Depth From Defocused Motion

Zarina Myles
University of Central Florida

DEPTH FROM DEFOCUSED MOTION

by

ZARINA MYLES
B.Sc. Madras University, Madras 1977
M.Sc. Mathematics Madras University, Madras 1980
M.Tech. Computer Science Indian Institute of Technology N. Delhi 1984

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2004

Major Professor:
Niels da Vitoria Lobo

ABSTRACT

Motion in depth and/or zooming causes defocus blur. This work presents a solution to the problem of using defocus blur and optical flow information to compute depth at points that defocus when they move.

We first formulate a novel algorithm which recovers defocus blur and affine parameters simultaneously. Next we formulate a novel relationship (the *blur-depth relationship*) between defocus blur, relative object depth and three parameters based on camera motion and intrinsic camera parameters.

We can handle the situation where a single image has points which have defocused, got sharper or are focally unperturbed. Moreover, our formulation is valid regardless of whether the defocus is due to the image plane being in front of or behind the point of sharp focus.

The *blur-depth relationship* requires a sequence of at least three images taken with the camera moving either towards or away from the object. It can be used to obtain an initial estimate of relative depth using one of several non-linear methods. We demonstrate a solution based on the Extended Kalman filter in which the measurement equation is the *blur-depth relationship*.

The estimate of relative depth is then used to compute an initial estimate of camera motion parameters. In order to refine depth values, the values of relative depth and camera motion are then input into a second Extended Kalman Filter in which the measurement equations are the discrete motion equations. This set of cascaded Kalman filters can be employed iteratively over a longer sequence of images in order to further refine depth.

We conduct several experiments on real scenery in order to demonstrate the range of object shapes that the algorithm can handle. We show that fairly good estimates of depth can be obtained with just three images.

To Paramahansa Yogananda, my closest and dearest friend and guide

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A machine vision system analyzes images and produces descriptions of what is imaged. These descriptions must capture the aspects of the objects being imaged that are useful in carrying out some task. Several tasks require the user to form a 3-D model of the object being imaged. Alternately, a *depth map* can be obtained. This gives the depth corresponding to each point imaged and is occasionally referred to as a 2.5-D model.

Several techniques have been employed to extract depth information from an image or a sequence of images. These have included extracting information from the variation in *shading* in a smooth object as well as the analysis of the deformation of *texture* of a uniformly textured object. Others include the analysis of the displacement of points in the two images of a *stereo* pair. Another technique is based on the following idea: when an object gets out of focus, the level of *defocus blur* gives a cue to the depth of the corresponding object point.

Yet other techniques involve the extraction of useful information from time-varying images. Brightness patterns in the image move as the objects that give rise to them move. *Image flow* is the apparent motion of the brightness pattern. Image flow is often referred to as *optical flow* when the motion of image points is assumed to be infinitesimal. Hereafter, the terms image flow and optical flow will be used interchangeably. This flow depends both upon the camera and/or object motion as well as the object shape. Hence image flow (2-D motion) has been employed to extract both 3-D camera motion as well as depth. However, translation and depth can be recovered only up to a scale factor. This can be inferred from the fact that a small translation of a small object close by would give rise to the same sequence of images as a large translation of a magnified version of the same object further away. Image flow is also observed when the *zoom* feature of the camera is employed and there is neither camera nor object motion.

There are limitations to each technique listed above and the problem of recovering depth has not been robustly solved. In particular, depth-from defocus and depth-from-motion/zooming address complementary issues. When there is substantial image flow caused by motion or zooming, the real or virtual motion along the optical axis causes the image to get out of focus. The corresponding blur has traditionally been treated as noise. On the other side of the coin, traditional depth from defocus techniques have not taken into consideration the effects of image scaling and/or distortion when the blurring is due to a change in focal length or a change in the distance from the lens to the image plane.

This thesis solves the problem of using defocus and motion information to compute depth at points that defocus when they move. This provides a shape from defocus method for moving points, a capability that previously did not exist. Traditional techniques to compute shape (depth) from motion/zooming can be enhanced because previously unavailable estimates of depth, are now available from the defocus information. We term this *Depth from Defocused Motion*.

The computation proceeds along the following main steps. It simultaneously recovers the level of defocus blur as well as the image flow in a sequence of images.

The level of blur is then used to obtain an initial estimate of depth and the camera parameters. Those are then used in an iterative technique employing the Extended Kalman Filter to recover both the depth of the imaged object as well as 3-D motion of the camera. The main stages of the depth from defocused motion algorithm are listed below:

1. With the camera parameters (focal length f, aperture size D and zoom) being fixed, obtain a set of 3 or more images with the camera positioned as shown in Figure 1.

2. Obtain blur and image flow information between an image and its two successive images as shown in Chapter 3.

3. Compute an initial approximation for the relative depth Z for all points in the image as shown in Chapter 4.

4. Given relative depth, compute an initial estimate of camera motion parameters by solving the motion equations.

5. With these initial values of relative depth and camera motion, use the Extended Kalman filter to refine the relative depth and 3-D motion parameters as elaborated in Chapter 5.

6. Refine depth values by repeating the steps 2-5 for a larger sequence of images.

The rest of this thesis is organized as follows. Chapter 2 provides an overview of background work, Chapter 3 derives a novel algorithm to obtain optical flow and defocus blur simultaneously. This formulation is validated by conducting experiments on both artificial as well as real images. Chapter 4 derives a novel relationship between defocus blur/sharpening and relative depth as well as fixed camera-related parameters. This relationship is used in a Kalman Filter in order to obtain an initial estimate of depth. Chapter 5 shows how this estimate of depth can be refined by incorporating the optical flow values and initial estimates of depth within a second Kalman filter. Chapter 6 summarizes up this work and enumerates the major strengths and weaknesses of this research.

