracking, quaternions must sometimes be re-normalized due to rounding errors, to make sure that they correspond to valid rotations. This however, does not affect the estimation process and can be done outside the main algorithm, as an auxiliary correction step.

1.2.2 Homography

Relating two images $\mathbf{x}$ and $\mathbf{x}'$ of a 3D point $\mathbf{X}$ becomes impossible without the knowledge of the camera parameters and the value of $\mathbf{X}$ itself. However, when the point $\mathbf{X}$ lies on a plane $\mathbf{\Pi}$ (Figure 1.3), a simple geometric entity suffices to map one image point ($\mathbf{x}$) to another ($\mathbf{x}'$). This geometric entity is called the *homography* subtended by $\mathbf{\Pi}$, and is represented by $\mathbf{H}_{3\times3}$. Thus, as in the case of perspective projection, a homography maps one image point $\mathbf{x}$ to another $\mathbf{x}'$, upto scale.

$$\mathbf{x}' = \frac{1}{\lambda}\mathbf{H}\,\mathbf{x}$$

where λ is the normalizing factor. For the purpose of computation, the homography $\mathbf{H}$ is parametrized by 8 parameters, since multiplying it by a scale factor can be nullified by multiplying λ with the same factor. Thus, without loss of generality $\mathbf{H}(3,3)$ can be assumed to be 1. Since the above equation is a linear one, 8 equations are required to solve for the value of $\mathbf{H}$ in the minimal case,

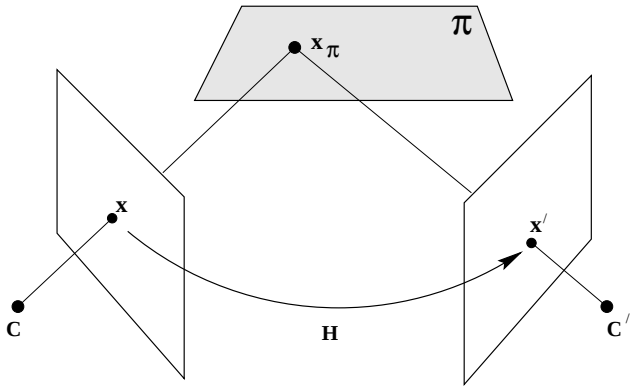


Figure 1.3: Images of a point $\mathbf{X}$ are related *directly* by the homography $\mathbf{H}$ in the case when $\mathbf{X}$ lies on the plane. Image courtesy [2]

which results in 4 image-to-image correspondences (each correspondence giving 2 equations in x and y coordinates). Although this minimal case is hugely sensitive to errors in correspondence, current feature extraction algorithms like SIFT [23] ensure that homography estimation is quite accurate when the camera poses aren't too far apart. Thus, a RANSAC based approach [2] suffices to weed out incorrect correspondences as they are usually only outliers of the actual function.

1.2.3 Homography and Camera Parameters

This section details the relationship between the homography $\mathbf{H}$ relating two images and the relative pose between their corresponding cameras. For the sake of simplicity, let us assume that the two cameras are given by $\mathbf{P}_1 = [\mathbf{I} \,|\, \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{R} \,|\, \mathbf{t}]$. Let $\mathbf{X}$ be the 3D point belonging to the plane $\mathbf{\Pi} = \begin{bmatrix}\mathbf{n}^\top & 1\end{bmatrix}$, and let $\mathbf{x}$ and $\mathbf{x}'$ be its projections respectively. Then

$$\mathbf{x} = \mathbf{P}_1\mathbf{X} = [\mathbf{I} \,|\, \mathbf{0}]\,\mathbf{X} \tag{1.1}$$

$$\mathbf{X} = \begin{bmatrix}\mathbf{x}^\top & \rho\end{bmatrix}^\top \tag{1.2}$$

Different values of ρ represent different points on the 3D line joining camera centre $\mathbf{C}$ and 3D point $\mathbf{X}$ (Figure 1.3). Thus the value of ρ that satisfies the above Equation (1.2) is $(-\mathbf{n}^\top\mathbf{x})$.

Substituting the value of ρ in the projection equation for the second image, we get

$$\mathbf{x}' = \mathbf{P}_2\mathbf{X} = [\mathbf{R} \,|\, \mathbf{t}]\,\mathbf{X} \tag{1.3}$$

$$= \mathbf{R}\mathbf{x} - \mathbf{t}\mathbf{n}^\top\mathbf{x} = (\mathbf{R} - \mathbf{t}\mathbf{n}^\top)\mathbf{x} \tag{1.4}$$

When the internal parameters cannot be assumed to be identity but are known to be different for the two images, the modified equation of the relationship is as follows

$$\mathbf{H} = \mathbf{K}'(\mathbf{R} - \mathbf{t}\mathbf{n}^\top)\mathbf{K}^{-1} \tag{1.5}$$

where $\mathbf{K}$ and $\mathbf{K}'$ are the internal parameters of the two camera respectively.

1.2.4 Infinite Homography

One key observation that is related to Equation (1.5) the plane parameters $\mathbf{n}$ are completely attached to the translational component of the relative pose between the two cameras. Thus when the

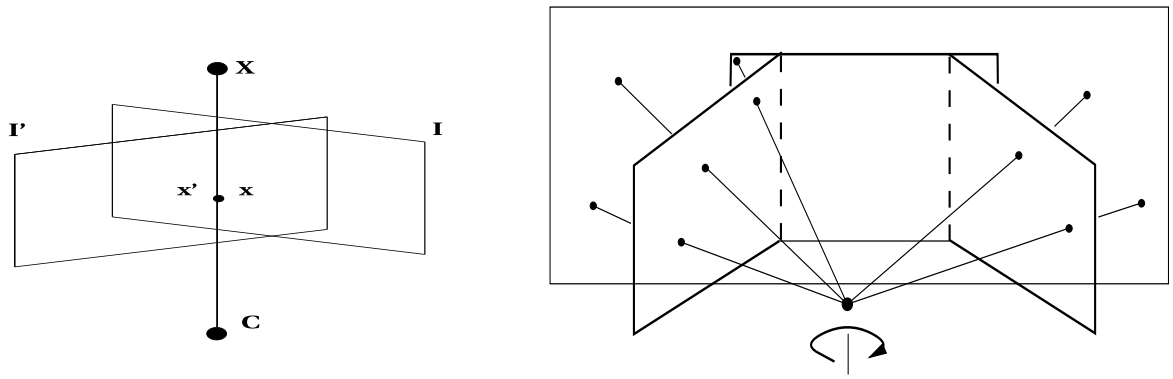


Figure 1.4: The left figure represents the fact that when the camera is panning, the same ray joining the 3D point **X** and the camera centre **C** is captured in both the images and hence all points are related by the same homography **H** irrespective of depth. The right figure indicates how a mosaic may be created using 3 images taken from a rotating camera, by back-projecting the two extreme images onto the reference plane of the middle one. The right image is courtesy [2]

camera undergoes no translation while taking images, Equation (1.5) holds for points on *any plane irrespective of its position and orientation*. In the case of such a panning camera, the homography so produced is called the **Infinite Homography**, and is given as

$$\mathbf{H} = \mathbf{K}'\mathbf{R}\mathbf{K}^{-1} \tag{1.6}$$

Following its definition, any 4 general image-to-image correspondences may be used to compute the Infinite Homography.

1.3 Multiple View Geometry: Applications

Given the mature state of research in MVG, it is but natural to explore the various applications of it. The applications of MVG, addressed in this thesis, may be classified in two ways. The first is the classification into the *types* of algorithms being proposed. From this perspective, algorithms proposed in this thesis fall into three categories. The first is the act of *Inference* whereby MVG is used to evaluate the state of a scene, *for eg.* the position of an object like a book in the various frames of a video. This task is called *Tracking* in the Vision literature, and research in it forms a considerable amount of both Vision and Robotics communities.

The second is the act of *Manipulation*. Manipulation is a term that can be easily associated with Robotics, although as we shall see later, it can also be used in the context of videos/images. Indeed, many tasks like grasping are its sub-parts. From the robotics perspective, a camera is a sensor and information obtained from it has to be channeled through computer vision for use in manipulation. Note that in our case, we consider *Navigation* to be a kind of manipulation; it is considered to be the case where the robot manipulates itself. Applications of manipulation related activities range from underwater and ariel unmanned vehicles, to robots that aid the elderly and perform surgery.

The last type is that of *Prediction*, where the system tries to predict a particular scenario from the data available. Note that this is different from the task of inference, in the sense that the predicted scenario need not actually be corroborated by real evidence captured later. Tasks like removing a scene object from a video in a consistent manner would fall in this category (*Predicting*

how the video would look like *without* the object). An exaggeration of this direction is *Extrapolation*, which is a sort of prediction of the *future* of a scene using available content about it. Here the *whole* content of an images is predicted using data from various other views.

The second perspective is to view the algorithms proposed in this thesis in the light of *applications* to *Robotic* and *Video/Image Manipulation,* which as the name suggests, represents a set of algorithms for manipulating *images/videos and robotic tasks.*

1.3.1 MVG for Robotic Vision

The field of Robotic Vision is a recent phenomena when compared to the other sensors like sonar and radar that have been used for more than 3 decades now. The main advantage of using cameras as sensors as opposed to others is that (i) cameras are *passive* sensors. Unlike radar and sonar, that have to generate a lot of information *first* in order to successfully receive information about the environment, vision systems only *receive* information; they do not transmit any. This ensures increased levels of portability, durability etc. as compared to other sensors. (ii) cameras provide a much cheaper mechanism of obtaining *accurate* 3D information about the world as compared to other sensors. The second reason has gained weight only recently with many accurate 3D reconstruction algorithms being proposed in the vision literature. Within Robotic Vision, the primary usage of Multiple View Geometry centres around

- Navigation - The basic goal of navigation systems is to analyze images of a scene taken by a camera to execute two important tasks. The first is the task of *localization.* Localization refers to the task of identifying *where* a robot lies (in terms of geometry) with respect to a *pre-defined global coordinate system.* The second task is that of *planning*, where the robot tries to identify a *most optimal* path to follow in order to reach a particular position in the world coordinate system. Robotic systems proposed in the past have employed both traditional cameras for the purpose of navigation, as well as *omni-directional cameras*, which are cameras with high field of view. Within regular cameras, usage of a single camera leads to *monocular* vision, while the usage of two cameras leads to the usage of *stereoscopic* vision. The second task can alternatively be *map building* which looks at the task of building a 3D map of the environment, which can be used later for navigation. There has also been research on coupling the task of localization and map building together, in the name of *simultaneous localization and mapping* (SLAM). The part of SLAM research which uses vision sensors is termed vSLAM, and has recently received much attention. Another perspective to view vision based navigation is its use for indoor and outdoor tasks. Tasks like landmark detection and following have been done quite successfully in the indoor environment, and has potential for producing tour-guiding robots like Minerva. In outdoor navigation, the NAVLAB project forms the mainstay, where road detection is used to guide a vehicle to follow traffic and navigate between two sites. Recently, the CyCab project to study car-following for autonomous navigation has also gained popularity.

- Positioning - The task of positioning is derived from the task of navigation in a limited scenario. Instead of considering an object that can navigate through an environment, positioning concerns itself with the task of manipulating the position of a *robotic arm* within a limited *workspace.* Potential applications are in industrial settings where a high degree of accuracy is required: for tasks like, say, automobile manufacturing. The field of Visual Servoing was first introduced for this purpose in the 1990s, and continues to attract a lot of research to this date. Most visual servoing tasks are set in the optimization framework, where the desired goal is achieved when a particular *objective function* reachs its global minima. The major problems

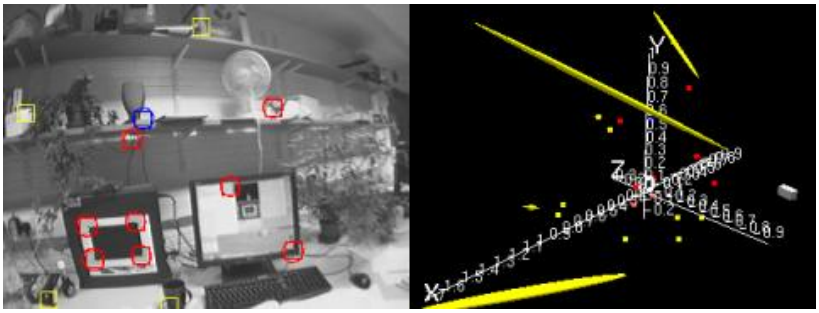


Figure 1.5: SLAM using a monocular vision sensor. [3]

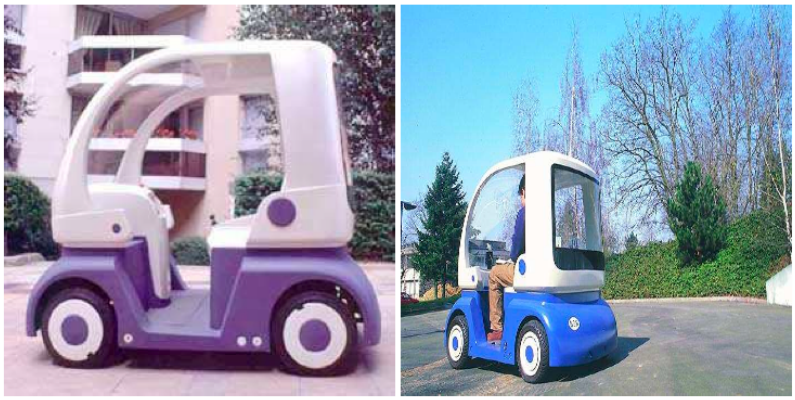


Figure 1.6: Projects like CyCab look at autonomous car following using vision sensors and Multiple View Geometry algorithms. [4]

in this area include algorithms for servoing systems (i) that can deal with different types of features, (ii) that can ensure accurate positioning (global minima) from many different initial positions, etc.

1.3.2 MVG for Image/Video Manipulation

The task of image and video manipulation has multiple usages primarily when it comes to enhancing the visual experience of the user 1.1. In this regard many fields like Image Mosaicing, Image Based Rendering, Augmented Reality etc. have attracted a lot of attention in the past 2 decades. Much of the work in these fields are based on advances of Multiple View Geometry, with many new fields like Single View Reconstruction [20] heavily borrowing from both the Multiple View Geometry and the Machine Learning literature.

- Mosaicing - The task of mosaicing can be defined as the *combination* of multiple photographs based on their *topological* relationship into one big *super-photograph* that encompasses the entire field of view captured by all the individual images combined. In this case, topology is understood as the spatial position of photographs on, say, a drawing board, where the positions actually depend on what part of the scene is being viewed by a photograph. Mosaicing solutions have matured to the extent that commercial products are now available even on standard digital cameras, where users can *pan* across a scene and recover the mosaic with a single click of a button. It is less known that the concept behind the creation of such mosaics

Figure 1.7: Algorithms like *Autostitch* provide Mosaicing solutions. This *mosaic* has been created using a collection of 57 images. [5]

(a) One of 27 input images

(b) Novel views

Figure 1.8: Image Based Rendering involves predicing how an image will look from a novel view point. [6]

is infact the *infinite homography*, which is illustrated in Section 1.2.4.

- Image Based Rendering (IBR) - As the name suggests, the field is concerned with producing images of a scene from view points, from which *no photograph* has been captured. Thus, the idea is to *predict* how a scene will look *from a novel viewpoint*, given existing photographs of it. Another way to look at the same task is to ask the question "What color does each pixel of a novel photograph take, given a set of photographs of the scene ?". The underlying concept is that if a scene is captured from multiple view points, all the information of the scene is captured in these images; so in order to describe how a new image would look, we simply have to define the appropriate function that transfers data from the given images to the novel imaeg. Approaches to image based rendering have used Multiple View Geometry in both the monocular camera case, and the stereopsis (two cameras attached together, viewing a scene from various angles) case. When monocular images are concerned, quantities like the *Fundamental Matrix* are used to transfer information across images.

- The task of video manipulation, has received interest in the commercial stream in the form of visual effects in movies. Movies like *The Matrix* involved scenes where multiple cameras were placed in circular arcs to show *still images* of a freezed scene from various view points. Since, cameras cannot be placed at the close proximity that was required for the shot to

Figure 1.9: Results for full frame video stabilization. [7]

look smooth, a *scene geometry* based interpolation technique was reqiured in order to give a smooth visual experience. This category of effects technically overlap with those of Image Based Rendering, though many other factors like the illumination and lighting of a scene have to be considered in this case. Another set important video manipulation algorithms come in the *video stabilization* category. Videos of scenes taken with an ariel camera, for example, suffer from "jitters" caused due to various factors, like turbulence, for example. In both robotic applications for unmanned ariel vehicles and in movies for the shooting of ariel sequences, such stabilization algorithms are dearly necessary. One class of stabilization algorithms, perform *camera tracking*; it is the task of estimating the pose of the camera throughout the video, which is then given as feed back into the stabilization system that produces a new *stabilized video*, using Multiple View Geometry based algorithms. The commercial tracker *Boujou* [24], falls under this category of solutions.

1.4 Contributions of this Thesis

The following contributions have been made during the course of this thesis.

- A Tracking framework which deals with the task of inferring the position of an object made up of *scene planes* both in 2D and 3D. In 2D, this results in estimation of the *homography* and in 3D this results in the estimation of the *pose* of the camera involved. Homography based tracking is posed in a Particle Filter framework to handle non-linearities in the system dynamics and non-Gaussian process and observation noises. This is extended to a homography based framework for tracking multiple planes in a piecewise planar static environment observed with a moving camera. The proposed system exploits the multi-view relationships between a set of planes to efficiently and robustly predict the position of each plane in a time-sequenced set of images, given appropriate initialization. The Unscented Kalman Filter is used for the purpose of state estimation. Two effective solutions for the initialization are also proposed.

- Many robotic systems operate in the *inference* and *manipulation* loop [25]. Since the earlier contributions dealt with the *inference* part, we next investigate the use of MVG for *visual servoing*. We present a framework for servoing based on Fourier Transforms of the contours of planar objects, that alleviates the need for correspondence between images, which is a pre-requisite for most transformation estimation algorithm. Although earlier this framework was

only applicable for a limited set of transformations (affine), we present a servoing scheme that operates in the *projective* space, along with a set of applications where the framework proves advantageous.

- Our third contribution is to the field of *video manipulation*. We develop an algorithm for video *inpainting*, where specific objects from a video are removed and resulting space-time *holes* are filled in a consistent manner. The algorithm is fully automatic unlike traditional image and video inpainting algorithms, and takes as input two functions; one function defines the object to be removed, and the other defines the background model that is used for hole-filling. This concept is then extended to a more general inpainting based framework for Image Based Rendering (IBR). Image Based Rendering (IBR) concerns itself with algorithms for an image based representation of the 3D information of a scene. Novel views of the scene can then be rendered with this information. We extend IBR to include cases when 3D information about a particular scene is incomplete, by incorporating information about the *type* of scene being viewed (*for eg. the face of a person*). We then devise algorithms to transfer specific semantic characteristics to the current scene from similar scenes available to us.

Alternatively, with the exception of Chapter 7, the contributions of this thesis may also be viewed from the perspectives of a *Planar World*. In the first couple chapters, we present algorithms that infer the position and parameters of *scene planes*. The chapter on servoing concentrates on manipulating a robotic arm with respect to a planar object, which is being constantly viewed by a camera placed on the palm of the arm. The chapter on video completion deals with modifying elements of a video, that can be ascertained to *not* belong to a previously identified planar object. It is also shown that a panning camera shares some properties with a planar scene, which allows manipulation of dynamic objects in a video taken with the camera only rotating about an (multiple) axi(e)s.

1.5 Organization

The remainder of this thesis is organized as follows:

- In Chapter 2, we give an over view of the basic mathematical concepts needed for this thesis. First we give an overview of Particle Filters and Unscented Kalman Filter, that are used for Tracking as explained in Chapter 3 & 4. This is followed by a brief explanation of the task of visual servoing, which is the main subject of Chapter 5. Next we give an introduction to Successive Over Relaxation (SOR), a class of iterative algorithms that are used to solve linear equations involving large number of variables (of the order of millions). This class of algorithms are used in the computation of Optical Flow and Poisson Blending. We use Optical Flow as a means of non-linear registration between two images in Chapter 7, while Poisson Blending is a technique used to merge two images so that the result is seamless.

- In Chapter 3, we first give an overview of an algorithm for tracking a *single* plane in multiple views using different cues like texture an edge, from [4, 26]. This framework is then analyzed to point out *why* a particle filter framework is necessary in the first place, followed by analysis of what type and amount of particles are necessary to accomplish the above task with minimal computational load. The analysis requires an understanding of the homography space.

- In Chapter 4, the single plane framework is then extended to use geometry based cues that enables the tracking of multiple planes. The extended framework is implemented using Unscented Kalman Filters for reasons explained later. Issues of initialization, robustness and implementation details are then discussed.

- In Chapter 5, we first start with an overview of the applications of Fourier transforms to multiple view geometry, which is then extended to a robotic manipulation task called *visual servoing.* Starting from simple extensions, we show how the Fourier framework may be extended to include complex tasks with the aid of examples.
- In Chapter 6, we formulate Video Completion in a Noise Removal framework. The object(s) to be removed is defined as an outlier of a basic objective function, and holes produced while removing these outliers from the video are replaced with the extrapolation of the desired function at the given time (images are time-sequenced). Results are shown on two basic scenarios. Extension of the framework to other objective functions is also discussed.
- In Chapter 7, we extend the idea of IBR to scenes with incomplete information. A database of similar scenes is used as a base to transfer semantic and missing information to the current scene to create novel views that are otherwise impossible using current IBR techniques. The presence of a database also allows us to introduce different subtle variations into the current scene.
- In Chapter 8, we conclude the thesis. We summarize the contributions of this thesis and comment on the limitations and future work.

Chapter 2

Preliminaries

2.1 Introduction

In this chapter, we focus on presenting the mathematical preliminaries that underlie the content developed in the rest of this thesis. First we review the essentials of two important Bayesian filters in Sections (2.2,2.3), which are some of the most widely used filters for Tracking (Chapter 3). These are used in Chapters 3 & 4 for the purpose of inferring the position of a desired object. This is followed by a description of *Visual Servoing,* which is an area in Robotics devoted to the positioning of a robotic arm with respect to an object of interest. Such positioning algorithms find use in applications like surgery or cinematography. The algorithms presented in Chapter 5 are built over the concepts of this section.

Next is a discussion of important tools and techniques that are used in Chapters 6 & 7. First we discuss the concept of Successive Over Relaxation (SOR), which is a pre-requisite for algorithms like Optical Flow and Poisson Blending. SOR is a collection of optimization algorithms that iteratively solve huge sets of linear equations, and are being increasingly used in Computer Vision for many problems like Optical Flow [13]. The basic algorithms of SOR are followed by the description of an implementation of [13], a popular algorithm for computing the Optical Flow (pixel-wise correspondence) between two images, that is used as an algorithm for non-linear registration in Chapter 7. Finally, an algorithm for seamlessly blending images, called Poisson Blending, is described in Section (2.2). This technique finds wide usage in image editing applications [27], and is useful when merging information from one image into another in a seamless manner.

2.2 Unscented Kalman Filter

One of the most popular estimation techniques used in the Computer Vision literature is the Kalman filter. Kalman filters are based on linear dynamical systems discretised in the time domain. They are modelled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the visible outputs from the hidden state. The Kalman filter model assumes the true state at time k is evolved from the state at $(k-1)$according to

$$\mathbf{x}_k = \mathbf{F}(\mathbf{x}_{k-1},\ \mathbf{u}_{k-1},\ \mathbf{v}_{k-1}) \tag{2.1}$$
$$\mathbf{y}_k = \mathbf{H}(\mathbf{x}_k,\ \mathbf{n}_k) \tag{2.2}$$
$$\mathbf{v}_k \sim N(\mathbf{0},\ \mathbf{Q}_k) \tag{2.3}$$
$$\mathbf{n}_k \sim N(\mathbf{0},\ \mathbf{R}_k) \tag{2.4}$$

where the functions $\mathbf{F}(\cdot,\cdot,\cdot), \mathbf{H}(\cdot,\cdot)$are linear in nature. Many real dynamical systems do not exactly fit this model, either because of non-linearity of the state transition or observation functions, or because of the non-Gaussian nature of the noise involved. Especially in Computer Vision, because of the perspective projection function, most estimation equations are highly non-linear in nature. On the other hand, a Gaussian assumption for the noise in data proves to be an adequate assumption in most cases. Given that the Kalman filter produces the most optimal solution in the linear case, it is desirable to extend the technique to deal with non-linear systems.

2.2.1 Kalman Filter for Nonlinear Systems

While using Kalman filter for state estimation, the main quantities estimated are the mean and covariance of the state vector at the current time instant, given current inputs, noise, transition and observation functions and the mean and covariance of the state vector in the previous instant. The major problem in extending the filter to non-linear systems is the difficulty in the propagation of the error covariance through the non-linear transition and observation functions. For the case when the non-linearity of the system is not significant, a Kalman filter coupled with a first order approximation of the system may provide a sub-optimal solution that is very close to the actual state space values. This is precisely the technique employed by the *Extended Kalman Filter* (EKF), under the only assumption that the non-linear functions involved be differentiable. Many earlier estimation algorithms in Computer Vision have employed the EKF to estimate quantities like the pose of a camera and the 3D position of feature points, given correspondences. When more precision is required or when the underlying state dynamics is highly non-linear, however, another variant of the Kalman filter may be used to obtain more accurate estimates. This algorithm, the *Unscented Kalman Filter* (UKF) address the non-linearity by using a deterministic sampling approach to estimating the mean and covariance after propagation through the non-linear system. This approach is termed the *Unscented Transform*.

2.2.2 The Unscented Transform

In all the above filters, the error covariance of the state vector is represented by a Gaussian Random Variable (GRV). In case of the UKF, the state distribution is again approximated by a GRV, but is now represented using a covariance matrix as well as a minimal set of points sampled from the matrix. These sample points capture the true mean and covariance of the GRV, and, when propagated through the true nonlinear system, captures the posterior mean and covariance accurately to second order for any non-linearity. The EKF, in contrast, only achieves first-order accuracy. The process of propagating covariances through a non-linear system is called the Unscented Transform. Surprisingly, the computational complexity of the UKF using the *Unscented Transform* (UT) is the same order as that of the EKF.

Consider propagating a random variable $\mathbf{x}$of dimension L through a non-linear function, $\mathbf{y} = f(\mathbf{x})$. Assume $\mathbf{x}$has mean $\bar{\mathbf{x}}$and covariance $\mathbf{P}_{\mathbf{x}}$. To calculate the statistics of $\mathbf{y}$, we form a matrix $\mathcal{X}$of $2L+1$sigma vectors $\mathcal{X}_i$according to the following:

$$\begin{aligned}
\mathcal{X}_0 &= \bar{\mathbf{x}} && (2.5)\\
\mathcal{X}_i &= \bar{\mathbf{x}} + (\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}})_i, \qquad i = 1, \ldots, L, && (2.6)\\
\mathcal{X}_i &= \bar{\mathbf{x}} + (\sqrt{(L+\lambda)\mathbf{P}_{\mathbf{x}}})_{i-L}, \qquad i = 1, \ldots, 2L, && (2.7)
\end{aligned}$$

where $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. The constant αdetermines the spread of the sigma points around $\bar{\mathbf{x}}$, and is usually set to a small positive value (e.g., $1 \leq \alpha \leq 10^{-4}$). The constant $\kappa$is a secondary scaling parameter which is usually set to $3 - L$(see reference [28]for details), and β is used to incorporate prior knowledge of the distribution of $\mathbf{x}$ (for Gaussian distributions, $\beta = 2$ is optimal).$(\sqrt{(L+\mathbf{P}_{\mathbf{x}})})_i$ is the ith column of the matrix square root (e.g., the lower triangular Cholesky factorization). These sigma vectors $\mathcal{X}_i$ are propagated through the nonlinear function

$$\mathcal{Y}_i = f(\mathcal{X}_i), \qquad i = 0, \ldots, 2L,$$

and the mean and covariance for $\mathbf{y}$are approximated using a weighted sample mean and covariance of the posterior sigma points $\mathcal{Y}_i$,

$$\begin{aligned}
\overline{\mathbf{y}} &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_i, && (2.8)\\
\mathbf{P}_{\mathbf{y}} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_i - \overline{\mathbf{y}})(\mathcal{Y}_i - \overline{\mathbf{y}})^\top, && (2.9)
\end{aligned}$$

with weights W_i given by

$$\begin{aligned}
W_0^{(m)} &= \frac{\lambda}{L+\lambda},\\
W_0^{(c)} &= \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta, && (2.10)\\
W_i^{(m)} &= W_i^{(c)} = \frac{1}{2(L \mid \lambda)}, \qquad i = 1, \ldots, 2L.
\end{aligned}$$

This method differs from the general Monte Carlo sampling methods, which require orders of magnitude more sample points since they make no assumptions about the underlying distribution (which is assumed to be Gaussian in our case). The *Unscented Transform* (UT) produces mean and covariances that are accurate to the third order for Gaussian inputs for all non-linearities.

2.2.3 UKF Parameter Estimation

The *Unscented Kalman Filter* (UKF) is a straightforward extension of the UT to the recursive estimation equations of the traditional Kalman Filter, where the state Random Variable (RV) is redefined as the concatenation of the original state and noise variables $\mathbf{x}_k^a = \left[\mathbf{x}_k^\top \ \mathbf{v}_k^\top \ \mathbf{n}_k^\top\right]^\top$. When the process and measurement noises are purely additive however (zero mean), the system state need not be augmented with the noises and the computational complexity of the UKF is reduced to that of the EKF since $\mathbf{x}_k^a = \mathbf{x}_k$. The UKF estimation algorithm is summarized below (Algorithm 8).

The Kalman Filter is one of the most widely used filters in the Robotics community, with major usage in fields like Simultaneous Localization and Mapping (SLAM) [3]. However, the non-Gaussian assumption of Kalman Filters is one of the major reasons why its use in computer vision is restricted (Section 3.5). For problems in computer vision like human body tracking, filters like Particle Filters are used to model the state dynamics of the system. We now proceed to give a description of Particle Filters.

2.3 Particle Filters

Particle filters, also known as sequential Monte Carlo methods (SMC), are sophisticated model estimation techniques based on recursive Bayesian filtering. Particle filters are more powerful than the Kalman filter in that they escape both from the assumption of a linear model for state dynamics and a Gaussian model for the state vector noise distribution. They are often an alternative to the Extended Kalman filter (EKF) or Unscented Kalman filter (UKF) with the advantage that, with sufficient samples, they approach the Bayesian optimal estimate, so they can be made more accurate than either the EKF or UKF.

2.3.1 State Estimation for Non-Gaussian Random Variables

Particle methods, like all sampling-based approaches (e.g., MCMC), generate a set of samples that approximate the filtering distribution $p(\mathbf{x}_k|\mathbf{y}_{0,\ldots,}\mathbf{y}_k)$ using a set of samples $\hat{\mathcal{X}}_k^{1\cdots n}$, where the probability distribution represents the posterior distribution of a state vector computed using Bayesian filtering estimates. Along with the set of samples, a set of positive weights $\pi_k^{1\cdots n}$ are also maintained that approximately represent the probability of each of the sample representing the true value of the system state at the kth instant. This particle system then evolves along the time according to the state equation, with a number of particles representing the evolving pdf as the end result.

2.3.2 Basic Filter Equations

To define the problem of Bayesian Tracking or Filtering, consider the evolution of the state sequence $\{\mathbf{x}_k, k \in \mathbb{N}\}$ of a target, given by

$$\mathbf{x}_k = \mathbf{f}_k(\mathrm{x}_{k-1}, \mathbf{v}_{k-1}) \tag{2.11}$$

where $\mathbf{f}_k : \Re^{n_x} \times \Re^{n_v} \longrightarrow \Re^{n_x}$ is a possibly non-linear function of the state $\mathbf{x}_{k-1}$, $\{\mathbf{v}_{k-1}, k \in \mathbb{N}\}$ is an i.i.d process (independent identically distributed) process noise sequence, n_x, n_v are dimensions of the state and process vectors, respectively and $\mathbb{N}$ is the set of natural numbers. The objective of tracking is to recursively estimate $\mathbf{x}_k$ from measurements

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{2.12}$$

where $\mathbf{h}_k : \Re^{n_x} \times \Re^{n_n} \longrightarrow \Re^{n_z}$ is a possibly non-linear function, $\{\mathbf{n}_k, k \in \mathbb{N}\}$ is an i.i.d measurement noise sequence and n_z, n_n are dimensions of the measurement and measurement noise vectors, respectively. In particular we seek filtered estimates of $\mathbf{x}_k$ based on the set of all available measurements $\mathbf{z}_{1:k} = \{\mathbf{z}_i, i = 1, \ldots, k\}$ up to time k.

From a Bayesian perspective, the tracking problem is to recursively calculate some degree of belief in the state $\mathbf{x}_k$ at time k, taking different values, given the data $\mathbf{z}_{1:k}$up to time k. Thus, it is

required to construct the pdf $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. It is assumed that the initial pdf, $p(\mathbf{x}_0, \mathbf{z}_0) \equiv p(\mathbf{x}_0)$, of the state vector, also known as the prior, is available ($\mathbf{z}_0$ being the set of no measurements). Then, in principle, the pdf $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ may be obtained recursively in two stages: prediction and update.

Suppose that the required pdf $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ at the time $k-1$ is available. The prediction stage involves using the system model (2.11) to obtain the prior pdf of the state at time k via the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{2.13}$$

Note that in the above equation, use has been made of the fact that $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$ as Equation (2.11) describes a Markov process of order one. The probabilistic model of the state evolution, $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, is defined by the system Equation (2.11) and the known statistics of $\mathbf{v}_{k-1}$.

At time step k, a measurement $\mathbf{z}_k$ becomes available, and this may be used to update the prior (update stage) via Bayes' rule:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})}, \tag{2.14}$$

where the normalization constant

$$p(\mathbf{z}_k|\mathbf{z}_{1:k}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})d\mathbf{x}_k \tag{2.15}$$

depends on the likelihood function $p(\mathbf{z}_k|\mathbf{x}_k)$, defined by the measurement model Equation (2.12) and the known statistics of $\mathbf{n}_k$. In the update stage (Equation (2.14), the measurement $\mathbf{z}_k$ is used to modify the prior density to obtain the required posterior density of the current state.

The recurrence relation equations (2.13, 2.14) form the basis of the optimal Bayesian solution. This recursive propagation of the posterior density is only a conceptual solution in that in general, it cannot be determined analytically.

2.3.3 Sampling Issues

The Sequence Importance Sampling (SIS) algorithm is a Monte Carlo (MC) method that forms the basis for most sequential Monte Carlo filters developed over the past decades [29] It is a technique for implementing a recursive Bayesian filter by Monte Carlo simulations. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this Monte Carlo characterization becomes an equivalent representation to the usual functional description of the posterior pdf, and the SIS filter approaches the optimal Bayes estimate (represented by Equation 2.14).

In order to describe the details of the algorithm, let $\{\mathbf{x}^i_{0:k}, w^i_k\}_{i=1}^{N_s}$ denote a *Random Measure* that characterizes the posterior pdf $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$, where $\{\mathbf{x}^i_{0:k}, i = 0, \ldots, N_s\}$ is a set of support points with associate weights $\{w^i_k, i = 1, \ldots, N_s\}$ and $\mathbf{x}_{0:k} = \{\mathbf{x}_j, j = 0, \ldots, k\}$ is the set of all states up to time k. The weights are normalized such that $\Sigma_i w^i_k = 1$. Then, the posterior density at k can be approximated as

$$p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i) \tag{2.16}$$

We therefore have a discrete weighted approximation to the true posterior, $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$. The weights are chosen using the principle of *Importance Sampling* [29].

Sequential Importance Sampling

This principle relies on the following: Suppose $p(x) \propto \pi(x)$ is a probability density from which it is difficult to draw samples, but for which $\pi(x)$ can be evaluated (and so $p(x)$ up to scale). Also, let $x^i \sim q(x)$, $i = 1, \ldots, N_s$ be samples that are easily generated from a proposal $q(\cdot)$, called the *Importance density*. Then, a weighted approximation to the density $p(\cdot)$ is given by

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i) \tag{2.17}$$

where

$$w_k^i \propto \frac{\pi(x^i)}{q(x^i)} \tag{2.18}$$

is the normalized weight of the i^{th} particle.

So, if the samples $\mathbf{x}_{0:k}^i$, were drawn from an importance density, $q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ then the weights in Equation (2.16) are defined by Equation (2.18) to be

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}^i|\mathbf{z}_{1:k})} \tag{2.19}$$

At each iteration, one could have samples constituting an approximation to $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, want to approximate $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ with a new set of samples. If the importance density is chosen to factorize such that

$$q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \tag{2.20}$$

then one can obtain samples $\mathbf{x}_{0:k}^i \sim q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ by augmenting each of the existing samples $\mathbf{x}_{0:k-1}^i \sim q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$ with the new state $\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})$. To derive the weight update equation, $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$ is first expressed in terms of $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, $p(\mathbf{z}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Note that Equation (2.14) can be derived by integrating Equation (2.21).

$$\begin{aligned}
p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) &= \frac{p(\mathbf{z}_k|\mathbf{x}_{0:k}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \\
&= \frac{p(\mathbf{z}_k|\mathbf{x}_{0:k}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \qquad (2.21) \\
&= \frac{p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \\
&\propto p(\mathbf{z}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \qquad (2.22)
\end{aligned}$$

By substituting Equations (2.20, 2.22) into Equation (2.19), the weight update equation can then be shown to be

$$\begin{aligned} w_k^i &\propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)p(\mathbf{x}_{0:k-1}^i|\mathbf{z}_{1:k-1})}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i,\mathbf{z}_{1:k})q(\mathbf{x}_{0:k-1}^i|\mathbf{z}_{1:k-1})} \\ &= w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i,\mathbf{z}_{1:k})} \end{aligned} \tag{2.23}$$

Furthermore, if $q(\mathbf{x}_k|\mathbf{x}_{0:k-1},\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{z}_k)$, then the importance density becomes only dependent on $\mathbf{x}_{k-1}$ and $\mathbf{z}_k$. This is particularly useful in the common case when only a filtered estimate of $p(\mathbf{x}_k|\mathbf{z}_{1:l})$ is required at each time step, as in our algorithm of Chapter 3. In such scenarios, only $\mathbf{x}_k^i$ need be stored, and so one can discard the path, $\mathbf{x}_{0:k-1}^i$, and history of observations, $\mathbf{z}_{1:k-1}$. The modified weight is then

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i,\mathbf{z}_k)} \tag{2.24}$$

and the posterior filtered density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be approximated as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{2.25}$$

where the weights are defined in Equation (2.24). It can be shown that as $N_s \longrightarrow \infty$ the approximation represented by Equation (2.25) approaches the true posterior density $p(\mathbf{x}_k|\mathbf{z}_{1:k})$.

The SIS algorithm thus consists of recursive propagation of the weights and support points as each measurement is received sequentially. A pseudo-code description of the algorithm is given in Algorithm 9.

One of the main reasons for the popularity of Particle Filters is the ability to represent *any* noise distribution in the process and measurement functions. This makes room for various robustness measures to be incorporated into the measurement process, which are otherwise difficult to include. For example, as we will see in Section 3.4.3, several robustness measures like point-wise maximum functions and "good-ness" weights are introduced to remove noise and spurious features from the measurement process, for the purpose of robust tracking. In robotics, the problem of tracking is a useful preliminary for other tasks like positioning, specifically called *Visual Servoing*. In the next section, we outline the major traits involved in positioning a robotic arm with respect to a particular object of interest, which forms the mainstay of visual servoing algorithms.

2.4 Visual Servoing

In the robotics literature, many sensors like radar, sonar, laser etc. are used to infer knowledge about the environment which is then used for tasks like navigation and manipulation. Recent applications of robotic systems in industrial and medical settings has resulted in the demand for a fine control over the accuracy of the dynamics of the concerned robot. Computer vision may be used to fill this demand of accuracy. Research in the last few decades has established that Vision is a very versatile sensor for robotic applications, since algorithms for processing images has reached a mature stage

where robust, accurate solutions are available. In addition, vision sensors are cheaper than most other robotic sensors like laser, with competitive accuracies.

Visual servoing pertains to a class of algorithms that use visual feedback to guide the *arm of a robot* accurately in an environment (called the *workspace* of the robot). Typical uses of visual servoing are in positioning the robotic arm with respect to an object for tasks like manipulation of the object, active exploration of the object etc. Thus visual servoing can be defined as a *visual feedback based class of robotic arm positioning algorithms*. This positioning may be with respect to an biological object like an organ for surgery, or with respect to an industrial object like a conveyor belt. Recently applications have also been proposed for automatic navigation through car following.

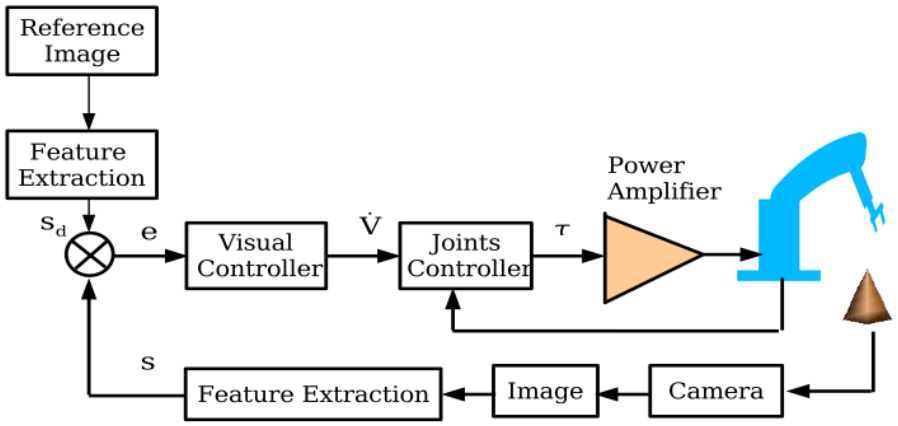


Figure 2.1: Figure representing the block diagram of the visual servoing control loop. Features are extracted from the current and reference images, and compared to generate the task function error that needs to be regulated to zero.

2.4.1 Robot Motion as a Control Law

The main objective of the visual servoing algorithm is to *position* the robotic arm with respect to an object in a *supervised* framework. This means that the user gives as input to the system, an image (and features extracted from the image) of *how* the scene (or equivalently object) would look like from the *desired* position (also called reference position). Visual servoing algorithms operate by posing the positioning problem as regulating an error function of the state dynamics (camera pose) to zero. This is a closed loop system where feedback is received from the camera placed on the robot. An illustration is given in Figure 2.1 The design of a visual servo control thus primarily involves (i) the extraction of appropriate features, and (ii) the design of a control law that minimizes an appropriate error function. At any instant of time, features are captured by the system and the direction of motion is estimated by computing the gradient of the pose-error function as described in the next few sections.