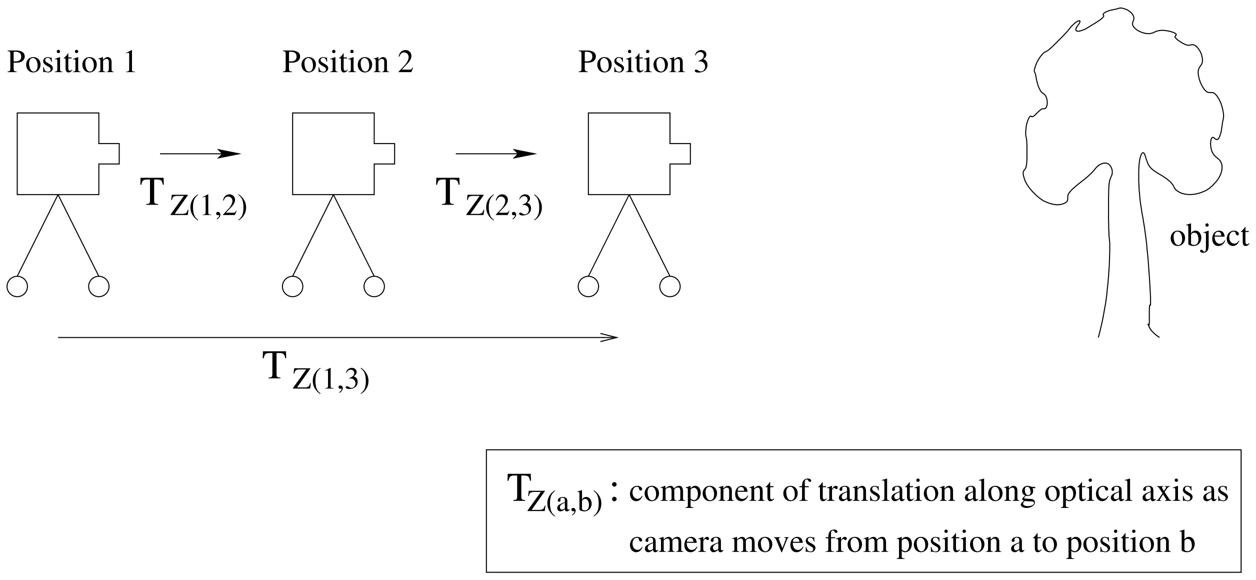

Position 1
Position 2
Position 3
object
$T_{Z(1,2)}$
$T_{Z(2,3)}$
$T_{Z(1,3)}$
$T_{Z(a,b)}$: component of translation along optical axis as camera moves from position a to position b

Figure 1.1: Positions of Camera for Three Different Images of the Same Object

CHAPTER 2

LITERATURE REVIEW

Substantial work has been done in the fields of *depth-from-defocus*, *depth-from-zooming*, and *depth-from-motion*. Most techniques which compute depth from motion first require the *image flow* (optical flow) information to be obtained. The next four sections (Sections 2.1 through 2.4) list some of the seminal works in these four areas.

Finally, inspired by our pioneering work to compute affine transformation and defocus blur simultaneously [28], researchers have pursued this new approach to the recovery of depth cues. The fifth section (Section 2.5) describes some of the work in this area.

2.1 Depth from Zooming

Depth from zooming corresponds to the situation in which both the scene as well as the camera are stationary and optical flow is induced by zooming. Zooming is similar to a translation of the camera along the optical axis and induces optical flow in images along radial lines emanating from the optical center of the image (i.e., the point where the optical axis pierces the image plane).

Advantages over binocular systems include the fact that it doesn't suffer from the occlusion problem and that accurate knowledge of stereo geometry is not required.

2.1.1 Thin Lens Model

Ma and Olsen [23] have developed an algorithm based on the thin lens model. They first derive a *gradient-based model*, which assumes an infinitesimal change in focal length and the induced image displacement. The depth Z of an object is given by

$$Z = \frac{f^2 r_t}{f r_t - r f_t}$$

	f	$=$	the focal length
where	r_t	$=$	the rate of change of the radial distance of an image point
	f_t	$=$	the rate of change of the focal length

In order to compute r_t they derived the zooming-image-flow equation which is an enrichment to Horn's "constant brightness equation".

The computation of the flow field r_t is strongly affected if the irradiance function is not smooth (eg. if the scene surface is sharply textured or if the images are noisy). Moreover, small errors in the computation of flow field yield large errors in depth. Hence, the above model, based on instantaneous changes in focal length and image radial distance is inherently unstable.

They then derived a *feature-based scheme* where the model was derived assuming discrete changes in focal length and the induced image motion. The model assumes two images taken by a camera with two distinct focal lengths f_1 and f_2. Here the depth Z is given by

$$Z = f_2 \frac{V}{V - \alpha r}$$

$$\begin{array}{lrcl} & r & = & \text{radial position of a point in the } f_1 \text{ image} \\ \text{where} & V & = & \text{image displacement magnitude} \\ & \alpha & = & \frac{f_2-f_1}{f_1} \end{array}$$

The above equation is further modified such that lines in the images are matched instead of points.

The algorithms above make the assumptions that there is *a priori* knowledge of the optical center (the point where the optical axis cuts the image plane) as well as the two focal lengths. It also assumes that the optical center does not move during zooming, although that has been proved [20] to be an inaccurate assumption. It requires that the camera is precalibrated and that there are known matches between image frames.

Moreover, they do not account for deformation due to zooming although in reality a pixel in the first image would expand into a region within the second (zoomed in) image. Perhaps their greatest weakness is that they assume a thin lens model which is not appropriate for the set of lenses in a zoom camera. The validity of their model has been demonstrated only on synthetic images only.

2.1.2 Thin Lens Model Handling Image Deformations

Mobasseri and Doraiswamy [26] address the problem of deformations of points and lines caused by zooming by applying shape correspondence across focal lengths. The thin lens model is employed. Object depths are described in terms of the ratio of feature translation on the image plane in response to changes in focal length. The algorithm then searches for the focus of expansion (FOE) as well as the above ratio iteratively by studying the intensity in corresponding patches of both images. In their experiment they looked for the FOE in a 15×15 pixel uncertainty region. However, the underlying thin lens model is one of the reasons for the suboptimal quality of their results.

2.1.3 Thick Lens Model

Lavast et al [20, 7, 19] address some of the limitations of the above paper and provide solutions which are progressively more flexible and accurate. In their work, they use the *thick lens model*, thus more accurately modeling the optical phenomena occurring during a focal change.

[20] requires high zoom-lens quality but this restriction is lifted in [7]. [19] further extends the work by using correlation techniques to handle deformation (expansion/contraction) of image features. However, all their work requires a calibration step which involves the imaging of high-precision grids with the help of a micrometric table and precise optical control and assumes a simple model of image deformation. Results were shown on real images of a cube and a pebble.

The computation of depth from zoom is more reliable when there are large flow vectors. However, these make feature matching more difficult. Also, the large variation in focal length causes defocusing in the image, which has not been addressed in the depth-from-zoom algorithms above.

2.2 Depth from Defocus

It has been noted that for any camera setting there is a limited depth of field within which an object has to lie in order to be in perfect focus. As an object moves beyond that range it gets in increasingly poorer focus. Hence the level of defocus is a cue for depth. This fact has been exploited in many algorithms in order to obtain depth-from-defocus(DFD). In order to be able to model blur one needs to know something about the point spread function (PSF) i.e., the function which describes the blurred image of a point source of light. A blurred image is thus considered to be the result of a convolution of the sharp image with the point spread function. Like depth-from-stereo, DFD is easily implemented in parallel, but unlike stereo, the problems of correspondence can be avoided.

An alternate approach for computing depth using focus information is referred to as *depth-from-focus*. In this approach, the space of physical parameters of a camera is searched to find a set of camera parameter values which focuses the object. The object distance is then determined based on the values of the camera parameters. The search usually requires a large number of images acquired at different camera parameter settings. Below are enumerated different techniques for obtaining depth from defocus.

2.2.1 Single Image

Earlier work has explored the possibility for obtaining depth using only one image. However, the corresponding algorithms assumed strong texture in the scene and required that all edges be step edges. Thus there was no need to distinguish between “soft” and blurred edges.

Pentland [32] approximated the PSF by a two-dimensional Gaussian and assumed dense texture, all consisting of step edges. He then developed an algorithm to find the level of blur in *vertical* edges. *Lai et al* [18] extended this algorithm to handle edges in *any* orientation. With increasing levels of blur the results became more unreliable, indicating the limit of the Gaussian blur model. *Subbarao* [38] extended the approach by allowing the PSF to be any rotationally symmetric function, which is a reasonable assumption when the camera aperture and lens are circular. He derived a relationship between the distance of an object and the second central moment (variance) of the PSF. He then outlined a technique for obtaining the line spread function and gave a closed form solution for computing depth from the spread parameter.

The methods above are restricted to isolated step edges. The presence of other edges nearby (within a distance of about twice the spread parameter) would affect the depth estimation. One technique to ensure step edges at convenient distances has been to impose a known texture on the scene; another has been to allow for any texture and to use more than one image to measure the difference in texture amongst them.

2.2.2 Two Images

Pentland [32] [33] [34] proposed a depth-from-defocus algorithm using two images. He shows how the difference in localized Fourier power between two images at corresponding image locations is a monotonic increasing function of the difference of blur. By using Parseval's theorem he describes a simple implementation based only on local convolutions. He obtains quick and reliable but coarse estimates of range and suggests that they can be used for initial target acquisition or obtaining the initial course estimates for stereo disparity in a coarse-to-fine stereo algorithm. The algorithm is designed to handle two images, one of them taken by a pinhole aperture to ensure perfect focus everywhere. However, a small aperture increases diffraction effects, thus distorting the acquired image and necessitating an increase in the exposure period, which therefore slows down the system.

Subbarao and Wei [39] compute the first few coefficients of the 1-D Fourier transform of an image and use them to compute depth. Working in the 1-D Fourier domain makes the method more robust to noise, besides being computationally economical. They quote an RMS error of 4% at 0.6 meter distance and this changed linearly to about 30% RMS error at 5 meters.

The methods above, based on inverse filtering, have certain limitations. It is difficult to accurately estimate the frequency spectra of local regions. As depth changes throughout the scene, it is desirable for the regions to be as small as possible, yet accurate spectral analysis benefits from larger local areas. Also, windowing introduces artifacts (spurious high frequency components generated by the discontinuity at the window boundary) which can cause large errors if not addressed. Also, methods which depend upon studying a preselected frequency band may cause a very large error if there is not enough energy of the image content within the frequency band.

Xiong and Shafer [48] propose an elegant way to cope with this problem. A moment filter set is introduced to compensate for the frequency distribution within the pass-band of each of the narrow-band filters. This translates to five times as many convolutions as needed in a typical filter bank, which renders their approach computationally more expensive.

Subbarao and Surya [40] developed a spatial domain convolution/deconvolution transform, which they call the S-transform. The image is modeled with a third order polynomial and a simple and elegant equation is derived as follows:

$$i_2(x,y) - i_1(x,y) = \frac{1}{4}(\sigma_2^2 - \sigma_1^2)\nabla^2 \left(\frac{i_2(x,y) + i_1(x,y)}{2}\right)$$

Here i_1 and i_2 are the two images and the blur circle radii can be expressed in terms of the second central moments σ_2 and σ_1 respectively. This method produces accurate averaged depth estimates for a large planar object although it does not yield depth maps with high spatial resolution.

Ens and Lawrence [9] proposed a method based on a spatial domain analysis of two blurred images. It estimates the convolution matrix which is convolved with one of the images to produce the second. They demonstrated their algorithm on a slanted planar scene and computed the depth map where a single depth value is computed using a 64×64 window. It produces accurate depth maps, but the iterative nature of the convolution matrix estimation makes it computationally expensive.

Mudenagudi and Choudhury [27] introduce a *depth-from-defocused stereo* technique which fuses two depth cues to obtain a superior depth map. Depth estimates from defocus are traditionally not as accurate as those obtained from a stereo pair of images. On the other hand stereo has to address the correspondence problem and traditionally produces a sparse depth map. Their method uses two pairs of images. The same camera parameters are applied for a pair of left and right images. Their technique requires the camera parameters to be known. They first compute the relative defocus blur in either the left pair of images or the right pair of images. This value is then input into their model which relates it to the disparity between a pair of left and right images. This two-step process yields a dense depth map without the need for an separate feature matching step.

2.2.3 Active Rangers

One of the inherent weaknesses of the DFD algorithms listed above is that it requires that the scene have high frequency texture. A texture-less surface appears the same focused or defocused and the resulting images do not contain the information necessary for depth computation. This problem has been addressed by active rangers which project dense structure on the scene. *Pentland* [34] developed an active range camera consisting of a structured light source (implemented by a slide projector) and a sensor (video camera) to measure the defocus of the structured light. Once again, he obtains quick but coarse depth maps.

Nayar et al. [30] refine the above approach by determining an illumination pattern which optimizes both the accuracy as well as the robustness of depth measurements. The implementation yields good results in real-time but it requires a very precise hardware configuration. Firstly, the camera is made telecentric by attaching an additional aperture at the front focal plane of the camera. This ensures constant image size and avoids the correspondence-like problem which occurs when defocus is caused by anything other than a change in aperture size. Also, the illumination pattern is fabricated using micro-lithography and incorporated into the sensor. This pattern is incorporated into the scene via the same optical path used to image the scene. Errors due to lens distortions are not present since the light rays which are projected out through the imaging optics are subjected to similar geometric distortions as rays reflected back to the sensors. Lastly, light rays passing through the lens are split in two directions using a beam-splitting prism. This produces two images that are simultaneously detected using two cameras. Cross-polarized filters can be attached to the illumination and imaging lens to filter out specularities.

2.3 Image Flow

Image flow refers to the study of 2-D motion of points on the image plane as a result of 3-D motion in space. Research in the field of image flow has spawned many algorithms in the past few decades. It is often referred to as *optical flow* when the motion of image points is assumed to be infinitesimal.

Algorithms to compute image flow differ in the image flow motion model employed and in the technique used.