2.4.2 Image Based Visual Servoing (IBVS)

Visual servoing systems can be classified into different categories [4] based on sensor configurations, number of sensors, control architecture, initial knowledge about target object and camera, etc. In our work, however, we only consider a specific category called Image Based Visual Servoing

(IBVS) which is also one of the most widely studied sub-areas. The IBVS framework is set over the general task function approach with the addition that the features being considered are image points (corresponding points in the current and reference images). The associated error function to be minimized is thus a function of the pose of the camera and time and is defined as

$$e(r,t) = \begin{bmatrix} x_1^C - x_1^R \\ y_1^C - y_1^R \\ \vdots \\ x_n^C - x_n^R \\ y_n^C - y_n^R \end{bmatrix} \tag{2.26}$$

where the subscript denotes the image point index and the superscript (C, R) denotes the current and reference views. The error minimization proceeds by following a simple (proportional) control law that allows the system to behave like a first order decoupled system while regulating the task error to zero. The behaviour of a first order system is described as

$$\dot{e}(r,t) = -\lambda e(r,t) \tag{2.27}$$

where λ is the rate at which we would like the error of the system to decrease. The first derivative of the task function is given as

$$\dot{e}(r,t) = \frac{de}{dr} V_E + \frac{de}{dt} \tag{2.28}$$

$$\dot{q} = \left(\frac{dr}{dq}\right)^{-1} V_E \tag{2.29}$$

where $V_E = \frac{dr}{dt}$ is the velocity of the robot arm, and $\dot{q}$ is the joint velocities. The term $\frac{de}{dt}$ represents the dynamism in *the elements of the scene* and is useful while servoing with respect to dynamic objects. For our case, this term has value zero. Thus, at any given position and time (r,t) the velocity of the robotic arm can be computed by equating the error dynamics Equation (2.28) and the first order behaviour Equation (2.27) to get

$$\&V_E = \left(\frac{\hat{de}}{dr}\right)^{-1} (-\lambda e(r,t)) \tag{2.30}$$

$$\dot{e}(r,t) = -\lambda \frac{de}{dr} \left(\frac{\hat{de}}{dr}\right)^{-1} e(r,t) \tag{2.31}$$

which gives an estimate of the velocity in terms of the current image error.

Uptil now, we have looked at the essentials that are required for application of MVG in robotics. The tracking Chapters 3 & 4 use the Kalman and Particl Filters for modeling, while Chapter 5 presents a few novel visual servoing algorithms. The rest of this chapter is focused on tools and techniques used in the computer vision community for the purposes of image and video editing, and in general for inferencing and modifying video data. Over the years, the field has matured from using ad-hoc algorithms and even filters for estimation quantities like the "shift" in images

capturing a scene from two nearby viewpoints, to mathematically rigorous optimization frameworks for estimation. Since many of these tasks have quality as the paramount concern, little or no interest is shown in the time taken for the actual algorithm to process data. This is in stark contrast to robotics, where real-time performance is essential. We describe two such algorithms used in this thesis. The first is an optical flow computation algorithm [13]. Optical flow refers to the task of estimating the dense correspondence (correspondence for every pixel) between two images. The current approach sets optical flow estimation in an optimization framework where minimizing the intensity difference between the first image and the transformed second image is the objective function. It is minimized using a standard technique, to obtain the transformation. The second algorithm is that of Poisson blending [27], which deals with the task of *blending* one image into the other. Blending refers to the task of modifying the intensity of one image so that it can be placed at a specified position in the second image, without *seams* occuring in the second image at the border of the region where the first image is pasted. Seams usually result because of intensity difference between images. In the following sections, we first start with the description of Successive Over Relaxation (SOR), an algorithm for solving linear equations in a *large* number of variables, typically of sizes around 1 million, where traditional matrix inverse based solutions cannot work because of memory constraints. SOR is an integral part of both the optical flow algorithm and Poisson blending algorithm presented in Sections 2.6 & 2.2.

2.5 Successive Over Relaxation

Many problems in Computer Vision require large systems of linear equations to be solved. For example, in Optical Flow, each pixel contributes two variables to the overall system. Thus, an image of size 512×512 gives rise to $\sim 5 \times 10^5$ variables. Thus costly matrix computations like inverse becoming infeasible, requiring alternate solutions.

Successive Over Relaxation (SOR) is a technique to compute an iterative solution to the linear equation

$$\mathbf{A}\,\mathbf{x} = \mathbf{b} \tag{2.32}$$

where the variable $\mathbf{x}$ is first initialized to a value $\mathbf{x}^{(0)}$. SOR pertains to the category of *Stationary methods* that can be expressed in the simple form $\mathbf{x}^{(k)} = B\ \mathbf{x}^{(k-1)} + c$, where (B, c) are variables that are independent of the iteration, denoted by the supersrcipt k. Other computation methods pertaining to this category include *Jacobi, Gauss-Seidel etc.* [1].

The most basic of stationary methods is the *Jacobi iteration*. At each iteration, each variable of $\mathbf{x}$ is solved for, by freezing the value of all other variables to their previous iterations value. Thus

$$\mathbf{x}_i^{(k)} = (\mathbf{b}_i - \sum_{j \neq i} \mathbf{A}(i,j)\ \mathbf{x}_j^{(k-1)})/\mathbf{A}(i,i) \tag{2.33}$$

The *Gauss-Seidel* method improves upon the *Jacobi* method by using current iteration values of $\mathbf{x}_j$ whenever available. Thus, the Equation (2.33) is split into two parts with superscripts (k) and $(k-1)$. Finally, SOR improves over the Gauss-Seidel method by introducing an *extrapolation factor* ω, to take a weighted average between solutions of the $(k-1)$ and the $(k)^{th}$ iteration.

$$\mathbf{x}^{(k)} = \omega\ \bar{\mathbf{x}}^{(k)} + (1-\omega)\ \mathbf{x}^{(k-1)} \tag{2.34}$$

The SOR algorithm is illustrated in Algorithm 10.

Being an optimization technique that minimizes over large amount of variables, it finds a lot of application within the many problems of computer vision like optical flow, where the idea is to

estimate the motion between two images, for *every pixel* of the image. Thus in a typical image of size 1000x1000, we have 1 million variables to whom flow values need to be assigned. In the next section, we describe one such algorithm [13].

2.6 Optical Flow

The **Optical Flow** between a pair of images represents a vector field of motion that takes the pixels of one image onto the pixels of another. In short, it establishes dense correspondences between the given images, unlike feature extraction and matching algorithms like SIFT [23] that compute a sparse correspondence map. Unlike feature extraction algorithms, which work only on images with sufficiently high texture content, optical flow algorithm are required to give dense maps even for plainly textured surfaces. This has forced recent papers [13] to pose the computation of optical flow as an optimization problem. In the subsequent sections, we look at how the optical flow between two images is cast as the vector field minimizing the difference between the input images in a suitable feature space. The SOR technique (Algorithm 10) is used for obtaining the flow minimizing the objective function.

2.6.1 Optical Flow as a Minimization Problem

Under the assumption that the illumination of the scene does not change between the capturing of the two images, the optical flow is estimated by minimizing the functional [30]

$$E(u,v) = \int_{\Omega} \left((|I_t(\mathbf{x}+\mathbf{w}) - I_{t-1}(\mathbf{x})|)^2 + \alpha(|\bigtriangledown u|^2 + |\bigtriangledown v|^2) \right) \mathbf{dx} \tag{2.35}$$

where $\mathbf{w} = [u\, v\, 1]\,, \mathbf{x} = [x\, y\, 1]$, Ω denotes the entire image region and t denotes time. The left term in the above equation represents the intensity difference between the first image and the warped second image that needs to be minimized, and the term on the right hand side represents the "smoothness" of optical flow, ensuring that optical flow in texture less regions is an interpolation of optical flow from its surroundings. An optimum must satisfy the following two Euler-Lagrange equations

$$\begin{aligned} (I_t(\mathbf{x}+\mathbf{w}) - I_{t-1}(\mathbf{x}))I_x - \alpha\Delta u &= 0 & (2.36)\\ (I_t(\mathbf{x}+\mathbf{w}) - I_{t-1}(\mathbf{x}))I_y - \alpha\Delta v &= 0 & (2.37) \end{aligned}$$

where (I_x, I_y) denote x and y derivatives of I_t. At this point, [30] chooses to linearize I_t around $\mathbf{x}$ in order to obtain linear equations from (2.36, 2.37).

$$\begin{aligned} (I_z + I_x u + I_y v)I_x - \alpha\Delta u &= 0 & (2.38)\\ (I_z + I_x u + I_y v)I_y - \alpha\Delta v &= 0 & (2.39) \end{aligned}$$

where $I_z = I_t(\mathbf{x}) - I_{t-1}(\mathbf{x})$. The above system of Equations (2.38, 2.39) can now be solved using linear methods like SOR (Section 2.5).

2.6.2 Extension to Brox's Formulation

One of the major drawbacks of the methodology of [30] is the linearization, which produces only an approximate solution to the actual flow. In the case of large displacements, the image does not change linearly along the displacement vector. Another drawback is its sensitivity to changes

in illumination and noise. In order to overcome these difficulties, the authors of [13] proposed a robustified variational function, whose linearization was postponed until a later stage. The following functional is a modified version of the one proposed in [13].

$$E(u,v) = \int_\Omega \left(\Psi\left((I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))^2\right) + \alpha\Psi\left(|\bigtriangledown u|^2 + |\bigtriangledown v|^2\right)\right)\mathbf{dx} \tag{2.40}$$

The subscript t has been dropped for convenience (Refer Equations (2.36,2.37). The Euler-Lagrange equations corresponding to Equation (2.40) are

$$\begin{aligned}
\Psi'((I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))^2)\,(I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))I_x - \alpha\,\mathrm{div}(\Psi'(|\Delta u|^2 + |\Delta v|^2)\Delta u) &= 0 \quad (2.41)\\
\Psi'((I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))^2)\,(I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))I_y - \alpha\,\mathrm{div}(\Psi'(|\Delta u|^2 + |\Delta v|^2)\Delta v) &= 0 \quad (2.42)
\end{aligned}$$

In order to deal with the non linearity of the above equations, a multi-resolution strategy is adopted wherein a Gaussian pyramid of images is built and optical flow, initialized to $\mathbf{0}$ at the coarsest level is propagated downwards, with each level's result providing the initialization of the next finer level. Let k be the index for this first set of iterations. Then $\mathbf{w}^0 = [0\,0\,1]$ and $\mathbf{w}^{k+1} = \begin{bmatrix}u^{k+1}\ v^{k+1}\ 1\end{bmatrix}$ is the solution of the following equations

$$\begin{aligned}
\Psi'((I_z^{k+1})^2)\,(I_z^{k+1})I_x^k - \alpha\,\mathrm{div}(\Psi'(|\Delta u^{k+1}|^2 + |\Delta v^{k+1}|^2)\Delta u^{k+1}) &= 0 \quad (2.43)\\
\Psi'((I_z^{k+1})^2)\,(I_z^{k+1})I_y^k - \alpha\,\mathrm{div}(\Psi'(|\Delta u^{k+1}|^2 + |\Delta v^{k+1}|^2)\Delta v^{k+1}) &= 0 \quad (2.44)
\end{aligned}$$

where $I_z^{k+1} = (I^{k+1}(\mathbf{x}+\mathbf{w}) - I(\mathbf{x}))$. The solution $\mathbf{w}^{k+1}$ is then upsampled using bilinear interpolation and used as initialization for the optimization process at the next finer level of the Gaussian pyramid. The trouble however, is that the Equations (2.43, 2.44) are still nonlinear because of the addition of the robust kernel Ψ and the symbols I_*^{k+1}. Hence, this is the stage when linearization is introduced to simplify the above equations into linear ones.

$$I_z^{k+1} \approx I_z^k + I_x^k du^k + I_y^k dv^k \tag{2.45}$$

where $u^{k+1} = u^k + du^k$ and $v^{k+1} = v^k + du^k$. Since linearization is performed at each step after a fixed point has been computed, the nonlinear constraint is approximated by a sequence of linear approximation than a single approximation.

$$\begin{aligned}
(\Psi')_D^k &= \Psi'((I_z^k + I_x^k du^k + I_y^k dv^k)^2) && (2.46)\\
(\Psi')_S^k &= \Psi'(|\Delta(u^k + du^k)|^2 + |\Delta(v^k + dv^k)|^2) && (2.47)\\
(\Psi')_D^k\,(I_x^k(I_z^k + I_x^k du^k + I_y^k dv^k)) - \alpha\,\mathrm{div}((\Psi')_S^k\Delta(u^k + du^k)) &= 0 && (2.48)\\
(\Psi')_D^k\,(I_y^k(I_z^k + I_x^k du^k + I_y^k dv^k)) - \alpha\,\mathrm{div}((\Psi')_S^k\Delta(v^k + dv^k)) &= 0 && (2.49)
\end{aligned}$$

Again, another fixed point iteration needs to remove the nonlinearity of Ψ in order to compute the values of (du^k, dv^k). Denoting the fixed point index by l, and initializing $du^{k,l}$ and dvk, l to 0, we get the following linear equations.

$$\begin{aligned}
(\Psi')_D^{k,l}\,(I_x^k(I_z^k + I_x^k du^{k,l+1} + I_y^k dv^{k,l+1})) - \alpha\,\mathrm{div}((\Psi')_S^k\Delta(u^k + du^{k,l+1})) &= 0 \quad (2.50)\\
(\Psi')_D^{k,l}\,(I_y^k(I_z^k + I_x^k du^{k,l+1} + I_y^k dv^{k,l+1})) - \alpha\,\mathrm{div}((\Psi')_S^k\Delta(v^k + dv^{k,l+1})) &= 0 \quad (2.51)
\end{aligned}$$

Finally, discretization of terms like div results in a linear system of equations, which can be solved by SOR (Algorithm 10). Thus the algorithm has 3 levels of iteration, 2 in the equations shown above and 1 in the actual SOR algorithm.

2.6.3 Improvement over Brox's Algorithm

Since in this thesis, we use optical flow for estimating a non-linear registration function between two unrelated images, it makes sense to include more features in the actual objective function than just intensity difference. Specifically, we modify Equation (2.40) to obtain [31]

$$E(u,v) = \int_\Omega \left(\Psi \left(\sum_c (I^{[c]}(\mathbf{x}+\mathbf{w}) - I^{[c]}(\mathbf{x}))^2 \right) + \alpha \ \Psi \left(|\Delta u|^2 + |\Delta v|^2 \right) \right) \mathbf{dx} \tag{2.52}$$

where each of the c channels represent one feature. The features considered in this thesis include gray value, $x-$ and $y-$ gradients, rgb channels, and other oriented gradients whenever necessary. As a consequence, Equations (2.50, 2.51) get modified to

$$(\Psi')^{k,l}_D \ (\sum_c (I^k_x (I^k_z + I^k_x du^{k,l+1} + I^k_y dv^{k,l+1}))) - \alpha \ \mathrm{div}((\Psi')^k_S \Delta(u^k + du^{k,l+1})) \ = \ 0 \tag{2.53}$$

$$(\Psi')^{k,l}_D \ (\sum_c (I^k_y (I^k_z + I^k_x du^{k,l+1} + I^k_y dv^{k,l+1}))) - \alpha \ \mathrm{div}((\Psi')^k_S \Delta(v^k + dv^{k,l+1})) \ = \ 0 \tag{2.54}$$

where $(\Psi')^{k,l}_D$ is modified to

$$(\Psi')^k_D = \Psi'(\sum_c (I^k_z + I^k_x du^k + I^k_y dv^k)^2) \tag{2.55}$$

Optical flow algorithms typically find use in areas where it is required to find dense correspondences (or *register*) different images of a scene. Since there is no internal model of the scene, optical flow algorithms are capable of registering images even with non-linear deformations, like registration *across* photographs of faces, which we use in Chapter 7.

2.7 Poisson Blending

In imaging applications like mosaicing or inpainting, it is often desirable to copy-and-paste a region from one image to another. However, because of reasons like illumination variation, intensity variation occurs between images and as a result such a copy-pasting often leaves a noticeable seam at the boundary of the grafted region in the second image. Blending algorithms [27, 32] tend to merge two or more regions seamlessly by trying to get a measure of this intensity variation and nullifying it at the boundary of the two image regions. In this section we describe in detail an algorithm for blending images known as **Poisson Blending**.

2.7.1 Blending as Equation Solving

Given a known function $\mathbf{f}^*$ defined over a source region $\mathbf{S}$ which contains a closed region Ω that needs to be filled, we want to compute a new function $\mathbf{f}$ that is defined on $\mathbf{\Omega}$ and its boundary $\partial\mathbf{\Omega}$, such that $\mathbf{f}$ takes the value of $\mathbf{f}^*$ on boundary, while inside the region it satisfies some properties (Figure 2.2) like being close to a given vector field $\mathbf{v}$, possibly the gradient of the image to be blended in.

Thus, we want to find a function $\mathbf{f}$ such that $\| \bigtriangledown \mathbf{f} - \mathbf{v} \|^2$ is minimum over the whole of $\mathbf{\Omega}$ and $\mathbf{f} = \mathbf{f}^*$ at the boundary $\partial\mathbf{\Omega}$. Thus we define the objective function to be minimized as

$$\min_{\mathbf{f}} \int \int_{\mathbf{\Omega}} \| \bigtriangledown \mathbf{f} - \mathbf{v} \|^2 \qquad \text{with} \qquad \mathbf{f}|_{\partial\mathbf{\Omega}} = \mathbf{f}^*|_{\partial\mathbf{\Omega}} \tag{2.56}$$

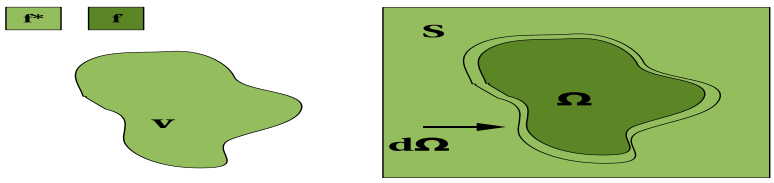


Figure 2.2: The source region $\mathbf{S}$ has a "hole" $\mathbf{\Omega}$ that needs to be filled in a manner both consistent with the boundary of $\mathbf{\Omega}$, $\mathbf{d\Omega}$ and the guiding gradient field $\mathbf{v}$. The function $\mathbf{f}^*$ is prevalent in the source region everywhere except in the interior of $\mathbf{\Omega}$, and we want to find an appropriate function $\mathbf{f}$ for the interior.

The optimum of the above equation must satisfy the associated Euler-Lagrange equation

$$\Delta \mathbf{f} = \text{div}\mathbf{v} \text{ over } \mathbf{\Omega} \qquad \text{with} \qquad \mathbf{f}|_{\partial\mathbf{\Omega}} = \mathbf{f}^*|_{\partial\mathbf{\Omega}} \tag{2.57}$$

which is also known as the Poisson equation with Dirichlet boundary conditions. The above equation is a linear equation since div and Δ are linear operators, and a discretization of the system provides the following equation

$$\begin{aligned} &\mathbf{f}(x+1,y) + \mathbf{f}(x-1,y) + \mathbf{f}(x,y+1), + \mathbf{f}(x,y-1) - 4\mathbf{f}(x,y) = \\ &\quad v_x(x+0.5,y) - v_x(x-0.5,y) + v_y(x,y+0.5) - v_y(x,y-0.5) \end{aligned} \tag{2.58}$$

where (x,y) is a pixel location and (v_x, v_y) are the $x-$ and $y-$components of $\mathbf{v}$, with the boundary condition imposed on pixels belonging to $\partial\mathbf{\Omega}$. The above system of equations is finally solved using Successive Over Relaxation (Algorithm 10).

Blending equations are very useful for producing "seamless results" when combining a set of images is concerned. This typically happens in image and video manipulation algorithms like those in Chapter 6 and 7.

2.8 Summary

In this chapter, we have given a detailed account of the mathematical preliminaries that the work in subsequent chapters build upon. We started with a description of Multiple View Geometry (MVG), that covered imaging fundamentals with a bias towards a particular entity called homography. Next two filters: The Unscented Kalman Filter and the Particle Filter were reviewed. The Successive Over Relaxation method for solving large linear systems iteratively was introduced next. A method of computing dense pixel correspondences between two images called Optical Flow was introduced in Section 2.6, finally followed by Poisson Blending, used to seamlessly blend one image into another.

Chapter 3

Multiple Feature Tracking

3.1 Introduction

Visual tracking is the task of estimating the position of a known object in various frames of a video, given an initialization. In the context of robotics, visual tracking algorithms help in estimating the position of objects in the immediate environment of the robot. Such algorithms are useful for positioning the robot relative to the object for purposes like navigation [21], since in order to navigate with respect to an object in the environment, the robot has to get constant feedback about the *position* of the object in the environment. The tracking problem in the robotic literature has been modeled as a motion estimation problem. Tracking may be done both in 2D, which involves predicting the pixel coordinates in every frame that contain the object of interest, or in 3D, which involve prediction of the 3D *pose* of the object. There are two major sources of visual features that are used in visual tracking: edges/contours and texture. Both visual features have advantages and disadvantages that make them suitable/unsuitable in many scenarios.

For scenes with sharp edges and high spatial gradients, contour features are very informative. Active contours can be used to track complex shapes in 2D tracking systems as in [33] and [34]. Straight lines are more suitable for 3D tracking applications [35]. When the scene is very cluttered or textured, contours may be absent, or may be difficult to extract. For example, although building are objects with distinct edges, they become difficult to extract and track in the presence of clutter, like trees. Illumination effects like shadows may also considerably affect the edge detector performance. In such cases, it is not advisable to identify features like contours that are spread over a large portion of the image, and hence are difficult to identify. Texture-based methods are required in such situations.

Interest (or texture or feature) points are useful for tracking of textured scenes. An interest point may be loosely defined as an entity comprising an image point and a small image "patch" in the points' "neighbourhood". The patch is selected so that it includes the image point. (Typical patches are of size 11×11 around the interest point). The Shi-Tomasi algorithm [8] for detecting "good" feature points is considered an effective algorithm for identifying texture points that are track-able. Unfortunately, tracking of interest points is sensitive to the quality of the image. In the presence of poor and noisy images, the tracking process fails. Another inconvenience is the effect of the size of the tracking window. Errors in the tracking process may also occur as a result of large displacements or changes in illumination. Since, interest point tracking algorithms formulate tracking as the minimization of error only between successive images, such algorithms are known to "drift" over time [8, 36].

In order to achieve robust tracking, an effective strategy has been to include both types of fea-

tures in the estimation process. Traditionally, the integration of different visual cues for tracking has been modeled as a probabilistic estimation problem, particularly set in a generative framework [37]. In this framework, the state of the object of interest (modeled in our case by a homography) is assumed to be a statistical quantity whose *maximum aposteriori* value is estimated in each frame. Prior information for the tracking process comes from a knowledge about the *motion model* of the object, and *likelihood* information is obtained from the visual cues estimated from each frame [38]. Such a model is effective since it is difficult to model the non-linear nature of the homography space and the non-Gaussian nature of the process and measurement errors (Section 3.5). Several ways to model the likelihood [38, 39, 40, 41] have been proposed earlier.

In the current work, we wish to address the problem of tracking an object through large changes of perspective and illumination. Perspective changes are obtained when the camera *does not* point at the object of interest and is in fact quite far away in terms of direction. The resultant view of an object is a skewed one, similar to the view obtained when humans look at objects from *"the corner of the eye"*. This problem is of relevance to the robotic community since tracking objects in such scenarios becomes erroneous and hence contributes to an incorrect estimation of the position of the robot with respect to the environment. This, in turn, affects tasks like navigation.

Given the existence of methods to produce accurate correspondences [23] and optimal methods to compute homographies [18], it is prudent to ask why homography computation should be posed as a tracking problem rather than a direct computation between two views. There are two main reasons

- Feature descriptors like SIFT [23] perform poorly under large changes of perspective or illumination. In a video, such changes are gradual and can be overcome/sidestepped using a tracking approach.
- Regions with textures of lower quality [8] are generally not matched accurately. In such cases, features from other parts of the image belonging to a different plane might be used to improve homography computation (Chapter 4).
- In presence of cluttered background with similar textures, descriptors like [8] tend to "drift". This undesirable characteristic can be overcome using other cues like edges (Section 3.3.3) or geometry (Chapter 4), although only partially.

Considering the complementary advantages and inconvenience of edges and texture, it is matured to consider both together [21], [42], [43]. In the literature, most works in cue integration are sequential in nature. For example, [44] uses the result from the texture point tracker to provide better positioning of the edge location. In contrast, in this paper, cue integration is done in parallel, assuring that the misgivings of the texture tracker are not propagated to the edge tracker.

The remainder of the paper is organized as follows. Section 3.2 presents work related to tracking using texture and edge based cues using particle filters. Section 3.3 outlines the proposed tracking framework to incorporate multiple cues, and motivates the use of the particle filter for the task at hand. Section 3.4 describes the particle filter approach to integrating multiple cues for tracking a single plane. Experimental results on challenging sequences (Section 3.6) under varying conditions of pose and illumination show the robustness of the tracker. We answer the question of why a particle filter framework is well suited to the problem of tracking in Section 3.5. Finally, we summarize with elucidation of future avenues for research in Section 3.7. Sections 3.3 & 3.4 have overlap with earlier works [26, 4].

3.1.1 Notation

In the remaining sections, we follow a few conventions. Capitals like (C, F, I, J, H, R, Z) are used to represent quantities like matrices, or images, or contours, or covariances. Their notation is defined as and when they are used. Small letters either donate elements of a list, or vectorized form of matrices denoted by capitals, or scalars. Thus h denotes a vector form of the homography matrix H, and so on. When a particular small letter denotes a vector form or an element is made clear from the context of the presentation. For example, r_0 represents an element of the rotation matrix R. However, r_x represents rotation about the x axis. Bold letters like $\mathbf{x}$ are used to represent an image point, while $\mathbf{l}$ is used to represent a line in the image. Since probabilities are involved, the notation $\hat{}$ is used to represent *estimated* quanities of the true variable with the same corresponding letter. p denotes probability. Other symbols used are clarified at appropriate places.

3.1.2 Contributions

In this chapter, we present an application of MVG to the task of tracking. Specifically, we show how a particle filter based algorithm for the tracking of planes can be set in a homographic framework, in order to track the position and shape of a planar object over time. Our contributions include

- A framework for tracking using particle filters, in which the motion of a planar object is tracked by modeling it as a homography. A Bayesian filtering approach is developed with two kinds of measurements, texture or interest points and edge or contour measurements.
- A set of heuristics for (i) determining the importance weights of each of the *measurements*, based on the quality of texture or edge being tracked as the case may be, (ii) and determining robust log likelihoods of the measurements based on a point-wise maximum model.
- An analysis of the particle filter framework, to provide adequate justification as to why there is a need for using particle filters, over other, more simpler filters like the Kalman filter variations.

As we shall see in Chapter 4, the current Bayesian framework can actually extended to the case where *multiple* planes need to be tracked, where cues from each plane can be used to *correct* the errors in the tracking of other planes.

3.2 Related work

Texture Based Tracking Texture based tracking methods in the literature [8, 45, 11, 46, 47, 48] can be grouped into two categories. The first category is the model-less approach to texture tracking [8]. In [8], interest points are chosen in the image that represent sufficient texture information. Now, in a hierarchical scheme, the local intensity profile around each interest point is matched with the next frame of the image sequence by minimizing a suitable objective function. The region of maximum overlap is declared as the patch corresponding to the interest point in the next frame. Such an algorithm is more robust to partial occlusion of the object since interest points are tracked in a local coordinate frame. Improvements to the algorithm include effective pruning of outliers to produce more accurate tracking [49] and extension to tracking under large deformations and illumination changes [50].

The other approach to tracking is to consider the whole object as one big texture template that needs to be tracked over time. Approaches in this category [11, 48, 45] basically model tracking as a motion estimation problem with the measurement being the intensity difference between successive

frames. Thus, the difference between the current frame and the reference template warped with the latest parameter values of the motion model, is minimized. Different approaches in this regard involve simply using the Jacobian [45, 48] to tackle the non-linear minimization as opposed to an efficient second-order minimization [11]. Such algorithms are a bit more robust to illumination changes since minimization happens over a large area of the object. However, they are not suited for large perspective changes. Recently, hybrid algorithms that use both texture patches of small size and the whole template have been proposed to deal with large perspective and illumination changes [47].

Edge Based Tracking Like texture tracking, approaches based on edge based (or contour based) cues can also be categorized into model-less and model-based methods [51, 38, 52]. One of the first methods for contour tracking was the "Moving Edges Algorithm" [51]. This algorithm, essentially similar to [8], tracked contours by minimizing the intensity difference of points along the contour with the next image. The search for correspondence was done along a line normal to the contour. Finally the cost function minimized was a sum of the intensity of the point in the previous frame and the points along the normal direction. The major problem with this approach was that multiple local minima could be found when the object was surrounded by cluttered background.

The CONDENSATION algorithm [38] presents a model-based approach to contour tracking. In this case, B-splines are used to represent the contour with a non-linear motion model to explain non-rigid deformations of objects. Also, since the measurement functions are non-linear, a standard tracking framework like Kalman filters is not applicable. Instead to model the multi-modal posterior and likelihood functions, particle filters are used. Unlike model-based approaches to template matching however, the CONDENSATION algorithm builds upon the Moving Edges Algorithm. Another alternative to contour tracking was suggested in [52]. Instead of modeling the motion by a suitable representation, the authors resort to learning the possible non-rigid deformations of the object in an offline learning stage. Thus the actual motion of the object is parametrized by these learnt states and a similarity model. The tracking problem is then formulated as a minimization of distances from a edge map computed for every frame.

Particle Filter Tracking One of the first algorithms to employ particle filters for visual tracking was the CONDENSATION algorithm [38]. Particles were employed to sample the space of B-spline curves respecting a particular motion model, and each hypothesis was evaluated by the edge measurements available from the current image. The main advantage of particle filters is that they allow representation of non-linear non-Gaussian models of motion and state. For example, the homography space is non-Gaussian. This means that given the homography between the reference and current frames, the probability distribution of process and measurement errors between the reference frame and the current frame does not follow a Gaussian distribution in the homography space (Section 3.5). In such a case, filters like the Kalman filter are rendered ineffective. Additionally, the measurement model (reprojection error) is non-linear in nature. Thus, particle filter provides an effective strategy to model such motion. Additionally, particle filters also provide a framework for a disciplined integration between various cues available for tracking [53, 37].

3.3 Tracking: Framework

In this section, we develop the process and measurement models for the tracking framework. In the following sub-sections, we first develop the motion model that we use (Section 3.3.1), and describe the process of obtaining and integrating measurements (Section 3.3.2). This is followed by the

development of *likelihoods* that describe the process of integrating measurements based on texture and contour detectors into the framework. Finally the whole process is explained in Algorithm 1 and demonstrated in Figure (cite reference).

3.3.1 Motion Model: 2D

Given an initial image I_0 and an image I_t of a planar object at time instant t, there exists a homography h_t relating these images. If the vector $\mathbf{x}_0 = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix}^\top$ represents the homogeneous coordinates of a point in the first image and the vector $\mathbf{x}_t = \begin{bmatrix} x_t & y_t & 1 \end{bmatrix}^\top$ represents a point in the second image, the relation between these two points is written as $\mathbf{x}_t \sim H_t\mathbf{x}_0$ or

$$\mathbf{x}_t \sim \begin{bmatrix} h_t^1 & h_t^2 & h_t^3 \\ h_t^4 & h_t^5 & h_t^6 \\ h_t^7 & h_t^8 & h_t^9 \end{bmatrix} \mathbf{x}_0. \tag{3.1}$$

Estimating the homography can be posed as estimating the parameters of H_t, represented as a vector

$$h_t = \begin{bmatrix} h_t^1 & h_t^2 & h_t^3 & h_t^4 & h_t^5 & h_t^6 & h_t^7 & h_t^8 & h_t^9 \end{bmatrix}^\top. \tag{3.2}$$

This estimation is posed as a Bayesian inference problem in this chapter, along the lines of [37], to enable the inclusion of robust likelihood measurements as detailed in the next section. Thus we are interested in estimating the *aposteriori* value of h_t, at every instant. We make the assumption that the camera/object motion is smooth, and so h_t can be written as an increment over the homography computed in the previous frame. This is a reasonable assumption since most cameras capture images at 30 frames per second, which makes inter-frame motion small, and hence continuous. From the perspective of tracking, this assumption requires two concepts. The first is the concept of a Markov process, where the state at a particular time instant only depends on the state of the previous time instant. The second is the concept of a *Brownian* motion, which states that the state of a system in the current time instant is a random variable drawn from a normal distribution centered over the value of the state in the previous instant. Thus if $\hat{h}_t$ is the current homography estimate, we define change in this vector owing to inter frame displacement as

$$\hat{h}_t = \hat{h}_{t-1} + \Delta\hat{h}_t. \tag{3.3}$$

where, $\Delta\hat{h}_t$ is a random variable following a normal distribution. The parameters of the distribution are discussed in the implementation (Section 3.6). Now, let us consider a set F_0 of M visual features $F_0 = \{f_0^1, \ldots, f_0^M\}$ in the reference image I_0. This set of features is mapped by the transformation $\hat{H}_t$ to the estimated set of features $\hat{F}_t = \{\hat{f}_t^1, \ldots, \hat{f}_t^{\hat{M}}\}$ in the current image. The true estimate of the vector $\hat{h}_t$ can be computed by minimizing an error function of the form

$$\mathcal{G}(\hat{h}_t) = \mathcal{F}(\hat{F}_t - F_t), \tag{3.4}$$

where F_t is the measured visual feature vector and $\mathcal{F}$ is the distance measure. We define the optimal value $\hat{h}_t$ as the value that minimizes the above error over a *homography search space.* In a probabilistic sense, we define the *aposteriori* value of $\hat{h}_t$ to be

$$\hat{h}_t = \arg\min_{h_t} \mathcal{F}(\hat{F}_t - F_t), \tag{3.5}$$

We adopt a probabilistic model by defining the *homography search space* as the *prior* distribution, defined using Equation (3.3). Samples taken from this distribution are evaluated against

measurement *likelihoods* (Sections 3.3.3 & 3.3.3) to obtain an *aposteriori* state estimate for the current image $\hat{h}_t$. Derivations involving the measurement process to convert this prior estimation to posterior is explained in the next sub-section. It is important to note here that there exists no direct linear analytical method that can minimize this error function 3.5(detailed in Section 3.5). Particle Filter algorithm (Section 2.3) or what is called Condensation algorithm [38] is preferred here because it provides an efficient probabilistic framework to take care of such uncertainties.

3.3.2 Bayesian Tracking

This section outlines the Bayesian tracking formulation that employs Particle Filters (Section 2.3). Specifically, we show how the prior distribution represented in Equation (3.3) can be converted to the *aposteriori* distribution described by Equation (3.5).

Let h_t represent the state variable to be estimated at the current iteration of the algorithm. Let $\pi(h_t)$ be the belief of the random vector h_t at time t represented by posterior probability $p(h_t \mid F_{1,\ldots,t})$ based on features $F_{1,\ldots,t}$. Expanding using Bayes rule

$$p(h_t \mid F_{1,\ldots,t}) = \frac{p(F_t \mid h_t)p(h_t \mid F_{1,\ldots,t-1})}{p(F_T \mid F_{1,\ldots,t-1})}. \tag{3.6}$$

which reduces to

$$p(h_t \mid F_{1,\ldots,t}) \propto p(F_t \mid h_t)p(h_t \mid F_{1,\ldots,t-1}) \tag{3.7}$$

considering $p(F_T \mid F_{1,\ldots,t-1})$ to be a constant. The above equation states that the *aposteriori* measurement of the current instant, depends on the *likelihood* of the features in the current image, given a homography hypothesis ($p(F_t|h_t)$ and a *prior* on the homography in the current instant, depending on features measured *in the past* ($p(h_t|F_{1\ldots t-1})$). However, the above equation means that the search space for evaluating homographies encompasses the *entire* 9 dimensional space, since no restriction is put on its value here. Thus, we need to introduce the prior (Equation (3.3)) into the above equation. To do this, we make the observation that since feature measurements of the past are responsible for the *aposterori* estimation of the previous instant h_{t-1} and bear *no other* relation to the current state of the system because of the assumption of a *Brownian* motion and a Markovian model, we can further reduce the search space of the above equation. The next paragraph describes this process.

Particle Filter In order to reduce the search space, we show that based on our earlier assumptions, searching the entire homography space is equivalent to searching within the *apriori* space defined in Equation (3.3). To do this, we marginalize the probability $p(h_t \mid F_{1,\ldots,t-1})$ and apply Bayes' rule again to obtain the Bayesian estimation $p(h_t \mid F_{1,\ldots,t})$ as

$$\propto p(F_t \mid h_t) \int p(h_t \mid h_{t-1})p(h_{t-1} \mid F_{1,\ldots,t-1})dh_{t-1}. \tag{3.8}$$

Since the value $p(h_{t-1}|F_{1\ldots t-1})$ represents the *aposterior* estimate of homography from the previous instant, its probability space is a dirac delta function, with a probability of ∞ at the estimate value and 0 everywhere else. Thus, the above equation reduces to only estimating $p(h_t|h_{t-1})$, which is exactly our prior (Equation (3.3)). For more details, the reader is referred to [37].

The basic idea of particle filters is to approximate posterior density $p(h_t \mid F_{1,\ldots,t})$ by a set of samples (particles) h_t^i with associated weights or importance factors w_t^i. Thus each particle represents a potential homography hypothesis, and the corresponding weight represents the *closeness* of that particular hypothesis to the *true value*. The M particle-weight pairs $\{h_{t-1}^i, w_{t-1}^i\}_{i=1}^M$,chosen

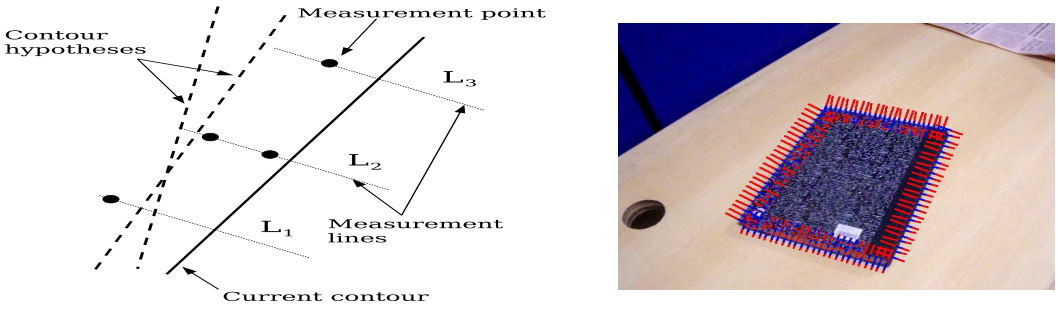


Figure 3.1: Left: Contour features in the current frame are obtained by searching along lines perpendicular to the contour estimated in the previous frame. Right: Contour extraction shown in practice.

to approximate density $p(h_{t-1} \mid F_{1,\ldots,t-1})$, are propagated to pairs $\{h_t^i, w_t^i\}$ using the motion model prior $p(h_t \mid h_{t-1})$. The weights $\{w_t^i\}_{i=1}^M$ associated to the particles h_t^i are computed proportional to the likelihood function in case of using the bootstrap filter as

$$w_t^i = \alpha \; p(F_t \mid h_t^i). \tag{3.9}$$

These particles, propagated from the posterior distribution of the previous frame to the current through the prior model are *condensed* based on the likelihood function (Equation (3.9)) to form an estimate of the posterior distribution of the current frame.

3.3.3 Texture and Edge Cues

In order to condense the particles generated from the previous instant's posterior estimate, and passed through the prior model, we need to define the *likelihood functions* $p(F_t|h_t^i)$. Depending on the feature set F_t different likelihood models may be derived. We use texture and contour features in this chapter, because of reasons explained in Section 3.1.

Contour Likelihoods

Let C_{t-1} be the contour representing the object in the previous frame at instant $(t-1)$. This contour, for simplicity, has been assumed to be an edge. Let h_{t-1} be the vector representing the homography that relates C_{t-1} to a reference contour C_0. A suitable discretization of the both contours C_{t-1} and C_0 is represented by the set of image points $\{p^m\}_{m=1}^M$. Points belonging to the current contour C_t can be searched along lines l_m perpendicular to the previous contour C_{t-1} centered at points $\{p_{t-1}^m\}_{m=1}^M$. Points $\{p_{t-1}^m\}_{m=1}^M$ are called principle points and lines l_m are called measurement lines.

Figure 3.1 demonstrates the measurement process to estimate the contours in the current frame. The object contour represents the object edge at the previous frame. The measurement lines are drawn normal to, and centered around the previous edge. In the figure, there are three normal measurement lines with one or more edge points detected on each line. Two edge proposals are shown and only the nearest edge point measurement to the edge proposal is considered. In case the measurement line does not intersect the edge proposal, as the line L_3, the data from this line will be nullified and will not be used in the likelihood function. Note here that by using short measurement lines, we reduce the effect of spurious contour points in cluttered environments.

A generic model of the contour likelihood was proposed in [34]. We start from this model to develop our robust contour likelihood model. Let the hypothesis h_t^i be a proposal of the state of the

contour C_t that intersects the measurement line l_m at a distance d_m from the same principle point (Figure 3.1). Then the generic likelihood model is given as

$$p(F_C|h^i) \quad = \quad \prod_{m=1}^{M} b(n_m - 1) \sum_{k=1}^{n_m} \frac{exp(-\frac{(v_k - d_m)^2}{2\sigma^2})}{n_m L^{n_m - 1}} \tag{3.10}$$

In words, the above equation states that if $n_m$features are obtained while searching along a measurement line of length L, then each of these measurements contribute a value $(v_k - d_m)^2$, measured in a Gaussian distribution of variance σ to discourage features too far from the hypothesized contour in the current image, C_t. Here, $v_k$represents the distance of the current feature from C_t and d_m represents the perpendicular distance of C_t from p^m_{t-1}. This estimation is then weighted down by a probability function $b(\cdot)$ that denotes the probability of finding n_m features.

Starting from the generic likelihood model, we assume that only 1 feature is detected along each measurement line, and that the feature selected is the one nearest to the hypothesized contour C_t. This basically ensures significant computational gains, since the measurement process is now simplified. Thus we ignore the probability function $b(\cdot)$ to integrate multiple measurements along a line [34], and derive our likelihood as

$$p(F_C \mid h^i) = \prod_{m=1}^{\bar{M}} \left(\frac{1}{\sqrt{2\pi}\ \sigma} \exp(-\frac{(D_m)^2}{2\sigma^2}) \right) = \prod_{m=1}^{\bar{M}} Q_m, \tag{3.11}$$

where $D_m = \min\{v_k - d_m\}_{k=1}^{n_m}$ represents the sum along each line approximated by its large value to speed the process. The number of measurement lines that intersect the contour C_t is $\bar{M}$. Taking log to simplify multiplication, we can write

$$Q_C = \log(p(F_C \mid h^i)) = \sum_{m=1}^{\bar{M}} \log(Q_m). \tag{3.12}$$

Texture Likelihoods

Harris detector is a method that detects interest point in scale-space based on the Laplacian. A popular tracker of Harris points is Shi-Tomasi-Kanade tracker [8]. In fact, the point locations of the features in the initial frame are selected as those points that show more track-ability than others. The higher singular values are, the more interesting the point feature is. Note that the most detected features are the corners and similar entities where its high spatial gradient gives robust information about its 2D properties.

The model used for the texture likelihood is the point-wise re-projection error, most suitable for Harris [54] points. Let a set of texture features F_T be extracted from the image at time t using Harris' detector. These features are matched to the corresponding features F_0 in the initial frame. Given a proposed motion model h^i, the probability density function of the likelihood is given as

$$p(F_T \mid h^i) = \prod_{k=1}^{N_p} \left(\frac{1}{\sqrt{2\pi}\ \sigma} \exp(-\frac{(D_k)^2}{2\sigma^2}) \right) = \prod_{k=1}^{N_p} Q_k, \tag{3.13}$$

Here, N_p is the number of texture points under consideration, the error D_k is the Euclidean distance $D_k = F_{kh_t} - F_{kT}$ between the kth measured point F_T and the projection of the kth point F_0 from

the initial frame to the current frame(F_{h_t}). Again taking log, Q_T can be written as

$$Q_T = \log(p(F_T \mid h^i)) = \sum_{k=1}^{N_p} \log(Q_k). \tag{3.14}$$

3.4 Multi-Feature Single Plane Tracking

In the previous section, we outlined the likelihood functions that are used in the estimation process (Equation 3.9). Since the measurement of texture and contour features for obtaining likelihoods is a process that is dependent on the quality of the image being used, it is unnatural to model the likelihood functions assuming all the measurements are equally important. For example, a certain interest point may be tracked better than others in not so textured regions. Thus, it is important to introduces *weights* that indicate *how good* a particular measurement is. These weights can then be integrated into the likelihood models (Equations (3.14) & (3.12)).