2.3.1 Image Flow Motion Models

These can be characterized as local, global or quasi-parametric models.

- *Local Motion Model:* These schemes were implemented by most earlier algorithms. They assume a uniform translational motion, use simple filtering schemes and small filter support. This assumption breaks down at depth discontinuities and at the center of expansion/contraction.
- *Global Motion Model:* As the support of the filters was increased the motion within it became more complicated. The model needed to take into consideration the divergence, curl, deformation and translation caused by the perspective projection of fast moving objects. This motivated a *global* motion scheme whereby the flow-field was parameterized by a small number of unknown variables. Examples of global motion models include affine and quadratic flow fields. Global methods are most useful when the scene has a particularly simple form, eg. when the scene is planar.

- *Constrained (quasi-parametric) Motion Models:* fall between local and global methods. Typically, these use a combination of global ego-motion parameters with local shape(depth) parameters. Examples of this approach include the *direct methods* which compute depth and 3-D motion from the normal component of image velocity and bypass the computation of optical flow.

2.3.2 Solution Techniques

Techniques to compute image flow can be roughly characterized as belonging to one of the following categories: differential methods, feature-based matching methods, region-based matching methods, energy-based methods and phase-based methods.

- *Differential Techniques:* Differential techniques compute image velocity from spatio-temporal derivatives of image intensity, or filtered versions of the image. These algorithms employ the *gradient constraint equation*,

  $$\nabla I(\vec{r}, t).\vec{v} + I_t(\vec{r}, t) = 0 \tag{2.1}$$

  which follows from the *brightness constancy equation* $I(\vec{r} + \delta\vec{r}, t + \delta t) = I(\vec{r}, t)$ where $\vec{r}$ is an image point $(x, y)^T$, $I(\vec{r}, t)$ is the intensity of point $\vec{r}$ at time t, and $\vec{v}$ is the image velocity. The latter assumption means that the intensity of an image point does not vary over time. The gradient constraint equation yields an under-constrained system and can only compute the component of image velocity along the normal to an edge (vernier component). Various techniques have been proposed in order to introduce further constraints. These have included first and second order constraints, as well as local and global methods of combining the local constraints. Since differentiation is an ill-posed problem, the performance of differential methods is substantially improved with both spatial and temporal smoothing which is either performed implicitly or achieved in an explicit preprocessing step.

- *Feature-based Matching Methods:* These first select some features in the image frames and then match these features and calculate the disparities between frames. The issues here are what to match, how to select candidate matches, and how to determine the goodness of a match. The main drawbacks of this approach are that they yield velocity vectors at sparse points and that the problem of establishing correspondence is potentially computationally explosive.

- *Region-Based Matching Methods:* These techniques avoid the instability of differential methods by searching for a *shift* $\vec{d}$ which yields the best fit between image regions in two images. This amounts to *maximizing a similarity measure* such as the normalized cross-correlation (NCR) or *minimizing a distance measure* such as the sum-of-squared difference (SSD), where

$$\begin{aligned} NCR(\vec{d}, x, y) &= \frac{\sum_{j=-n}^{n} \sum_{i=-n}^{n} I_1(x+i,\ y+j)\ I_2(x+i+d_x,\ y+j+d_y)}{\sum_{j=-n}^{n} \sum_{i=-n}^{n} (I_2(x+i+d_x,\ y+j+d_y))^2} \\ SSD(\vec{d}, x, y) & \\ &= \sum_{j=-n}^{n} \sum_{i=-n}^{n} W(i,j)[I_1(x+i,\ y+j) - I_2(x+i+d_x,\ y+j+d_y)]^2 \end{aligned}$$

where W denotes a 2-D window function, and $\vec{d} = [d_x, d_y]^t$ is usually restricted to a small integer number of pixels.

- *Energy-Based Methods:* This refers to the study of the output energy of velocity-tuned filters.

- *Phase-Based Techniques:* This refers to the study of phase behavior in band-pass filter outputs. It refers to both Fourier transform-based techniques using phase information as well as zero-crossing techniques.

2.3.3 Sample Approaches

Selected candidates of the different models and techniques described above are briefly described below.

Horn and Schunk [16] combined the gradient constraint (Equation 2.1) with the constraint that the velocity varies smoothly over the whole image. This does not handle motion discontinuities as the smoothness assumption is not then satisfied. *Nagel* [29] addressed this by introducing the concept of *oriented-smoothness*, where smoothness is not imposed across steep intensity gradients (edges) in an attempt to handle occlusion. His results are sparser but more accurate.

The methods above use a global smoothness constraint. A technique employing a local smoothness constraint would yield results which are superior in both accuracy and reliability besides being more computationally efficient.

Uras, Girosi, Verri and Torre [45] assume that the velocity is constant in a local spatial and temporal neighborhood of a pixel. This produces good results for translational motion but degrades rapidly with higher order geometric deformation such as rotation or dilation. *Lucas and Kanade* [22] addressed this by offering a global motion model and parameterizing the flow field in a small spatial neighborhood. Since the computation is local, the algorithm is more efficient.

Barnard and Thompson [3] have a feature matching technique in which they use Moravec's *interest operator* (which produces a high measure at corner-like positions) to obtain candidate image features for matching. An iterative technique is then employed to find possible matches in a pair of images.

Weber and Malik [47] convolve the image sequence with a *set of linear, separable, spatio-temporal filters* which consist of functions of the Gaussian and its derivatives. The brightness constancy equation is then applied to each of the resulting images. Deficiencies in the model due to stochastic errors (due to sensor noise) and systematic errors are identified and addressed.

The brightness constancy assumption which has been made so far is violated when there is illumination non-uniformity and shading, surface reflectance changes due to inter-reflection, the motion of the object, camera or light source, etc.. *Negahdaripour and Yu* [31] address this shortcoming by allowing for a linear transformation of the brightness of an image point.

Liu, Hong, Herman and Chellapa [21] advocate a large filter support in order to alleviate the aperture problem, smooth out noise, avoid aliasing and reduce the quantization and truncation error of the filter. The image window sizes selected in their experiments were as large as 21 x 21 and as many as 9 consecutive frames were required for the computation. They recognize the inadequacy of the simple translational model and introduce *3D Hermite polynomial differential filters*. These possess several advantages: the orthogonality and Gaussian derivative properties of the filters insure numerical stability and the approach is generalizable to any desired higher order derivatives. Confidence measures were identified and a technique to integrate them was proposed.

Black [4] developed a robust algorithm to obtain the optical flow which accounts for motion discontinuities. Minimization is done for a non-convex objective composed of a data conservation constraint, a spatial coherence constraint and a temporal continuity constraint. A robust estimation framework was introduced and it handled the situation when any of the above three assumptions was violated. The resulting algorithm uses a fixed amount of computation per frame, incrementally improves the motion estimates over time and adapts to scene changes.

Anandan [1] and *Singh* [35] use different *coarse-to-fine SSD-based region matching* strategies. However they are computationally expensive and use a local model for optical flow. Hence they handle translational motion but degrade rapidly with motion which has a rotational or dilational component.

Szeliski and Coughlan [41] introduce a constrained motion model which consists of an affine patch-based feature tracker using *splines*. Both a two-frame as well as a multi-frame approach are introduced. It is computationally more efficient than traditional correlation-based approaches since the bulk of the computation is done at sparsely placed spline-control vertices.

Heeger [46] introduced an *energy-based* method which employs a *family of motion-sensitive Gabor filters*. A Gaussian pyramid scheme speeds up computation. The Gabor filters require a large spatial extent which implies instability at motion boundaries. The approach involves solving a non-linear system which is very sensitive to initial conditions. *Fleet and Jepson* [10] employ a *phase-based technique*. This decomposes the input signal according to scale, speed and orientation using a set of velocity-tuned filters. The optical flow is computed from the outputs of these filters. It produces good results when applied to relatively simple image sequences involving fairly small translation but it requires 46 3-D convolutions and 21 frames per computation. *Waxman, Wu and Bergholm* [46] study the motion of edges using second order derivatives of the edge map. All three techniques result in a bias in the component of velocity estimates towards the filter tunings.

Tian and Shah [43] develop a technique based on the Markov random model to obtain optical flow and motion segmentation from edge maps. Their technique can handle occlusion and provides superior results even across motion boundaries.

2.4 Depth from Motion

It has been observed that when a camera moves with respect to an object the corresponding displacement of image points depends both upon the camera motion as well as upon the shape of the object being imaged. Hence a study of the image displacement (optical flow) has often been employed to recover both camera motion parameters as well as image shape in the form of a depth map. If the image motion is considered as infinitesimal it is termed as the *2-D instantaneous motion field* or *optical flow* and if it is computed over discrete time intervals it is referred to as a *displacement map.*

2.4.1 Motion Models: Instantaneous versus Displacement

Consider a camera-centered coordinate system in which the origin is the center of projection, the Z-axis coincides with the optical axis and the image plane is located at $Z = f$, where f is the focal length. Let an object point X, Y, Z be imaged at pixel location (x, y) and the relative motion of the camera with respect to the scene be composed of a translational velocity (T_x, T_y, T_z) and a rotational velocity $(\omega_x, \omega_y, \omega_z)$. Techniques for computing egomotion can be categorized either as instantaneous-time methods or as discrete-time methods depending on whether input is image velocity (optical flow) or image displacement. The *instantaneous motion equations* represent the *optical flow* $(v_x, v_y)^T$ in terms of motion and depth as

$$\begin{aligned} v_x &= \frac{-T_x f + xT_z}{Z} + \frac{\omega_x xy}{f} - \omega_y(\frac{x^2}{f} + f) + \omega_z y \\ v_y &= \frac{-T_y f + yT_z}{Z} + \frac{\omega_y xy}{f} - \omega_x(\frac{y^2}{f} + f) - \omega_z x \end{aligned} \tag{2.2}$$

Alternately, the *displacement motion equations* represent *discrete displacement* $(v_x, v_y)^T$ as

$$v_x = f\left(\frac{r_1 x + r_2 y + r_3 f - \frac{T_x}{Z} f}{r_7 x + r_8 y + r_9 f - \frac{T_z}{Z} f}\right) - x$$
$$v_y = f\left(\frac{r_4 x + r_5 y + r_6 f - \frac{T_y}{Z} f}{r_7 x + r_8 y + r_9 f - \frac{T_z}{Z} f}\right) - y$$

where the rotation matrix $\mathbf{R}$ given by

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\omega_y \cos\omega_z & -\cos\omega_y \sin\omega_z & \sin\omega_y \\ \sin\omega_x \sin\omega_y \cos\omega_z + \cos\omega_x \sin\omega_z & -\sin\omega_x \sin\omega_y \sin\omega_z + \cos\omega_x \cos\omega_z & -\sin\omega_x \cos\omega_y \\ -\cos\omega_x \sin\omega_y \cos\omega_z + \sin\omega_x \sin\omega_z & \cos\omega_x \sin\omega_y \sin\omega_z + \sin\omega_x \cos\omega_z & \cos\omega_x \cos\omega_y \end{bmatrix}$$

In the latter case the rotation $\mathbf{R}$ refers to a sequence of clockwise rotations about the axes Z, Y and X in that order. There are two equations corresponding to every image point and they have seven unknowns (T_x, T_y, T_z, ω_x, ω_y, ω_z and Z). The addition of another image point moving with the same 3-D motion would add two equations but increase the number of unknowns by one, i.e., the depth Z of the new point. Not only are the equations non-linear but the translations T_x, T_y and T_z are scaled by depth Z and hence both can only be recovered up to a scale factor.

The real difference between instantaneous-time and discrete-time methods is that the instantaneous formulation is an approximation that is valid only for short time steps. Since a sequence of photographs are taken at discrete points in time, the displacement model is more accurate. However, the instantaneous model has the attractive property that the rotational and translational components can be separated.

2.4.2 Feature-based versus Optical-Flow-based Solution Techniques

The approaches used to solve the above problem can be classified according to whether they are based upon *features* or *optical flow*. Feature-based methods first find corresponding features between two images. Since the number of features obtained usually number in the hundreds, the results from those systems yield very sparse shape results.

In an optical-flow-based technique, the motion of all pixels in an image is first computed. There is a degree of uncertainty associated with these optical flow values and these uncertainties are carried over to the subsequent motion and depth values. A feature-based method can accumulate structural information for features because they are tracked across many frames and hence the motion/depth information can be labeled as totally unreliable (corresponding to those object points not tracked) or very reliable. However, due to the uncertainty associated with optical flow based techniques, there is a spectrum of reliability associated with the computed motion and depth values.

2.4.3 Representative Solution Techniques

Several techniques have been employed to obtain shape and camera motion from a sequence of time-varying images. Below are outlined a handful of the more recent ones with special emphasis to those which are similar to the technique employed in this research.

Huang and Netraveli [17] present a review of techniques for determining motion and structure of rigid bodies by knowing the locations of their corresponding features at different times or when they are projected on two different cameras. Three major categories of problems are considered:

- 3D to 3D: locations of corresponding features in 3D-space are known at 2 different times;
- 2D to 3D: locations of features in 3D-space and their projection on the camera plane are known;
- 2D to 2D: projections of features on the camera plane are known at two different times.