For this reason, we first look at an algorithm that measures how a particular interest point's intensity varies over time, across images. Thus interest points with less variation are good features that can be tracked reliably, since most feature tracking algorithms assume minimal intensity variation over time. Shi and Tomasi [8] developed a method that measures the dissimilarity between image point features. They found out a measurement matrix whose eigenvalues are large when the dissimilarity is less. In other words, if the measurement matrix has a large enough eigenvalue, the feature is considered as a good feature to track. We generalize this concept about points, given by Shi and Tomasi, to the case of edge features located on a measurements line. Thus, we start from these goodness measurements to define a function that associates a probabilistic weights for the edge (contour) and texture features.

3.4.1 Good Features to Track

Similar to [8], the affine image motion between successive frames is computed as one that minimizes the intensity *dissimilarity*

$$\epsilon = \int\int_W [J(A\mathbf{x} + \mathbf{d}) - I(\mathrm{x})]^2 w(\mathbf{x}) d\mathbf{x} \tag{3.15}$$

where W represents the feature window around a Harris corner and $w(\mathbf{x})$ is a weighting function. Using Taylor expansion and after a few simplifications, we arrive at the following equation for determining a "good" feature.

$$Z\mathbf{d} = e$$

where Z represents the covariance of the image derivative

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$$

For a tractable feature, it is required that the matrix Z has large eigenvalues.

Good Texture Features:

Texture features essentially represent a pattern in which intensity change within a feature window is present along both x and y-directions. Thus, the two eigenvalues are both large and comparable in

magnitude. Thus ensuring that the minimum eigenvalue is above a threshold suffices to find "good" texture points. This is expressed by the equation

$$\min(\lambda_1, \lambda_2) > \lambda_p \tag{3.16}$$

where (λ_1, λ_2) are the eigenvalues and λ is a chosen constant.

Good Edge Features:

In case of edge features, the intensity pattern in the feature window is unidirectional. Since the feature window needs large eigenvalues to be resilient to noise, the above condition for texture applies here as well. However, due to the uni-directional nature of edges, one eigenvalue is significantly smaller than the other. Thus, we may impose the following additional constraint to acquire "good" edge features

$$\max(\lambda_1, \lambda_2) > \lambda_e. \tag{3.17}$$

3.4.2 Assigning Feature Weights

Given a set of good texture and edge features, we may define the goodness of each type of feature by measuring the amount of features that are tracked along the sequence of frames. Let us remember that we consider here two types of visual features. They are contour feature F_C and texture feature F_T. Texture feature is essentially an image point; while contour feature is a gradient peak along a measurement line. We develop a goodness function for texture point features and the one for contour features is analogous.

Assume we select N_0 texture point features in the initial frame. Only N_t features in the current frame have been selected as good features and matched to its corresponding points in the initial frame. Let the set $\mathcal{N}_t = \{n_i \mid i = 1, \cdots, N_t\}$ be the set good features tracked in the current frame. The probability that a point feature n_i is in this set can be given as a function of the dissimilarity measurement given in (3.15) as

$$W_T = \; p(\mathcal{N}_t \in \mathcal{N}_0) = p(\mathcal{N}_t \in \mathcal{N}_{t-1})\; p(\mathcal{N}_{t-1} \in \mathcal{N}_0) \tag{3.18}$$

$$p(\mathcal{N}_t \in \mathcal{N}_{t-1}) = \frac{1}{N_{t-1}} \sum_{i=1}^{N_{t-1}} \frac{1}{2\pi\sigma^2} \exp\left[-\frac{\epsilon_i^T \epsilon_i}{2\pi\sigma^2}\right] \tag{3.19}$$

To simplify the computation, let $p(\mathcal{N}_{t-1} \in \mathcal{N}_0) \approx \frac{N_{t-1}}{N_0}$ and finally we write

$$W_T = \frac{1}{N_0} \sum_{i=1}^{N_{t-1}} \frac{1}{2\pi\sigma^2} \exp\left[-\frac{\epsilon_i^T \epsilon_i}{2\pi\sigma^2}\right]. \tag{3.20}$$

For edges, assume that there are N_0 measurement lines on the object contour in the initial frame and N_t matched good feature measurement lines in the current frame. Analogous to texture point features, the weighting function for edge features can be written as

$$W_C = \frac{1}{N_0} \sum_{i=1}^{N_{t-1}} \frac{1}{2\pi\sigma^2} \exp\left[-\frac{\epsilon_i^T \epsilon_i}{2\pi\sigma^2}\right]. \tag{3.21}$$

This allows us to get a quantitative evaluation of the feature's reliability. The higher the weight, the more reliable a feature is. By comparing the weights, we may decide upon the most reliable feature.

Algorithm 1 Visual Tracking based on Goodness Weight

Input: $I_0, \ldots, I_t$, $F_0 = (f_i^0 \mid i \in [1, \ldots, n])$
Output: $h_t = \left[h_t^1, \ldots, h_t^9\right]$
NumParticles: $M = (M_C + M_T)$
Set the number of particles needed to sample the space effectively
for $(k \in 1, \ldots, t)$ **do**
$F = \text{ExtractFeatures}(I_k)$
$F_k = \text{TrackFeatures}(I_k, F, F_{k-1})$
$\left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M_C} = \text{DrawSamples}(h_{k-1}, M_C)$
$\left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M_T} = \text{DrawSamples}(h_{k-1}, M_T)$
$\left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M} = \left(\left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M_C}, \left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M_T}\right)$
$h_{k|k-1} = \text{MotionPrior}(F_k, F_{k-1})$
$\left(\mathrm{h}_{\mathrm{k}}^{\mathrm{i}}\right)_{i=1}^{M} = \text{PropogateParticles}(\left(\mathrm{h}_{\mathrm{k}-1}^{\mathrm{i}}\right)_{i=1}^{M}, h_{k|k-1})$
$(Q_C)_{i=1}^{M_C} = \log(\text{CLikelihood}(F_k, \left(\mathrm{h}_{\mathrm{k}}^{\mathrm{i}}\right)_{i=1}^{M_C}))$
$h_k^C = \arg\min_i (Q_C)_{i=1}^{M_C}$
$(Q_T)_{i=1}^{M_T} = \log(\text{TLikelihood}(F_k, \left(\mathrm{h}_{\mathrm{k}}^{\mathrm{i}}\right)_{i=1}^{M_T}))$
$h_k^T = \arg\min_i (Q_T)_{i=1}^{M_T}$
$W_C = \text{GoodEdges}(F_k, F_{k-1})$
$W_T = \text{GoodTextures}(F_k, F_{k-1})$
$h_k = W_C * h_k^C + W_T * h_k^T$
Democratic Integration
end for

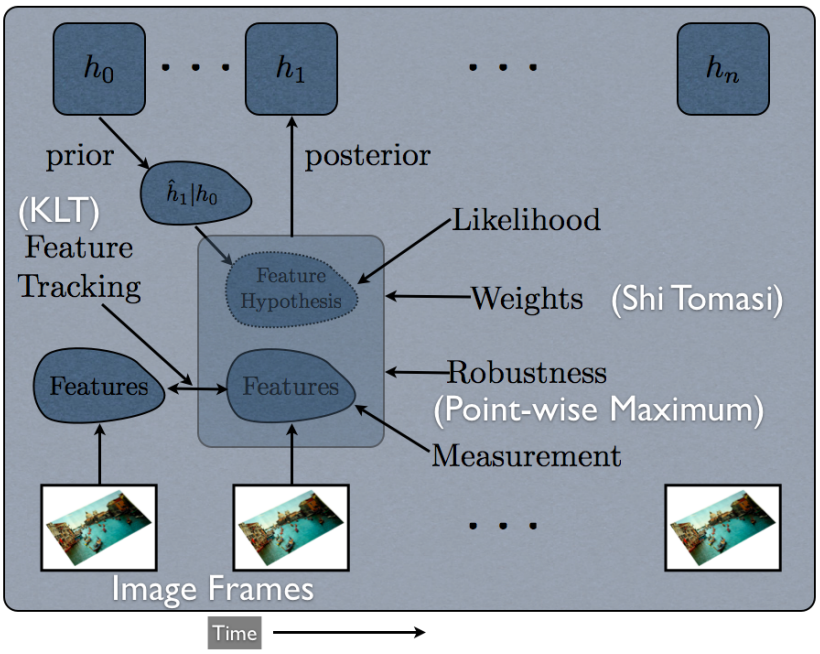


Figure 3.2: The Particle Filter Tracking Algorithm.

3.4.3 Robustness

The likelihood functions Q_C and Q_T presented in Equation (3.12) and Equation (3.14) are robust to Gaussian noise but are sensitive to outliers. To handle outliers, we use Q_C^m and Q_T^n. These error functions are the mth and nth smallest values of the error vectors Q_C and Q_T respectively.

$$Q_C^m = m\text{th}_i \{Q_C\}_i \quad Q_T^n = n\text{th}_i \{Q_T\}_i \tag{3.22}$$

The objective functions that belongs to this class are highly robust to outliers. For example, when $m = M/2$ and $n = N_p/2$, these are the median operator and the minimization process leads to least median optimization [55], which can handle noisy measurements with upto 50% outliers.

Till now, we have discussed the prior and likelihood models and developed the tracking framework. We discussed approaches to weigh the different feature measurements based on an estimate of their track-ability. Based on robust statistics, these measurements are then included into the tracking process to estimate the likelihood of hypothesis generated using particle filters. The overall algorithm is now summarized in Algorithm 1.

3.5 Why Particle Filters ?

In this section, we explore the reasons that justify the use of particle filter for homography based tracking. In order to do so, we look at the following aspects about our algorithm

- Can the process noise in the tracking framework assumed to be Gaussian ?
- Can the measurement noise assumed to be Gaussian ?
- Can the non-linearity of the process and measurement functions be handled by an approximation like the Unscented Transform of the UKF ?

3.5.1 Process Noise

Let us first consider the process dynamics. In our current work, we have assumed that the motion model is a Brownian motion in the homography space. This essentially means that in the *apriori* estimate of the current state is the same as the *aposteriori* estimate of the previous state with the addition of some noise. In order to analyze the effect of camera motion on the computed homography, let us first recall that the homography induced by a plane is given by

$$H = R - \frac{t\, n^\top}{d} \tag{3.23}$$

where (R, t, n, d) are the camera and plane parameters respectively. Examining each element of the matrix gives us

$$H(i,j) = R(i,j) - \frac{t(i)n(j)}{d} \tag{3.24}$$

where (i, j) represent row and column indices respectively. Considering that for our task the values (n, d) representing the plane parameters are assumed to be constant, the above equation represents a linear transformation of the pose parameters. From literature, we know that pose has often been modeled using a Brownian motion with a Gaussian process noise [3], which means that modeling the elements of rotation and translation in the same manner (rotation being a non-linear function of the pose parameters) can be acceptable, and since Equation (3.24) is a linear transformation of the rotation and translation parameters, assuming a Gaussian process noise seems to be a reasonable choice. Our argument is further supported by Figure (3.4) which shows that bounded pose parameters produce bounded rotation parameters that can be modeled as a Gaussian in most cases. Although some of the elements follow a non-Gaussian transformation, the error seems to be acceptable, as our experiments (Figure 3.5) show.

3.5.2 Measurement Noise

Given a homography H between two images, feature points $\mathbf{x} = [x \quad y \quad 1]$ and lines $\mathbf{l} = [l_x \quad l_y \quad 1]$ transform as

$$\mathbf{x}' = H\mathbf{x} \tag{3.25}$$
$$\mathbf{l}' = H^{-\top}\mathbf{l} \tag{3.26}$$

where $(\mathbf{x}', \mathbf{l}')$ are the corresponding points and lines in the second image. Since, the measurement noise should give an estimation of the possible deviation of the actual observation from the measured value, it should also account for the process noise propagated through the observation functions (Equations (3.25,3.26)). Although it is well known that a Gaussian variable retains its characteristics when passed through a linear system, analysis of the effects of a non-linear system on Gaussian variables is complex and at best can only be approximated to a degree [9]. As verification for our theory, Figure 3.3 shows how a line is transformed by a homography that is a direct result of perturbation of the system dynamics by Gaussian error. As can be seen, the "Gaussian-ness" of the error is not preserved. Thus, although the process noise may be modeled by a Gaussian, it is not an effective modeling strategy for the measurement noise.

3.5.3 Non-linearity of the Measurement Function

Since the state transition function assumes a Brownian motion and hence is a linear function, we need to establish the *nature* of the non-linearity of the measurement function in order to ascertain why an approximation like the EKF or the UKF may not be as suitable for the task at hand as the particle filter. In order to do that, we consider the measurement functions in Equations (3.25,3.26) instead of the actual Equations (3.13, 3.11) for simplicity. This choice wont affect our argument since we are only concerned with the non-linearity of the measurement process, and Equations (3.13, 3.11) are highly non-linear as compared to the equations used here. We proceed to compare the performance of a Particle Filter based tracker for homography against an Unscented Kalman Filter for the same state dynamics.

The following are the state space equations that we consider

$$
\begin{aligned}
H_k &= H_{k-1} + \Delta H && (3.27)\\
\Delta H &= \mathcal{N}(0, Q) && (3.28)\\
x_k &= Hx_0 + \Delta x && (3.29)\\
\Delta x &= \mathcal{N}(0, R_x) && (3.30)\\
l_k &= H^{-\top} + \Delta l && (3.31)\\
\Delta l &= \mathcal{N}(0, R_l) && (3.32)
\end{aligned}
$$

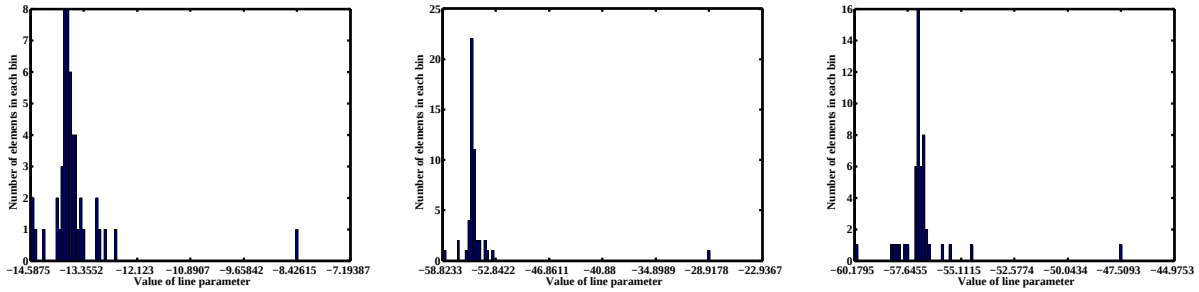
where $\mathcal{N}$ represents a Gaussian, in this case with mean 0 and covariances (Q, R_x, R_l) . Figure 3.5 shows the comparison between a UKF tracker and a Particle Filter tracker, for tracking a single plane. As can be seen, because of the considerable amount of non-linearity involved in the inverse of a matrix, the UKF is not adept at tracking the measurements of the lines correctly. It is important to note here, that the above measurement functions are *projective* in nature, and so the *actual* innovation process (difference between estimated observation and measured observation) involves yet another non-linearity of division.

Since both the measurement process being non-Gaussian, and the non-linearity being too high for approximation are factors that make the UKF unsuitable, the EKF as an alternative is ruled out as UKF is easily a more sophisticated tracker [28]. This justifies the use of Particle Filters in our approach, with an added advantage that a Particle Filter framework allows for additional robustness functions that are difficult to incorporate otherwise.

3.6 Experimental Results

We conducted a number of experiments in order to both ascertain our decision to employ particle filters for plane tracking, as well as to analyze the robustness of the tracker.

Since particle filters are a tool that perform state estimation *without* any model of the noise covariance of the system under consideration, it is important to first determine that the standard noise models *do not* apply to the problem at hand. It is to verify this claim that we tested the process and measurement noise covariances for the current approach, to determine their Gaussian nature (Figures 3.3 & 3.4). While Figure 3.4 ascertained that a Gaussian modeling of the process noise is an acceptable approximation for homography tracking, Figure 3.3 shows decisively that the measurement model has non-Gaussian noise. In fact it is so very highly non-linear, (because of a 3 degree matrix inverse function coupled with the perspective projection function), that a Gaussian distribution from the homography space, ends up clustering almost like a dirac delta



(a) (b) (c)

Figure 3.3: Monte-Carlo test: A line transferred by multiple homographies, that are generated as a result of perturbation of a "mean" homography. The noise introduced is Gaussian in nature. Observe how the resulting variation in the parameters of the line, cannot be modeled as a Gaussian. The non-linearity is introduced because of the presence of the matrix inverse, which is a non-linear operation (it involves computation of the determinant, which in itself is a non-linear operation of degree 3).

function, around certain chosen values of the transformed space (in this case the space of lines). The distribution in Figure 3.3 is also multi-modal which means that there is an existence of multiple points, where the Gaussian distribution from the homography space condenses.

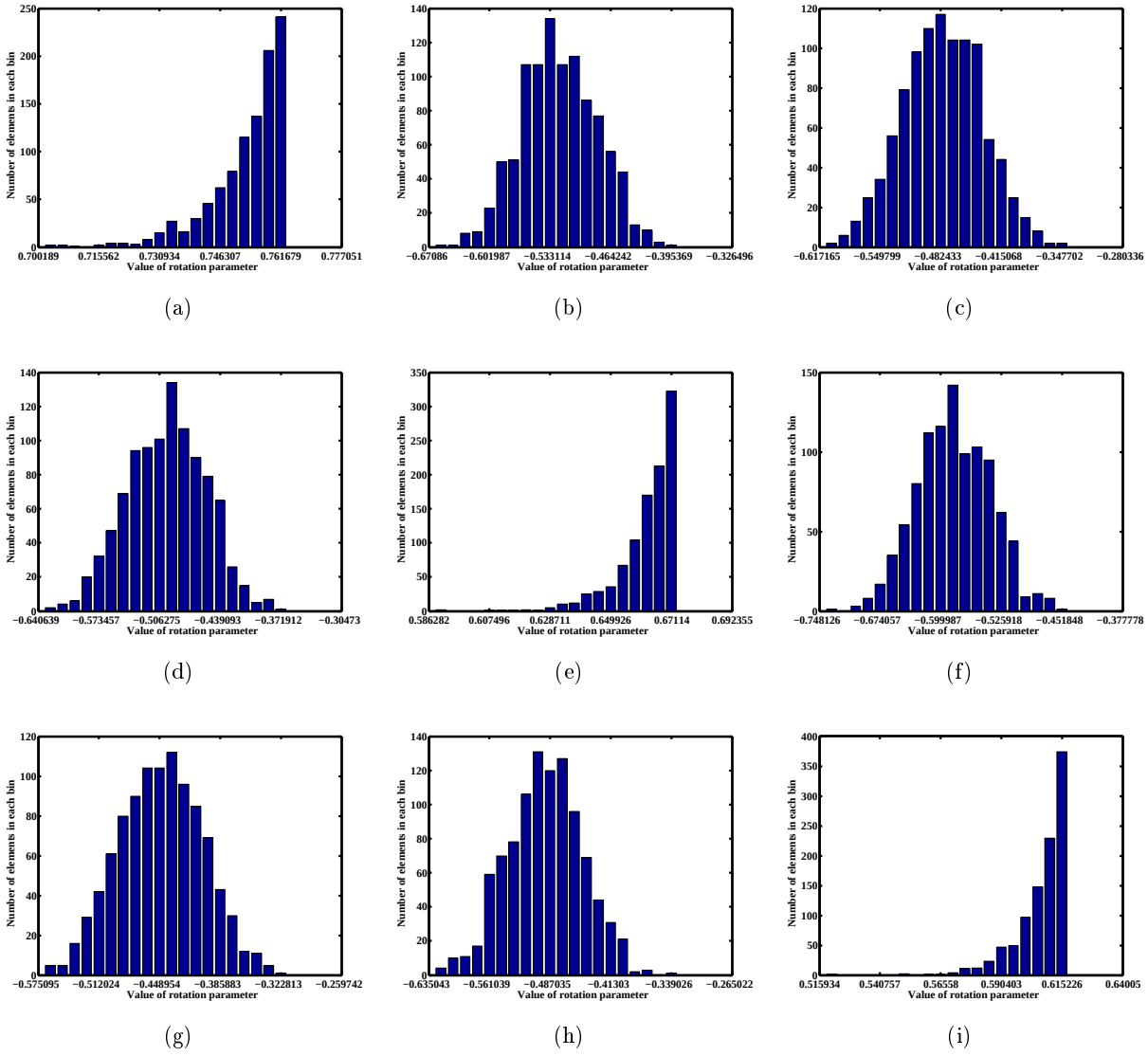


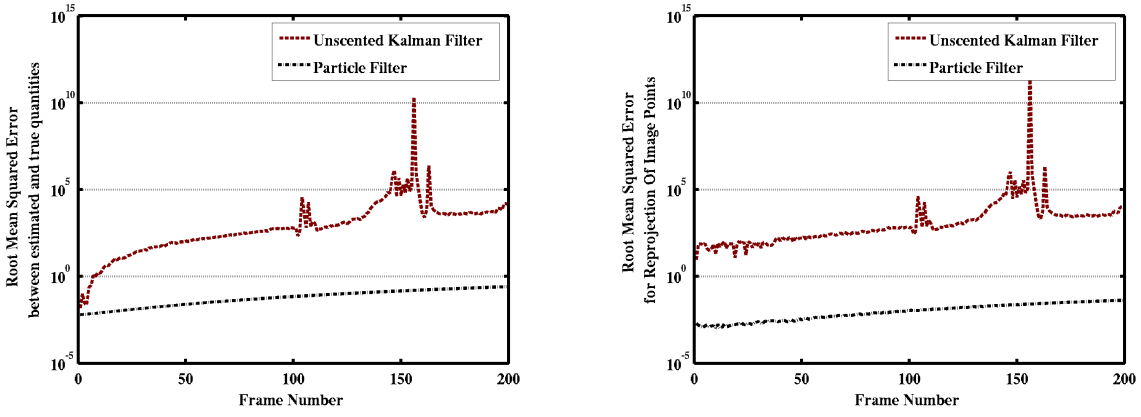
Figure 3.4: Monte-Carlo test: This figure show the variance of the elements of the rotation matrix, when Gaussian noise is introduced into the corresponding representation of rotation (Euler angles). Each of the figures show a histogram plot where the X axis represents rotation values, and the Y axis represents the number of times that values attained. Most of the elements can be modeled as a Gaussian, as can be seen. The data has been collected from 1000 samples of the rotation space.

The next experiment we conducted was to verify our theory on the non-Gaussian nature of the measurement process, by comparing a UKF based tracker for homography computation, with a tracker with *exactly the same* process and measurement functions, but employing a particle filter. The UKF and PF implementations were obtained from standard sources like the ReBEL toolkit [56]. In order to test the theory, we first computed a random set of 3D points, over 10 random planes in 3D. We then generated a random camera path, based on Brownian motion, and projected all these points onto images. Some other 3D points were generated in rectangular grids and lines were fit to these points. This corresponded to the line measurements. All the measurements were finally perturbed by Gaussian noise in order to generate the observations.

(a) (b) (c) (d) (e) (f)

Figure 3.6: Image samples of testing the integration-based tracker on the "Panoramic-book" Sequence. A tracker is aligned to the object boundaries (red color).

For the state dynamics, a Brownian motion model was incorporated into both the UKF and the PF, with exactly the same process noise covariance. While the UKF's parameters α, β and κ were set to 10^{-2}, 2 and -6 respectively (these are ideal values, following norms described in Section 2.2.3), only the amount of particles to be generated at each step was defined for the PF to be 200. This is a very minimal set of particles considering that we are sampling a 9 dimensional space. Figure 3.5 compares the performance of the two filters. Observe that the two graphs in this figure are plotted with a log scale in y axis, suggesting that the error in tracking for the UKF is *many orders of magnitude* higher than that of the PF. In our experiments, we typically found that these errors are heightened because of the non-linearity of line measurement model. Even with different variations for the parameters of the UKF, we were unable to obtain any improvement over the presented results. This shows conclusively that our argument for the use of PF for homography based tracking is fully justified.



(a) (b)

Figure 3.5: Comparison between UKF and Particle Filters for tracking a synthetic homography sequence. The sequence itself is generated by projected randomly generated points (on a 3D plane) onto a camera sequence, that follows a straight path. Since the motion model is a Brownian motion, the resultant estimate of the UKF tracker has "jitters". However, as can be seen, Particle Filters are more adept at tracking the homography parameters. (a) shows the norm error in tracking computed as the root mean square error between the tracked homography and the ground truth value for a set of 10 planes. (b) shows the reprojection-error for the sequence.

Having tested the theoretical foundations of our algorithm satisfactorily, we now move towards real world examples for demonstrating the robustness of our approach. Our tracker has been tested

(a) (b) (c) (d) (e) (f)

Figure 3.7: Image samples of testing the integration-based tracker on the "Notebook" Sequence. A tracker is aligned to the object boundaries (red color).

using different video sequences to show the performance over different qualities of texture and edges. As shown in Figure 3.6, the tracker is first aligned manually with the object in the initial frame to provide an estimate of the location of the object. The objective of our tracking is to show results where the tracker faithfully follows the object borders along the sequence, in the presence of various environmental factors. The proposed integration tracker is compared to texture-based one and edge-based one using the same video sequences. The edge tracker is based on the moving edge algorithm and the texture tracker is based on Harris points. The state vector is estimated and tracked probabilistically using particle filter. (Section 3.2). The tracker works approximately on 15-18 frame per second speed using laptop system with 1.6 GH AMD processor and 256 MB RAM memory.

The *"Book-newspaper"* sequence is a video sequence which includes a book with highly textured front face. The book is also surrounded by a textured background. The sequence also contains a considerable amount of motion. In addition, changes in the illumination was introduced with a good amount of shadow. Figure 3.8 shows six sample images from a total 94 images. As can be seen, in the first frame of the sequence, the book is far from the camera and texture points detected using KLT are shown overlaid. Over time, the camera performs the double act of moving towards the object, *while* performing drastic rotation as seen in the last image of the sequence, shown above. The first row (a) of the results shows the results from the edge tracker. Notice how because of the color similarity between the boundary of the book and its interior, the tracker could not help but deviate *into* the book (shown in the middle frames). The second row (b) contains the texture tracker. Here we can clearly see how the initial set of interest points detected on the book diminish over time, although since sufficient texture is present, the texture tracker performs reasonable well with the help of robustly detected points. The observation to note is that at the end of the sequence, the sudden change in illumination causes the texture tracker to be jerked off its trajector. The results of the proposed integration tracker are shown in the last row (c). It can be seen that the integration tracker gives more perfect and precise performance than either of the two trackers, since it combines the complementary advantages of both trackers. In the central images, the texture tracker provides robustness because of the highly textured areas of the book, while at the end, the edge tracker provides robustness due to increased contrast in the image, which heightens image borders and edges. Figure 3.9 shows the edge goodness weight along frames. Observe the final peak in the "good-ness" weights of the edge tracker, at the time when the illumination changes.

Figures 3.7 and 3.6 show the results of testing the integration based tracker using *"Not-book"* and *"Panoramic-book"* sequences respectively. In the former the tracker works properly due to the texture feature availability. The later one skewed a little due to the absence of the texture and the presence of considerable amount of edge shadow.

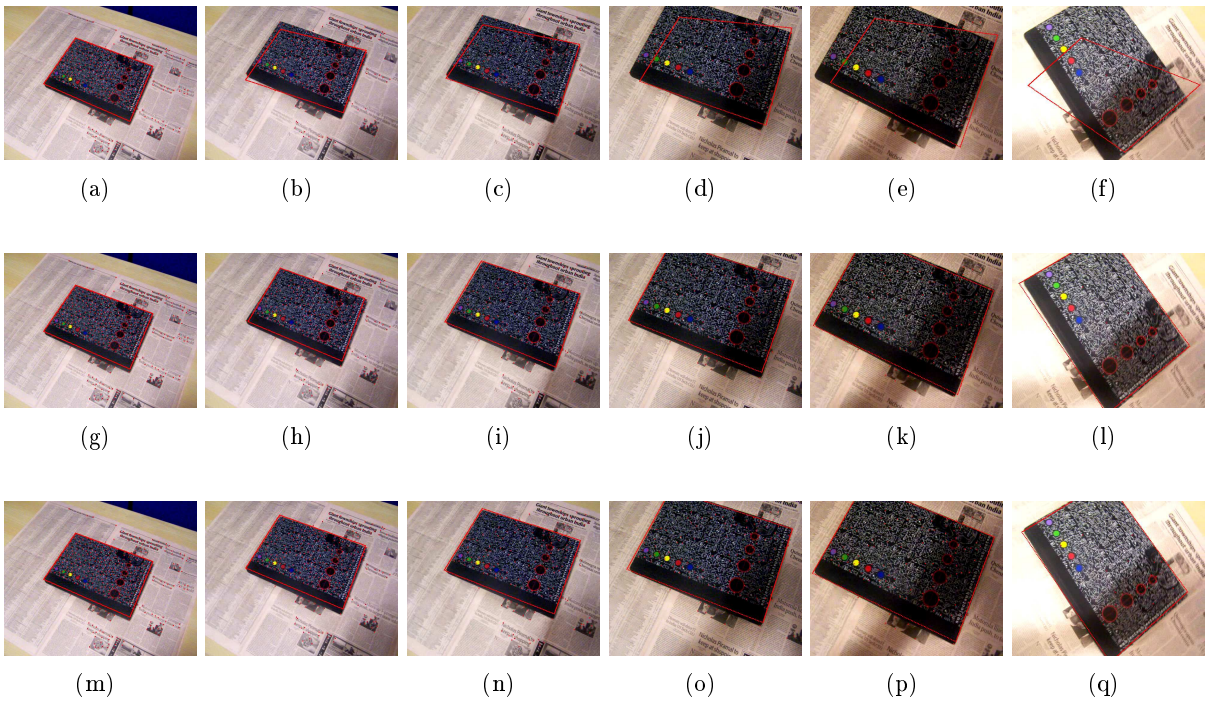


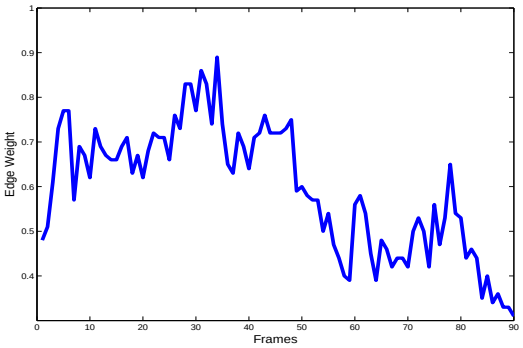
Figure 3.8: Image samples of testing the three edge-based (a-f), texture-based (g-l), and integration-based (m-r) trackers on the "Book-newspaper" Sequence. A tracker is aligned to the object boundaries (red color). The edge tracker failed to follow the object due to shadows and changes in illumination. Texture tracker gives better alignment in case of changes in illumination and motion. The integration-based tracker outperforms the other two.

3.7 Summary

In this chapter, we first looked at the problem of 2D tracking using interest points and edges as measurement, with the tracking framework itself embedded in probabilistic estimation theory. We presented an integration framework that integrates edge and texture points for planar object visual tracking. The integration process is done between two sub-trackers using democratic integration based on goodness weights. The weights are computed with respect to each type of feature adaptively. Selection of "good" features ensures reliability of the tracker under a variety of conditions. On the other hand, robust likelihood models ensure accurate computation of motion. The results show that such a framework helps perform robust tracking in the presence of a large amount of perspective and illumination change.

Secondly, we analyzed the need for a particle filter based approach. We justified the reason that the measurement noise in the current framework is non-Gaussian, and hence the popular line of Kalman filters are unfit for this particular problem. To further the argument, we also compared our approach to the Unscented Kalman Filter, which is best suited to the purposes of tracking in our context among the available Kalman filters. We showed that a particle filter framework has superior performance.

The problem of tracking itself is an inference problem, whose major task is to estimate the position of the object, needed for other tasks like manipulation. In the next chapter, we propose a solution for the 3D counterpart to the current problem, which is *pose tracking*. Together, these



(a)

Figure 3.9: Goodness weights for edges in the "Book-newspaper" video sequence. The decrease in weights explain the failure of the edge tracker.

two chapters represent a tracking framework for estimating the position and orientation of planar, and piece-wise planar (objects made up of planes) objects. Applications of this tracking are in manipulation tasks like visual servoing (Chapter 5).

Chapter 4

Multiple Plane Tracking

4.1 Introduction

Tracking the position of a camera with respect to a static environment is an important part of various applications, a task which is called *localization* in the robotics literature. In Chapter 3, we looked at the application of MVG to tracking the pose in the 2D case, which is represented by a homography. In this chapter, we extend our application to the case where a general 3D estimation of pose is essential. Note here, that pose comprises six parameters: 3 parameters denoting the position, and 3 parameters representing the orientation of the camera.

The determination of the position and orientation of the camera viewing a scene, is of great importance in Computer Vision and related areas [3, 57, 58]. The various applications of Pose Tracking range from Simultaneous Localization and Mapping (SLAM) in Robotics to Structure from Motion (SfM) and subsequently Augmented Reality (AR) in Computer Vision and Graphics. Although commercial camera trackers exist [24], pose tracking is still a challenging problem when the scene and consequently the video experiences changes in illumination, occlusion or related artifacts. At this point, let us note that tracking the pose of an object in the scene and tracking the pose of the camera are equivalent problems: when the scene is stationary with respect to a world coordinate system and only the camera is in motion, the pose of any object in the scene is related to the pose of the camera with respect to the world coordinate system, by a fixed transformation that does not change with time. In this chapter, we would like to address the pose tracking problem from a planar perspective.

As described in Chapter 3, tracking of a single plane corresponds to pose tracking in 2D. A straightforward extension of this problem to 3D is to track multiple planes simultaneously. Tracking multiple planes is a pre-requisite to problems like piecewise-planar reconstructions [59], which normally find applications in architecture modeling [60]. In addition, multi-planar tracking also finds use in Augmented and Virtual Reality [61], since it is the one of the simplest scenarios in which near complete reconstruction of a scene is possible. When a scene consists of multiple planes, it can be shown that the homographies induced by the planes belong to a subspace [62]. Also, from the definition of a plane induced homography, we find that homographies induced by the planes in a scene are inter-related through the pose of the camera in case of rigid scenes. By formulating these relationships as constraints, it is possible to reduce the number of parameters involved in multiple plane tracking, while simultaneously making the process more robust.

Since tracking camera pose and scene planes individually have many advantages, it is highly desirable to consider tracking them simultaneously, given that it is much more convenient to track both together than separately. In this chapter, we investigate a method to track the pose of the

camera by incorporating homographies induced by scene planes into an Unscented Kalman Filter [28] framework. Each individual homography might be computed using robust methods like those in Chapter 3. By combining information from multiple sources, we show how the resulting pose tracking is not only accurate, but also serves to *correct* the underlying homographies in case of errors due to illumination changes, occlusion etc.

The remainder of the paper is organized as follows. Section 4.2 presents works in the literature that our relevant to our case. Section 4.3 outlines the proposed tracking framework to incorporate multiple cues, and motivates the use of the Unscented Kalman Filter [28] for the task at hand. Section 4.4.4 describes the UKF approach for tracking multiple planes simultaneously by keeping track of the pose and plane normals. Experimental results on challenging sequences (Section 4.5) under varying conditions of pose and illumination show the robustness of the tracker. Finally, we summarize with elucidation of future avenues for research in Section 4.6.

4.1.1 Notation

In the remaining sections, we use the following notation. Capitals like (R, F) denote the rotation matrix and feature set respectively. Since the probability theory continues from the previous chapter p is used to represent probability, and thus $\mathbf{p}$ is used to represent the pose of the camera. Λ represents covariances, while notations like $\hat{}$ are used to represent estimated quantities like the previous chapter. Since many notations have been borrowed from works like feature uncertainty [9] or convex optimization [63], notations at these places differ from the standard conventions. In such cases, these notations have been elaborately explained.

4.1.2 Contributions

,

In the previous chapter, we saw how a Bayesian filtering framework could be used to integrate multiple measurements like edges and textures into a tracking framework for a single plane. In this chapter, we extend this formulation to the case where *multiple* planes need to be tracked. In such a case, we show that the interlinking between the plane and pose parameters that results in the various homographies across views can be used to robustly and efficiently track multiple planes. Specifically, our contributions in this chapter include

- A tracking framework that formulates the problem of tracking multiple planes as tracking the pose of the camera coupled with the normals of the individual plane parameters. This is followed by a derivation of the measurement noise based on theories of feature uncertainties [8].
- A set of heuristics that provide robustness to the system in the presence of varying conditions like illumination, occlusion etc.
- Two solutions for the initialization of the tracker with one providing a *globally* optimal minima, something that has never been done before.

4.2 Related Work

Setting the problem of pose recovery in a piece-wise planar scene offers multiple advantages both in terms of the modeling of the problem as well as in practical application of the algorithms developed. A piece-wise planar model of a scene is a model that assumes that the *entire* environment being

tracked is made up of a set of planes. This is a very valid assumption when dealing with scenarios like urban environments. Planar structures and related geometric quantities form some of the most studied areas of MVG [62, 64, 65], and hence both the understanding of the homography space and methods for robust computation are well studied areas. In [62], the authors prove that the space of all the homographies relating planes in two images can be restricted to a subspace that has a rank of 4 at best. In [64], the authors derive linear subspace constraints on the relative homographies of multiple (≥ 2) planes across multiple views. They first show that the collection of all relative homographies (homologies) of a pair of planes across multiple views, spans a 4-dimensional linear subspace. This constraint can be extended to the case of multiple planes across multiple views and even for a single plane across multiple views. In short, many of the constraints relating homographies across multiple views and planes are well known and can be used in tracking efficiently, for example, to constrain the search space for a particular plane in an image, given the location of the other planes in that image. From the computational perspective, extremely robust solutions also exist to compute the homography induced by a plane in two cameras [18], and while computation of robust Fundamental Matrices [66] still remain a tricky issue, homographies are comparatively simpler to compute accurately (A Fundamental Matrix can be thought to be a *non-planar* counterpart of the homography, applicable to generic 3D scenes). However, like the Fundamental Matrices, homographies can also be expressed as a function of the camera pose, and can be decomposed in a similar manner [10, 67]. Given that a piece-wise planar environment is an acceptable model for a wide class of scenarios, and that algorithms for automatically 'recognizing' planes in a video exist [68], we believe that a Homographic framework shows a lot of potential.

With the advent of constraints relating homographies over multiple views, tracking of multiple planes in a scene has attracted interest [69, 70]. Applications include pose estimation [58] for visual slam and augmented reality [71]. The main advantage of tracking multiple planes is the fact that, under the assumption of a static rigid scene, various constraints based on the scene geometry may be added to simplify the tracking formulation [70] or to increase the accuracy of tracking [69]. Additionally tracking multiple planes is much more robust to occlusion and illumination change since information about the pose of the scene obtained from planes that are tracked accurately might be used to obtain accurate motion information about planes that are not tracked well.

In [69], the authors propose methods to track the full camera position and orientation from intensity differences measured in a video sequence. The camera pose is calculated based on plane equations relating the homography induced by a plane and the pose of the camera, thus avoiding dependency on point correspondences. The plane based formulation also allows additional constraints to be naturally added, e.g. perpendicularity between walls, floor and ceiling surfaces, co-planarity of wall surfaces etc. The resultant system is similar to ours, in the sense that multiple planes are related through the homography to the pose of the camera, and so the only dynamic factor in the tracking is the change in pose. Thus, tracking multiple planes is reduced to tracking the pose of the camera *assuming the plane parameters are known before hand.* Tracking is done using an inverse compositional approach that uses a gradient descent optimization function over the pose space for the tracking. Since, the pose change of the camera between consecutive frames is minimal, a local minima obtaining method like gradient descent suffices. However, the drawback of their approach is that since they do not rely on feature correspondences but on techniques closer to image registration, their method cannot be sufficiently robust to changes in illumination.

In [70], the authors make use of the alternate fact that homographies induced by multiple planes between two views of a camera are embedded in a lower dimensional space [62]. By posing the problem of multiple plane tracking as a function of this lower dimensional space's parameters, the tracking of multiple planes is done simultaneously. Although they propose that this method is robust to both occlusion and illumination change, there is still a drawback because their method is based

on minimizing intensity differences between images using a Bundle Adjustment like minimiation procedure. Since difference in intensity is the measurement, a change in illumination beyond a point will cause the *entire tracker* (*i.e.* tracking of all the planes) to deviate, even if illumination change affects only one plane of the several present. A similar case can be argued with the occlusion claim.

Since a by-product of our tracking is the pose of the camera, some of the past works in pose estimation are releveant to us. For example in [3], the authors propose an algorithm based on Extended Kalman Filters to track the pose of the camera. KLT [8] features are detected and tracked through normalized cross correlation. These tracked features are then used as measurements to estimate the pose of the camera, using the standard pinhole projection equations (Section 1.2.1). The by-product of their process is an estimate of the 3D structure of the feature points. To further enhance the accuracy of the system, the depth of these tracked feature points is measured using particle filters, and the resultant information is pumped back into the Kalman Filter to reduce the error covariance of the pose over time. Our current chapter takes the same analogy of Kalman Filters, but introduces an *intermediate* level where a *mid-level* primitive like homography is computed and subsequently used as measurement. This provides additional robustness to illumination and occlusion, which the algorithm in [3] lacks. Alternatively, the authors of [58] use the relationships of [62] to construct a set of linear equations with pose as the unknowns. These equations are then solved using singular value decomposition to obtain the least squared pose estimate for every frame.

4.3 Tracking Framework

In Chapter 3, we looked at how a Particle Filter framework could be used to track a single frame by integrating measurements from multiple features like edges and cues. In the following sections, we introduce a complementary framework that allows tracking of multiple *planes* using a single feature. Although one solution is to individually track the planes using the method as suggested in Chapter 3, we wish to add robustness to the tracking process, by borrowing information not only from different features, but also from different *planes*. This is particularly useful when one of the planes has very good features and hence can be tracked very accurately. This accuracy, as we show later in this chapter, can be trasferred through the camera pose to obtain accurate tracking of *other* planes in the scene, which may not have such good features on them.

In the following sub-sections, we first develop a motion model for the tracking process, along with a measurement model that takes as input homographies computed using interest points tracked over time. By incorporating camera pose as one of the parameters to be estimated, we show how cues from *several* planes can be interlinked to improve the tracking of each of them, over individual plane tracking approaches. We believe that this contribution together with the contribution of the previous chapter lays the ground work for the generic task of tracking *multiple planes using multiple features,* which results in a robust tracker for scenes with multiple planes.

4.3.1 Motion Model: 3D

Assume that we have m planes being tracked using feature points, and at any time instant t we wish to estimate the homographies with respect to each of these planes. Thus we have multiple instances of homography $H_t = \{h_t^1, \ldots, h_t^m\}$ that may be computed from point correspondences, where h_t^i represents homography related to the i^{th} plane. Let us recall that the homography relating two views of a plane is related to the geometry of the views through the following relation (Equation 1.5)

$$h_t^i \sim K \begin{bmatrix} R_t^i & \frac{t_t^i n^{i\top}}{d^i} \end{bmatrix} K^{-1} \tag{4.1}$$

where $\begin{bmatrix} R_t^i \; t_t^i \end{bmatrix}$ represents the relative pose between the first and t^{th} views, n^i represents the plane normal at distance d from the coordinate system attached to the first view, and K represents the internal parameters of the camera.

Unlike the previous chapter, in which no assumption is made about the rigid-ness of a scene, this chapter requires us to make the assumption that the scene is rigid, since only then is the above equation valid *across* all homographies. Like the previous chapter however, we assume here that the internal parameters of the camera, K are known. Since all the homographies are related through the pose of the camera, its advantages to consider the state vector consisting of the pose and the plane parameters, which is much more reduced in number than a concatenation of the individual homographies. Also, a common parameter like pose ensures that measurements from multiple planes may be fused together. We thus define the state vector to be estimated at any instant t as

$$\mathbf{s}_t = \begin{bmatrix} \mathbf{p}_t & \mathbf{n^1}_t & d_t^1 \ldots & \mathbf{n^m}_t & d_t^m \end{bmatrix} \tag{4.2}$$

$$\mathbf{p}_t = \begin{bmatrix} r_t^1 & r_t^2 & r_t^3 & t_t^1 & t_t^2 & t_t^3 \end{bmatrix} \tag{4.3}$$

$$\mathbf{n^i} = \begin{bmatrix} n^{i_1} & n^{i_2} & n^{i_3} \end{bmatrix}, \quad \| \mathbf{n^i} \| = 1 \tag{4.4}$$

where $\mathbf{p}_t$ represents the current estimate of pose. Here the rotation is represented using Euler angles. Thus for m planes a total of $3m + 7$ parameters need to be tracked (plane normals are represented by only 2 parameters because of the norm constraint). Thus if $\hat{\mathbf{s}}_t$ represents our current estimate of the state vector, then change owing to inter-frame displacement can be written as

$$\hat{\mathbf{s}}_t = \hat{\mathbf{s}_{t-1}} + \Delta\hat{\mathbf{s}}_t. \tag{4.5}$$

following a Markov process and *Brownian* motion model, as explained in Chapter 3.

Given the state vector outlined above, it is fruitful to argue about the measurement model of the tracker. At any instant, based on the value of the state parameters, multiple *homography hypotheses* can be generated in this model. These can then be used to transfer interest points between frames, and a likelihood model like Section 3.3.3 could be applied to obtain measurements needed for the filter. We choose to deviate from this approach for two reasons. The first is that this measurement process is computationally costly, since for each hypothesis, hundreds of feature points have to be transferred and their respective errors measured. Secondly, in the presence of robust homography estimation algorithms [9] that are reasonably fast (though not accurate), it makes less sense not to make use of them.

Thus in our current model, we use homographies computed from the interest points as measurements. Given that at each iteration of the algorithm, we estimate the homographies induced by each plane as $\hat{H} = \{\hat{h_t^1}, \ldots, \hat{h_t^m}\}$ the estimate of the state vector $\hat{\mathbf{s}}_t$ can be calculated by minimizing the error function

$$\mathcal{G}(\hat{\mathbf{s}}_t) = \mathcal{F}(\hat{H}_t - H_t) \tag{4.6}$$

where $\mathcal{F}$ is a suitable distance measure, and H_t may be obtained by the robust algorithm outlined in Section 3.2. The optimal value is now given by

$$\hat{\mathbf{s}}_t = \arg\min_{\hat{\mathbf{s}}_t} \mathcal{F}(\hat{H}_t - H_t) \tag{4.7}$$

Any tracking algorithm requires the modeling of noise estimates in the state dynamics. In our case, the two types of errors: process noise and observation noise, correspond to errors in the estimation of pose, and errors in the estimation of homographies. It has been observed earlier that a Gaussian assumption for process noise is suitable for tracking the pose of the camera [3]. For homographies, the assumption of a Gaussian noise source is only an approximation of the true noise [9], to the first order. It is however, desirable to represent both noises using a Gaussian model because

- In practice, a Gaussian approximation for noise in homography has been found to be a useful model for cases when the underlying images are captured from similar view points (like successive frames of a video).
- a Gaussian model allows us to employ the widely used class of Kalman filters, that are a close approximation to the Bayesian filter, which is considered the most optimal estimator of states, given data.

Since the state transition and observation functions are highly non-linear in nature (Equation (4.1)), and since a first order approximation to the observation noise is used, the Extended Kalman Filter (EKF) seems to be a poor choice for the filtering. It is for these reasons that an Unscented Kalman Filter [28] is employed in the current work.

4.3.2 Bayesian Tracking

We now present the Bayesian filter formulation for computing pose and plane parameters. The filter is similar to the one outlined in Chapter 3, with the only difference coming in the modeling of noise in the process, and consequently the choice of filter for the estimation process.

Let h_t represent the state variable to be estimated at the current iteration of the algorithm. Let $\pi(h_t)$ be the belief of the random vector h_t at time t represented by posterior probability $p(h_t \mid F_{1,\ldots,t})$ based on features $F_{1,\ldots,t}$. Expanding using Bayes rule

$$p(h_t \mid F_{1,\ldots,t}) = \frac{p(F_t \mid h_t)p(h_t \mid F_{1,\ldots,t-1})}{p(F_T \mid F_{1,\ldots,t-1})}. \tag{4.8}$$

which reduces to

$$p(h_t \mid F_{1,\ldots,t}) \propto p(F_t \mid h_t)p(h_t \mid F_{1,\ldots,t-1}) \tag{4.9}$$

considering $p(F_T \mid F_{1,\ldots,t-1})$ to be a constant.

Unscented Kalman Filter The representation of pose and plane parameters as a Gaussian Random Variable (GRV), allows us to use an Unscented Kalman Filter which produces the minimum mean squared error estimate at each iteration. In the case of tracking multiple planes, we assume our measurements (homographies) to be independent both over time and from each other.

Thus,

$$p(F_t \mid h_t) = \prod_{i=1}^{m} p(F_t^i \mid h_t) \tag{4.10}$$

The prior $p(h_t \mid F_{1,\ldots,t-1})$ can be modeled to account for the motion of the camera by defining it as a Gaussian variable centered around the state estimate of the previous representation, with the

assumption that the underlying process is Markovian. This model is in fact the representation of a simple Brownian motion

$$\begin{aligned} p(h_t \mid F_{1,\ldots,t-1}) &= p(h_t \mid h_{t-1}) p(h_{t-1} \mid F_{1,\ldots,t-1}) \\ &= \mathcal{N}(h_{t-1}, R) \end{aligned} \tag{4.11}$$

Alternatively, the state vector can also be augmented with a velocity and acceleration model [72] but in the current implementation, we have considered a Brownian motion model for simplicity.

Having defined the process and measurement models (equivalently the prior and likelihood functions in a Bayesian filtering terminology), we now turn to the noise models, which serve to constrain the prior and likelihood functions to a reduced region in the state space. Since the process noise is determined to be a GRV (reasons for which had been outlined earlier in this section), we now turn to the description of the measurement noise.

4.3.3 Cues From Multiple Planes

In case of multiple planes, the likelihood model is implicitly defined in the Unscented Kalman Filter algorithm [28]. However, one of the main factors contributing to the likelihood is the measurement noise covariance. In order to represent the true process faithfully, we need a noise covariance that is close to the true covariance in the measurement process. Assuming that the tracked feature points are perturbed with Gaussian noise, first order uncertainty analysis may be used to measure the variance of the computed homography as a function of the variance of the individual features [9]. We thus define the measurement noise, which is the covariance of the measured homography as

$$\Lambda_h = JSJ \tag{4.12}$$

$$J = -\Sigma_{i=1}^{n} \frac{u_k u_k^\top}{\lambda_k} \tag{4.13}$$

where (u_k, λ_k) are the eigenvectors and eigenvalues of the matrix A in the objective function $Ah = 0$, which is solved using singular value decomposition to obtain the least square homography solution h.

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 & -X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 & -Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 & -X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nX_n & -y_nX_n & -X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nY_n & -y_nY_n & -Y_n \end{bmatrix} \tag{4.14}$$

$$h = \min_h \|Ah\| \tag{4.15}$$

where (x_i, y_i) and (X_i, Y_i) are corresponding points. The value of S is obtained from A and the covariances of the individual features [9], which represent the uncertainty in the feature matching process. In order to define the uncertainty in the feature matching process we observe that its accuracy depends on the intensity pattern around each feature, which often has strong directionality and is location-dependent. Such directional uncertainty can be characterized by the following inverse covariance matrix, modeled as a function of the image gradients in their local neighbourhoods [73].

Thus we define uncertainty in the feature matching process for each interest point as a covariance computed as

$$\Sigma_{x_i,y_i} = \begin{bmatrix} \frac{\partial^2 I(x_i,y_i)}{\partial x^2} & \frac{\partial^2 I(x_i,y_i)}{\partial x \partial y} \\ \frac{\partial^2 I(x_i,y_i)}{\partial x \partial y} & \frac{\partial^2 I(x_i,y_i)}{\partial y^2} \end{bmatrix}^{-1} \tag{4.16}$$

where $I(x_i, y_i)$ represents the image under consideration. The covariance is computed over a small window around the matched point features. A more accurate method to estimate feature position uncertainty is presented in [74], which takes into account not only the image pattern but also the image pixel noises. But for this investigation, the simple formulation expressed above suffices.

4.4 Single-Feature Multi-Plane Tracking

Given the process and measurement noise (Equations 4.11,4.12) we now define the process and measurement functions that are used in the Unscented Kalman Filter [28], under the assumption that m planes are being tracked.

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathcal{N}(0, R) \tag{4.17}$$

$$\begin{bmatrix} h_t^1 & \dots & h_t^m \end{bmatrix} = o(\mathbf{s}_t\ ,\ v_t) \tag{4.18}$$

$$v_t = \begin{bmatrix} \Lambda_{h_t^1} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \Lambda_{h_t^m} \end{bmatrix} \tag{4.19}$$

The observation function o is defined as Equation (4.1). In order to initialize the filter, a rough estimation of the plane normals is obtained by decomposition of the homography between the first two frames for each plane [10, 75]. The whole algorithm is summarized in Algorithm 2. Figure (cite reference) illustrates the whole process.

We now discuss important implementation issues like the parametrization of the observations (homographies), normalization of the image coordinates used to compute the homography, initialization of the UKF and finally coordinate normalization.

4.4.1 Initialization

Traditional methods for obtaining the camera pose and plane normals from the homography matrix rely on the Singular Value Decomposition (SVD) of homography to provide solutions [10, 55]. In both methods, eigenvalues of either the homography matrix H or $\mathrm{H}^\top\mathrm{H}$ are used to get upto 8 solutions for $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$ and then 6 solutions are weeded out based on many constraints. Finally, the 2 remaining solutions may be disambiguated by either considering a third view or a second plane.

Faugeras SVD-based decomposition

The algorithm starts with the singular value decomposition of the homography matrix, followed by the equation relating the diagonal matrix thus produced to a new set of variables as

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top \tag{4.20}$$

$$\mathbf{\Lambda} = \mathbf{R}_{\mathbf{\Lambda}} + \mathbf{t}_{\mathbf{\Lambda}}\mathbf{n}_{\mathbf{\Lambda}}^\top \tag{4.21}$$

Algorithm 2 The UKF tracking algorithm

Extract features for first frame, segment into planes.
Track features to second frame, compute homographies and decompose to obtain (R, t, n, d).
Pass to coordinate normalization function (Section 4.4.3).
Compute covariances of features.
Initialize UKF tracker.
for $(k = 3, \ldots, n)$ **do**
  Extract features in current frame (KLT Tracker [8]).
  Normalize and compute homography (Section 4.4.2).
  Transfer covariance from initial frame to frame $k-1$ (Section 4.4.2).
  Compute homography covariance (Section 4.3.3).
  Assign robustness weights (Section 4.4.4). Discard homographies with large covariances.
  Pass measurements (homographies) and covariances to UKF tracker and obtain best estimate of the current pose and plane parameters.
  (Optional) Compute *corrected* homographies through the measurement function model.
  (Optional) Correct the underlying correspondences produced by KLT.
end for

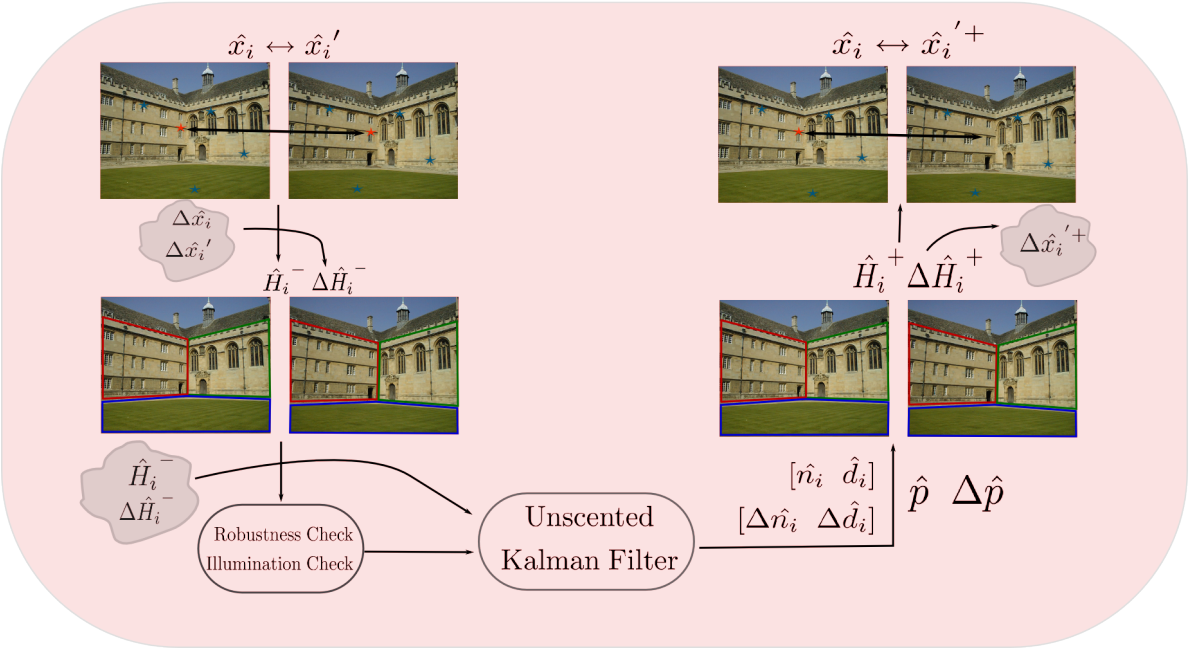


Figure 4.1: One step of the UKF based algorithm with covariance transfer.

Computing the components of the rotation matrix, translation and normal vectors is simple when the matrix being decomposed is a diagonal one. First, $\mathbf{t}_\Lambda$ can be easily eliminated from the three vector equations coming out from (4.21) (one for each column of this matrix equation). Then, imposing that $\mathbf{R}_\Lambda$ is an orthogonal matrix, we can linearly solve for the components of $\mathbf{n}_{\mathbf{\Lambda}}$, from a new set of equations relating only these components with the three singular values (see [10] for the detailed development). As a result of the decomposition algorithm, we can get up to 8 different solutions for the triplets: $\{\mathbf{R}_{\mathbf{\Lambda}}, \mathbf{t}_{\mathbf{\Lambda}}, \mathbf{n}_{\mathbf{\Lambda}}\}$. Then, assuming that the decomposition of matrix $\mathbf{\Lambda}$ is done, in order to compute the final decomposition elements, we just need to use the following expressions:

$$\mathbf{R} = \mathbf{U}\,\mathbf{R}_{\mathbf{\Lambda}}\,\mathbf{V}^\top \tag{4.22}$$
$$\mathbf{t} = \mathbf{U}\,\mathbf{t}_{\mathbf{\Lambda}} \tag{4.23}$$
$$\mathbf{n} = \mathbf{V}\,\mathbf{n}_{\mathbf{\Lambda}} \tag{4.24}$$

Zhang SVD-based decomposition

Zhang *et. al* [67] take a different approach by first computing the eigenvalues of $\mathbf{H}^\top\mathbf{H}$, and then using it for further computation of the quantities $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$.

$$\mathbf{H}^\top\,\mathbf{H} = \mathbf{V}\,\mathbf{\Lambda^2}\,\mathbf{V}^\top \tag{4.25}$$
$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3) \tag{4.26}$$
$$\mathbf{V} = [\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}] \tag{4.27}$$
$$\lambda_1 \geq \lambda_2 = 1 \geq \lambda_3 \tag{4.28}$$

In the first step, values of $\{\mathbf{t}^*, \mathbf{n}\}$ are computed, where $\mathbf{t}^*$ is the normalized translation vector. Subsequently, the rotation matrix is obtained as $\mathbf{R} = \mathbf{H}\,(\mathbf{I} + \mathbf{t}^*\,\mathbf{n}^\top)^{-1}$. Eight solutions are obtained in the following manner

$$\mathbf{t}^* = \pm\frac{\mathbf{v'_1} \pm \mathbf{v'_3}}{\zeta_1 \pm \zeta_3} \tag{4.29}$$
$$\mathbf{n} = \pm\frac{\zeta_1\mathbf{v'_1} \pm \zeta_3\mathbf{v'_3}}{\zeta_1 \pm \zeta_3} \tag{4.30}$$

where equations in numerators and denominators share the same sign in all variations. The variables $\{\mathbf{v'_1}, \mathbf{v'_3}, \zeta_1, \zeta_3\}$, are functions of the eigenvalues $\mathbf{\Lambda}$ [67].

4.4.2 Data Normalization and Covariance Transfer

Two of the important steps in this algorithm are the normalization of image correspondences before the computation of the homography from Equation (4.15), and the transfer of covariance of the image features across a homography from initial frame to the previous frame. A typical operation involved in this transformation is the conversion from homogeneous to in-homogeneous coordinates for the transferred feature points resulting in a change in the estimated/transferred covariance. We describe the equations involved in these transformations below

Data Normalization

It has been established very early [76] that numerical conditioning of the input variables is important for an SVD like solution described in Equation 4.15. This argument has been further enhanced in [77]. The algorithm in [76] has 3 distinct steps: 1) Apply an initial affine transformation to each of the image points in order to normalize them. 2) Compute the resulting homography using equation 4.15. 3) Apply the inverse transformation to the compute homography. Let (x_i, y_i) and (X_i, Y_i) be the points involved (un-normalized). Let H be the homography that takes (x_i, y_i) to (X_i, Y_i). Let $(x_{i,}, y_i)$ be normalized by T_1 to get (x_i^{est}, y_i^{est}) and (X_i, Y_i) be normalized by T_2 to get (X_i^{est}, Y_i^{est}). Then the estimated homography H_{est} is related to H as $H = \mathrm{inv}(\mathrm{T}_2) * H_{est} * \mathrm{T}_1$, which can be re-cast as an affine transformation of H_{est}. The resulting covariances are then estimated as follows.

$$\begin{aligned} \Lambda_{\mathbf{x}}^{est} &= \mathbf{J}_{xest} \Sigma_{\mathbf{x}} \mathbf{J}_{xest}^{\top} && (4.31) \\ \mathbf{\Lambda}_{\mathbf{X}}^{est} &= \mathbf{J}_{Xest} \Sigma_{\mathbf{X}} \mathbf{J}_{Xest}^{\top} && (4.32) \\ \mathbf{\Lambda}_{H} &= \mathbf{J}_{H} \Lambda_{Hest} \mathbf{J}_{H}^{\top} && (4.33) \end{aligned}$$

where $\mathbf{J}_{xest} = \partial \mathbf{x}^{est} / \partial \mathbf{x}$, $\mathbf{J}_{Xest} = \partial \mathbf{X}^{est} / \partial \mathbf{X}$, $\mathbf{J}_H = \partial H / \partial H_{est}$, and Λ_{Hest} is computed from equation 4.12. The above equations are actually approximations found by a Taylor expansion of the corresponding functions. However, since the underlying transformations are affine in nature, these approximations are exact. The effect of normalization on the estimated homography parameters is shown in Figure 4.6.

Covariance Transfer

Once the pose estimate for the current frame has been established, the measurements (homographies) are corrected in order to ensure that the feature matching process for detecting features in the next frame is initialized properly [8]. Assume that the features in the previous frame are obtained as $\mathbf{x}' = \hat{\mathbf{H}}\mathbf{x}$, where $\hat{\mathbf{H}}$ represents the estimated measurement using the UKF's measurement noise estimation process [28]. Given that the estimated homography has variance $\Lambda_{\hat{\mathbf{H}}}$, and the features in the first frame have variance $\Lambda_{\mathbf{x}}$, the variance of the point $\mathbf{x}'$ may be obtained as [9]

$$\begin{aligned} \Lambda_{\mathbf{x}'} &= \begin{bmatrix} \mathbf{B} & \vdots & \hat{\mathbf{H}} \end{bmatrix} \begin{bmatrix} \Lambda_{\hat{\mathbf{H}}} & \vdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \vdots & \Lambda_{\mathbf{x}} \end{bmatrix} \begin{bmatrix} \mathbf{B}^{\top} \\ \cdots \\ \hat{\mathbf{H}}^{\top} \end{bmatrix} && (4.34) \\ \mathbf{B} &= \begin{bmatrix} \mathbf{x}^{\top} & \mathbf{0}^{\top} & \mathbf{0}^{\top} \\ \mathbf{0}^{\top} & \mathbf{x}^{\top} & \mathbf{0}^{\top} \\ \mathbf{0}^{\top} & \mathbf{0}^{\top} & \mathbf{x}^{\top} \end{bmatrix}_{3 \times 9} && (4.35) \end{aligned}$$

Finally, in order to convert homogeneous coordinates to in-homogeneous coordinates, a scaling operation is imposed on the transferred points $\mathbf{x}'_{3\times 1}$, where each point is divide by its 3rd coordinate to make it 1. The corresponding change in the covariance matrices is given by [9]

$$\begin{aligned} \Lambda_{\mathbf{x}'}^{2\times 2} &= \nabla f \Lambda_{\mathbf{x}'} \nabla f^{\top} && (4.36) \\ \nabla f &= 1/W^2 \begin{bmatrix} W & 0 & -X \\ 0 & W & -Y \end{bmatrix} && (4.37) \end{aligned}$$

where $\mathbf{x}' = [X\ Y\ W]^\top$. Since this function is a perspective function, and the above Equation (4.36) is a first order approximation of the Taylor series expansion around the in-homogeneous point, the approximation can be poor for large values of $(\frac{X}{W}, \frac{Y}{W})$, which means an appropriate normalization like in Section 4.4.2 might be required to obtain a better performance. Figure 4.7 illustrates this effect.

4.4.3 Coordinate Normalization: Two solutions

Each decomposition by the algorithms of Faugeras[10] and Zhang[67] produce estimates of $\{\mathbf{R}, \mathbf{t}, \mathbf{n}\}$ assuming a coordinate system in which the perpendicular distance between the origin and the plane in consideration is 1. Since we consider all the homographies computed with respect to a fixed reference frame, the origin in all the decompositions obtained is the same. Thus the difference in the various solutions obtained by SVD decomposition only differ in a scale factor. We call this process "coordinate normalization".

In the ideal scenario, it would be easy to eliminate this scale factor by comparing two estimates of the relative translation of a view given by decomposing homographies induced by two planes. However, in the presence of errors an optimized estimate of the scale is essential. There are two approaches to this problem.

Let us assume that m planes exist, with the perpendicular distance of the jth plane being denoted by d^j. Now, let the solutions of translation obtained by decomposition methods be denoted by t^j, which is the translation vector obtained by decomposing the homography H^j. Thus the actual translation vector is represented by $t = t^j d^{*j}$, where d^{*j} is the optimum of an objective function. Since, estimates obtained from the various planes must converge, we are interested in the optimum values $\left[\ d^{*1}, d^{*2}, \ldots, d^{*m}\ \right]$ such that

$$\left[\ d^{*1}, \ldots, d^{*m}\ \right] = \min \sum_{l=1}^{m} \sum_{j=1}^{m} \|t^j d^j - t^l d^l\|_2^2 \tag{4.38}$$

The above function is quadratic and can be reduced to the form $\|\mathbf{Ax}\|_2$ in the following manner

$$\begin{aligned}
\left[\ d^{*1}, \ldots, d^{*m}\ \right] &= \min \sum_{l=1}^{m} \sum_{j=l+1}^{m} \|t^j d^j - t^l d^l\|_2^2 && (4.39)\\
&= \min \sum_{l=1}^{m} \sum_{j=l+1}^{m} \| \begin{bmatrix} t_1^j & t_1^l \\ t_2^j & t_2^l \\ t_3^j & t_3^l \end{bmatrix} \begin{bmatrix} d^j \\ d^l \end{bmatrix} \|_2^2 && (4.40)\\
&= \min \| \begin{bmatrix} t_1^1 & t_1^2 & 0 & 0 & 0 & 0 & 0 \\ t_2^1 & t_2^2 & 0 & 0 & 0 & 0 & 0 \\ t_3^1 & t_3^2 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & t_1^{m-1} & t_1^m \\ 0 & 0 & 0 & 0 & 0 & t_2^{m-1} & t_2^m \\ 0 & 0 & 0 & 0 & 0 & t_3^{m-1} & t_3^m \end{bmatrix} \begin{bmatrix} d^1 \\ \vdots \\ d^m \end{bmatrix} \|_2^2 && (4.41)\\
&= \min \|\mathbf{Ax}\|_2 && (4.42)
\end{aligned}$$

which is convex only when $\mathbf{A}$ is positive definite. Although, a least squares solution can be obtained using an SVD (it is well known that a minima of the objective function $\|\mathbf{Ax}\|$ is obtained as the

eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^\top\mathbf{A}$ with the additional condition that $\|\mathbf{x}\| = 1$ which is not a problem since the reconstruction is anyway upto scale), a better solution may result from a Convex reformulation. For this task we introduce a new set of variables (t^*) which represent the *actual* translation of the view upto scale. The modified functions now become $f_{,j}(t^*, d^j) = \|t^* - t^j d^j\|_2$ and we wish to minimize the sum of such functions. Looking closely at f_j we find

$$\begin{aligned} f_{,j}(t_i^*, d^j) &= \|t^* - t^j d^j\|_2 && (4.43)\\ &= \| \begin{bmatrix} t^* & d^j \end{bmatrix} \begin{bmatrix} 1 \\ -t^j \end{bmatrix} \|_2 && (4.44) \end{aligned}$$

Equation (4.44) represents the composition of two functions. The first function is the Euclidean norm, while the second is a an affine function. The parameter vector is $x_{,j} = \left[(t^*)^\top \; d^j\right]^\top$. We can collect all such functions into a common framework.

$$\begin{aligned} x &= \left[(t^*)^\top, d^1, \ldots, d^m\right]^\top && (4.45)\\ g_{,j}(x) &= \|(A_j x + b)\|_2 && (4.46)\\ & x \in \mathbb{R}^{3m+k}, && (4.47)\\ & A_j \in \mathbb{R}^{j,3m+k}, b = \mathbf{0}_{1,1} \end{aligned}$$

The functions g_j in Equation (4.46) represents a set of convex function in the variables [78]. In order to minimize it, we finally introduce another variable γ that puts a bound on the maximum value taken by each of the convex functions. This results in a function to be minimized with convex constraints, defined by the following set of equations

$$\begin{aligned} \min_x \quad & \gamma && (4.48)\\ \text{such that} \quad & g_j(x) = \|A_j x + b\|_2 < \gamma && (4.49)\\ & x \in \mathbb{R}^{3m+k}, && (4.50)\\ & A_j \in \mathbb{R}^{j,3m+k}, b = \mathbf{0}_{1,1} && (4.51) \end{aligned}$$

There are excellent mathematical packages like SeDuMi[63] for obtaining *globally optimal* solutions for such cases.

4.4.4 Robustness

The idea of robustness is important to a tracking algorithm like Kalman filter, since it includes all measurements at every instant in its prediction of the state vector. Thus, although uncertainties are handle with a theoretical inclusion of the error covariance in the estimation process, outliers in the data or input do not have a representation in the estimation process. Thus, such a tracker will be sensitive to bad estimates of homography that are produced when there are very few feature correspondences. The solution to this problem is to *detect* and *exclude* such measurements from the estimation process, thereby obtaining a double advantage: (i) By not including outlier measurements, we ensure an accurate estimate of the state vector, and (ii) With an accurate estimate of the state vector, we can obtain an accurate *replacement* of the outlier measurements, by percolating

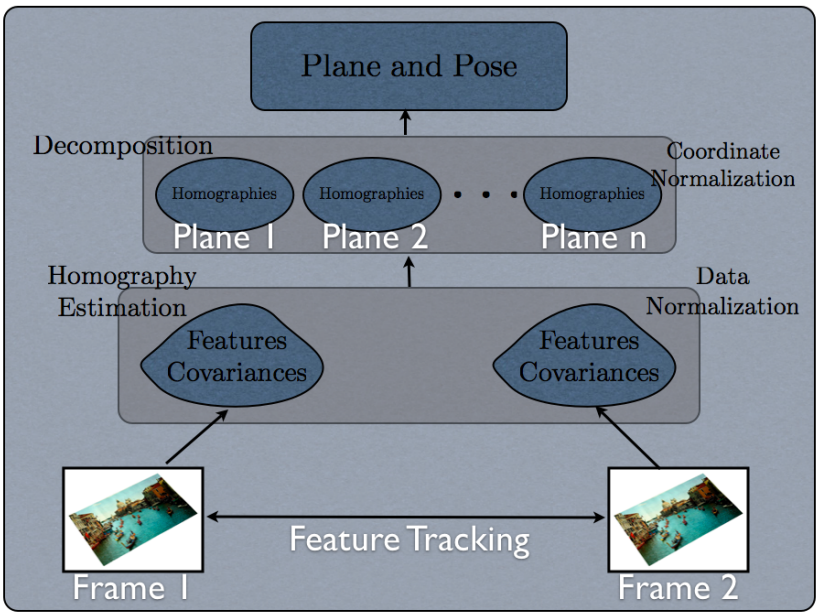


Figure 4.2: Initialization of the UKF tracker.

the state vector through the measurement process. As described in Section , this serves to not only estimate but also *correct* erroneous homographies.

A bad estimate of the homography may be obtained due to either incorrect feature correspondences, or due to bad conditioning of the underlying homography estimation process which may be the result of a smaller set of feature correspondences than desired.

- Incorrect feature correspondences may be of two types. When, there is a *reasonable* error (say 2-3 pixels), between correspondences across views, such an error may be accounted for in the covariance estimation process of the homography (Section). In order to get an estimate of *how reasonable* a particular feature correspondence is, we use the *Normalized Cross Correlation function*.

$$ncc(x_1, x_1') = \frac{\sum_{x \in w_{x_1}, y \in w_{x_1'}} (I_1(x) - \overline{I_1(x)}) \cdot (I_2(y) - \overline{I_2(y)})}{\sqrt{\sum_{x \in w_{x_1}} (I_1(x) - \overline{I_1(x)})^2 \sum_{y \in w_{x_1'}} (I_2(y) - \overline{I_2(y)})^2}} \tag{4.52}$$

where $\cdot$ represents element wise multiplication, w_{x_1} represents the immediate neighbourhood surrounding the feature at position x_1 in the first image I_1 and likewise for the corresponding feature and image. $\overline{I_1(x)}$ represents the mean of the image (grayscale) taken in the neighbourhood w_{x_1}. We can *redefine* the feature uncertainty covariance (Equation 4.16) by including the above measure

$$\Sigma_{x_i, y_i} = \frac{s}{ncc(x_1, x_1')} \begin{bmatrix} \frac{\partial^2 I(x_i, y_i)}{\partial x^2} & \frac{\partial^2 I(x_i, y_i)}{\partial x \partial y} \\ \frac{\partial^2 I(x_i, y_i)}{\partial x \partial y} & \frac{\partial^2 I(x_i, y_i)}{\partial y^2} \end{bmatrix}^{-1} \tag{4.53}$$

where $x_1 = [x_i \; y_i]$ and s represents a scale factor that needs to be adjusted empirically. We

found a value of 10 to be a reasonable estimate. An alternative way of estimating s might be the "good-ness" weight of Section 3.4.1.

- Incorrect feature correspondences may also arise, when there is a *good match* between the concerned patches. Such feature correspondences are *outliers* of the correspondence data, and need to be weeded out. Standard solutions for estimating homographies based on RANSAC (random sample consensus) exist for this purpose.

- The number of feature correspondences has a lot of importance in judging the accuracy of the obtained homography. Typically 50-100 correspondences form a good sample for obtaining accurate homography. The covariance of the obtained homography may also be weighted by a factor proportional to the number of accurate homographies, where the accuracy of a match may be included in the feature covariance as mentioned above.

With the above robustness measures, we may introduce suitable weights into the estimation process. When in the presence of illumination or occlusion, however, the homography estimation may fail even with the above robustness heuristics. It is here that the Kalman Filter takes over to *predict* the *maximum likelihood* observation (homography) based on the measured state vector values (pose and plane parameters).

4.5 Experiments and Results

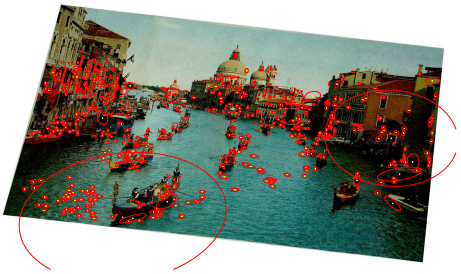
In order to test our theories, we conducted a combination of real and synthetic experiments. The synthetic experiments were aimed at empirically verying the *quality* of the results, while the real experiments demonstrate the efficacy of our approach.

Figure 4.6 shows the effect of normalization. First, a set of points were generated on a plane and the projected onto two randomly generated cameras. The resultant points in the second image were then perturbed by varying degrees of noise, and the homography was computed using both the normalized and un-normalized algorithms. The resultant homography was then used to project points in the first image, onto the second. Then the root mean square error between the projected points and the *actual* noiseless quantities was computed. As is seen in the figure, normalization not only produces a very good estimate of the homography, but is also very resilient to the noise introduced. Figure 4.6(b-c) illustrates this effect.

The second experiment was to analyze the coordinate normalization algorithm (Section 4.4.3). A set of 10 random planes were generated and projected on to two randomly generated cameras. The homographies computed after addition of noise to *both* points were then decomposed using the algorithm of [10] to obtain an initial estimate for the normals. This was then passed to both the algorithms to generate estimates of the perpendicular distances. Figure 4.5 shows the resultant errors in estimates of the perpendicular distances. The convex optimization solution has a very low root mean square error as compared to the SVD based solution (Figure 4.5(a) is shown in log space). Also, the convex optimization solution is *much* more resilient to error than the SVD based solution.

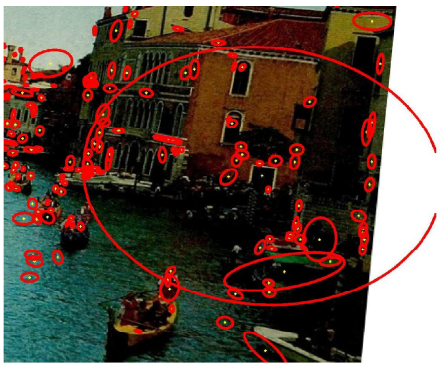
The next experiment was conducted to test the overall estimation for the plane and pose parameters. First a set of points were generated on 3 random planes. Then 200 camera positions were generated following a Brownian motion model, where the current camera position is extracted from a Gaussian centered around the previous camera position. The computed homographies were then perturbed with noise, and then used for estimation of the plane and pose parameters. The initialization of the plane parameters was highly erroneous because of the noisy homography. Figure

(a) Original Image



(b) Covariances Overlaid

(c) Original Enlarged



(d) Covariances Enlarged

(e) Second Image

(f) Transferred Covariance

Figure 4.3: Covariances of image features extracted by employing 4.16. The ellipses have been scaled for better viewing. Notice (d) how the size of the ellipses is a function of how "highly textured" the underlying feature is, which in itself depends on the gradient profile. [8]. (e-f) Second image and corresponding covariances obtained after transferring from the homography. Notice how the bottom of the image in (f) contains larger directional error, since the first and second images are rotated versions of each other with the axis of rotation somewhere near the top left. Thus as points move away from the axis of rotation, their uncertainty increases [9].

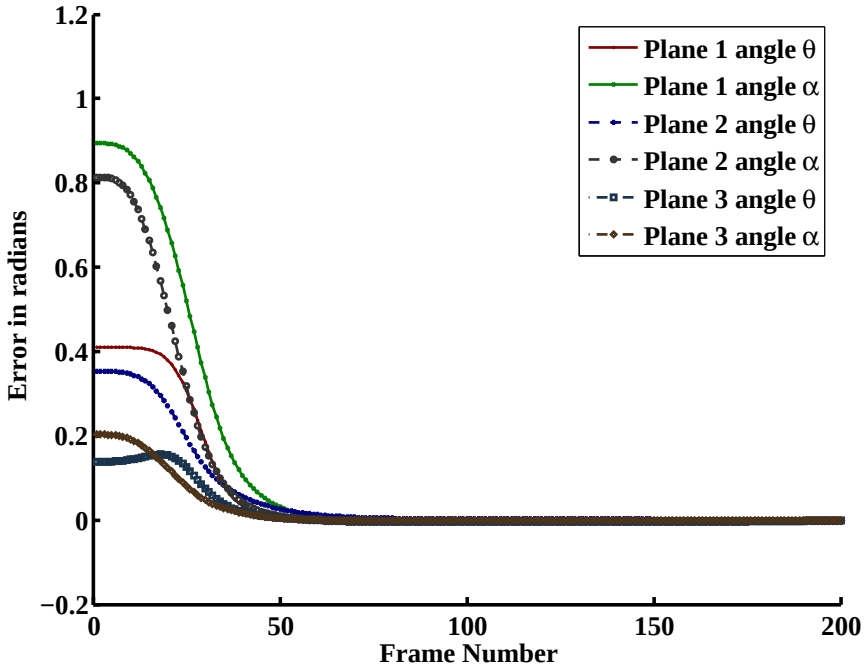
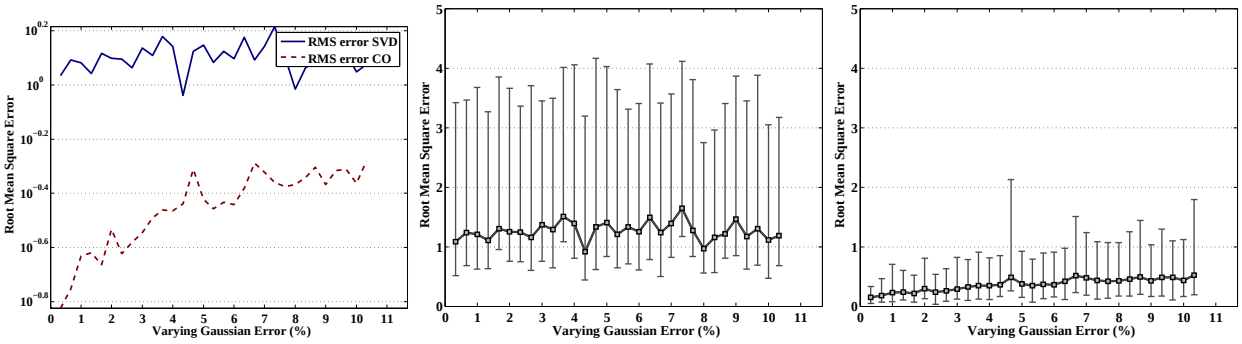


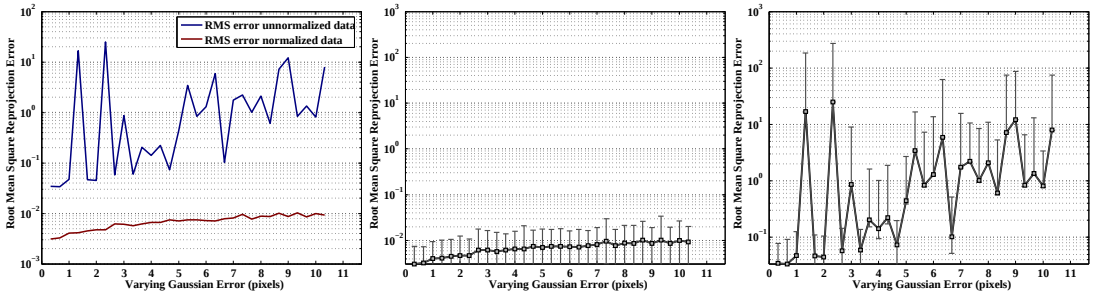
Figure 4.4: The plot shows the decrease in the error of plane parameters, estiamted using the UKF for a synthetic sequence. Notice how the initial estimates have large error, that remains constant for a while. This is because, we start with a very large covariance estimate of the homography, which accounts for the large difference between initial estimates of the plane parameters obtained using the decomposition algorithm [10]. However, even in this case, because of the presence of multiple planes, the error in the plane parameters reduces very quickly (almost exponentially) once the homography covariance stabilizes.

4.4 illustrates the results of UKF based tracking of the plane and pose parameters. The error in the estimation of pose parameters is very low, although "jitters" occur because our motion model has a lot of randomness. However, the estimation of plane parameters shows a sharp decrease over time, suggesting that even with a large initialization error in a plane parameter due to noisy homography, the other planes contribute to a decrease in error by keeping the pose error, which in turn "stabilizes" the value of the plane parameters. Figure 4.4(a) illustrates clearly.



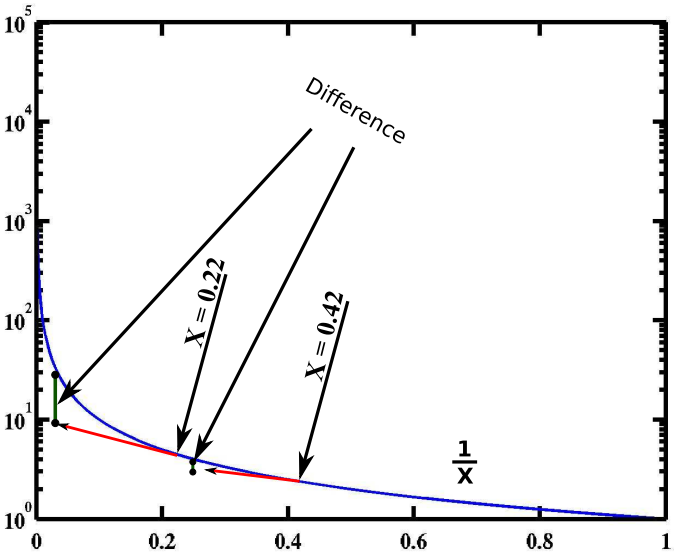
(a) (b) Noise variation of SVD initialization. (c) Noise variation of convex optimization based initialization.

Figure 4.5: The graphs show the superiority of the convex optimization solution over the traditional SVD based solution. Both figures (b-c) plot the root mean squared error between the estimated and ground truth parameters, along with bars to indicate the variation of the error, estimated through a Monte-Carlo test. Although the error eventually increases with increasing variance, the convex optimization solution nevertheless shows excellent resilience.

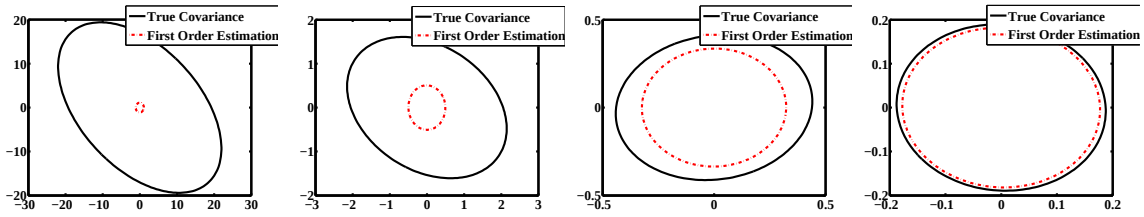


(a) RMS errors plotted. (b) Noise variation of normalized estimation. (c) Noise variation of un-normalized data.

Figure 4.6: The above figure plots the effect of normalization on homography estimation. The root mean square error is plotted between estimated and true homographies, against varying noise level in the feature correspondences. Observe how the error bars show that normalization is not only more accurate, but more resilient to noise as opposed to the un-normalized solution.