The features considered include points, lines, curves and corners. Since the problems are essentially non-linear, appropriate formulation is extremely important in order to avoid difficulties in either the numerical computation of the solution or the determination of non-uniqueness and multiple solutions. The paper identifies various problem formulations, efficient algorithms for their solution, the existence and uniqueness of solutions and the sensitivity of solutions to noise in the observed data.

Heeger and Jepson [14] show how the non-linear equation describing the optical flow field can be split by an exact algebraic manipulation to form three sets of equations. The first set relates the flow field to only the translational component of 3D motion. Thus, depth and rotation need not be known or estimated prior to solving for translation. Once the translation has been recovered, the second set of equations can be used to solve for rotation. Finally, depth can be estimated with the third set of equations. The algorithm is highly parallel and does not require iteration or an initial estimate.

Tomasi and Kanade [44] develop a factorization method based on the orthographic projection model. An image sequence is represented as a $2F \times P$ measurement matrix $\mathbf{W}$, which is made up of the horizontal and vertical coordinates of P points tracked through F frames. If image coordinates are measured with respect to their centroid, the measurement matrix *is of rank three.* As a consequence it can be factored into the product of two matrices $\mathbf{R}$ and $\mathbf{S}$, which represent camera rotation and object shape (depth) respectively. The two components of camera translation are computed as averages of the rows of the measurement matrix. The method can also handle and obtain a full solution from a partially filled-in measurement matrix that may result from occlusions or tracking failure. It employs many points and frames, and for most sequences, a large amount of object rotation (usually 360^o).

Szeliski and Sing Bing Kang [42] compute both shape and camera motion by performing a batch analysis of *image streams*, the temporal tracks of distinguishable image features. A least squares formulation allows for perspective or any arbitrary camera model, partial and/or uncertain tracks and even to simultaneously use point and line correspondences. The technique is simple, since a general purpose optimization technique, the Levenberg-Marquardt method, is employed. This involves only an error computation and an error propagation at each step.

Points are tracked from frame to frame using a relatively simple algorithm based on the *monotonicity operator* which computes the number of neighboring pixels whose intensity is less than that of the central pixel. Computational costs are reduced by using a sparse matrix technique. The algorithm converges to the correct solution (after typically 100 frames), even with a poor initial estimate of the true shape (constant depth plane).

Broida and Chellapa [6] introduce the concept of the Extended Kalman filter to solve for both motion and relative depth. A recursive procedure for parameter estimation is advantageous over batch procedures for the following reasons:

- Much less computation is required for each new set of data;
- Optical flow values can be extrapolated ahead in time to aid in preprocessing the next set of data;
- The authors claim that since both the plant and measurement equations are non-linear, their linear approximation preferentially weights the computation towards the more recent data. This ensures that minor deviations from the constant velocity model are correctly tracked. On the other hand a batch method would have given the best "straight line" (constant velocity) estimate for all the data.

Their algorithm assumes that the camera motion is smooth and hence it can be represented by a Taylor's series approximation. In addition, a set of point correspondences is assumed to be available.

An object-centered coordinate system is used. Rotation is represented by quarternions since the differential equation describing its time-derivative is simpler than the one involving Euler angles. The Kalman filter is initialized with the output of a batch algorithm run on the first few frames.

The algorithm was successfully demonstrated on both synthetic and real data. However, convergence of the algorithm depends upon good initial values for the parameters to be determined. Further, the performance is very highly dependent upon the values selected for the plant noise matrix. This was selected by trial and error for each experiment. Also, the matrix inversion involved in the computation of Kalman gain is guaranteed to be well-conditioned only if the measurement noise covariance is a sufficiently large multiple of the identity matrix. This condition is not satisfied when there is no prior knowledge of the motion and depth parameters.

Azarbayejani and Pentland [2] extended Broida and Chellapa's algorithm to recover focal length (as well as camera motion and object depth) from a set of point correspondences. Representational changes allow for improved stability and accuracy. The unknown focal length f is replaced by an unknown $\beta = \frac{1}{f}$ that allows the model to handle both orthographic ($\beta = 0$) as well as perspective projection. Moreover, an object-centered coordinate system allows for each object point to be fully represented by a single parameter instead of three.

Xiong and Shafer [48] have developed a robust algorithm to compute camera motion parameters and depth from a dense optical flow field. They too use an Extended Kalman Filter

(EKF) based technique which keeps track of the uncertainties associated with every value computed. Traditional EKF techniques deal with $N \times N$ matrices where N is proportional to the number of features tracked. In the case of dense optical flow measurements in a typical image of size 640×480, the value of N would be 307200.

However, the special properties of the specific problem are exploited in order to reduce the computation to $O(N)$.

A number of novel techniques were introduced. Key features are listed below:

- The uncertainty $\mathbf{P}$ of the state vector is decomposed into 4 manageable components by making use of the fact that the uncertainty associated with the depth of N points has a limited number of degrees of freedom.
- The Sherman-Morrison-Woodbury inversion formula is employed to reduce the problem of inverting an $N \times N$ matrix into one of inverting an $m \times m$ matrix where $m \ll N$.
- The dimension of the state uncertainty matrix is not allowed to increase over time. This is ensured by taking a principle value decomposition of the matrix and using the k (k fixed) largest principle values.
- Stability is improved by dynamically redefining the motion parameters so that they are equally sensitive to state uncertainty.

We employ the Xiong and Shafer ideas in the final step of this work. Further details are covered in section 4.2.2.

The techniques listed above do not address the situation when motion is accompanied by defocus blur. A small degree of blur is factored in as noise but large degrees of blur cause the techniques to fail.

2.5 Recovery of Optical Flow and Defocus Blur

Inspired by our earlier work [28], the following list work that has been done to recover depth cues when there is defocus blur in addition to optical flow, as in the case of defocused-motion or defocused-zoom.

Zhang et al [49] [50] introduce a multi-step technique to compute affine motion and defocus blur. They employ blur invariant moments defined by *Flusser and Suk* [11], [12] to normalize an image to a standard form. Affine parameters are then computed on these normalized images. The blur difference is then recovered by studying the difference between the second image and the first image which has been artificially affine transformed with the recovered parameters. They recommend the method for image restoration. However, experimental results on real images were only shown for the case of planar objects.

Deschenes et al [8] have derived a homotopy based method to compute image displacement and defocus blur simultaneously using a homotopy-based method. The unknowns are expressed in terms of partial derivatives of the two images in a non-linear relationship. They use the Levenburg-Marquardt method to solve the equations. Although their system of equations require a single scale as compared to [28], the complexity of computation is comparable. Moreover, they model the point spread function (PSF) of defocus blur as a Gaussian, which is not realistic for large degrees of blur. The experiments do not include real image pairs with substantial defocus blur as well as affine transformation. Finally, the complete affine transform is not recovered and hence the recovered parameters cannot be used for image restoration.