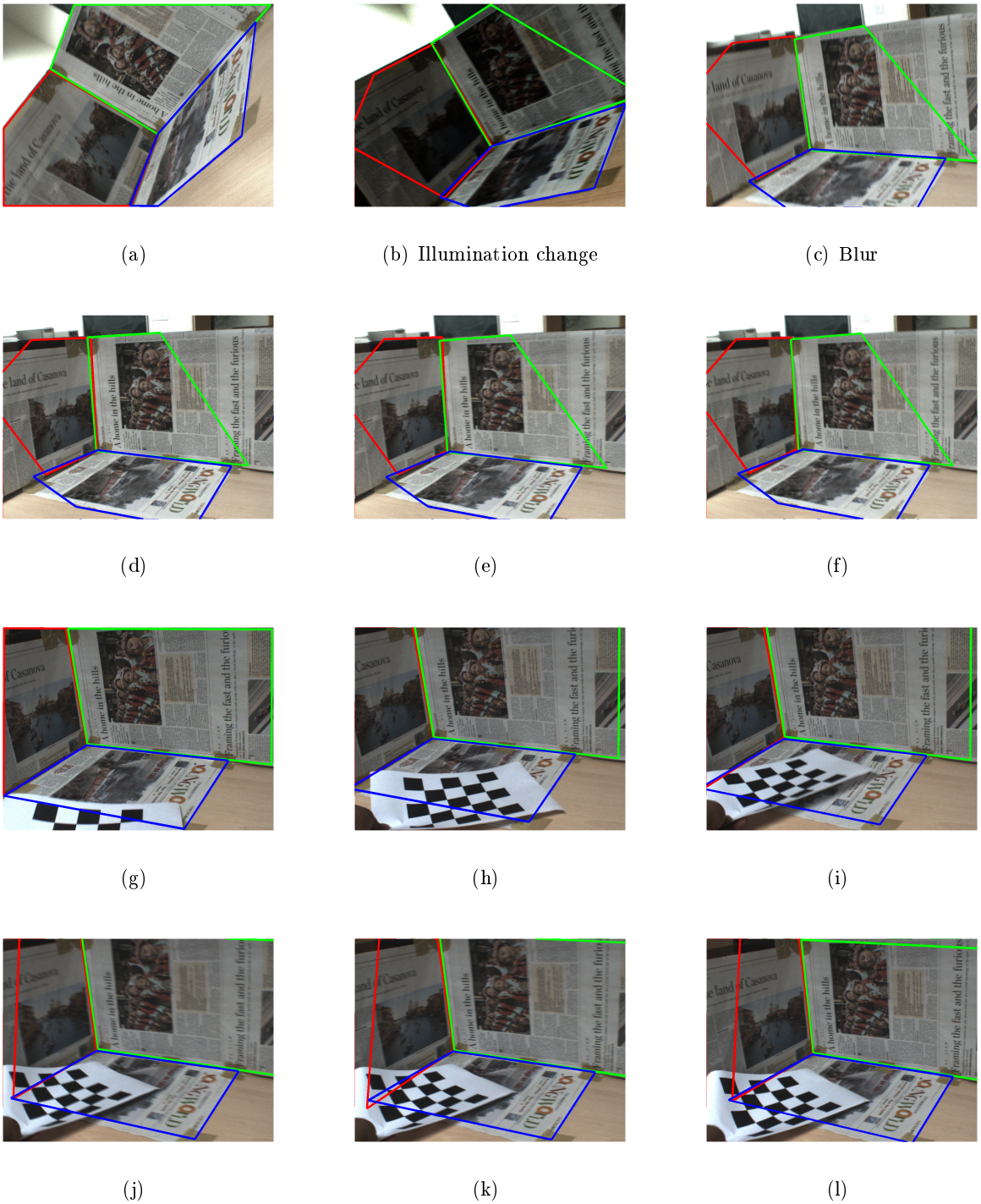


(a) The function 1 / w



(b) (c) (d) (e)

Figure 4.7: Approximation error of the perspective projection function 4.36.(a) The function $1/x$ is shown along with the error in first order approximations at two positions $x = 0.22$ and $x = 0.42$. (b-e) Actual covariance (black solid) and first order approximation (red dot-dash) after perspective projection of a 3 dimensional Gaussian, when the scale of the denominator with respect to the numerator increases from (b-e).



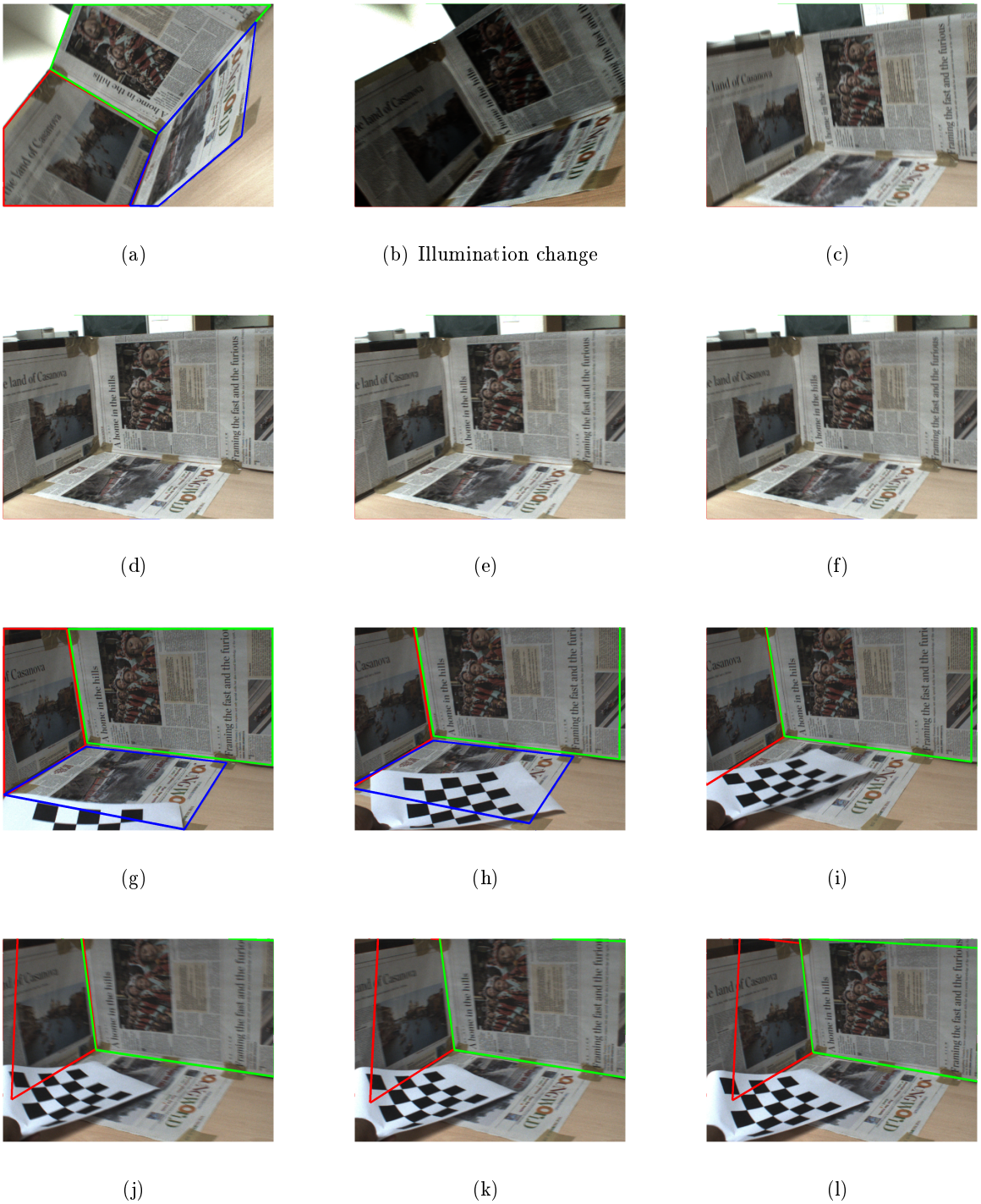
(a) (b) Illumination change (c) Blur

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 4.8: Results of UKF tracking on two sequences. Figures (a-f) are when there is considerable change of perspective, illumination and even blur in some cases. The KLT tracker will fail through most of these scenarios, but our tracker is robust to such changes. Figures (g-l) show the case when there is a large occlusion of one of the planes. Our tracker still robustly tracks the plane, showing that coupled estimation of the planes can be *much* more robust than estimation of the planes individually. The "jitters" in the homography estimation are due to the random motion model we use. It can be smoothened out either by using our result as an initialization to algorithms like [11] or by using a better motion model [3].



(a) (b) Illumination change (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 4.9: Results of tracking for the KLT tracker. In the first set of figures (a-f), the KLT tracker looses all its planes because of the illumination change. In the second set (g-l), the tracker loses one plane because of the occlusion.

Finally, two experiments on real data were conducted to illustrate the robustness of the tracker to conditions like change in perspective, change in illumination and occlusion. In the first, a video was

taken of three planes as shown in Figure 4.8 using a robotic camera. The intrinsic parameters of the camera were calibrated separately using a checkerboard pattern. The planes are so positioned with respect to the camera such that motion of the camera produces a large change in the perspective of the planes. To further complicate the video, a large illumination change occurs in the frames which normally renders the KLT tracker ineffective, as shown in Figure 4.9. Our algorithm, however, handles both the large change in perspective as well as the illumination change. This is because by the time both the perspective and the illumination change, the plane parameters for our algorithm stabilize and this helps us obtain good homographies even when many features of a plane are lost. Thus, all the lost KLT features can be re-initialized using the obtained homography and since KLT operates in a local neighborhood, this further helps in producing better estimates of homographies in the next iteration, and this recursive estimation produces a nice fit as can be seen in Figure 4.8(a-f).

The second experiment was conducted to solely test the power of *prediction* of the homography, in the absence of it. An object was introduced into the scene while the video is being captured in such a manner that the third plane was blocked to a large extent. This rendered the KLT tracker absolutely helpless as it could not track the features any longer. However, since by then the estimates of the plane normals had stabilized, the pose estimation was enough to accurately predict the position of the 3rd plane. Figure 4.8(g-l) shows this result.

The conclusion to draw is that the proposed tracker achieves a good amount of robustnes in the presence of many artefacts.

4.6 Summary

This chapter extends the problem of tracking introduce in the previous chapter to the case when multiple planes exist in the scene, and when 3D pose of the camera or equivalently the object needs to be estimated. We exploit the dual advantage of the concept that not only are plane tracking and pose tracking individually advantageous with useful applications (Section 4.4.4), but are interrelated. With this concept as the centrepiece of our argument, we develop a framework for tracking multiple planes and camera pose, which is robust to both illumination changes and occlusion. We model uncertainties in the feature detection process in a sound manner with focus on the texture content in the image(s), and model uncertainties in the measurement process using theoretically sound arguments [9]. This results in a tracker that both tracks accurately, and corrects errors in the measurement process.

The larger goal accomplished through this chapter is the integration of 2D and 3D modules to give rise to a generic tracking framework for *inferring* the position of the object in both 2D and 3D scenarios. As mentioned in Chapter 1, any task of inferring is a useful pre-requisite for other more complex tasks like *manipulation*. In the next chapter, we introduce a new scheme for positioning a robotic arm attached with a camera with respect to a planar object. The task, known as visual servoing, assumes that the tracking of the object is done and starts with that assumption. As described in chapter 1, navigation or positioning can be thought off as a manipulation task where the robot manipulates its position based on information available from the environment. In the next chapter, we present several algorithms for this positioning task, in a frequency domain framework, with analysis of how it can be a promising approach for future developments.

Chapter 5

Frequency Domain Visual Servoing Using Planar Contours

5.1 Introduction

In the previous Chapters (3 & 4), we looked at how MVG could be used to model the motion of an object in the case of 2D and 3D geometry. For any robotic system employing vision for guidance, this performs the task of localization or pose determination with respect to an immediate environment (more technically a *world coordinate system*). The current chapter builds over the previous chapters in that we now *assume such inference* to be available to us in the form of correspondence between the objects images across time. This information is then used to develop algorithms for *positioning* the robot with respect to the object under consideration. The task of positioning is useful for various reasons. For example, the simple task of an automatically driven car requires the car to constantly monitor its position with respect to the road, and to constantly evaluate what turn to make and when; all these tasks amount to the *positioning* of the car with respect to various points on the road. Thus the task of positioning can be defined as the *constant manipulation of the position of a robot with respect to object (s) of the environment, while receiving and incorporating constant feedback about localization from its sensors.*

In this chapter, we look at a positioning task called visual servoing. Visual servoing can be explained as the task of manipulating the motion of a robotic arm so that it moves from one position known as the start position to the desired or end position, with respect to a desired object (or target). Along this path, constant feedback in terms of visual input is utilized which confirms whether the robot is on the correct path, or corrects it when the robot goes astray. The nature of the path followed depends on the control law (objective function whose outcome determines the direction of motion of the robotic arm at each instant. It takes as input feature correspondences) designed, which in turn depends on the image features being utilized for feedback. Visual servoing approaches model robot motion as an error minimization process [79, 25, 80, 81], which is a function of the pose of the camera and the features used for navigation. Error models employed in the past include image and pose errors [80], errors in homography [82] and moments [83], etc. The path the robot has to follow at any instant is directly derived as motion along the gradient of this pose-feature (or pose-error) space. The estimation of the direction of motion is usually done using a Jacobian which is a first order approximation of the gradient in the pose space at that particular point.

Had the error function been convex, this approach would have lead to a global minimization algorithm, which takes the robot manipulator or end-effector from the initial to final positions. However, the inherent non-linearity of perspective projection gives rise to problems such as local

minima of the error function which means that there exists points in the pose-error space where the error is non-zero, yet the above algorithm would be incapable of giving a particular direction for the robot to proceed resulting in stagnation. Another popular problem is the non-straight Cartesian paths for the robot, during the minimization process. This means that when the resulting error is a difference in image coordinates (image points are considered as features) of an object captured in the initial and final poses, the error minimization ensures that the robot proceeds in a straight path in the *image space*. Because perspective projection is non-linear, this straight path in the image space results in a *non-straight* Cartesian path, which means the manipulator travels more than it needs to and runs the risk of reaching its joint limits in the process (joint limits are reached when the robot cannot stretch its arm beyond a point in order to reach a particular position in space). In addition, exact correspondence between points in the current and desired views need to be estimated for accurate convergence.

One approach to solve / avert one or many of these problems in the visual servoing community has been to design algorithms [84, 85, 82, 86] that differ primarily in the features used for the minimization process. The nature of features used leads to different formulations of the control law, which in turn result in different paths in which the robot performs the desired minimization. Of the several methods, some of the most popular features are extracted from the image space. Image based visual servoing (IBVS) has proved stability and convergence around the desired view [87], and the features tend to remain within the camera field of view throughout the minimization process. Hence, defining the error functions in the image space has gained popularity. However, the path followed by the robot to reach the target position, using IBVS, need not be straight in the Cartesian space. Directly in contrast is the Position Based Visual Servoing (PBVS) [88] framework, that extracts the 3D pose of the camera at each instant and uses it for servoing. It has the advantage of the straight Cartesian path, but suffers from the fact that it is highly sensitive to calibration errors, and can cause the object to go out of the field of view during the servoing process [89].

In order to counter the potential drawbacks of non-straight Cartesian paths, hybrid methods [79], comprising quantities both in image and pose space, have also been proposed. Recently, a new class of methods have tried to exploit higher order primitives in the image space. Such methods try to incorporate the advantages of position based visual servoing (PBVS), while retaining the IBVS framework. Tahri and Chaumette [83], used moments of planar objects to derive the control law for image based visual servoing. Advantages of such an approach over traditional IBVS include decoupling of the translational and rotational degrees of freedom and removing the need for explicit correspondence computation.

Another popular class of higher order primitives that may be extracted from an image is Fourier descriptors. Fourier transform has seen wide use in the computer vision community for geometry related computations. Specifically, Fourier based methods have been used to estimate quantities such as the Fundamental matrix for registering cameras [90], and homography for registering images of contours [91]. Geometric computations based on Fourier transforms are robust to image noise, since each frequency has a collection of measurements from the entire image or contour. Recently, Fourier based techniques have also been applied for correspondence estimation [91]. Thus, visual servoing approaches set in a Fourier based framework have the potential of bringing these advantages into the servoing process. However, the restriction of Fourier based geometric computations to affine transformations [92] has been a bottleneck in applying them to visual servoing. In this chapter, we resolve this problem in the context of visual servoing, and present an approach for servoing based on Fourier transforms.

5.1.1 Notation

In the following text, the capital letter X is used to denote a contour, while $\mathcal{F}$ represents the Fourier transform. Subscripts are used to denote the index number in a set, like a set of points in a contour. Thus X_i denotes the i^{th} point of a contour, whose x- and y- coordinates are defined as x_i and y_i. Rotation matrices are represented as R and Jacobians are represented as J. Finally, weight matrices are represented by W and error vectors by e. Since probabilities are not involved in this chapter, p represents the pose of the camera, while v represents the velocity screw or rate of change of pose. Finally, Z represents the depth of the object, while A represents an affine transformation.

5.1.2 Contributions

In Chapters 3 & 4, we developed a Bayesian filtering framework to track a single plane using multiple measurements, and multiple planes by exploiting their inter-relationships. In this chapter, we focus at the core of robotic manipulation, for which the previous two chapters are pre-requisite algorithms, and propose several new control schemes to manipulate the robotic arm. Specifically our contributions in this chapter include

- A correspondence-less visual servoing algorithm, built over the traditional IBVS framework, which is capable of positioning the robotic arm at a desired position *without* knowledge about explicit point-to-point correspondences between the object contour imaged in the initial and final views.
- A method employing the Taylor series expansion of the Fourier transform about a point in the rotation space, showing how even in the case of out of plane rotation beween initial and final views of a 5 DOF robot, straight Cartesian path may be achieved in visual servoing.
- A control law employing weighted Fourier features, capable of following a user given path *while* minimizing the corresponding error funtion over time.

Currently, we consider servoing in the case of 1D Fourier transforms of planar object contours, since it simplifies the underlying geometric transformations to the extent that approximations provide reasonable solutions. Future work might include extension to both non-planar objects and 2D Fourier transforms.

5.2 Related Work

Some of the initial works that inspired our approach was the application of frequency domain techniques to the estimation of multiple view geometric quantities like the homography [91, 93]. In [93], the authors propose the correspondence-less estimation of an affine transformation between planar contours imaged in two views as the estimation the transformation between their respective frequency transforms. This problem is then solved by imposing a rank constraint on a measurement matrix obtained from the respective Fourier coefficients. Additionally the correspondence between the two images is obtained by computing the shift between the Fourier components of the respective contours. This has been extended in [91], where the authors start from the solution provided by [93] and iteratively estimate the *projective* transformation between the input images of the planar contour, again with correspondence as the by-product.

The closest work to our approach is the work by Chaumette *et. al.* [83, 94], where the authors define a set of moments as features for the purposes of visual servoing. Moments are higher order

equations in terms of the image coordinates, and are shown to be able to capture certain properties about the shape of the contour in consideration that is independent of the angle of view of the camera. These moments have nice properties like allowing the individual control of each of the 6 degrees of the pose of the camera. The representation of Fourier transforms as exponentially weighted image pixels is close to the representation of image moments. Indeed, this leads to several similarities like the linearization of projective transformations in order to decouple rotation and translation. However, unlike [83], our attempt is to *improve* IBVS by using Fourier based processing rather than proposing a novel control scheme for servoing. This in turn, allows us to make use of stability and convergence properties of IBVS. Again, for example, it is possible to combine the path following method with IBVS to get a control law that follows a given path while keeping features in the field of view.

5.3 Fourier Based Visual Servoing

Since in visual servoing, the basic aim is to get a direction to move the robot at every instant, the application of Fourier analysis seems promising. In this section, we show how the properties of Fourier representation of contours may be used to do three important tasks in visual servoing.

5.3.1 Correspondence-less Servoing

Image Based Visual Servoing (IBVS) defines the path followed by the manipulator as descent along a pose-image error function. This function is described as the sum of squared differences between the corresponding points in the initial and final views. This function, however, implicitly makes the assumption that correspondence between the two views has been established *apriori*. This is a problem when correspondence between views cannot be established easily or accurately. Correspondence-less visual servoing is particularly useful when the two views of the object are widely separated leading to inaccurate correspondences. Traditional IBVS cannot correct this error, hence there is a need to address this problem. We show now that in case of planar contours, this problem may be addressed by the use of frequency domain techniques [91, 93].

Let $X = \{x_i\}, Y = \{y_i\}$ and $X\_s = \{x\_s_i\}, Y\_s = \{y\_s_i\}$ be the x and y coordinates of a set of two contours that are images of a planar contour in 3D, imaged in two cameras. Let us assume that the correspondence between the two images is unknown, however, by creating X, Y such that the points of the contours are either stacked in a clockwise direction or anticlockwise direction, we can reduce the correspondence estimation problem to an estimation of *shift* between the two contours [91]. We wish to be able to servo so that the camera moves from the first view to the second view in the absence of knowledge about this shift. Let $\mathcal{F}$ represent the 1D fourier transform. This change in shift comes up as a multiplicative factor in the Fourier transform

$$\mathcal{F}_k(X\_s) = \mathcal{F}_k(X)\, exp^{2\pi jk\lambda} \tag{5.1}$$

$$\mathcal{F}_k(Y\_s) = \mathcal{F}_k(Y)\, exp^{2\pi jk\lambda} \tag{5.2}$$

where $\mathcal{F}_k$ represents the k^{th} frequency component. When the views are related by an affine transformation, several methods are available to estimate the correspondence between contours [93]. But in the case where the contours are transformed by a more general *projective* transformation, these algorithms only give an *approximate* estimate of the correspondence. However, this approximation gives rise to a simple algorithm that can be used to perform servoing in a Fourier based framework.

We define correspondence less servoing as a two step process, when a generic projective transformation exists between the initial and desired images. In the first step, shift between the two contours is estimated using the algorithm of [93]. In the next step, IBVS is performed by using an estimation of this shift to generate the error vector (Equations (2.26,2.27)). As the servoing process converges the robot to the desired view, the two views (current and desired) approach each other, and in such a case, the transformation between the two planes is known to be affine [2], for which estimation of shift is accurate. The reasoning behind our approach is the following. Affine transformations are known to be good approximations of projective transformations [9], since effects of perspective projection get prominent only when the object under consideration is very close to the camera or the camera moves drastically[2]. Thus, estimation of shift assuming affine transformations is a close approximation to the actual shift present between the contours. Since IBVS is robust to such small errors in feature correspondence, the servoing is minimally affected because of this erroneous correspondence and so the robot moves in the correct direction. Eventually, when the views get close enough, the affine part of the transformation takes over and more accurate correspondences are obtained. The overall algorithm is summarized in Algorithm 3.

In practice, in comparison to the algorithms proposed in [93] for shift estimation, we found the following derivation to be more robust in estimating the correspondence under projective transformations.

$$\mathcal{F}_k(X\_s) = \mathcal{F}_k(X) * exp^{(2\pi ik\lambda/n)} \tag{5.3}$$

$$\log(\frac{\mathcal{F}_k(X\_s)}{\mathcal{F}_k(Y)}) = 2\pi ik\lambda/n \tag{5.4}$$

$$\lambda = \max_{\lambda} |(\mathcal{F}^{-1}\left(\frac{n}{2i\pi k}\log(\frac{\mathcal{F}_k(X\_s)}{\mathcal{F}_k(x)}\right)|), \tag{5.5}$$

which in words, means that we are trying to minimize the *maximum possible shift* between the two contours in consideration. This is particularly useful, when the contour does not have a random shape but has a specific pattern to it, like the one shown in Figure 5.1.

Algorithm 3 Algorithm for correspondence-less visual servoing.

Input: Initial and desired contour images.
Extract contour points, $X = (x_i)$, $Y = (y_i)$ and $X\_s_k = (x\_s_i), Y\_s_k = (y\_s_i)$.
for $k = 1 \ldots \infty$ **do**
  Estimate shift λ between the two contours using Equation (5.5).
  Construct error matrix using λ (Equations (2.26,2.27)).
  If contours are sufficiently close, exit.
end for

5.3.2 Straight Cartesian Paths for 5 DOF Servoing

In the previous section, we formulated a simple algorithm for correspondence-less servoing that can work even in scenarios where correspondence is difficult to obtain. In this section, we focus on the straight Cartesian trajectory problem, and propose a new algorithm that can achieve near-straight Cartesian paths in the case of a 5 DOF (degrees of freedom) visual servoing process. Since we achieve straightening of the path by a linearization in the rotation space, it becomes exceedingly complex to handle a 6 DOF robot with 3 degrees each for translation and rotation. Thus we consider only 2 degrees of rotational freedom in our current work.

Let the image of a 3D contour in the initial and desired views be represented by $\{X_i^2\}$ and $\{X_i^1\}$, where superscript denotes the frame number and the subscript denotes the i^{th}point. Each point $X_i^k = \begin{bmatrix} x_i^k & y_i^k \end{bmatrix}$. Let us further assume that the two contours are related by a homography f.

In the case of planar contours, an affine transformation is comprised of the effects of 4 parameters of pose out of the existing 6 [65]. These are translation in x (t_x), y (t_y) and z (t_z), and rotation about the z axis (r_z). These parameters result in shifting, scaling and rotation in the image space respectively, and so servoing with respect to these parameters results in straight Cartesian paths [4]. The *non-linearity* of the image motion - Cartesian motion relationship is primarily due to the remaining rotation parameters (r_x, r_y). Thus, in order to ensure a straight Cartesian path, we need an estimate of these rotation parameters. Our aim is thus to get an estimate of r_x or r_y, given the contours, since in a 5-DOF robot, the other pose parameters can be readily decoupled.

For the sake of simplicity, assume the two given views only differ in r_x in their respective camera poses. This means that the values of (r_8,r_9) are the main parameters to be estimated in Equation (5.11). The case for r_y can be made on similar lines. Extension to a generic pose change is discussed later. The points in the two images are related by f which is an *infinite homography* [65].

$$
\begin{aligned}
X_i^2 &= f(R, X_i^1) && (5.6)\\
&= \begin{bmatrix} f_x(R, X_i^1) & f_y(R, X_i^1) & 1 \end{bmatrix}^\top && (5.7)\\
f_x(R, X_i^1) &= x_i^2 = \frac{R_1 X_i^1}{R_3 X_i^1} && (5.8)\\
f_y(R, X_i^1) &= y_i^2 = \frac{R_2 X_i^1}{R_3 X_i^1} && (5.9)\\
R &= \begin{bmatrix} R_1 & R_2 & R_3 \end{bmatrix}^\top && (5.10)\\
&= \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}, && (5.11)
\end{aligned}
$$

where R represents the rotation between the two views under consideration. We now perform a Taylor expansion of f with respect to R gives us a linear representation of the rotation. A convenient point about which R may be linearized is the identity matrix

$$
I = \begin{bmatrix} I_1 & I_2 & I_3 \end{bmatrix}^\top = \begin{bmatrix} i_1 & i_2 & i_3 \\ i_4 & i_5 & i_6 \\ i_7 & i_8 & i_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.12)
$$

Linearizing the right hand side of Equations (5.8 & 5.9) about the identity matrix, we get

$$
\begin{aligned}
f_x(R, X_i^1) &= \frac{I_1 X_i^1}{I_3 X_i^3} + \frac{\partial f(I, X_i^1)}{\partial r_i}(r_i - i_i) && (5.13)\\
f_x(R, X_i^1) &= (2 - r_9) x_i^1 - r_8 x_i^1 y_i^1 && (5.14)\\
f_y(R, X_i^1) &= y_i^1 - r_8 y_i^1 y_i^1 + r_6 && (5.15)
\end{aligned}
$$

where the values of R are set according to a rotation about x-axis. Notice that the major cause of worry is the parameter r_6, which is conveniently decoupled from the rest of the parameters, and

present just as an addition term. This ensures that its effect can be removed by subtracting the DC component from the Fourier transform.

We now take the Fourier transformation of both sides of Equations (5.14 & 5.15), to obtain

$$\begin{aligned} \mathcal{F}(x_i^2) = \mathcal{F}(f_x(R, X_i^1)) &= (2 - r_9)\mathcal{F}(x_i^1) - r_8\mathcal{F}(x_i^1) * \mathcal{F}(y_i^1) \quad (5.16) \\ \mathcal{F}(y_i^2) = \mathcal{F}(f_y(R, X_i^1)) &= \mathcal{F}(y_i^1) - r_8\mathcal{F}(y_i^1) * \mathcal{F}(y_i^1) + \mathcal{F}(r_6) \quad (5.17) \end{aligned}$$

where $*$ represents convolution.

The above equations expand into 2 equations *per* frequency of the Fourier transform. Since all the points share the same values of the rotation parameters, and since the Fourier parameters $(\mathcal{F}(X_i^j))$ are known to us, the estimation of the remaining parameters (r_8, r_9) becomes a linear estimation problem. By collecting the parameters to be estimated in the above two equations into a matrix and inverting the sides of the equations, we get the following

$$\begin{bmatrix} \mathcal{F}(x_i^1) & \mathcal{F}(x_i^1) * \mathcal{F}(y_i^1) \\ 0 & \mathcal{F}(y_i^1) * \mathcal{F}(y_i^1) \end{bmatrix} \begin{bmatrix} (2 - r_9) \\ -r_8 \end{bmatrix} = \begin{bmatrix} \mathcal{F}(x_i^2) \\ \mathcal{F}(y_i^2) - \mathcal{F}(y_i^1) \end{bmatrix} \quad (5.18)$$

where $\mathcal{F}(r_6)$ is assumed to be cancelled out due to subtraction of the DC component. The above equation can be collected over all the frequencies except the base frequency to obtain an *over-determined* set of equations that can be solved in the least squares sense using SVD [2].

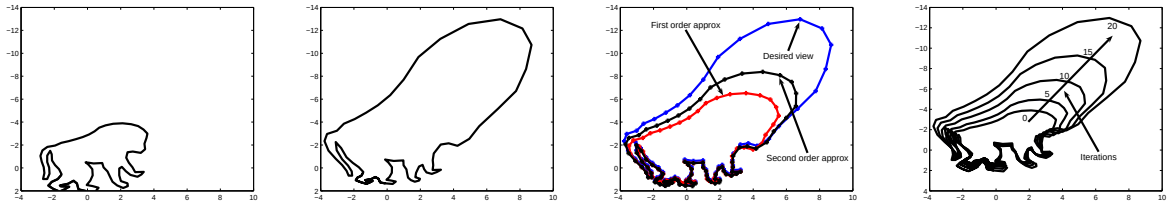
Since this derivation is based on a linearization about the current image, iterative estimation and warping is used for convergence. As stated earlier, the effect of r_6 in the equations may be compensated by discarding the DC component of the transforms. The summary of our approach can be found in Algorithm 4.

Algorithm 4 Simple iterative estimation of rotation, between contours.

Input: Two contours, represented by point sets (X_i^1) and (X_i).
$(X_i^3) = (X_i^2)$
while convergence is not achieved upto acceptable error **do**
  Take Fourier transform of (X_i^3) and (X_i^2). Discard the DC component.
  Build an error matrix consisting of Equations (5.16 & 5.17), like in Equation (5.18).
  Minimize the error using an algorithm like SVD [2].
  Rotate (X_i^3) to obtain the new contour. Assign it to (X_i^3) for the next iteration.
  Collect the rotation matrix produced as solution.
end while
Composition of all the collected rotation matrices provide the entire transformation between (X_i^2) and (X_i^1).

In order to illustrate the efficacy of the above algorithm, we conduct a simple experiment. Two contour images of a scene are obtained, and the rotation between the two views is iteratively estimated. At each step of the iteration, we also obtain a new contour, that is the rotated version of the first one. Figure 5.1 shows the result of our estimation algorithm. As can be seen in Figure 5.1(d), the first image *iteratively moves* towards the second image, ascertaining our theory that iterative convergence is possible for our algorithm.

Extension to general pose change The challenge now is to estimate r_x or r_y when the remaining pose parameters (t_x, t_y, t_z, r_z) are not assumed zero. Since these parameters form an affine



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

Figure 5.1: (a) Current view (b) Desired view (c) First and second order Taylor approximations of the rotation about x-axis. (d) Different stages of the iterative minimization process.

transformation, results from [92] may be leveraged to our advantage. Specifically, when a contour undergoes an affine transformation $H = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$, the corresponding Fourier components undergo the same transformation. Since (t_x,t_y) primarily contribute to the DC component and (t_z) to scale, higher frequencies are suitable to estimate (r_x) or (r_y) robustly. Now consider an intermediate view X_i^3, which differs from the current view (X_i^1) in a rotation about the x or y axis. Thus Fourier transform of the relationship between X_i^3 and X_i^2 may be described by an affine transformation.

$$\mathcal{F}(X_i^2) \quad = \quad \lambda(A\mathcal{F}(X_i^3) + \mathcal{F}(b)) \tag{5.19}$$

The unknown scale factor λ approximates the change t_z, and may be described as the ratio of areas of the contour in the two images [81]. Expanding the relationship between X_i^3 and X_i^1 in Equation (5.6) using the Fourier relationship derived earlier and ignoring the base frequency to eliminate r_6, we get a matrix along the lines of Equation (5.18) as

$$\mathcal{F}(X_i^2) \quad = \quad \lambda(\begin{bmatrix} a_{11}(2 - r_9)\mathcal{F}(x_i^1) + a_{12}\mathcal{F}(y_i^1) - r_8 a_{11} \\ \mathcal{F}(x_i^1) * \mathcal{F}(y_i^1) - r_8 a_{12}\mathcal{F}(y_i^1) * \mathcal{F}(y_i^1) \\ \\ a_{21}(2 - r_9)\mathcal{F}(x_i^1) + a_{22}\mathcal{F}(y_i^1) - r_8 a_{21} \\ \mathcal{F}(x_i^1) * \mathcal{F}(y_i^1) - r_8 a_{22}\mathcal{F}(y_i^1) * \mathcal{F}(y_i^1) \end{bmatrix}) \tag{5.20}$$

These set of equations in 6 unknowns, 4 of A and 2 of R can be solved in the least squares sense using the equations resulting from the various frequencies in the Fourier transform. The parameter λ may be eliminated by imposing the condition that A and R are rotation matrices and hence have unit norm. Once the rotation values are estimated, the translational degrees of freedom may be controlled separately leading to a decoupled system. The entire method is summarized in Algorithm 4, with the only change that step 4 is replaced by Equation (5.20).

5.3.3 Path Following

Having considered two problems with traditional visual servoing in the previous sections, we now turn our attention to a novel problem, that of *altering the path* of minimization in the visual servoing process.

In traditional visual servoing, research focuses on proposing different types of features and algorithms for positioning [4]. Thus, different algorithms concentrate on the advantages of using different

kinds of features and algorithms for stability of the servoing process, or for ensuring convergence in the presence of local minima etc. The bottleneck of this approach is that once the features and algorithms used for servoing are set, the robot is constrained to move along a predefined path corresponding to the one obtained using gradient descent [95, 25]. Thus when artefacts like obstacle come in the way of the robot, major changes to the control algorithm or features are required to alter the path of the robot. Another problem with traditional servoing algorithms like IBVS is that if the path dictated by the control law takes the robot outside the limits of the joint angles, the IBVS process will fail in such cases. Hence it is desirable to be able to control the visual servoing process such that the end-effector travels through a specified set of poses or is within certain prespecified limits. Thus it would be useful if we could control the trajectory of the robot end-effector *while* performing IBVS. We believe this approach is particularly useful in scenarios where the robot cannot be trained manually by first showing it the entire path to follow.

In this sub-section, we overcome this problem by desigining a control algorithm capable of handling this issue. We show how a minimization of the weighted Fourier error can be used as a method to follow *different paths* in the minimization process, while the robotic manipulator goes from the *same* initial position to the *same* final position. Thus, such an algorithm can take as input, say the output of a path planning algorithm that plans the path of a robot through an environment taking into consideration the obstacles present along the way. Our algorithm will then *follow* the obtained path, while minimizing the error between current and desired frames.

We start the derivation of our control law by first observing a snapshot of the equations involved in traditional IBVS, which is shown below

$$
\begin{aligned}
e &= s - s^* && (5.21)\\
\dot{e} &= \frac{\partial e}{\partial p}\frac{\partial p}{\partial t} && (5.22)\\
\frac{\partial p}{\partial t} &= \left(\frac{\partial e}{\partial p}\right)^{+}\dot{e} && (5.23)\\
J &= \left(\frac{\partial e}{\partial p}\right)^{+} && (5.24)
\end{aligned}
$$

where $(e, \dot{e})$ represent the image error and the rate of decrease of image error, respectively. p represents the pose of the robot manipulator, while J represents the Jacobian that ultimately decides the direction of movement at every step of the convergence.

In traditional image based visual servoing (IBVS), the control law is specified in terms of the image error $e = s - s^*$, and the Jacobian J_s (Equation (5.24)). The velocity screw of the robot end-effector at any time is then given by

$$
\begin{aligned}
J_s &= \begin{bmatrix} J_1 & \dots & J_n \end{bmatrix}^{\top} && (5.25)\\
J_i &= \begin{bmatrix} 1/Z_i & 0 \\ 0 & 1/Z_i \\ -v_i/Z_i & -u_i/Z_i \\ -u_i v_i & -(1+v_i^2) \\ 1+u_i^2 & u_i v_i \\ -v_i & u_i \end{bmatrix}^{\top} && (5.26)\\
v &= J^{+} e && (5.27)
\end{aligned}
$$

We now define the Fourier error to be

$$e_{\mathcal{F}} = \mathcal{F}(e) = \mathcal{F}(s) - \mathcal{F}(s^*), \tag{5.28}$$

where $\mathcal{F}(s)$ is a concatenation of the Fourier transforms of the x and y coordinates. Since Fourier transforms are linear and invertible, the Jacobian of the Fourier error can be computed in a manner similar to the traditional IBVS error. It turns out to be

$$J_{\mathcal{F}} = \frac{\partial F}{\partial s} J_s \tag{5.29}$$

$$J_{\mathcal{F}}^{+} = J_s^{+} \left(\frac{\partial \mathcal{F}^{-1}}{\partial s} \right) \tag{5.30}$$

$$= J_s^{+} \frac{\partial (\mathcal{F}^{-1})}{\partial s}, \tag{5.31}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. By defining the Fourier error and related control law, we have ensured that the feature vector used for the minimization process is the Fourier descriptor of the image points, which may be a contour. The advantage of Fourier descriptors is that each frequency component captures *global* properties about the contour in consideration, unlike image points, which are local in nature. Since we have already seen earlier that Fourier components tend to capture the geometric properties of the transformation between initial and final views [93, 91], we use this concept to argue that changing each of these components has a different effect on the overall image error of traditional IBVS, and hence the minimization process.

Thus, to control this effect, we now introduce a *weight* matrix W, that weights each component of the Fourier error. Thus our new error model can now be defined as

$$e_{\mathcal{F}}^{W} = W e_{\mathcal{F}} \tag{5.32}$$

$$W = \begin{bmatrix} w_1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & w_i & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & w_{2n} \end{bmatrix} \tag{5.33}$$

$$v = J_s^{+} \frac{\partial \mathcal{F}^{-1}}{\partial s} (W e_{\mathcal{F}}) \tag{5.34}$$

The velocity screw at each iteration is then given by Equation (5.34). Since each Fourier component is composed of contributions from all the points in the image, weighting the Fourier components has a global effect on the error minimization. This allows IBVS to follow different paths within the same minimization framework.

To demonstrate the application of the above control law, assume that we have a sequence of relative pose estimates $\{\Delta p_i\}$ with respect to the destination frame available to us. These estimates may be the output of a constrained optimization problem used in path planning [84]. In order to ensure that the servoing algorithm makes the robot pass through these given poses, we need to find the weight matrix W that will guide the robot appropriately. In order to do this, we observe that in Equation (5.34), the left hand side represents the screw velocity, which is in turn defined as change

in pose over time (between successive frames, this takes the value of one instant). Thus if we define $v = \Delta p_i$ in the above equation, the inversion of the right hand side will give the *difference* between the estimated pose change in the current instant and the *desired* pose change, represented by Δp_i . We can now define the desired weight matrix as the value that *minimizes* this difference as

$$W = \min_W (-J_F^+ W e_{\mathcal{F}} - \Delta p_i)^2, \tag{5.35}$$

where the negative sign for error indicates the Jacobian at the desired pose. Reordering the terms in the equation by substituting W for a vector of its diagonal elements and $e_{\mathcal{F}}$ for a matrix with elements in its diagonal, we get

$$W = \min_W (-J_{\mathcal{F}}^+ e_{\mathcal{F}}^* W^* - \Delta p_i)^2 \tag{5.36}$$

$$W^* = (-J_{\mathcal{F}}^+ e_{\mathcal{F}}^*)^{-1} * \Delta p_i \tag{5.37}$$

$$W^* = -e_{\mathcal{F}}^{*-1} J_{\mathcal{F}} \Delta p_i \tag{5.38}$$

As can be seen, although the relationship between pose change and the different components of the Fourier transform are not completely known, nevertheless we are able to control the path of the robot by weight individual components. This equation holds as well near the desired pose as the approximation of the Jacobian for IBVS. The entire method is summarized in Algorithm .

Algorithm 5 Path following using weighted Fourier servoing.

```
Input: initial and desired images for the servoing.
Input: A list of poses (Δp_i) that the robot has to achieve.
Append desired pose Δp_desired to the given list.
Set k = 1
while convergence to desired pose not achieved within bounds do
    Compute weight matrix for servoing to position Δp_k, using Equation (5.36).
    Minimize weighted Jacobian error, to obtain servoing direction and velocity.
    Servo.
    if Δp_k achieved within error bounds then
        k = k + 1
    end if
end while
```

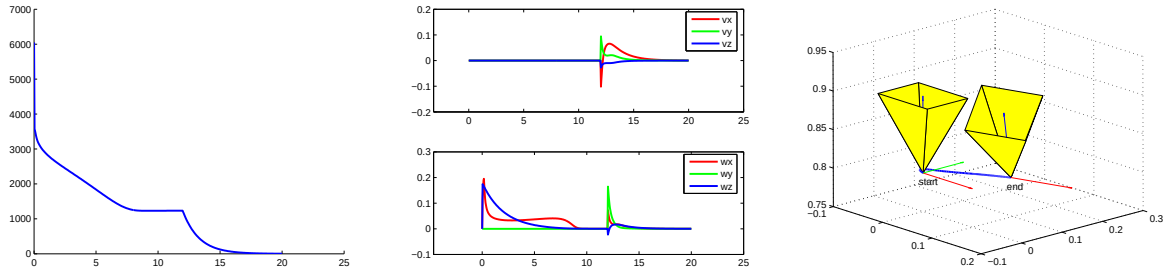
In the current section, we saw how Fourier analysis may be applied to the problem of visual servoing to generate new control laws that have several advantages over traditional servoing algorithms like IBVS. In the next section, we present and analyse the results of the propose methods.

5.4 Results and Analysis

In the next few paragraphs, we present the results of our three algorithms on simulation data. This kind of simulation based experiments is actually a standard in the visual servoing literature [96], with simulation toolkits in Matlab being released by some of the pioneers in the field.

5.4.1 Straight Cartesian Paths for 5 DOF Servoing

Figure 5.1 shows the setup in which the current and desired views are separated by 10 degrees of rotation about the x-axis and z-axis. The decoupling involves minimizing the error in rotational components of the views first, followed by translation. Each iteration of the minimization process consists of finding the least square solution for the angle (r_x) and minimizing the rotational error by rotating the camera appropriately. Since the initial and final views are widely separated, at first this minimization is approximate. However, as the camera moves nearer to the desired pose, the relationship between the two views moves towards an affine transformation, which can be handled better by Fourier transforms. It is interesting to note that each frequency component gives 2 equations for a total of $2(n-1)$ equations barring the base frequency. Had correspondences been unknown, we would still have $(n-1)$ equations following the idea in Section 5.3.1 which opens up the possibility of pose computation without correspondence in planar scenes, useful for PBVS.