CHAPTER 3

COMPUTATION OF IMAGE FLOW AND DIFFERENCE IN DEFOCUS BLUR

In this section, we compute affine parameters as well as blur simultaneously from a pair of input images. We first introduce the theoretical model of blur and affine motion. Then we propose an iterative solution method. We demonstrate the validity of our theory and the proposed solution by reporting experimental results with real scenery. Note that in this work we are not concerned with motion blur since the underlying model is that of a stop-and-shoot sequence.

3.1 Theoretical Formulation

It has been shown[21] that in the case of negligible out-of-plane rotations and in the case of planar patches, the optical flow can be expressed as an affine transformation $\mathbf{A} + \vec{T}$ given by

$$\mathbf{A} + \vec{T} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{3.1}$$

Our theoretical formulation has been developed to handle scenes consisting of large, planar patches within which the affine transform model for optical flow is appropriate. Assume we have two images I_1 and I_2 where the second image is obtained after a large camera motion consisting of zooming in or out and rotation in the image plane. Even with improved technology there is a mechanical limit to the speed with which a camera can refocus itself.

Hence the large changes in depth cause the camera to get out of focus and it is appropriate to include defocus blur in the model of transformation that describes the geometric changes undergone between the two images. In the formulation below we develop a linear relationship amongst the unknowns, namely the parameters of the affine transformation as well as the change in the level of blur between the two images. The linear approximation is valid if the unknowns take small values, and hence an iterative technique is developed to correctly recover *large* unknown parameters.

For the sake of clarity in the development of the formulation we will first assume only a four-parameter affine transformation $\mathbf{A}$, where

$$\mathbf{A} = \mathbf{I} + \mathbf{B} = \begin{bmatrix} 1 + b_{11} & b_{12} \\ b_{21} & 1 + b_{22} \end{bmatrix}$$

Here the matrix $\mathbf{B}$ is composed of small elements which indicate the difference of $\mathbf{A}$ from the identity matrix. The formulation is then extended to the case of the general six-parameter affine transform which includes image translation as well. In other words the optical flow $[u, v]^T$ in that case can be expressed as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \vec{r} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

where $\vec{r} = [x, y]^T$ represents the underlying coordinate axes and $\vec{T} = [T_x, T_y]^T$ represents the pure translation component of optical flow.

3.1.1 Four-Parameter Affine Transformation

Let image I_a be an affinely transformed version of image I_1 where the transformation can be represented by the four-parameter matrix $\mathbf{A}$. Since deforming an image is equivalent to deforming the underlying coordinate axes, we can write

$$I_1(\vec{r}) = I_a(\mathbf{A}\vec{r}).$$

[24] has shown that convolution of the first image I_1 with a Gaussian (G) is equivalent to the convolution of the second image I_a with a Gaussian which has been deformed by the same transformation $\mathbf{A}$. That is,

$$I_1(\vec{r}) \otimes G(\vec{r}, \sigma^2) = I_a(\mathbf{A}\vec{r}) \otimes G(\mathbf{A}\vec{r}, \sigma^2\mathbf{A}A^T)$$

where σ is any arbitrarily chosen value for standard deviation, $\otimes$ denotes convolution, the ordinary Gaussian $G(\vec{r}, \sigma^2)$ is

$$G(\vec{r}, \sigma^2) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\vec{r}^2}{2\sigma^2}\right)$$

and the generalized Gaussian $G(\mathbf{A}\vec{r}, \sigma^2\mathbf{A}A^T)$ (i.e., the Gaussian deformed by an affine transformation $\mathbf{A}$) is

$$G(\mathbf{A}\vec{r}, \sigma^2\mathbf{A}A^T) = \frac{1}{2\pi det(\mathbf{A})\sigma^2} \exp\left(-\frac{\vec{r}^{\,T}(\mathbf{A}A^T)^{-1}\vec{r}}{2\sigma^2}\right) \tag{3.2}$$

In order to solve for the affine parameters in $\mathbf{A}$ an over-determined system can be obtained by convolving at several points $\vec{l_i}$ in both images [24] to get

$$\begin{aligned}\int I_1(\vec{r})G(\vec{r}-\vec{l_i}, \sigma^2\mathbf{I})d\vec{r} \;\; &\approx \;\; \int I_a(\mathbf{A}\vec{r})G(\mathbf{A}\vec{r}-\vec{l_i}, \sigma^2\mathbf{A}A^T)\; d(\mathbf{A}\vec{r}) \\ &+[(\mathbf{A}-I)\vec{l_i}]^T \int I_a(\mathbf{A}\vec{r})G'(\mathbf{A}\vec{r}-\vec{l_i}, \sigma^2\mathbf{A}A^T)\; d(\mathbf{A}\vec{r})\end{aligned} \tag{3.3}$$

where l_i is any point on the image plane and G' is the derivative of G with respect to the image coordinates. Equation (3.3) is Taylor's first order expansion of $G(\mathbf{A}(\vec{r}-\vec{l_i}), \sigma^2\mathbf{A}A^T)$ about the point $(\mathbf{A}\vec{r}-\vec{l_i})$.

Equation (3.3) cannot be used to solve for the unknown $\mathbf{A}$ since G on the right hand side cannot be evaluated. Hence we rewrite the above in a linear form of the unknown $\mathbf{B}$. To achieve this we express the generalized Gaussian in terms of the ordinary Gaussian and its derivatives using Taylor's expansion of Equation (3.3) about the matrix $\mathbf{B} = 0$ (which corresponds to no affine transformation) where

$$\mathbf{B} = \mathbf{A} - I = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The symbolic processing language Macsyma [5] was used to expand Equation (3.2) up to first order terms in $\mathbf{B}$ about the point $\mathbf{B}$=$\mathbf{0}$ to yield

$$\begin{aligned} G(., \sigma^2\mathbf{A}A^T) \;\; &\approx \;\; G(., \sigma^2) + \sigma^2 b_{11} G_{xx}(., \sigma^2) + \sigma^2 b_{12} G_{xy}(., \sigma^2) \\ &+\sigma^2 b_{21} G_{yx}(., \sigma^2) + \sigma^2 b_{22} G_{yy}(., \sigma^2) \end{aligned}$$