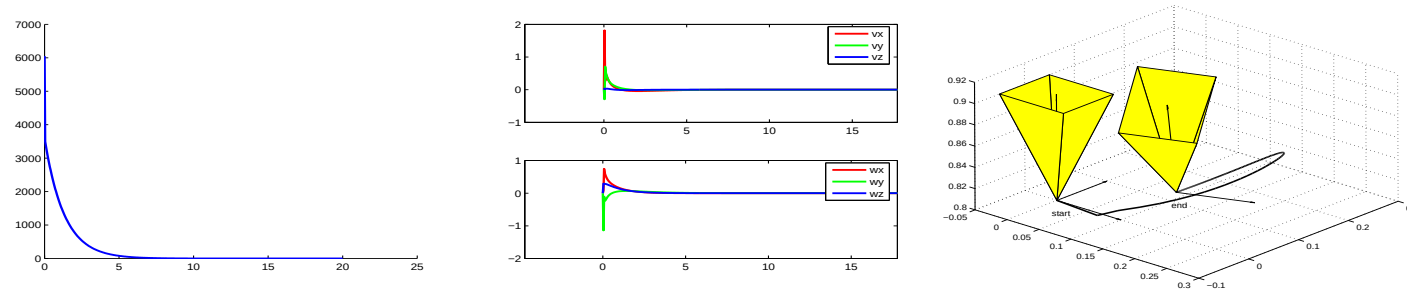


(a) (b) (c)

Figure 5.2: Cartesian straight path for 5 DOF Servoing: (a) Image error increases during rotation. (b) Observe how the rotation is corrected first and then the translation. (c) The slight error towards the end of the path is due to the approximations used in the formulation.

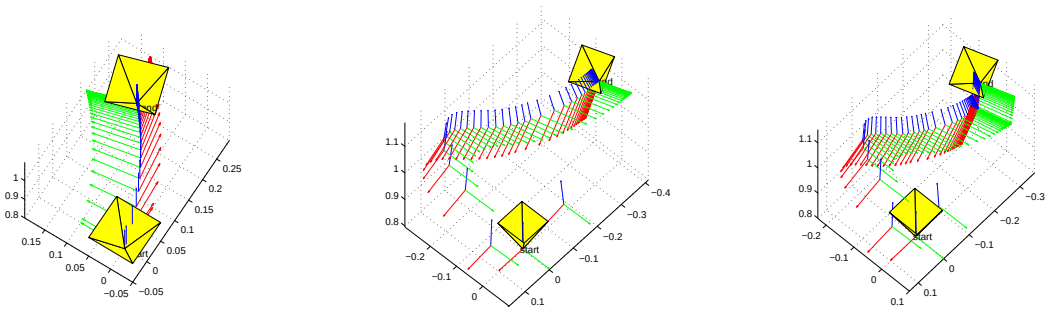
Figure 5.2 shows the visual servoing trajectory after plugging in the decoupling solution, above the normal visual servoing trajectory. Note how the decoupling results in a straight path in Cartesian coordinates when normal IBVS would result in a curved trajectory.

The main drawback of this method is the Taylor approximation that needs to be leads to small errors in the initial rotation error minimization. However, we observe that the minimization process



(a) (b) (c)

Figure 5.3: Image based visual servoing: (a) Image error decreases exponentially. (b) Observe how the corrections in rotation and translation lead to the path of the robot being curved in (c).



(a) (b) (c)

Figure 5.4: Path following: (a) Image based visual servoing (IBVS). (b) Specified path to be followed, and (c) Path of the end-effector following trajectory. Observe how closely the specified trajectory is followed.

converges even for huge perspective distortions (Figure 5.1), and hence believe this to be a promising result in spite of the minute error.

5.4.2 Path Following

Figure 5.4 shows the results of the path following approach. Figure 5.4(a) shows the camera trajectory during normal visual servoing, while Figure 5.4(b) the desired camera trajectory. Notice how, even with the linear approximation, the actual camera trajectory 5.4(c) closely mimics the desired trajectory. This suggests that the Fourier-pose space curves slowly around the desired pose, and thus the linearization approximates Fourier error for a large region around the desired pose.

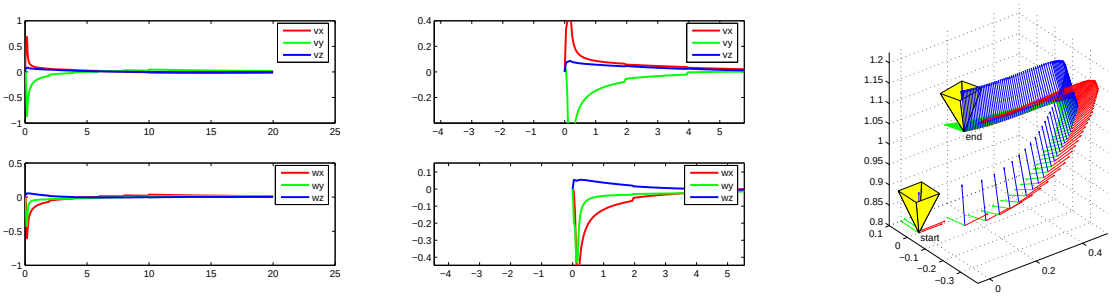
The path following approach is particularly useful for tasks like local minima avoidance in IBVS and for keeping features in field of view. However, since explicit relationships between each of the Fourier components and the elements of camera pose (rotation, translation) is difficult to analyze, only approximate relationships for both the tasks can be devised. However, the concept of weights allow for a smooth camera trajectory that can satisfy more than one constraint. This is a useful contribution considering joint and velocity limits of robots.

5.4.3 Correspondence-less Servoing

The results for correspondence-less visual servoing are shown in Figure 5.5. In this experiment, we have considered two contours in initial and desired views with 100 points sampled in each view. The correspondence misalignment is around 20 pixels. As expected, the actual shift has little consequence on the minimization procedure. The increasing accuracy in correspondence results in many artifacts in the minimization procedure, where the correspondence between the current and desired views shifts by a particular number. For example, at the start of the IBVS iteration, the correspondence is estimated as a 15 pixel shift, which gets refined to 20 over time.

5.5 Summary

The applications of MVG to the area of Robotics is amply visible. With this chapter, we have shown that a frequency domain based approach to applying MVG to robotic arm manipulation has several advantages over the traditional methods in terms of the path being followed by the robot,



(a) (b) (c)

Figure 5.5: Correspondence-less visual servoing:(a) Velocity screw in the servoing process. Notice the slight bumps in the screw (magnified in (b)). (c) Corresponding camera trajectory that is slightly more skewed than traditional visual servoing.

the constraints on its motion and the features it extracts etc. One of the major components required by an area like visual servoing is the *tracking* or *inferring* of the location of the target in the image, since although our method does not require *explicit point-to-point* correspondence, we do need to know that the two contours extracted in the current and target images are corresponding ones. In the earlier chapters, we looked at how this problem could be attempted in a 2D and 3D setting.

In this paper, we have presented several reasons to set visual servoing algorithms in a Fourier based framework. In the case of planar objects, Fourier based approaches can be used to combine correspondence estimation and servoing, to decouple rotational and translational components of the velocity screw and to do path following. Although some of our contributions closely parallel the work in [83], Fourier based approaches are particularly attractive since recent results in geometric vision [90] could be easily leveraged to advance visual servoing.

Several extensions to this approach may be planned. Firstly, extending the path following approach to non-planar objects is easily achievable since the only variable to be handled in such a case would be the considerable depth variation within the object. In the case of correspondence-less servoing, recent results in Fundamental matrix estimation [90] may be incorporated to extend visual servoing to general rigid objects. Extending the decoupling of rotation and translation to 6 DOF requires that we combine r_x and r_y estimation into one single algorithm. A side product of such an approach would also be approximate estimation of pose from planar objects, which could further be used for PBVS.

Emerging from the three chapters, current and previous, is a common theme for *inferring and manipulation* with respect to planar or piece-wise planar targets in an environment. A piece-wise planar model is one of the simplest and yet one of the most reasonable modeling strategies for the environment in computer vision. Through the course of the last few chapters we have shown how this modeling can be brought to robotics, by introducing the major components of a framework that can *localize* and *navigate or manipulate* objects in a piece-wise planar environment. We believe there is a huge scope for extending this approach to produce robots that can perform such navigation in real time to give, say guided tours of museums [97].

From a broader perspective, we have shown how MVG has good potential for application in robotic manipulation. We now extend this approach by analyzing how MVG could not only be used in robotics, but also in video and image editing and creation applications, by proposing two new algorithms for automatic object detection and removal (Chapter 6), and for image based rendering in the absence of complete information (Chapter 7). Finally we conclude by summarizing our

contributions and elucidating future directions of research (Chapter 8).

Chapter 6

Video Completion

6.1 Introduction

In the previous chapters, we proposed algorithms for inference and manipulation of the camera attached to a robot of of equivalently a rigid object in the environment. Having proposed contributions to the area of robotics, we now wish to extend the scope of applications for MVG. It is in this spirit that the next couple of chapters discuss the applications of MVG to the areas of *video and image manipulation.* A video may be thought off as a time sequenced collection of image pixels, which are related to each other *across frames* through the *geometry* of the scene. Thus, planar objects are related through planar homographies, and any point in an image is related to its corresponding point in any other image through the Infinite Homography, subject to the constraint that the camera viewing the scene should not translate (only rotate). Such relationships have a lot of power in them because they constrain the images produced to lie within a restricted domain rather than assume any random values.

Since MVG codifies the relationships between the various geometric quantities that represent the 3D world imaged by a moving or stationary camera, *inferred* values of such quantities may further be used to *manipulate or enhance* the content of an existing video, or to *produce* novel images. In this regard, the field of *Augmented Reality* [98, 99, 61, 43] made some of the first steps towards manipulating video content. The idea in Augmented Reality is to *augment* a real scene with a *synthetic* object (like an animation, for example) in such a manner that its presence in the scene is *geometrically* consistent through time (and hence images) with respect to the rest of the environment. An example of this step is to ensure that a particular animation looks like its *static* in a movie with respect to the surroundings. However, apart from the laws of image formation by perspective projection, not much constraint can be imposed on the *content* of the 3D world. Hence, in a general setting, there is a need to estimate the depth of *each and every* pixel of the world in order to manipulate the contents of a video satisfactorily. In this chapter, we look into the cases where the world or camera follow a specific *scene model* that allows the formulation of video manipulation algorithms with much simpler complexity and thus with much higher guarantees of accuracy. This class of problems have the term "Video Inpainting" or "Video Completion". In the next chapter, we extend our reasoning from the question "Can we manipulate a part of the video, generating novel video patches ?" to "Can we produce images that are entirely novel ?", which touches a field called *Image Based Rendering.*

The problem of video completion or inpainting deals with correction or replacement of selected parts of a video from content taken from the rest of the video. The underlying concept is that a set of images, along with the associated geometric quantities capture *all the essential 3D information of*

a scene. This information can then be used *in lieu* of the depth information for video manipulation. The fundamental problem underlying video completion is the registration of the frames of the video with respect to each other. Typical solutions define different ways of obtaining registration of local texture patches across frames. Although many approaches have produced good results, several assumptions about the camera motion or the input data need to be made, explicitly or implicitly. Additionally, most successful efforts include significant manual input. Problems like visual coherence of the output also need to be introduced in the formulation.

In subsequent sections, we first present our approach to the problem. We show how video completion can be posed as an unsupervised learning problem in the presence of noisy data. The definition of noise determines the exact structure of the algorithm, and two scenarios are presented to illustrate our approach. The first one consists of a planar object affected by illumination artifacts, and the second scenario consists of people moving in a 3D environment. Both situations are extremely challenging from the current stand point of the video completion community. Impressive results in both scenarios illustrate the efficacy of our approach.

6.1.1 Notation

Notations in this chapter differ slightly from the rest of the work in this thesis, due to the probabilistic origins for representation of video. Thus calligraphic quantities like $(\mathcal{V}, \mathcal{I}, \mathcal{P}, \mathcal{L})$ are used to represent a video, an image, the pose of the camera and a log likelihood function, respectively. Other conventions like using H to represent a homography, and using $\hat{}$ to represent estimated quantities, are standard. Finally, (Ψ, Φ) represent the 3D model and the registration function, while η represents noise.

6.1.2 Contributions

In previous chapters, we saw how MVG applications could be used in robotics for various tasks like tracking and servoing. In the following chapters, we focus on employing MVG techniques to the task of image and video processing. In this chapter, we focus on automatic manipulation of video content without any other external input from the user. The video is analyzed and patches to replace are determined, along with the determination of information from the rest of the video that will fill every patch to be replaced. The main contributions of this chapter include

- A framework that formulates the *detection* and *removal as well as replacement* of content in a video in a manner that is consistent with the overall geometry of the camera as well as the scene being viewed, with the requirement being that the user needs to define two functions that determine the behavior of the algorithm.
- Application of the framework in the case of a specific geometric construct (*homography*) in two distinct scenarios, the first one containing non-planar objects amongst planar ones that needs to be removed, and the second one containing dynamic objects that need to be detected and replaced by other static objects present in the background.

In the next chapter, we show how this concept may be extended to a related field called Image Based Rendering, and present some interesting results that may be obtained from this extension.

6.2 Related Work and Background

Traditionally, video manipulation has been thought to be the manipulation of three dimensional textures (spread in space and time). Given a video, with a space time "hole" to be filled, the idea is to

formulate this problem as that of minimizing a particular error, either using optimization techniques or otherwise. Texture based approaches rely on sampling techniques like the one developed by Efros et al. [100, 101]. Examples of such method include [102, 103]. In [102], the authors pose video completion as a global optimization problem which maximizes the a function over all patches in the neighborhood of the missing data. A video is thought off as a 3 dimensional texture, that has small "holes" in it that need to be filled. In order to determine the *most plausible* texture that fills each hole, the neighbourhood texture of the hole is analyzed. Particularly, the authors try to find a patch that has a similar neighbourhood as the "hole". Data from the closest patch is then transferred to "fill" the current hole. The searching of a patch and consequently the transferring of data is modeled as an optimization function. Alternatively, [104] uses a probabilistic video modeling approach. Large number of images are used to learn texture patches called "epitomes" that try to capture the essense of the texture. An epitome of a video patch is constructed by observing the distinct textures appearing in the video patch, which in turn is obtained by comparing different sub-patches of a video patch. The constructed epitome along with a function that mapes each part of the epitome to several similar patches in the original video can then be used to "edit" the video. Methods based on the structure of an image have also been proposed. In [105, 7], optical-flow based propagation is used to fill missing regions of images. In [7], the authors take a video sequence and compute the optical flow between frames (Section 2.6). Like [102], a global minimization is applied next to fill the missing patches. However, the minimization now happens in the *optical flow space*, instead of the texture space. This means that the actual texture patch is *never* transferred. Instead the correspondence properties of the similar region is applied to the current patch to borrow information from *its neighbours alone.* Interesting effects like video stabilization have also been shown. In [106], the authors employ PDE based approaches to capture the edge structure of a single image. This structure is then extrapolated to the missing regions of every image. Thus each image is restored independently, and information from multiple views is not used. In the other categories, the work of [107] uses a novel approach. They address video completion under varying illumination. The only assumption they make is of a static background and a cyclically moving foreground.

Current approaches concentrate on producing visually appealing results, neglecting the "correctness" of the results obtained. This severely restricts their applicability. For example, a video with multiple moving objects occluding each other is challenging because of the huge number of parameters that need to be considered. Ideally, however, we would like our formulation to be independent of such constraints. Thus it is desirable to have an algorithm that would a) automatically identify what parts of a video need to be replaced based on some cost function, b) be able to find physically correct patches to replace missing parts of a video, and c) inpaint objects of various and varying sizes and shapes.

To the best of our knowledge, the first two characteristics mentioned above are not handled by current video completion algorithms except [107], and the third characteristic poses problems to texture based approaches since non-parametric sampling techniques are sensitive to scale changes. Thus, scenarios are typically restricted to scenes where objects to be removed do not change scale significantly, or the camera movement is restricted.

We present a novel approach to the problem, by defining inpainting without manual intervention as an unsupervised learning problem. A single cost function specifies what parts of the video need to be replaced, and also identifies the appropriate physically correct replacement patches. In this paper, we consider two types of cost functions based on registration between views, and show how one helps in removing dynamic objects and the other in removing non-planar objects from a given scene.

This approach is along the lines of some recent papers [108, 109] which take a learning approach

towards geometric problems. In dynamic mosaicing [108], the basic problem is to generate a mosaic of the static scene in the presence of dynamic objects, which are treated as noise and need to be removed. This requires registration across scenes, as in our case. Super-resolution [109], on the other hand tries to estimate a high resolution video from a low-resolution one, and the problem is posed in a supervised learning framework.

6.3 Inpainting as Outlier Replacement

A video is a sequence of images produced by the projection of a 3D world onto a camera. Our interest lies in identifying and removing certain parts of this world from the images making up the input video. Thus, the input video and the true video may be represented as follows.

$$\begin{aligned} \mathcal{V}_I &= \{\mathcal{I}_{I1}, \ldots, \mathcal{I}_{IN}\} && (6.1)\\ \mathcal{V}_T &= \{\mathcal{I}_{T1}, \ldots, \mathcal{I}_{TN}\} && (6.2)\\ \mathcal{P} &= \{\mathcal{P}_1, \ldots, \mathcal{P}_N\} && (6.3)\\ \mathcal{V}_T &= \mathcal{P}(\Psi) && (6.4)\\ \mathcal{V}_I &= \mathcal{P}(\Psi) + \eta && (6.5) \end{aligned}$$

where $\mathcal{V}_T$ represents the video to be inferred, and $\mathcal{V}_I$ represents the input to our algorithm. $\mathcal{P}$ represent the camera viewpoints, Ψ is a function of the 3D world, and η represents noise, which may be blacked out or degraded pixels, illumination artifacts, effects of jerky camera motion or occlusion, occluding or dynamic 3D objects etc. Notice how our approach does not need to know the complexity of the occlusion. The only assumption we make is that the whole of Ψ is visible without noise, in parts, somewhere in $\mathcal{V}_I$. Also, the definition of Ψ changes with the problem under consideration. Finally, the solution boils down to the estimation and replacement of this noise, with data from the video. Thus, we pose the problem of video completion as extrapolation of a function after fitting the same to input data in the presence of noise. Several approaches to this problem may be found in the machine learning literature [110], and here we take the approach of maximum likelihood estimation.

The solution proceeds in three steps. Since we follow a maximum likelihood formulation, we first need to evaluate the probability that a candidate hypothesis $(\hat{\Psi}, \hat{\mathcal{P}})$ produces the given video $\mathcal{V}_\mathcal{I}$. Maximizing this likelihood over the space of $(\Psi, \mathcal{P})$, gives us the best hypothesis. Noise is then represented as outliers of the model $(\Psi, \mathcal{P})$, and is removed by extrapolating the model at appropriate points.

6.3.1 Estimating $\Psi, \mathcal{P}$:

Given a candidate hypothesis $(\hat{\Psi}, \hat{\mathcal{P}})$, the total probability of the observed video V_I is given as

$$\begin{aligned} p(V_I|\hat{\Psi}, \hat{\mathcal{P}}) &= \prod_i p(I_{Ii}|\hat{\Psi}, \hat{\mathcal{P}}) && (6.6)\\ &= \prod_i p(I_{Ii}|\mathcal{P}_i(\hat{\Psi})) && (6.7) \end{aligned}$$

where $\mathcal{P}_i$ represents the camera matrix corresponding to the i^{th} view. Thus, the probability that $\hat{I}_{Ti} = \mathcal{P}_i(\hat{\Psi})$ represents the "true" i^{th} view can be determined by maximizing the above equation. We assume $P(I_{Ii}|\hat{I}_{Ti})$ to be a Gaussian over pixel differences, with each pixel's contribution being independent of the others.

$$p(I_{Ii}|I_{Ti}) = \prod_{\forall x,y} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{I_{Ii}(x,y) - \hat{I}_{Ti}(x,y)}{2\sigma^2}\right) \quad (6.8)$$

In other words, every individual frame of the best hypothesis should explain the current image to the best possible extent. We proceed to minimize the corresponding log-likelihood function

$$\mathcal{L}(I_{Ii}) = -\sum_{\forall x,y} (I_{Ii}(x,y) - \hat{I}_{Ti}(x,y)) \quad (6.9)$$

$$\mathcal{L}(V_I) = -\sum_{i} \sum_{\forall x,y} (I_{Ii}(x,y) - \hat{I}_{Ti}(x,y)) \quad (6.10)$$

6.3.2 Noise estimation:

Once $(\Psi, \mathcal{P})$ are estimated from the data, the problem of estimating noise reduces to finding outliers that do not fit the model. Thus, for every frame i, we may classify pixels as noisy if they lie more than two standard deviations away from the predicted pixel color using the estimated values of $(\Psi, \mathcal{P})$.

6.3.3 Outlier replacement:

Given the model $(\Psi, \mathcal{P})$, outliers can be replaced by projecting the model onto parts of the image labeled as outliers. This may be thought of as extrapolating the learnt model to predict missing data.

6.3.4 Registration:

The problem of estimating $(\Psi, \mathcal{P})$ introduces registration into our framework. Registration of two frames of the input video $\mathcal{V}_{Ii}$ and $\mathcal{V}_{Ij}$ involves finding a correspondence between the frames, that best explains the visual data present in them. Thus, it may be represented by a function Φ, that takes $\mathcal{V}_{Ii}$ to $\mathcal{V}_{Ij}$. The function Φ is defined over $(\Psi, \mathcal{P})$.

$$\mathcal{V}_{Ii} = \Phi_{(i,j)}(\mathcal{V}_{Ij}) \quad (6.11)$$

$$\mathcal{V}_{Ij} = \Phi_{(j,i)}(\mathcal{V}_{Ii}) \quad (6.12)$$

The implicit assumption in this case, is that $\mathcal{V}_{Ii}$ and $\mathcal{V}_{Ij}$ have sufficient overlap of visual data, which is true for consecutive frames of a video. The main insight to note here, is that Φ is actually adequate to estimate and remove all the noise present in the input video. This is because we assume that (1) every part of the "true" video is present somewhere in the input and (2) that noise, unlike the rest of the visual data, is independently introduced in every frame of the video. In cases of occlusion, noise in frames are related, but not by the same function as the rest of the visual data. Thus, hypotheses of $(\Psi, \mathcal{P})$ may be replaced in Equation (6.6) by $\hat{\Phi}$. Additionally, since we do not know which frame of the input sequence contains the "true" image or its parts, Equation 6.7 has to be defined over all the images, for every image of V_I.

$$p(V_I|\hat{\Psi}, \hat{\mathcal{P}}) = \prod_{j} \prod_{i} p(I_{Ii}|\hat{I}_{Ij}) \quad (6.13)$$

In other words, every individual frame of the best hypothesis should not only explain the current image correctly, but should also be able to explain *all* the other images, when transferred through the registration function $\hat{\Phi}$. Thus, the overall log-likelihood to be minimized becomes

$$\mathcal{L}(I_{Ii}) = -\sum_{j}\sum_{\forall x,y}(I_{Ii}(x,y) - \hat{\Phi}_{(i,j)}I_{Ij}(x,y)) \tag{6.14}$$

$$\mathcal{L}(V_I) = -\sum_{i}\sum_{j}\sum_{\forall x,y}(I_{Ii}(x,y) - \hat{\Phi}_{(j,i)}I_{Ii}(x,y)) \tag{6.15}$$

Algorithm 6 3-step learning approach to completion

Input: Video $\mathcal{V}_I = [\mathcal{V}_{I1}, \ldots, \mathcal{V}_{IN}]$, with N frames.
Output: Learned video $\mathcal{V}_T = [\mathcal{V}_{T1}, \ldots, \mathcal{V}_{TN}]$.
{Step 1: Registration of frames.}
for (Every frame $\mathcal{V}_{Ii} \in \mathcal{V}_I$) **do**
  for (Every frame$\mathcal{V}_{Ij} \in \mathcal{V}_I$) **do**
    $\Phi_{(i,j)} = \text{register}(\mathcal{V}_{Ii}, \mathcal{V}_{Ij})$
  end for
end for
$\Phi = \left[\Phi_{(1,2)}, \ldots, \Phi_{(N,N-1)}\right]$
{Step 2: Estimate outlier data.}
for (Every frame $\mathcal{V}_{Ii} \in \mathcal{V}$) **do**
  $\text{outlier}_i = \text{estimate\_outliers}(\mathcal{V}_{Ii}, \Phi)$
end for
Step 3: Learn true data.
$\Psi = \text{estimateModel}(\mathcal{V}_{I1}(\text{inlier}_{I1}), \ldots, \mathcal{V}_{I1}(\text{inlier}_{IN}), \Phi)$
for (Every frame $\mathcal{V} \in \mathcal{V}_I$) **do**
  $\mathcal{V}_{Ti} = \text{replaceOutlier}(\text{outlier}_\text{i}, \mathcal{V}_\text{I}, \Psi)$
end for

6.4 Results and Analysis

In this section, we apply the theory developed earlier to two scenarios, which differ in their definitions of Ψ. This in turn defines noise, and hence determines what can be removed in each situation. The first scenario defines Ψ as a planar object and the second scenario defines Ψ as a bundle of rays. In each of these cases, we first compute SIFT [23] correspondences across frames, followed by a homography estimation based on the Gold Standard Algorithm [2]. Once pairwise homographies are computed, classification of each pixel of every image is done using homography based registration between frames. Computationally, the largest bottleneck is the feature extraction part which takes around 10 minutes for 500 frames of a video with resolution 640×480.

6.4.1 Scenario 1: Planar Object

Figure 6.1(a-f) shows different frames of a planar object being observed from different views. Illumination artifacts are observed due to the specular nature of its texture. For the case of a planar

Algorithm 7 3-step learning approach to completion: Planar object motion

Input: Video $\mathcal{V}_I = [\mathcal{V}_{I1}, \ldots, \mathcal{V}_{IN}]$, with N frames.
Output: Learned video $\mathcal{V}_T = [\mathcal{V}_{T1}, \ldots, \mathcal{V}_{TN}]$.
{Step 1: Registration of frames.}
for (Every frame $\mathcal{V}_{Ii} \in \mathcal{V}_I$) **do**
  for (Every frame $\mathcal{V}_{Ij} \in \mathcal{V}_I$) **do**
    $\Phi_{(i,j)} = \text{estHomography}(\mathcal{V}_{Ii}, \mathcal{V}_{Ij})$
  end for
end for
$\Phi = \left[\Phi_{(1,2)}, \ldots, \Phi_{(N,N-1)}\right]$
{Step 2 & 3: Learn true data, and estimate/remove outliers}
$\Psi = \text{estimateModel}(\mathcal{V}_{I1}, \ldots, \mathcal{V}_{I1}, \Phi)$
for (Every frame $\mathcal{V}_{Ii} \in \mathcal{V}_I$) **do**
  $\text{outlier}_i = \text{computeOutlier}(\Psi, \mathcal{V}_{Ii}, \Phi)$
  $\mathcal{V}_{Ti} = \text{replaceOutlier}(\text{outlier}_\text{i}, \mathcal{V}_\text{I}, \Psi)$
end for

scene, we define the registration function as a homography [2].

$$\Phi(i,j) = H(i,j) \tag{6.16}$$
$$\mathbf{x}_j = H(i,j)\mathbf{x}_i \tag{6.17}$$
$$\Phi(j,i) = H(j,i) = H^{-1}(i,j) \tag{6.18}$$

A homography between two frames may be computed from 4 accurate point correspondences. However, in the presence of noise, robust algorithms to estimate homographies exist [111]. Equation (6.17) describes the registration between points of the frames $\mathcal{V}_{Ii}$ and $\mathcal{V}_{Ij}$. Given pair-wise homographies, outliers are computed as points on the image whose colours are not consistent with other frames.

In practice, we do not evaluate over 255 grey levels, while computing Ψ to account for change in lighting conditions, and normalize each image before processing. The algorithm is summarized in Algorithm 7.

Experiment 1:

Figure 6.1(m-r) shows results of noise estimation and removal over a video sequence of a planar object. The camera moves arbitrarily over a board, while a light source produces a specularity on its surface. In this case, even the light source is in motion, though the board is stationary. However, our method equally applies to cases where any of the three objects are in motion. As can be seen, illumination artifacts are correctly identified (row 2 of figure), and replaced (row 3) to produce a video without the specular high light. The only input has been the nature of Φ. Everything else came out of the video automatically as the "noise".

Figure 6.3 shows results on the same scene, when different number of views are used to estimate and remove the noise in one particular frame of the sequence. In order to collect ground truth for this experiment, a frame without specularities was taken and view transferred by estimating homography using manually given correspondences. Intensity differences between this frame and the frames with noise removed, were used to derive the accuracy of our algorithm. As the graph shows, the accuracy reaches a saturation point after some threshold number of views.

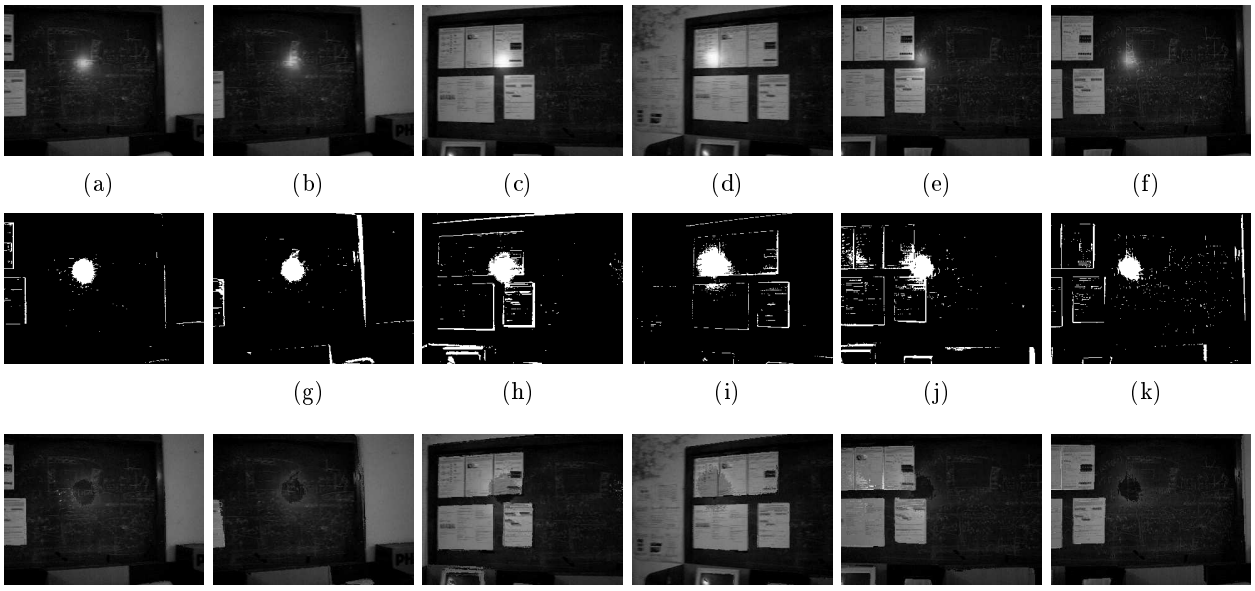


Figure 6.1: Frames from a video sequence of a planar object with a specular surface. The first row shows frames 40, 100, 150, 200, 250 and 300 of a 306 frame sequence. The second row shows the outlier detection result. The final row shows reconstructed images after outlier removal.

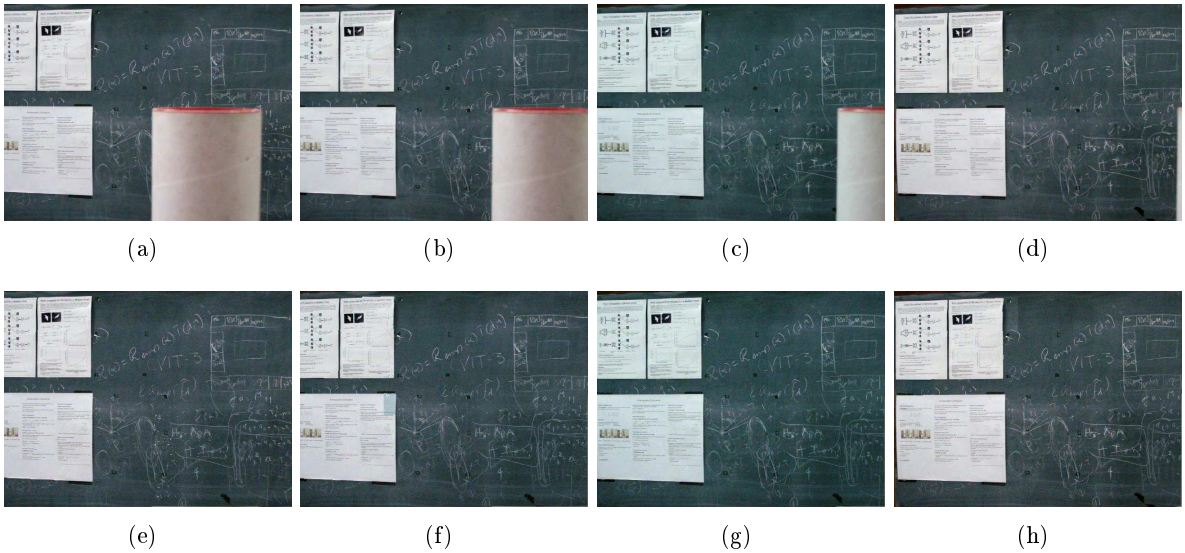


Figure 6.2: Results for a 3D object observed by a moving camera with a planar background. In this case, the 3D object is estimated as noise, and removed. Frames 120, 150, 18, 210 of a 300 frame sequence are shown.

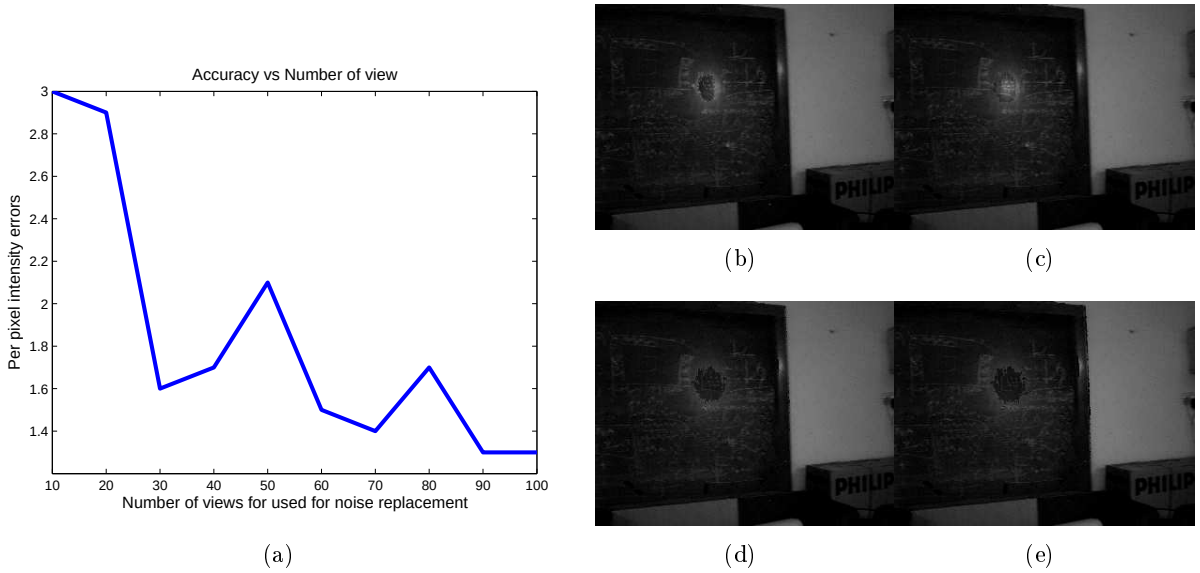


Figure 6.3: Figures on the left show removal of lighting artifacts when 20, 50, 80 and 100 views are considered to estimate outliers. The graph on the right shows decreasing error when compared to ground truth. The x-axis represents number of views, and the y-axis, per-pixel intensity differences.

6.4.2 Scenario 2: Rotating Camera

When a camera pans, different views capture the same bundle of rays corresponding to every 3D point observed in more than one image. Different views taken with such a camera are thus related by the infinite homography [2]. Unlike the previous scenario, however, the definition of noise changes in case of panning cameras, though registration between frames is still represented as a homography. Since the infinite homography explains objects present at any arbitrary depth, dynamic objects present in the scene are estimated as outliers, and hence represent the noise.

Figure 6.4 shows frames of a video with people walking. Notice how in this definition the number of people make no difference to our algorithm. The second and third rows show the estimated outliers and the corresponding recovered images. Also shows is a mosaic generated from the recovered images, and the noise, which may have further applications along the lines of [112].

6.5 Discussion

In this paper, we presented an approach to automatically identify and remove "noise" from a video, based on a function. We then showed how the video completion problem can be posed as the removal of outliers given a proper function to be satisfied by the completed video. The identification, removal, and completion are performed automatically with no interactive inputs. We demonstrated its application on several representative videos.

The main drawback of the approach is the need for a function to be satisfied by the "true" video. The constraint function could be simple homographies as in the examples shown here. Complicated videos with independent motion of the camera and multiple objects may be hard to handle as the the underlying constraint functions are complex. Our algorithm, being unsupervised, also requires

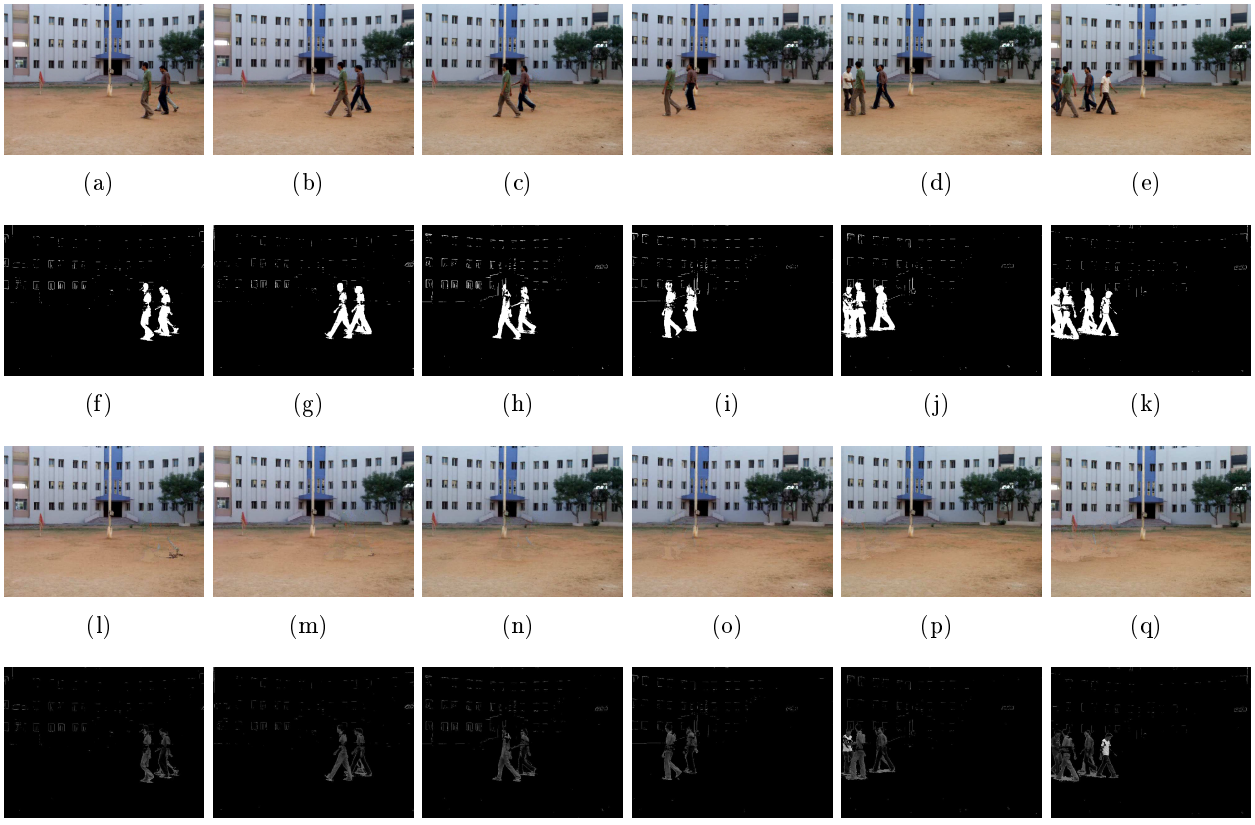
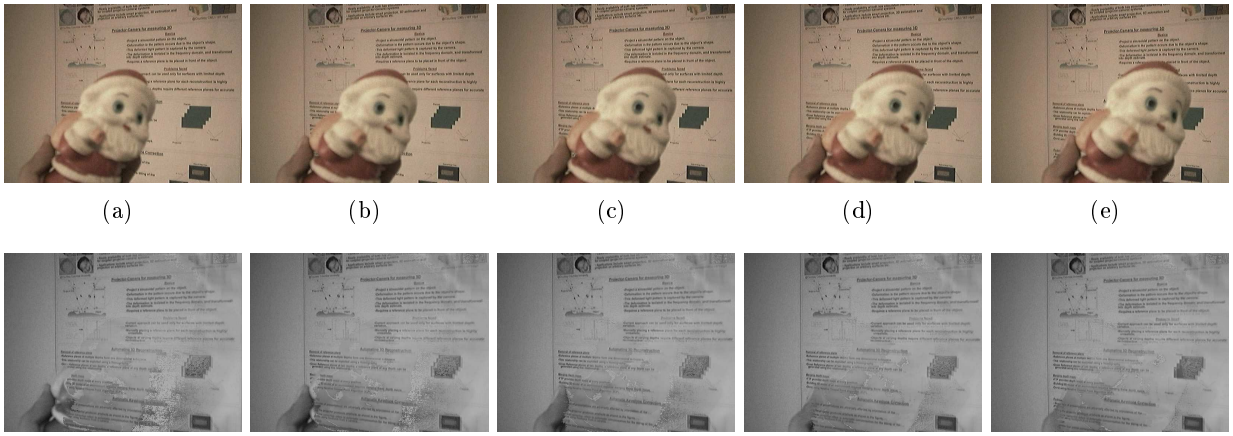


Figure 6.4: Frames 67, 70, 80, 90, 100 and 110 of a 200 frame video taken by a panning camera with multiple people walking. Notice how the last image has 4 people cluttered together. The third rows shows the estimated foreground.

(a)

Figure 6.5: A mosaic of the "true" video using the method of [5].



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

(f) (g) (h) (i) (j)

Figure 6.6: Various occlusions of a planar scene by a moving object. Two artifacts are observed. Since sufficient information is not available in the video, some part of the foreground remains. Secondly, outlier replacement is done by averaging over all the inliers in other frames. This leads to over-smoothing of the result, an undesirable trait.

the true video's pixels to appear more frequently than the noisy ones. Artifacts can be seen in Figure 6.6, where sufficient frames were not available for identifying and replacing pixels. The best replacement pixel might not be easy to determine as multiple candidates obeying the model can exist. Figure 6.6 shows results for one such replacement strategy.

However, we believe automating the completion process is possible and advantageous in many situations. In addition, the machine learning framework allows for further extension to a variety of scenarios, involving dynamic 3D scenes with occlusions.

6.6 Summary

In this chapter, we proposed algorithms for video manipulation that harness the power of MVG to detect and manipulate pixels that are *outliers* of a user specified objective function. The resulting videos and images are smooth in the sense that the objects are removed and resulting video patches are filled to ensure a seamless view of the entire scene. We demonstrated our approach on two different kind of scenarios with appealing results. The approach shows a lot of promise; however, it is not without drawbacks. Inherent drawbacks and possible future extensions are discussed in Chapter 8.

The key idea in this chapter was that 3D information of a scene embedded in its images can be *extracted and manipulated* for certain scene models using MVG without *directly* inferring the depth of the objects. The major assumption in this idea, is that the *entire* information of the 3D scene is *actually embedded* within its images captured by the camera. However, this might not be the case when phenomena like occlusion happen. Thus, in the next chapter, we proceed to answer the question "How should a scene look like from a viewpoint, when we do not have enough information about it ?". Note here that in the absence of *actual* information, it is impossible to predict how a novel view will look. What is possible, however, is to predict how a scene *can look*, by looking at *semantically similar* scenes. This is the central issue that we address in the next chapter.

Chapter 7

View Synthesis

7.1 Introduction

In the previous chapter, we showed how MVG could be employed for video manipulation, where the need was to *identify* and *replace specific parts of the scene*, which corresponded to identifying and replacing small to large sized patches in images of the video. In this chapter, we extend our exploration to answering the question "Is it possible to fill a *whole* image instead of just its parts ?". Typical applications of such a field of research is in producing immersive experiences for users, whereby limited data is sufficient for generating a comprehensive browsing capability for the user since new views not captured earlier can be *generated* using algorithms produced while answering this question. The difference between approaches presented in the previous chapter comes in the fact that in our case, we *do not* have complete knowledge about the scene embedded in its images. This means that, we have to answer the above question by showing that information to *fill* a *novel* view need not just come from other images of the *same scene,* but also from other images of *similar scenes.* To our knowledge, such a question has not been considered earlier. For demonstration purposes, we take the example of view synthesis for face images, since faces are a well studied subject in terms of the semantic understanding of content.

Given a set of images of an object/scene from known camera positions, Image Based Rendering (IBR) tries to predict the most likely image from a new view. IBR has been a widely researched area in recent years [6, 113]. The main underlying thread of all the algorithms in the literature has been to model the problem as a problem of *inference,* which answers the question "which image's pixel does the color of this pixel come from", for *every* pixel of the output image. This question has been answered in myriad different ways as will be seen in Section . Since typical applications of IBR include virtual reality, animations, etc., generating high quality images is of prime importance. Accomplishing this task is difficult because it requires complete knowledge of the geometry and photometry of the object. Additionally, producing high quality results seem to require elaborate camera setups [114] or produces visible holes in the novel view because of occlusion [113]. In this chapter, we present a method to compute novel views of *nearly* symmetric objects from positions impossible to current methods: views that require information *not* available in the input sequence. This is done by transferring semantic information from a database of "similar" objects, for which it is affordable to capture multiple views with precision.

7.1.1 Notation

We use the following notation in the rest of the chapter. p is used to denote the probability event. The capitals ($\mathcal{I}$, $\mathcal{S}$, $\mathcal{D}$) represent the image, the semantic scene and the depth map of a

scene respectively. W represents a weight matrix and Ψ represents a robustness function to ensure convergence [13]. Finally, F is used to represent a feature vector.

7.1.2 Contributions

In the previous chapter, we say how MVG could be used to formulate algorithms for the automatic manipulation of video content. Particularly, we saw how algorithms to transfer data across several images of a scene could be developed. We extend the concept of manipulation in this chapter by introducing methods to manipulate image content, by transferring data not just from the same scene but from *different* scenes with an underlying semantic commonality. The focus of this chapter is to produce novel images, by transferring *semantic* information from a database. Specifically, our contributions include

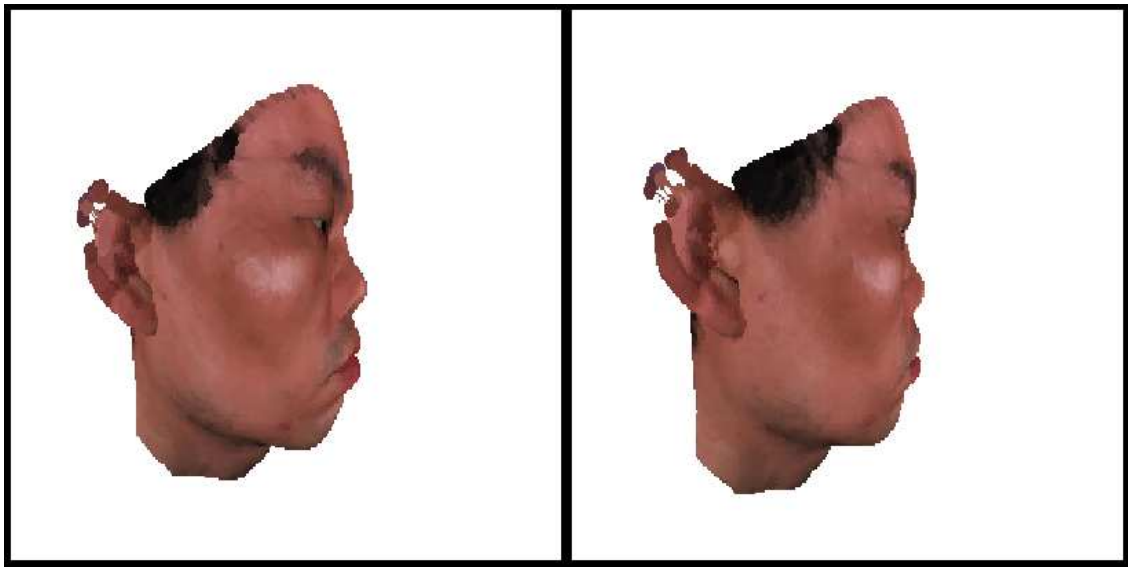
- An algorithm for matching a given scene with a database of *similar* scenes, using a suitable representation and matching function.
- An algorithm for transferring information from *semantically* similar scenes to the input scene, in order to *complete* it and produce *novel* views (views not captured by the camera).
- Two sets of semantic characteristics that can be transferred from scene to scene.
- A novel method for obtaining non-linear registration between two images.
- A method for producing *different variations* of a given scene, with variations for the unknown elements smoothly (*i.e.* in a visually pleasing manner) inserted.

Our results show that the current set of manipulation algorithms hold sufficient promise, and we discuss relevant issues for the future in Chapter 8.

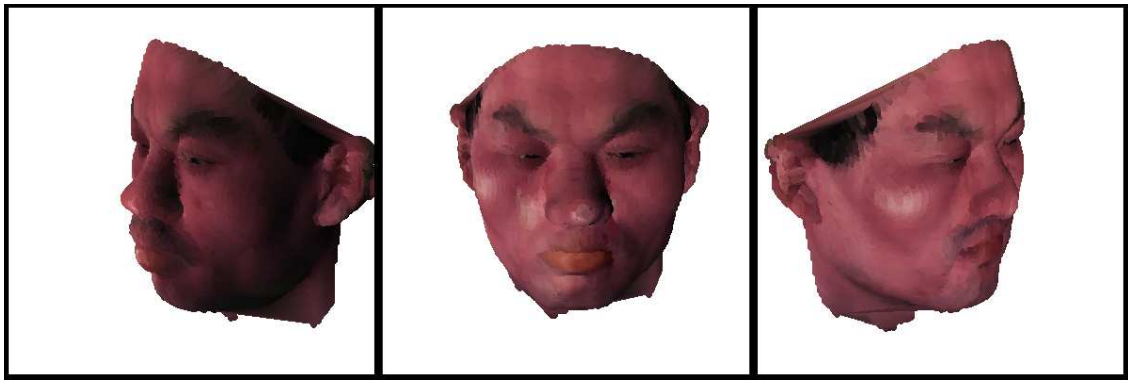
7.2 Related Work

IBR methods pertaining to 2 basic categories have been proposed earlier. The first set is to synthesize new views from few reference images using stereo vision techniques. The aim of methods following this approach is to construct correct synthetic views for a wide range of viewpoints (in the vicinity of the original views). These methods can be further subdivided into those that construct new views directly from the given images, and those that first build an explicit 3D model from which new views can be rendered using traditional computer graphics techniques like texture mapping. The first category includes methods like those of Wexler et al. [6, 115]. In [6, 115], the authors propose a method of transferring image patches to a novel view, through the estimation of epipolar geometry. Texture patches are then transferred by employing a texture database that consists of texture patches collected from different images of the scene. This reduces the search space for approipriate color for every color of the novel view. Techniques based on stereo reconstruction [113, 116] and volumetric techniques [117] fall in the second category, and rely on accurate and complete knowledge of the geometry of the scene for rendering purposes.

In contrast to these geometry based approaches, methods based on sampling the Plenoptic function [114] aim at capturing all rays required for rendering views within a specified 3D volume. Previous approaches include rendering algorithms that capture light rays within a 2D circle, within a 3D rectangular block, in an arc etc. Here, geometry is implicitly used to reconstruct the most probable texture/color given a view point. Such methods combine the precision of geometry and



(a) (b)



(c) (d) (e)

Figure 7.1: Top row: A sample of the set of images given as input to our algorithm. Notice that only one side of the face is shown. Bottom row: Reconstruction results showing both sides of the face. Each of the images in the bottom row corresponds to texture information for the nose transferred from different individuals. They all have the same symmetry information.

the realism of methods based on Plenoptic modeling. In this regard the work of McMillan et al. is important. McMillan and Bishop [118] introduce the term image-based rendering (now widely used for the field of view synthesis from real images). They describe a method for synthesizing new views from two cylindrical panoramic views created by mosaicing [119]. Image-based rendering is characterized as "reconstructing a continuous representation of the plenoptic function from a set of discrete samples of that function." Disparity maps are computed between adjacent cylindrical panoramas using a cylin- drical variant of the epipolar constraint, and new views are synthesized by warping (forward-mapping) the existing panoramas based on the disparities. The method does not deal with partially occluded points, and holes in the new view are simply filled by interpolation [120].

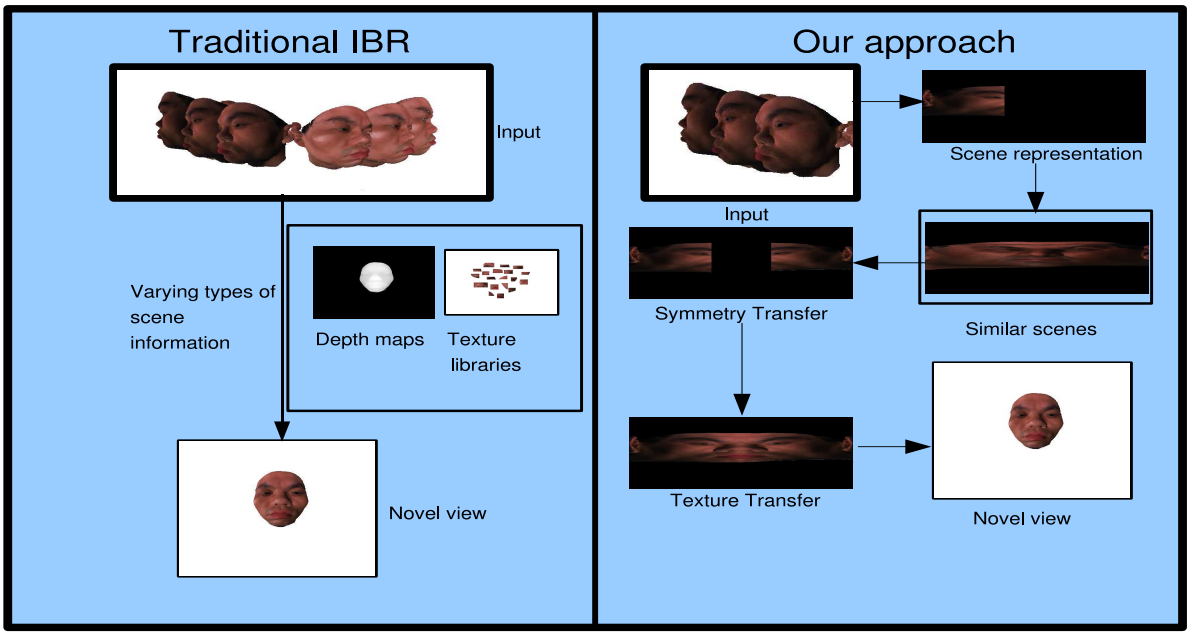
These techniques, however, suffer from the information bottleneck of the input sequence. Essentially, all the above methods are ways to represent the information of the novel view in terms of information available *only* in the input image/depth map sequence. By augmenting the current scene using "similar" scenes that might be available to us, this bottleneck may be overcome. Advantages of such an approach are three-folds. Since information from multiple sources is used to generate a novel view, potentially we have more data than current approaches. Thus, the gamut of novel views possible using the current system is be larger than any of the previous approaches. Secondly, only the database of "similar" scenes needs to be carefully captured. Thirdly, in the absence of knowledge about the scenes *actual* content, several alternatives may be substituted (Figure 7.8) from these similar scenes to produces different versions of a novel view, all consistent with the given input. Effectively, where other IBR approaches describe algorithms for view interpolation, we describe an algorithm for *view extrapolation*: the generation of plausible views given an input image sequence.

Image based applications that use semantic information from a database have gained popularity recently [121, 122]. Semantic information was used to fill holes for image inpainting [121, 122]. They transfer texture patches from similar scenes so as to produce full images that are also semantically correct. Although only texture information is transferred across images, such an approach opens up exciting possibilities in many areas. In this paper, we show how such transfer can be applicable to IBR, by posing view synthesis as an image completion problem. We transfer information about the *symmetry* of the object across images, along with texture information, to extrapolate views from the input sequence impossible to current IBR methods.

We demonstrate our approach using a representation that is a hybrid of Plenoptic modeling and stereo reconstruction based methods, although it uses more depth information than other hybrid approaches [123]. Our preference for this approach as our base is because it allows us to easily model semantic information about scenes, without having to restrict the rendering capability. Sections 7.3 discuss the problem and our approach, and we show how the representation of [123] allows us to transfer information like symmetry and texture across objects in Section 7.4. Details of implementation are presented in Sections (7.4,7.4.1) and results in Section 7.5. Finally we enumerate future directions of research and conclude with Section 7.6.

7.3 Problem Definition

We start with an objective function similar to [6]. Given a collection of n 2D images $\mathcal{I}_{1\ldots n} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ and their corresponding depth maps $\mathcal{D}_{1\ldots n} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ of an object, we wish to infer a new view $\mathcal{V}$, all of whose information cannot be supplied by the input images and depth maps. Additional information in the form of knowledge from similar scenes is available to us. Currently, let us denote this information by $\mathcal{S}_{1\ldots m} = \{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$, where m is the number of scenes available.



(a)

Figure 7.2: Traditional IBR vs Our Approach. While apart from the input image sequence, traditional IBR methods might use depth maps, texture patches or mosaics of the *same* scene, the input to our algorithm is MCOP images of *similar* scenes. This requires transferring semantic properties of the objects like symmetry from the input MCOP images to generate the novel view.

The most likely rendered view $\mathcal{V}$ would then be given by

$$p(\mathcal{V}) \quad = \quad p(\mathcal{V}|\mathcal{I}_{1...k}, \mathcal{D}_{1...k}) \cdot p(\mathcal{I}_{1...k}, \mathcal{D}_{1...k}) \tag{7.1}$$

where $(\mathcal{I}_{1...k}, \mathcal{D}_{1...k})$ represents the set of all images and depth maps needed by methods like [113] to render the novel view $\mathcal{V}$. The camera positions of these scenes are assumed to be chosen, based on the view to be rendered and the camera positions of the input images. Since, $k > n$, we need to supplement information provided by the set $(\mathcal{I}_{n+1...k}, \mathcal{D}_{n+1...k})$ from the set $\mathcal{S}_{1...m}$.

We use a Multiple Centre of Projection (MCOP) image [123] to represent the information about the geometry of the symmetric object (instead of the fundamental matrix used by [6]), for reasons explained in Section 7.4. This results in the modification of Equation (7.1) to give

$$p(\mathcal{V}) \quad = \quad p(\mathcal{V}|\mathcal{I}^{mc}, \mathcal{D}^{mc}) \cdot p(\mathcal{I}^{mc}, \mathcal{D}^{mc}|\mathcal{I}_{1...n}, \mathcal{D}_{1...n}, \mathcal{S}_{1...m}) \tag{7.2}$$

The term $p(\mathcal{V}|\mathcal{I}^{mc}, \mathcal{D}^{mc})$ is defined by Equation (7.5). However, semantic information about the scene may be introduced in the second term in Equation (7.2). Expanding it gives us

$$p(\mathcal{I}^{mc}, \mathcal{D}^{mc}|\mathcal{I}_{1...n}, \mathcal{D}_{1...n}, \mathcal{S}_{1...m}) \quad = \quad p(\mathcal{I}_{1...n}, \mathcal{D}_{1...n}, \mathcal{S}_{1...m}|\mathcal{I}^{mc}, \mathcal{D}^{mc}) p(\mathcal{I}^{mc}, \mathcal{D}^{mc}) \tag{7.3}$$

Here, we assume equal prior for MCOP images. Since we don't have a parametrization of the semantic space, only samples at various points, we do not want the semantic information incorporated into the Maximum Likelihood MCOP image to be different from all the images in the database. Thus it makes sense to formulate the likelihood function such that it penalizes deviation of the semantic information from the database. Again, since we also want to preserve information from the input images, we define the likelihood as

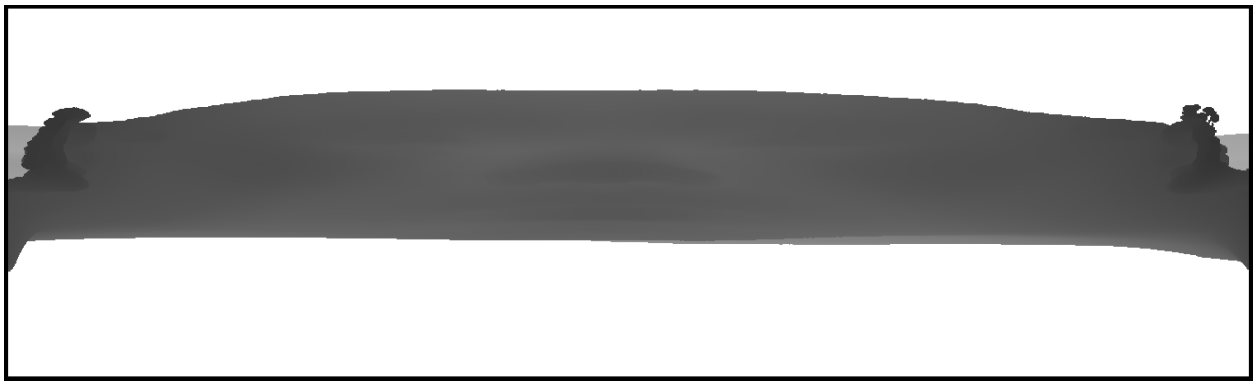
$$p(\mathcal{I}_{1...n}, \mathcal{D}_{1...n}, \mathcal{S}_{1...m}|\mathcal{I}^{mc}, \mathcal{D}^{mc}) \quad = \quad \min_{\mathcal{S}_i} f(\mathcal{I}^{mc}, \mathcal{I}^{mc}_{inc}, \mathcal{S}^{mc}_i) + s(\mathcal{I}^{mc}, S^{mc}_i) \tag{7.4}$$

where $f(\cdot, \cdot, \cdot)$ is a pair-wise sum of the Euclidean distance in a suitable feature space, and $s(\cdot, \cdot)$ is a measure of the similarity in semantic information between the two input images. $\mathcal{I}^{mc}_{inc}$ represents the incomplete MCOP image that can be formed from the input sequence $(\mathcal{I}_{1...n}, \mathcal{D}_{1...n})$ and S^{mc}_i represents the MCOP image of the ith scene in the database. In the following sections, we show two ways of introducing semantic information into this approach: using symmetry of scenes and texture similarity.

7.4 Approach

Equation (7.1) can be solved for $(\mathcal{I}_{n+1...k}, \mathcal{D}_{n+1...k})$ using texture and structure similarity between the scenes $(\mathcal{S}_{1...m})$ and the input images $(\mathcal{I}_{1...n})$, along with the correspondence information within the input images itself. Matching of range images of partial views of objects with a database of objects to account for occlusions, has been earlier used in 3D object recognition [124] and reconstruction [125]. We adopt a different strategy for two reasons. Firstly, this is a large optimization problem involving as many variables as there are pixels in the new views needed to be synthesized. It would rather be easier to directly generate the novel view $\mathcal{V}$ instead of generating these intermediate images and maps. Secondly, reconstruction of texture in this manner is a tricky issue since regularization is required to avoid high frequency artifacts, as shown in [6].

(a)



(b)

Figure 7.3: Sample MCOP texture and depth images. These images were taken by a camera moving in a circular arc around the face, although any other path symmetric about the axis of symmetry of the face, could be used. This image is constructed by concatenating the central column of each of these images in a sequential manner, as the camera moves through the scene. Notice how symmetry information may be extracted by registering the different parts of the texture image.

We use methods from the Plenoptic modeling literature to find a representation of the *entire* scene using a single image. Such images have been given many names, one of them being Multiple Centre of Projection (MCOP) Images [123], and are the collection of light rays and range maps computed using a slit camera and a laser range finder, moving along a pre-defined path through the scene. This representation is useful for three reasons. Firstly, as explained in Section 7.4.2 it allows us to model symmetry. Secondly, since such images need to be collected only for the scenes in the database, the user-defined camera paths may be modified to suit a variety of requirements. Some of them may be to have a compact representation of the complete scene, or to capture a specific semantic property of the scene like symmetry etc. Thirdly, such images represent the texture information of the *entire* scene and hence any incomplete MCOP image can be "filled" using standard texture transfer approaches [101]. The 3D model to be rendered is now given by the equation [123]

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \delta_{i,j} \begin{bmatrix} U_{i_x} & V_{i_x} & O_{i_x} \\ U_{i_y} & V_{i_y} & O_{i_y} \\ U_{i_z} & V_{i_z} & O_{i_z} \end{bmatrix} \begin{pmatrix} i \\ j \\ 1 \end{pmatrix} + \begin{pmatrix} C_{i_x} \\ C_{i_y} \\ C_{i_z} \end{pmatrix} \tag{7.5}$$

where for each pixel (i, j), C_i represents the centre of projection, O_i represents a vector from C_i to the image plane origin, U_i represents the horizontal axis of the projection plane, and V_i represents the vertical axis. Each pixel's depth is represented by $\delta_{i,j}$.

7.4.1 Algorithm

Since the database used to transfer semantic information contains a finite set of objects, Equation (7.4) can be minimized by considering all possible combinations of MCOP images that can be formed from the input images and the database. However, such a solution is computationally costly, and not scalable. Instead, we find that an alternate approach gives reasonable results, while being computationally quite fast.

In order to minimize the function $f(\cdot, \cdot, \cdot)$, we pick the MCOP image from the database such that it is similar in structure and depth over various scales around a fixed radius around the holes in the incomplete MCOP image $\mathcal{I}_{inc}^{mc}$. First, a feature vector containing gray value and gradient information for each MCOP image (texture and depth) is constructed. To ensure that color similarity is not given importance, a weighted cost function of the feature distance between input and database images is minimized.

$$\min_i \sum_s \sum_\Omega W * (F_{input}^s - F_i^s)^2 \tag{7.6}$$

where s denotes the scale, Ω denotes the region of interest, W denotes the weight matrix and F denotes the feature vector for each pixel of the image. Then, to ensure that the function $s(\cdot, \cdot)$ is obeyed, we directly transfer semantic information from the image chosen by minimizing equation (7.6) to the incomplete MCOP image. Such a nearest neighbour based approach also allows us to pick between the first k neighbours, allowing for variation in the semantic information, since the actual content of the input image is unknown.

7.4.2 Symmetry Transfer

Symmetry of 3D objects about various axes is a major cue in 3D computer vision to recognize and reconstruct partially occluded objects [124, 125]. The use of symmetry in images is however

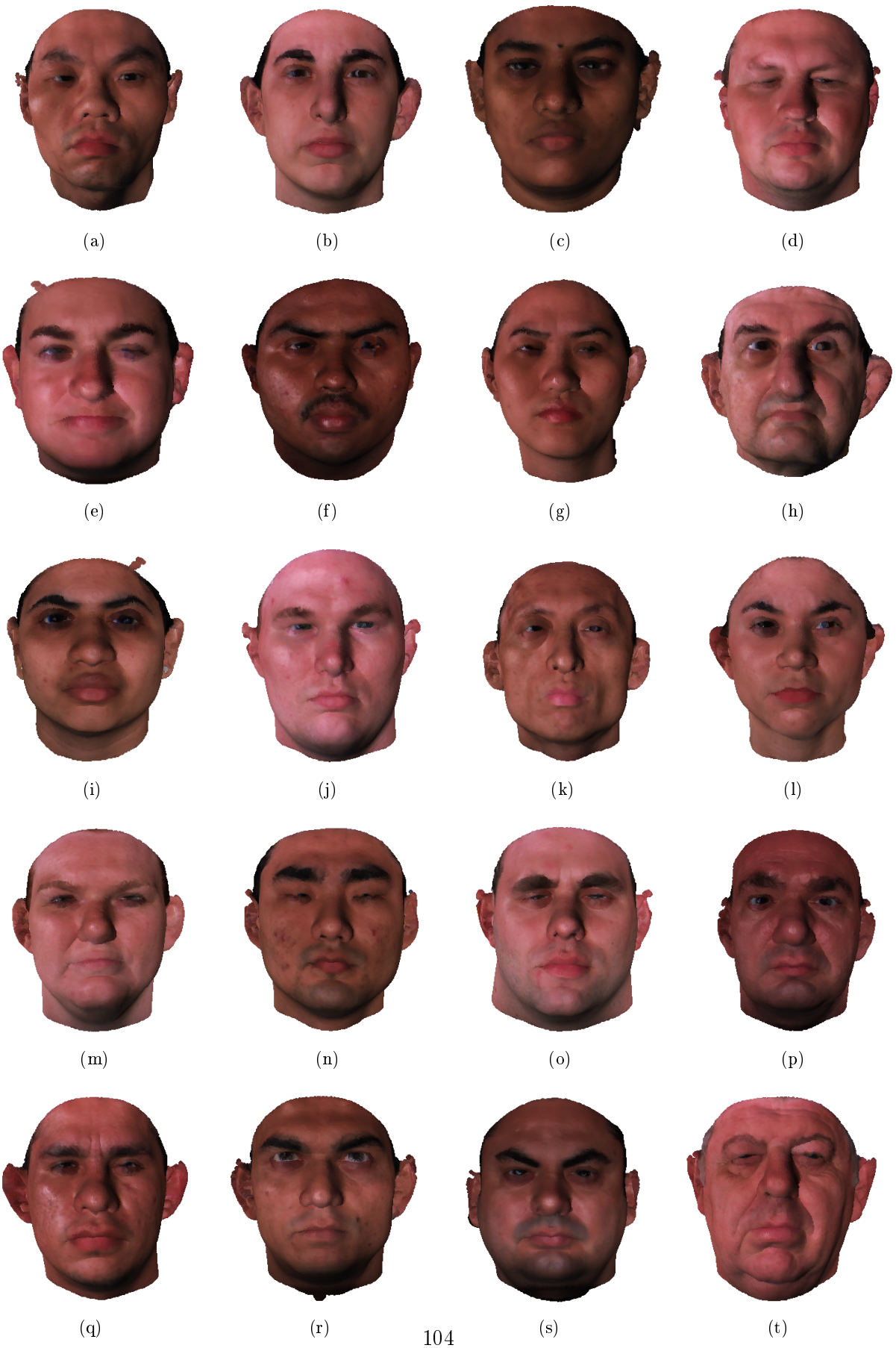


Figure 7.4: A Sample of the Face Database [12], used for our experiments

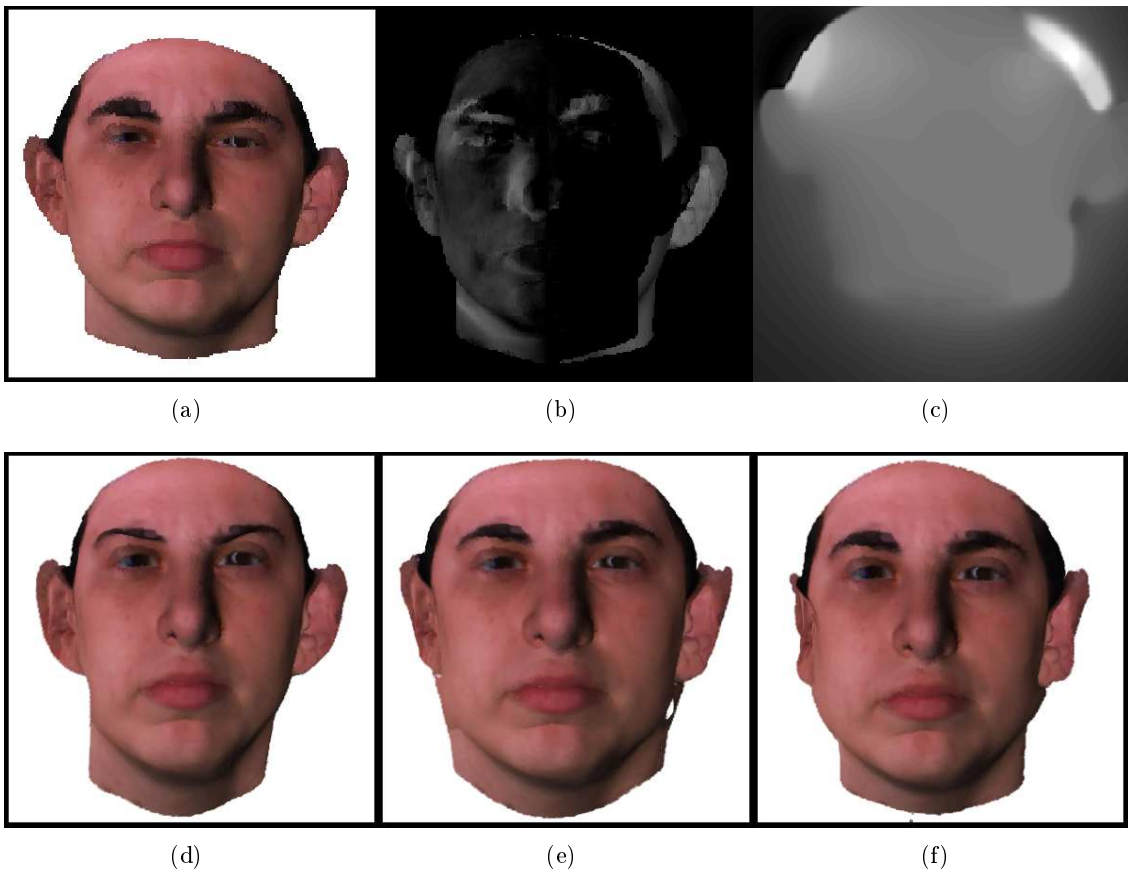
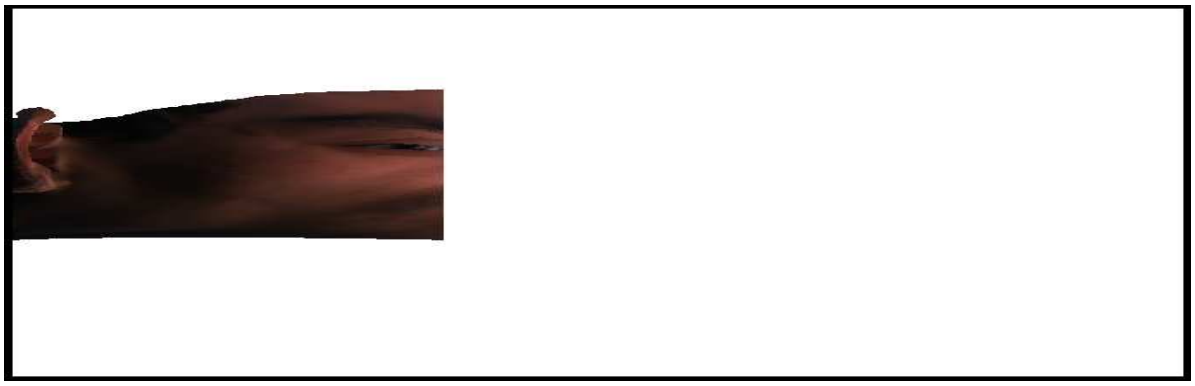


Figure 7.5: Symmetry extraction: Top row: Original image, difference between original and flipped images, optical flow information between the original and flipped version computed using a variation of [13]. This flow information is used to register an image with its symmetric counterpart. Bottom row: A frontal image formed by using symmetry (or asymmetry) information from different faces. Please refer to the attachment for the high resolution version.

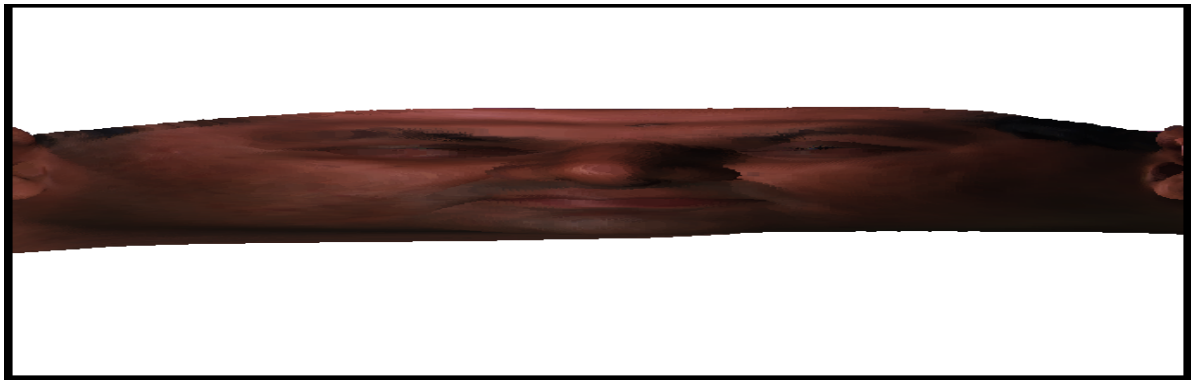
restricted, since symmetry is essentially a 3D property of objects. Perspective projection has adverse effects on the symmetry of an object, depending on the type of symmetry (axial, plane based, line based etc). However, symmetry is a useful cue for view synthesis since, depending on the type of symmetry, only partial views of the object contain information to generate views from any position.

We exploit the property of symmetry in images by employing concepts about the Plenoptic function. Any symmetry in a 3D object is captured by its corresponding Plenoptic function. Since MCOP images capture a sample of the Plenoptic function, by cleverly designing the path of the camera employed to capture the desired MCOP image, we may capture the symmetry property in an image. For an axially symmetric object, an MCOP image captured along a camera path symmetric with respect to the axis of symmetry of the object itself will capture its properties in the image. For example, a sample MCOP image of a face captured by a camera moving along a circular arc with the face at its centre is shown in Figure 7.3. Observe how all the symmetry properties of the face is captured by the image. Another example of plane symmetry may be found in [123].

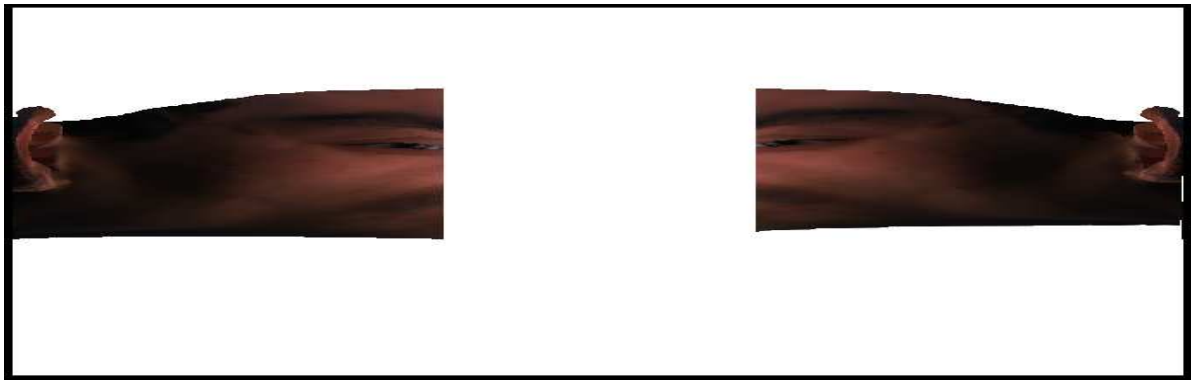
Real world objects are never perfectly symmetric. This is illustrated in Figure 7.5, which shows the slight asymmetry in a face. This asymmetry is sometimes different for objects of the same class (eg. faces). By registering the different parts of the object with each other, this slight deviation



(a) Input

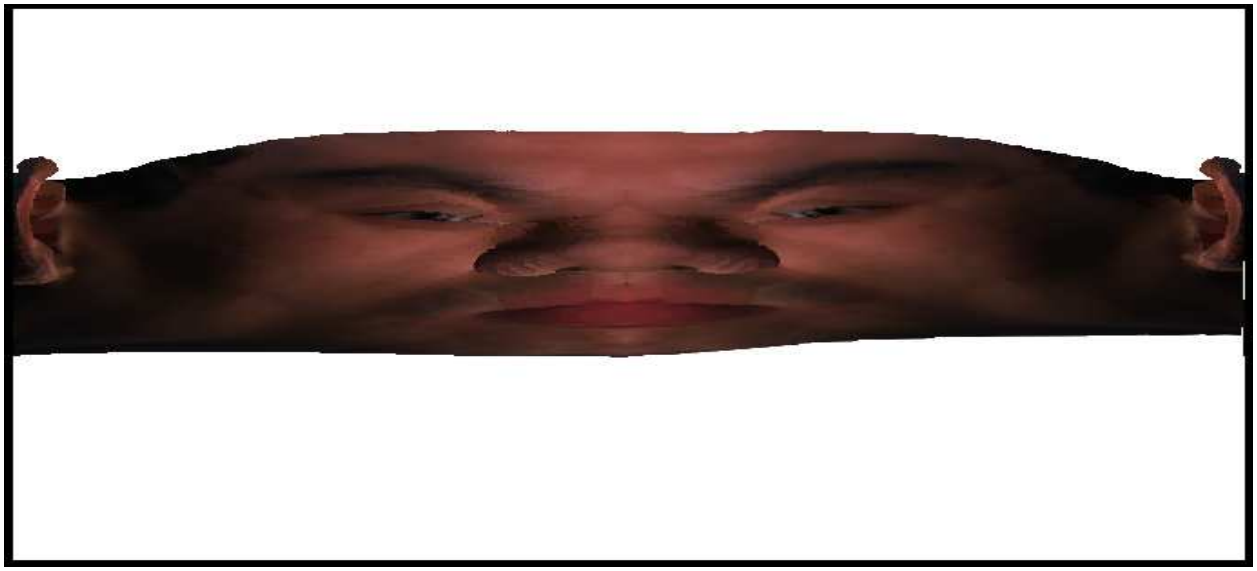


(b) Database Image



(c) Symmetry Transfer

Figure 7.6: Symmetry Transfer: (a) The input to symmetry based hole filling. (b) The database image used to transfer symmetry. (c) The generated result.



(a)

Figure 7.7: Texture Transfer: Result of grafting a nose onto the result in Figure 7.6(c).

from symmetry may be captured accurately. We register an MCOP image with its mirror image by using a variation of the optical flow algorithm in [13] presented in [31]. The only modification we add is to weight each of the different channels according to our need. In essence we try to find the flow that minimizes the following functional

$$Flow(u,v) = \int_{\Omega} (W * \Psi(I^{[k]}(\mathbf{x}+\mathbf{w}) - I^{[k]}(\mathbf{x})))^2 + \Psi(|\Delta u|^2 + |\Delta v|^2) \tag{7.7}$$

The different channels are gray value, R-G channel, R-B channel, and image derivatives (x and y). We use a weight of 10.0 for the derivative channels and a weight of 0.15 for the gray value channels. This is to ensure that any changes in lighting conditions do not influence the computed flow. The final linear equations are minimized using SOR templates [1]. Laplacian pyramids are created till the maximum image dimension is less than 25 pixels. A scale change of 0.95 between successive levels is chosen.

Once this registration information is given, images of the whole object may be generated from its partial views. In the absence of this information, it may be borrowed from other objects of the same class, as shown in Figure 7.5.

7.4.3 Texture Transfer

The MCOP images generated using few images will contain holes (Figure 7.6). Symmetry based methods can be used to fill some of these holes. In some cases however, enough information is not available for symmetry based methods to produce novel views without holes (Figure 7.6(c)). Thus, methods based on texture transfer from other images have to be employed. Given the closest MCOP image to the input as per Equation (7.6), this involves two tasks: computation of the closest texture patch to transfer texture from, and appropriate blending of the transferred texture with the input image.

In order to find the closest texture patch, feature vectors are constructed at various levels of the Laplacian pyramid out of the texture descriptors of the input and database image, with a scale change of 0.5 between levels, the lowest level having maximum dimension size of 100 pixels. These pixels are then matched with the database image and the SSD based score at multiple levels is used to find the most likely patch to transfer texture from. As mentioned in [121], such a simple measure is enough to determine the closest match accurately.

Like [121], we use to method of [27] to blend the textures from different images into the current one (Figure 7.7). Unlike [121, 122], pixels belonging to the MCOP image are considered hard constraints and are not modified in the process, since we want to preserve all information actually belongs to the scene. To ensure consistency, all the texture transferred to fill a hole is taken from one image. Finally, textures exported from the selected image are blended into the current image using the Poisson solver [27] available from [126].

7.4.4 Combining Approaches

Texture synthesis based approaches do not always produce meaningful results [121]. Mainly, when large holes in the MCOP images are created because, say, only one half of a face is captured, texture filling approaches may not produce desirable results. Thus using other information like symmetry along with hole-filling is a potent combination to address problems in image content. Figure 7.7 shows one such result.

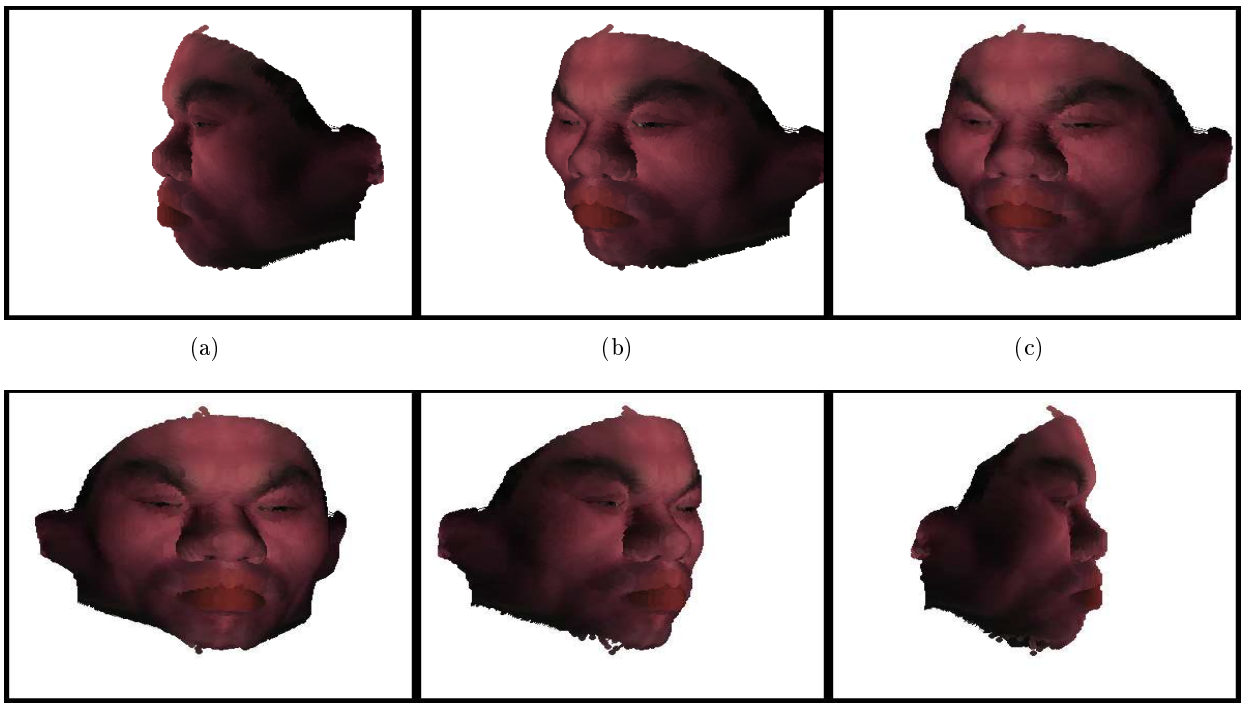
7.5 Results

Figure 7.1 demonstrates the results of symmetry transfer. In this case an MCOP image was created out of data captured by using a single depth map/texture map of the face. The depth map, of a profile view of the face was projected onto a 3D coordinate system, and an MCOP image was captured by rotating the camera in a circular arc around the face. This image was then completed by the symmetry information available from the closest matching face from the database. The fully formed MCOP image was then used to create the remaining views.

Results for symmetry texture synthesis are shown in Figure 7.8. Here, the nose of the subject was removed from the depth map before capturing its various views into an MCOP image. Figure 7.6 shows the MCOP image after symmetry information was transferred from the database. The remaining texture gap was then filled using the subject in Figure 7.7. Novel views of the subject are then rendered (Figures 7.8(a-f)). High resolution images of the rendering as well as the original images of the person from the same view are available in the attachment for comparison.

An advantage of such augmentation of information is the variety of modifications that are also available to the user. Figure 7.9 shows one such result. We call this "symmetrizing" an object, in the sense that the inherent asymmetry present in the object is removed. This is done by computing the optical flow between an MCOP image and its mirror image, and *warping* the MCOP image with the flow. Then, the warped MCOP image is merged with its intact mirror image to produce a "symmetric" version of the person. Essentially, this is equivalent to using half of the MCOP image, similar to the result presented in Figure 7.1. The difference here, is that symmetry information is not transferred from the database, it is supplied from the same object. Also, the symmetry information is inverted. Figure 7.9 compares the views of the symmetric object with its asymmetric version.

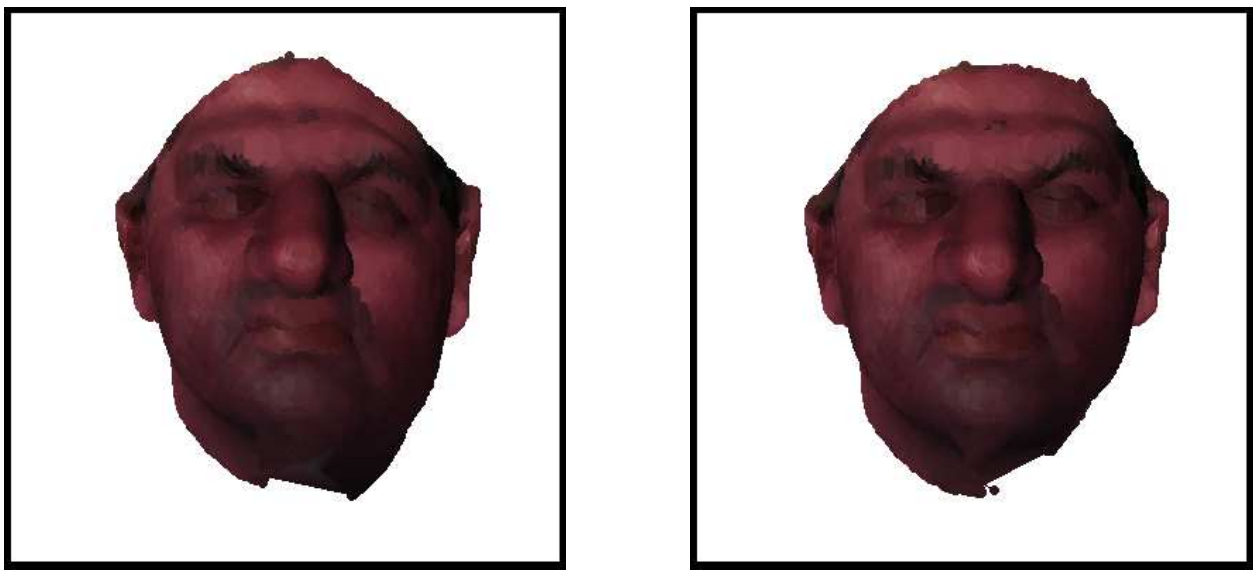
Finally, any missing information has several alternate sources to be borrowed from. Figure 8.3 shows the variations of the same object produced with information taken from the first 5 nearest neighbors. As can be seen, different sources not only produce change in texture information, but



(a) (b) (c)

(d) (e) (f)

Figure 7.8: Symmetry and Texture transfer: Novel views of the scene generated in Figure 7.7.



(a) (b)

Figure 7.9: (a): Symmetrized result. (b) Original sequence. Notice the change in the right half of the face.

also produce changes in the geometry of the object. All the results in this paper are on the USF Human ID 3D Face Database [12].

7.6 Summary

We presented a novel framework to extend view synthesis to situations where enough information for rendering the entire scene is not given as input. Semantic information is taken from images of similar objects or scenes that are available in the database to synthesize views that are impossible for traditional methods. We specifically showed the use of symmetry and texture transfer from similar faces to generate plausible all-around views of faces, given only partial views of it.

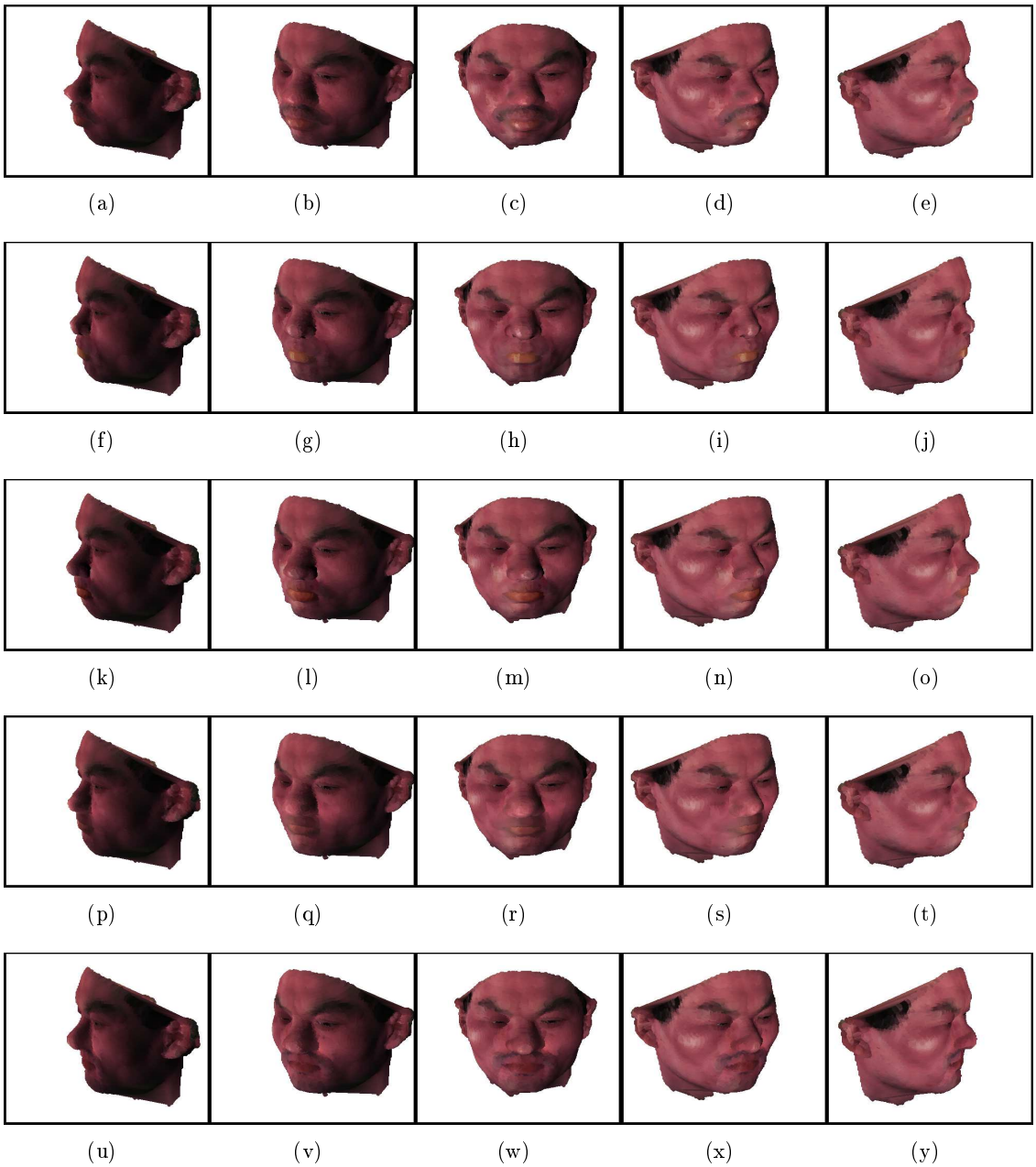


Figure 7.10: Various variations of the nose a mouth of a person by transferring information from the database. Notice how the MCOP image is a useful representation of a scene for such a kind of modification.

Chapter 8

Conclusions

8.1 Introduction

Multiple View Geometry (MVG) has reached a mature stage of research, where the theoretical understanding of underlying concepts is more or less complete, with emphasis increasingly shifting towards research on its various applications. Already, robust solutions for the *reconstruction* of objects from *handheld* camera images has reached commercial products [16]. However, products like *photosynth* fall under the category of *vision-only* technologies where computer vision occupies a central position. An alternate line of research is Robotics, where computer vision plays the role of a *sensor*. Over the course of years [3], advance in computer vision has ensured that cameras become some of the most reliable sensors for many robotics tasks like navigation and manipulation [25]. It is these recent initiatives to use computer vision as both image manipulation algorithms and as sensors in robotics that has inspired our current investigations.

8.2 Contributions

In this thesis, we focus on the use of a sub-field of computer vision called Multiple View Geometry, which concentrates on the interactions between the geometry of the world and its various images captured by a camera from different positions. Our contributions focus on two main themes: (i) to apply computer vision as a sensor for *inferring* and *navigating* problems in robotics. (ii) to apply computer vision as a tool for the *manipulation* of image and video content, with an extension to the *creation* of new content (Chapter 7). Specifically, our contributions include

- Algorithms for tracking the pose of an object in 2D (Chapter 3), which involves estimating the homography of an object with respect to a reference frame in a video. We introduce a particle filter framework for tracking that is robust to large changes in perspective and to illumination. We show that particle filters is the logical framework for this problem, with analysis and comparisons with other popular trackers like the Unscented Kalman Filter.
- Algorithms for tracking the pose of an object in 3D (Chapter 4), which involves estimating the pose of the camera (or equivalently of a rigid object) in a scene, when the camera is moving (equivalently th object is moving) and everything else is stationary. Based on the assumption of Gaussian noise models, which is common in the 3D pose tracking literature [3], and a piece-wise planar scene, we propose an Unscented Kalman Filter framework for tracking purposes. We also propose heuristics for robustness that are firmly grounded in theory, to make the tracker robust to both occlusion and illumination change, making it very suitable for various

robotic tasks like manipulation [4]. We also propose algorithms for the initialization of the tracker, with one algorithm providing a *globally* optimal solution with excellent performance in the presence of noise.

- Algorithms for guiding a robotic arm for positioning with respect to a target (Chapter 5), which involves guiding a robot with a camera towards a particular position with respect to an initially identified target object. We propose a frequency domain approach that alleviates the burden of correspondence from visual servoing algorithms, and then further develop this approach to argue that frequency domain techniques employing multiple view geometry have good potential for application visual servoing. To this end we propose servoing algorithms that are capable of tasks like guiding the robot along a straight Cartesian path instead of the traditional curved one (Section 5.3.2), and for tasks path following (Section 5.3.3).

- Algorithms for manipulating the content of a video (Chapter 6), which involve estimating and seamlessly removing objects from a video, based on a set of detection and removal *functions* that are defined by the user. We demonstrate the efficacy of our approach by considering two scenarios: one for detecting and removing *non-planar* objects from a video with a planar background, and the other for detecting and removing *dynamic* objects from a video of a generic scene when the camera is performing a panning motion.

- Algorithms for generating novel images of a scene with *unknown* 3D content (Chapter 7), which involves the *extrapolation* of a given scene using *semantic* information available from a database of similar looking scenes. The application is shown on a database of face images, where novel views of a given face are generated by identifying and importing content from a database using suitable functions for *semantic matching* and *information transfer*. Finally, the transferred information is then seamlessly blended into the existing scene representation to generate

8.3 Future Work

With applications research being a recent phenomena in MVG ([16, 20] were introduced in 2005) there is a lot of scope for future work, and we outline some of the issues in this thesis that may be the subject of future research.

- **Tracking:** The tracking framework proposed in this thesis is a novel one in the sense that it is a direction towards modeling *higher order primitives* like planes into the tracking framework, instead of have base level features like textures and edges. Indeed a promising direction of research is envisaging a framework where low-level features are used to obtain robust estimates of *mid-level* features like homography, which can be further used as the *basic measurements* for a tracking framework (Figure describes a tracking framework where basic features like interest points and edges are not *directly* absorbed by the tracker as measurements, but are fused to estimate mid level primitives like a homography). On a more immediate note, the UKF tracker proposed in Chapter 4 has an important limitation in the sense that it can only track planes that *have been observed in the first frame.* For extension to a tracker like the one proposed in Figure , it is necessary to incorporate planes observed during the tracking process as well. To do this, we observe that

$$H = R - t\frac{n^\top}{d}$$

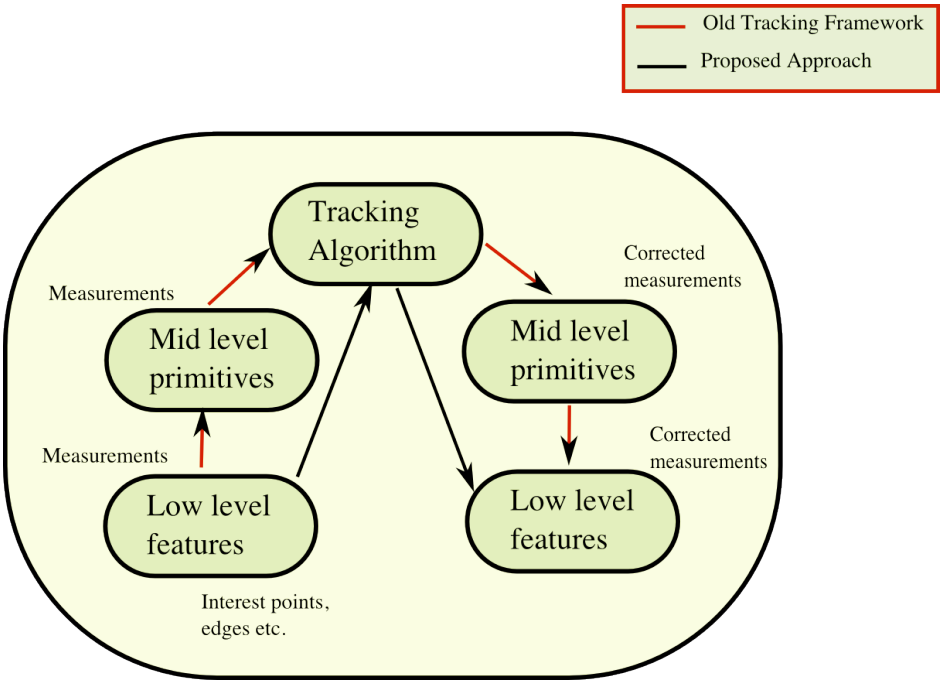


Figure 8.1: Tracking framework of the proposed approach and difference with traditional approaches.

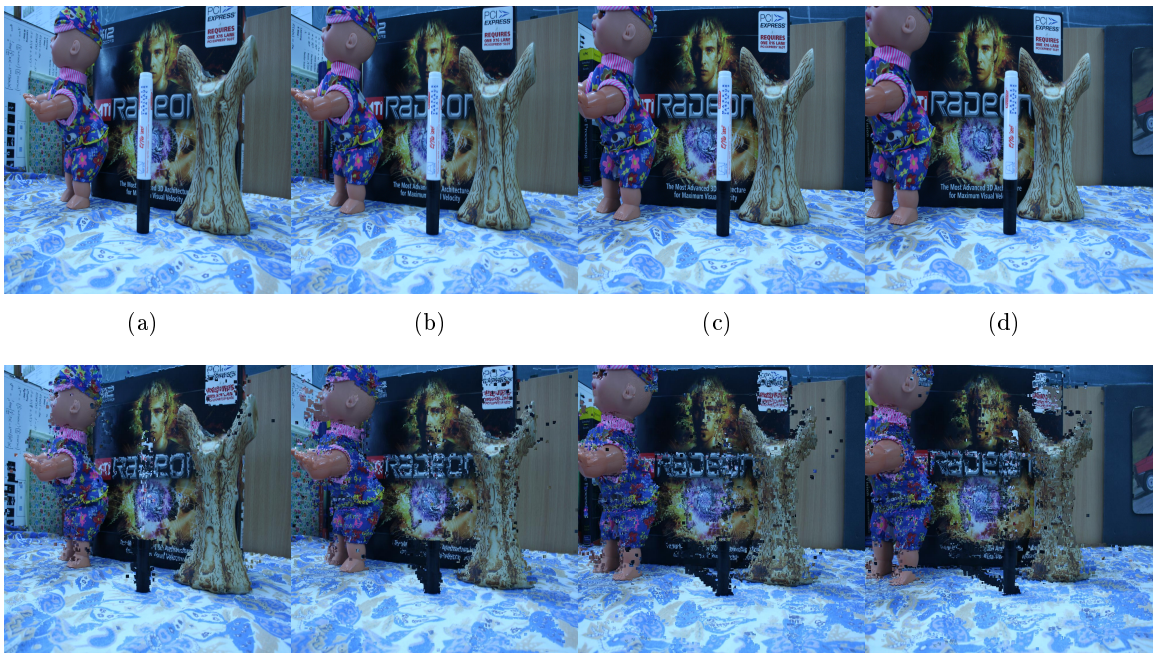
where (R, t) are obtained with respect to the reference coordinate system (first frame). If a plane has been found in the frame n, then in order to include that plane we will have to shift the coordinate system *only for the measurements from that plane,* which can be done by pre-multiplying the pose (R, t) (the state vector) by the value $[R_n\ t_n]^{-1}$, where (R_n, t_n) is the pose of the n^{th} frame as estimated by the UKF tracker. This allows measurements from a plane observed in *any frame* to be included.

$$H_n \quad = \quad R_n - t_n \frac{n_n^\top}{d_n}$$

where the subscript denotes that measurements and *consequently estimates* of both pose and plane parameters are obtained with respect to the nth frame. The conversion of plane parameters to the reference (first) frame is trivial.

- The chapter of video completion deals with the removal of objects from videos based on two functions defined by the user. Although results are pleasing when the underlying functions are homographies, extension to other functions like 3D points and cameras is not trivial. Particularly, the removal algorithm proposed in Chapter 6 is *highly* sensitive to the accuracy of the functions that the user defines for the inpainting algorithm. Thus there is further scope for how to make inpainting algorithms relatively insensitive to errors in the estimation process while making sure that the user achieves the desired output in a seamless manner.

- The algorithm proposed in Chapter 7 has a few limitations. (1) Symmetry as a cue restricts the applicability of the algorithm to nearly symmetric scenes. The use of MCOP as the representation limits the applicability to nearly-convex symmetric objects like faces, flower-vases, etc. It is challenging to transfer symmetry of symmetric objects like chairs due to the huge



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

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

Figure 8.2: Results of video completion when full 3D scenes are involved. The first shows a sample of the input images (16 images were used in total) and the bottom row shows the effect, when the user tries to remove the occluding pen. The underlying functions for detecting objects is a normalized cross correlation function coupled with an estimate of the depth of the object, reconstructed from multiple images. Notice how the accuracy of the 3D reconstruction has severe effects on the output quality.

variation of shape within the class. The transfer of symmetry and texture is dependent on the choice of representation for the scene. Representations that are more suitable that MCOP can, however, extend the method to other types of objects. (2) The use of the nearest neighbor algorithm for semantic similarity of scenes is quite limiting. It might work for objects like faces, but is not guaranteed to work for other objects. A comprehensive minimization for Equation (7.6) can, however, find a similar object from the database that perform more relevant matching and provide better results. (3) The transfer of texture can produce misalignment between the input and database images if they are not less similar. For example, the nose of a person, transferred from the database, may be incorrectly placed with respect to the position of the eyes. Extensions to this algorithm may be in two directions. The first is an extension to non-convex models like say, fishes (Figure cite reference) which needs improvement in the camera path that is used to capture the MCOP image model. Clever paths need to be designed that can capture the "non-convexity" of models like fishes with the rest of the algorithm being intact. Another direction of future research would be to extend our approach to the problem of "face-editing" where inpainting and IBR methods like ours are used to produce various effects like "face-touchups". A sample of such an approach is given in Figure of Chapter 7, where the nose of a person is modified. Recent work [127] on "face-swapping" outlines some of the prospective extensions to our work, that show promising results.

8.4 Conclusion

In this thesis, we proposed algorithms for the application of Multiple View Geometry for tasks like tracking, inpainting and image based rendering. The thesis focuses on two particular areas: robotic and video/image manipulation. In both these areas, we proposed several algorithms that are derived from MVG to demonstrate the power of the geometric theory of computer vision. We believe that not too far in the future, with research in MVG having matured to an advanced state, many of these applications might see commercial usage.

Appendix A

Related Publications

The following publications resulted from work in and related to this thesis.

- A. H. Abdul Hafez, Visesh Chari, C. V. Jawahar. Combining Edge and Texture Planar Trackers based on a Local Quality Metric. Proc. of IEEE International Conference on Robotics and Automation (ICRA'07), Roma, Italy, 2007.
- Avinash Kumar, Narendra Ahuja, John M. Hart, Visesh Chari, P. J. Narayanan, C. V. Jawahar. A Vision System for Monitoring Intermodal Freight Trains. Proc. of IEEE Workshop on Applications of Computer Vision (WACV 2007), Austin, Texas, USA, 2007.
- Visesh Chari, Avinash Sharma, Anoop Namboodiri, C. V. Jawahar. Frequency Domain Visual Servoing Using Planar Contours. Proc. of Indian Conference on Comupter Vision Graphics and Image Processing (ICVGIP'08) .
- Visesh Chari, C. V. Jawahar, P. J. Narayanan. Video Completion as Noise Removal. Proc. National Conference on Communications (NCC'08).
- Visesh Chari, Jag Mohan Singh, P. J. Narayanan. Augmented Reality using Over-Segmentation. Proc. National Conference on Computer Vision Pattern Recognition Image Processing and Graphics (NCVPRIPG'08).
- Visesh Chari, C. V. Jawahar. Multiple Plane Tracking using the Unscented Kalman Filter. IEEE International Conference on Robotics and Automation (ICRA'09) (Submitted).

Appendix B

Algorithms

B.1 Unscented Kalman Filter

The Unscented Kalman Filter described in Chapter 2, is mid way between the traditional Kalman Filter [128], and the Extended Kalman Filter [28]. While the traditional Kalman Filter models process and measurement functions as linear ones, the EKF *extends* Kalman Filter to non-linear systems by using a linearization of the non-linear functions. The UKF on the other hand, avoids the approximation of linearization by using the Unscented Transform to percolate noise variances through the non-linear system accurately [129]. We detail the UKF estimation algorithm below.

B.2 Particle Filters

An improvement over the UKF is required when the noise covariances governing the dynamics of a system are non-Gaussian, since the UKF makes a Gaussian assumption for noise. This is particularly useful in computer vision where many geometric quantities are highly non-linear functions, that cannot guarantee that an initial Gaussian noise covariance will remain Gaussian. An important component of the Particle Filter algorithm, as detailed in Chapter 2, is the Importance Sampling step, which allows the filter to *choose* particles from a distribution in order to propagate through the non-linear dynamics of the system. Importance sampling is usually done using a function that is different from the state transition (or process) and measurement functions, and it is proved that such a technique does not produce any information loss (Section 2.3). Below we outline the Sequential Importance Sampling algorithm, which is a variant of the Importance Sampling algorithm for the case of a Markov process.

B.3 Successive Over Relaxation

Successive Over Relaxation is a technique for obtaining accurate solutions to a large system of linear equations, for which tasks like matrix inverse become too computationally complex. The algorithm proceeds by starting with an initial guess for the solution, and iteratively estimating its values to inch closer towards zero error. Different variants of relaxation have been proposed, like Jacobi, Gauss-Seidel [1]. Here we outline the main algorithm for SOR, a technique that is increasingly being used for various tasks like image editing and optical flow (Chapter 2).

Algorithm 8 Unscented Kalman Filter (UKF) equations

Ensure:

$$\begin{aligned}\hat{\mathbf{x}}_0 &= \mathbb{E}\left[\mathbf{x}_0\right], \\ \mathbf{P}_0 &= \mathbb{E}\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)((\mathbf{x}_0 - \hat{\mathbf{x}}_0)^\top\right]\end{aligned}$$

for $k = (1, \ldots, \infty)$ **do**

Calculate the sigma points:

$$\mathcal{X}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}\hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}}\hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}\right]$$

The time-update equations are:

$$\begin{aligned}\mathcal{X}^*_{k|k-1} &= \mathbf{F}(\mathcal{X}_{k-1},\ \mathbf{u}_{k-1}) \\ \hat{\mathbf{x}_k^-} &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{k|k-1} \\ \mathbf{P}_k^- &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-)^\top + \mathbf{R}^{\mathbf{v}},\end{aligned}$$

Augment the sigma points with additional points derived from the matrix square root of the process noise covariance

$$\begin{aligned}\mathcal{X}_{k|k-1} &= \left[\mathcal{X}^*_{k|k-1} \quad \mathcal{X}^*_{0,k|k-1} + \gamma\sqrt{\mathbf{R}^{\mathbf{v}}} \quad \mathcal{X}^*_{0,k|k-1} - \gamma\sqrt{\mathbf{R}^{\mathbf{v}}}\right] \\ \mathcal{Y}_{k|k-1} &= \mathbf{H}(\mathcal{X}_{k|k-1}), \\ \hat{\mathbf{y}}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \\ L &\rightarrow 2L\end{aligned}$$

and the measurement update equations are

$$\begin{aligned}\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} + \hat{\mathbf{y}}_k^-)(\mathcal{Y}_{i,k|k-1} + \hat{\mathbf{y}}_k^-)^\top + \mathbf{R}^{\mathbf{n}}, \\ \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^\top \\ \mathcal{K}_k &= \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}\mathbf{P}^{-1}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathcal{K}_k\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\mathcal{K}_k^\top,\end{aligned}$$

where $\gamma = \sqrt{L + \lambda}$, λ is the composite scaling parameter, L is the dimension of the state, $\mathbf{R}^{\mathbf{v}}$ is the process-noise covariance, $\mathbf{R}^{\mathbf{n}}$ is the measurement-noise covariance and W_i are the weights as calculated in Equation 2.10.

end for

Algorithm 9 SIS Particle Filter

$\left[\left(\mathbf{x}_k^i, w_k^i\right)_{i=1}^{N_s}\right] = \text{SIS}\left[\left(\mathbf{x}_{k-1}^i, w_{k-1}^i\right)_{i=1}^{N_s}, \mathbf{z}_k\right]$
for $(i = 1 : N_s)$ **do**
  Draw $\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$
  Assign the particle a weight, w_k^i, according to Equation (2.24).
end for

Algorithm 10 The SOR Method, courtesy [1]

Choose an initial guess $\mathbf{x}^{(0)}$ to the solution $\mathbf{x}$
for $(k = 1, 2, \ldots)$ **do**
  for $(i = 1, 2, \ldots)$ **do**
    $\sigma = 0$
    for $(j = 1, 2, \ldots, i-1)$ **do**
      $\sigma = \sigma + \mathbf{A}(i,j)\mathbf{x}_j^{(k)}$
    end for
    for $(j = i+1, \ldots, n)$ **do**
      $\sigma = \sigma + \mathbf{A}(i,j)\mathbf{x}_j^{(k-1)}$
    end for
    $\sigma = (\mathbf{b}_i - \sigma)/\mathbf{A}(i,i)$
    $\mathbf{x}_i^{(k)} = \mathbf{x}_i^{(k-1)} + \omega(\sigma - \mathbf{x}_i^{(k-1)})$
  end for
  check convergence; continue if necessary
end for

Bibliography

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia, PA: SIAM, 1994.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[3] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, June 2007.

[4] A. H. A. Hafez, "Enhancing hybrid visual servo control by probabilistic techniques," Ph.D. dissertation, Osmania University, 2007.

[5] M. Brown and D. Lowe, "Recognising panoramas," in *International Conference on Computer Vision*, 2003, pp. 1218–1225.

[6] A. W. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image based rendering using image based priors," *International Conference on Computer Vision (ICCV)*, 2003.

[7] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum, "Full-frame video stabilization," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, Washington, DC, USA, IEEE Computer Society, 2005, pp. 50–57.

[8] J. Shi and C. Tomasi, "Good features to track," in *IEEE International Conference on Computer Vision and Pattern Recognition, CVPR'94*, Seattle, Washington, June 1994, pp. 593–600.

[9] A. Criminisi, I. Reid, and A. Zisserman, "A plane measuring device," *Image and Vision Computing*, vol. 17, no. 8, pp. 625–634, 1999.

[10] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Inteligence*, no. 3, pp. 485–508, 1988.

[11] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," *IEEE/RSJ Intelligent Robots and Systems*, vol. 1, pp. 943–948, 2004.

[12] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194.

[13] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *International Journal of Computer Vision*, vol. 67, no. 2, pp. 141–158, 2006.

[14] "Introduction to photogrammetry," *www.univie.ac.at/Luftbildarchiv/wgv/intro.htm*.

[15] "Machine perception of 3d solids," *http://www.packet.cc/files/mach-per-3D-solids.html*.

[16] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH Conference Proceedings*, New York, NY, USA, ACM Press, 2006, pp. 835–846.

[17] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," pp. 61–62, 1987.

[18] F. Kahl and D. Henrion, "Globally optimal estimates for geometric reconstruction problems," in *International Conference on Computer Vision*. 2005, pp. 978–985.

[19] F. Kahl, "Multiple view geometry and the l-infinity norm," in *International Conference on Computer Vision*, 2005.

[20] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH*, August 2005.

[21] M. Pressigout and E. Marchand, "Real-time planar structure tracking for visual servoing: a contour and texture approach," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'05*, vol. 2, Edmonton, Canada, August 2005, pp. 1701–1706.

[22] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, 2002.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal on Computer Vision*, pp. 91–110, 2004.

[24] "Boujou," *http://www.2d3.com*.

[25] P. Rives, B. Espiau, and F. Chaumette, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, 1992.

[26] A. A. Hafez, V. Chari, and C. Jawahar, "Combining texture and edge planar trackers based on a local quality metric," in *IEEE Int. Conf. on Robotics and Automation, ICRA'07*, Roma, Italia, April 2007.

[27] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, 2003.

[28] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," 2000, pp. 153–158.

[29] "Particle filter homepage," *http://www-sigproc.eng.cam.ac.uk/smc/*.

[30] B. K. P. Horn and B. G. Schunck, "Determining optical flow," in *Shape recovery*, USA, Jones and Bartlett Publishers, Inc., 1992, pp. 389–407.

[31] P. sand and S. teller, "Particle video," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[32] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Trans. Graph.*, vol. 2, no. 4, pp. 217–236, 1983.

[33] A. Blake and M. Isard, *Active Contours*. Springer-Verlag, April 1998.

[34] M. MacCormick, "Probabilistic models and stochastic algorithms of visual tracking," Ph.D. dissertation, University of Oxford, U.K., 2000.

[35] A. Comport, E. Marchand, and F. Chaumette, "Robust model-based tracking for robot vision," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, vol. 1, Sendai, Japan, September 2004, pp. 692–697.

[36] F. Jurie and M. Dhome. "Real time robust template matching,",", 2002.

[37] P. Li and F. Chaumette, "Image cues fusion for contour tracking based on particle filter," in *Int. Workshop on articulated motion and deformable objects, AMDO'04*, vol. 3179 of *Lecture Notes in Computer Science*, Palma de Mallorca, Spain, Springer-Verlag, September 2004, pp. 99–107.

[38] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, August 1998.

[39] K. Nummiaro, E. Koller-Meier, and L. J. V. Gool, "Object tracking with an adaptive color-based particle filter," in *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, London, UK, Springer-Verlag, 2002, pp. 353–360.

[40] J. Vermaak, P. Pérez, M. Gangnet, and A. Blake, "Towards improved observation models for visual tracking: Selective adaptation," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, London, UK, Springer-Verlag, 2002, pp. 645–660.

[41] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, London, UK, Springer-Verlag, 2002, pp. 661–675.

[42] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking.," in *IEEE International Conference on Computer Vision*, vol. 2, October 2005, pp. 1508–1511.

[43] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3d camera tracking," in *International Symposium on Mixed and Augmented Reality, Arlington, VA,*, November 2004.

[44] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust real-time visual tracking using a 2d-3d model-based approach," in *IEEE Int. Conf. on Computer Vision, ICCV'99*, vol. 1, Kerkira, Greece, September 1999, pp. 262–268.

[45] G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998.

[46] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 24, no. 7, pp. 996–1000, 2002.

[47] F. Jurie and M. Dhome, "Real time robust template matching," in *In British Machine Vision Conference 2002*, 2002, pp. 123–131.

[48] M. Pressigout and E. Marchand, "Real-time hybrid tracking using edge and texture information," *Int. Journal of Robotics Research, IJRR*, vol. 26, pp. 689–713, July 2007.

[49] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, "Making good features track better," in *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, IEEE Computer Society, 1998, p. 178.

[50] B. Georgescu and P. Meer, "Point matching under large image deformations and illumination changes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 674–688, 2004.

[51] P. Bouthemy, "A maximum likelihood framework for determining moving edges," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 11, no. 5, pp. 499–511, 1989.

[52] D. Freedman and M. S. Brandstein, "Contour tracking in clutter: A subset approach," *International Journal of Computer Vision*, vol. 38, no. 2, pp. 173–186, 2000.

[53] J. Vermaak, P. Perez, and A. Blake, "Data fusion for visual tracking with particles,"

[54] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Conference*, . 1988, pp. 189–192.

[55] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *International Journal of Computer Vision, IJCV*, vol. 27, no. 2, pp. 161–195, 1998.

[56] E. A. W. Rudolph van der Merwe, "Rebel: Recursive bayesian estimation library," 2006.

[57] C. Olsson, F. Kahl, and M. Oskarsson, "Optimal estimation of perspective camera pose," in *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, Washington, DC, USA, IEEE Computer Society, 2006, pp. 5–8.

[58] G. Simon and M.-O. Berger, "Pose estimation for planar structures," in *IEEE Computer Graphics and Applications*, 2002.

[59] C. Baillard, A. Zisserman, and O. O. England, "Automatic reconstruction of piecewise planar models from multiple views," in *In Proc. IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, 1999, pp. 559–565.

[60] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, 1996, pp. 11–20.

[61] G. Simon, A. Fitzgibbon, and A. Zisserman, "Markerless tracking using planar structures in the scene," in *Proc. International Symposium on Augmented Reality*, Oct. 2000, pp. 120–128.

[62] A. Shashua and S. Avidan, "The rank 4 constraint in multiple (≥ 3) view geometry," in *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, London, UK, Springer-Verlag, 1996, pp. 196–206.

[63] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

[64] L. Zelnik-Manor and M. Irani, "Multiview constraints on homographies," *IEEE Transactions on Pattern Analysis a d Machine Intelligence*, vol. 24, no. 2, pp. 214–223, 2002.

[65] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, 2003.

[66] P. H. S. Torr and D. W. Murray, "The development and comparison of robust methodsfor estimating the fundamental matrix," *International Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[67] Z. Zhang and A. R. Hanson, "3d reconstruction based on homography mapping," *Proceedings of ARPA*, pp. 1007–1012, 1996.

[68] A. Bartoli, "A random sampling strategy for piecewise planar scene segmentation," *Computer Vision and Image Understanding*, vol. 105, no. 1, pp. 42–59, 2007.

[69] D. Cobzas and P. Sturm, "3d ssd tracking with estimated 3d planes," in *Image and Vision Computing, CRV Special Issue*, 2005.

[70] O. Kähler and J. Denzler, "Rigid motion constraints for tracking planar objects," in *Annual Symposium of the German Association for Pattern Recognition, DAGM*, 2007.

[71] G. Simon and M. Berger, "Real time registration of known or recovered multi-planar structures : application to ar," in *British Machine Vision Conference*, 2002.

[72] N. Molton, A. Davison, and I. Reid. "Locally planar patch features for real-time structure from motion,",", 2004.

[73] Z. Sun, V. Ramesh, and A. M. Tekalp, "Error characterization of the factorization method," *Computer Vision and Image Understanding: CVIU*, vol. 82, no. 2, pp. 110–137, 2001.

[74] R. M. Steele and C. Jaynes, "Feature uncertainty arising from covariant image noise," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, Washington, DC, USA, IEEE Computer Society, 2005, pp. 1063–1070.

[75] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," in *INRIA Research Report - RR6303*, no. 6303, September 2007.

[76] R. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, pp. 580–593, Jun 1997.

[77] W. Chojnacki and M. J. Brooks, "Revisiting hartley's normalized eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 9, pp. 1172–1177, 2003.

[78] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[79] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 d visual servoing," *IEEE Trans. on Robotics and Automation*, 1999.

[80] S. Hutchinson, G. Hager, and Cork, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 18–27, 1996.

[81] P. Corck and S. Hutchinson, "A new partitioned approach to image-based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 507–515, Aug 2001.

[82] M. Vargas and E. Malis, "Visual servoing based on an analytical homography decomposition," *CDC-EDC '05, 44th IEEE Conference on Decision and Control*, pp. 5379– 5384, 2005.

[83] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *IEEE Transactions on Robotics*, vol. 21, pp. 1116–1127, December 2005.

[84] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. on Robotics and Automation*, vol. 18, pp. 534–549, August 2002.

[85] R. Mahony, P. Corke, and F. Chaumette, "Choice of image features for depth-axis control in image-based visual servo control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'02*, vol. 1, Lausanne, Switzerland, October 2002, pp. 390–395.

[86] C. Collewet and F. Chaumette, "A contour approach for image-based control of objects with complex shape," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'00*, vol. 1, Takamatsu, Japon, November 2000, pp. 751–756.

[87] E. Malis and P. Rives, "Robustness of image-based visual servoing with respect to depth distribution errors," in *IEEE Int. Conf. on Robotics and Automation, ICRA'03*, vol. 1, Taipei, Taiwan, Sept. 2003, pp. 1056–1061.

[88] W. Wilson, C. Hulls, and G. Bell, "Relative end-effector control using cartesian position based visual servoing," vol. 12, pp. 684–696, October 1996.

[89] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, D. Kriegman, G. . Hager, and A. Morse, Eds. LNCIS Series, No 237, Springer-Verlag, 1998, pp. 66–78.

[90] S. Lehmann, A. P. Bradley, I. V. L. Clarkson, J. Williams, and P. J. Kootsookos, "Correspondence-free determination of the affine fundamental matrix," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[91] P. K. Jain and C. V. Jawahar, "Homography estimation from planar contours," *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006.

[92] R. N. Bracewell, *Fourier Analysis and Imaging*. Kluwer Academic/Plenum Publishers, 2003.

[93] S. Kuthirummal, C. V. Jawahar, and P. J. Narayanan, "Fourier domain representation of planar curves for recognition in multiple views," *Pattern Recognition*, 2004.

[94] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Trans. on Robotics*, vol. 20, pp. 713–723, August 2004.

[95] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," in *Workshop on Geometric Reasoning for Perception and Action, LNCS 708*, Grenoble, France, September 1991, pp. 106–136.

[96] P. I. Corke, "A computer tool for simulation and analysis: the robotics toolbox for matlab," in *Proc. National Conf. Australian Robot Association*, 1995, pp. 319–330.

[97] "Minerva,"

[98] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 34–47, 2001.

[99] M. Pressigout and E. Marchand, "Hybrid tracking algorithms for planar and non-planar structures subject to illumination changes," in *ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'06*, Santa Barbara, CA, October 2006.

[100] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," *International Journal of Computer Vision*, pp. 1033–1038, 1999.

[101] A. Efros and W. T. Freeman, "Image quilting for texture synthesis," *ACM SIGGRAPH*, pp. 341–346, 2001.

[102] Y. Wexler, E. Shechtman, and M. Irani, "Space-time video completion," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 120–127, 2004.

[103] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting for occluding and occluded objects," *IEEE Conference on Image Processing*, pp. 69–72, 2005.

[104] V. Cheung, B. J. Frey, and N. Jojic, "Video epitomes," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 42–49, 2005.

[105] A. Kokaram, "Practical, unified, motion and missing data treatment in degraded video," *Journal of Mathematical Imaging and Vision*, pp. 163–177, 2004.

[106] G. S. M. Bertalmio, A. L. Bertozzi, "Navier-stokes, fluid dynamics, and image and video inpainting," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 355–362, December 2001.

[107] J. Jia, T. P. Wu, Y. W. Tai, and C. K. Tang, "Video repairing: Inference of foreground and background under sever occlusion," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 364–371, 2004.

[108] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaicing: Mosaicing of dynamic scenes," *IEEE Pattern Analysis and Machine Intelligence*, 2007.

[109] D. Capel and A. Zisserman, "Super-resolution from multiple views using learnt image models," *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[110] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[111] K. ichi Kanatani and N. Ohta, "Accuracy bounds and optimal computation of homography for image mosaicing applications," *International Confernce on Computer Vision*, pp. 91–110, 1999.

[112] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," in *ACM SIGGRAPH*, 2005, pp. 595–600.

[113] D. Scharstein, "View synthesis using stereo vision," *volume 1583 of Lecture Notes on Computer Science (LNCS), Springer-Verlag New York, Inc.*, 1999.

[114] M. Levoy and P. Hanrahan, "Light field rendering," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, 1996, pp. 31–42.

[115] M. Irani, T. Hassner, and P. Anandan, "What does the scene look like from a scene point?," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, London, UK, Springer-Verlag, 2002, pp. 883–897.

[116] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[117] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Comput. Vision*, vol. 38, no. 3, pp. 199–218, 2000.

[118] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system," in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, 1995, pp. 39–46.

[119] L. Mcmillan and G. Bishop, "Head-tracked stereoscopic display using image warping," in *Proceedings SPIE, volume 2409*, 1995, pp. 21–30.

[120] L. Mcmillan, "An image-based approach to three-dimensional computer graphics," Ph.D. dissertation, University of North Carolina at Chapel Hill, April 1997.

[121] J. Hays and A. A. Efros, "Scene completion using millions of photographs," in *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, New York, NY, USA, ACM, 2007, p. 4.

[122] M. Wilczkowiak, G. J. Brostow, B. Tordoff, and R. Cipolla, "Hole filling through photomontage," in *16th British Machine Vision Conference 2005 - BMVC'2005, Oxford, United Kingdom*, July 2005, pp. 492–501.

[123] P. Rademacher and G. Bishop, "Multiple-center-of-projection images," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, ACM, 1998, pp. 199–206.

[124] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[125] S. Thrun and B. Wegbreit, "Shape from symmetry," in *International Conference on Computer Vision (ICCV)*, Washington, DC, USA, IEEE Computer Society, 2005, pp. 1824–1831.

[126] A. Agrawal, R. Raskar, and R. Chellappa, "What is the range of surface reconstructions from a gradient field," *European Conference on Computer Vision (ECCV)*, 2006.

[127] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar, "Face swapping: automatically replacing faces in photographs," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, New York, NY, USA, ACM, 2008, pp. 1–8.

[128] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[129] S. Julier, "The scaled unscented transformation," *American Control Conference*, 2002.