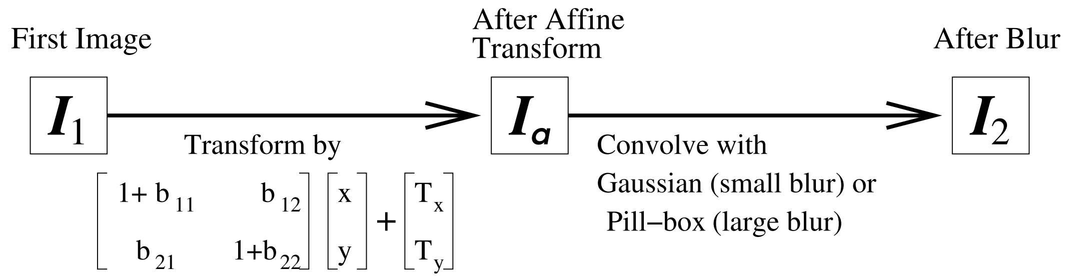

First Image
After Affine Transform
After Blur
I_1
I_a
I_2
Transform by
$\begin{bmatrix} 1+b_{11} & b_{12} \\ b_{21} & 1+b_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$
Convolve with Gaussian (small blur) or Pill–box (large blur)

Figure 3.1: Model Incorporating Affine Transformation and Defocus Blur

When the affine transformation is small, this is a reasonable approximation. Hence Equation (3.3) can be revised as

$$\begin{aligned} I_1(\vec{r}) \otimes G(\vec{r}-\vec{l}_i, \sigma^2) \quad \approx & \\ & I_a(\vec{r_1}) \otimes G(\vec{r_1}-\vec{l}_i, \sigma^2) + (\mathbf{B}\vec{l}_i)^T I_a(\vec{r_1}) \otimes G'(\vec{r_1}-\vec{l}_i, \sigma^2) \\ & +\sigma^2 b_{11}\ I_a(\vec{r_1}) \otimes G_{xx}(\vec{r_1}-\vec{l}_i, \sigma^2) + \sigma^2 b_{12}\ I_a(\vec{r_1}) \otimes G_{xy}(\vec{r_1}-\vec{l}_i, \sigma^2) \\ & +\sigma^2 b_{21}\ I_a(\vec{r_1}) \otimes G_{yx}(\vec{r_1}-\vec{l}_i, \sigma^2) + \sigma^2 b_{22}\ I_a(\vec{r_1}) \otimes G_{yy}(\vec{r_1}-\vec{l}_i, \sigma^2) \end{aligned} \tag{3.4}$$

where $\vec{r_1} = \mathbf{A}\vec{r}$. Second and higher order terms in the components of $\mathbf{B}$ are ignored since they are assumed to be small.

The derivation obtained earlier in [24], and our Equation (3.4) are now extended to include defocus blur. Denote the left hand side of Equation (3.4) by $\mathcal{I}_1(\vec{r})$ and the right hand side by $\mathcal{I}_a(\mathbf{A}\vec{r})$. Then

$$\mathcal{I}_1(\vec{r}) \approx \mathcal{I}_a(\mathbf{A}\vec{r}) \tag{3.5}$$

Now let us assume that image I_a is further transformed by a blur to yield another image I_2(see Fig. 3.1.1). The actual point spread function is given by the convolution of the diffraction limited point spread function (Airy disc) with the geometrically aberrated (defocused) blur function (circle) ([36] pages 147-150). At lower levels of blur, diffraction effects dominate and the blur can be approximated by the Gaussian, which is an approximation of the Airy disc. At higher levels of blur, the defocus effects dominate and the blur can be approximated by a convolution with a pill-box function. Hence our model assumes the Gaussian model for small levels of blur and the pill-box model for larger levels. The pill-box function (see

Fig. 3.1.1) has also been employed by [15] [37] [30].

$$Pillbox(\vec{r}, R) \quad = \quad \begin{cases} \frac{1}{\pi R^2} & for\, |\vec{r}| \leq R \\ 0 & otherwise \end{cases}$$

where increasing values of the radius R imply increasing levels of blur.

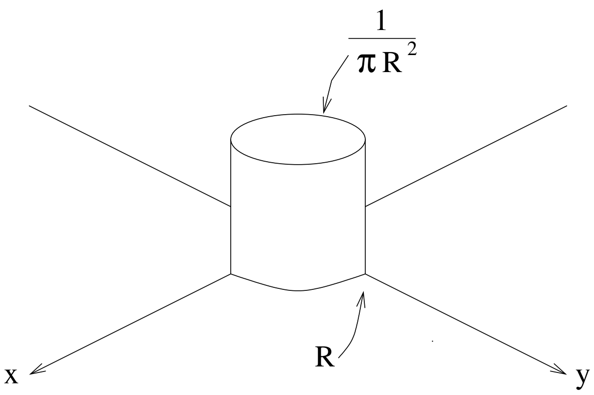


Figure 3.2: Pill-box Function

We define the generalized Pill-box function $P(\vec{r}, R, \mathbf{A})$ as

$$P(\vec{r}, R, \mathbf{A}) \quad = \quad \begin{cases} \frac{1}{\pi R^2 |\mathbf{A}|} & for\, \left|\mathbf{A}^{-1}\vec{r}\right| \leq R \\ 0 & otherwise \end{cases}$$

where $|\mathbf{A}|$ denotes the determinant of $\mathbf{A}$. The above is a Pill-box deformed by an affine transformation $\mathbf{A}$. Using a simple mathematical substitution we see that

$$P(\vec{r} - \vec{l_i}, R, \mathbf{A}^{-1}) = |\mathbf{A}|\ P(\mathbf{A}(\vec{r} - \vec{l_i}), R, \mathbf{I}) \tag{3.6}$$

In order to convolve the right hand side of Equation (3.5) with the appropriate Pill-box function, we first multiply Equation (3.5) by Equation (3.6) to yield

$$\mathcal{I}_1(\vec{r})\ P(\vec{r} - \vec{l_i}, R, \mathbf{A}^{-1}) \approx \mathcal{I}_a(\mathbf{A}\vec{r})\ |\mathbf{A}|\ P(\mathbf{A}(\vec{r} - \vec{l_i}), R, \mathbf{I})$$

Integrating both sides with respect to $\vec{r}$,we get

$$\begin{aligned}\int \mathcal{I}_1(\vec{r})\ P(\vec{r}-\vec{l_i}, R, \mathbf{A}^{-1})\ d(\vec{r}) & \\ &\approx \int \mathcal{I}_a(\mathbf{A}\vec{r})\ |\mathbf{A}|\ P(\mathbf{A}(\vec{r}-\vec{l_i}), R, \mathbf{I})\ d(\vec{r}) \\ &= \int \mathcal{I}_a(\mathbf{A}\vec{r})\ P(\mathbf{A}(\vec{r}-\vec{l_i}), R, \mathbf{I})\ d(\mathbf{A}\vec{r}) \qquad (3.7)\end{aligned}$$

The last step follows from the fact that $\int d\vec{r} = \int\int dxdy = \frac{1}{|\mathbf{A}|}\ \int\int d(x_1)d(y_1)$, where $[x_1, y_1]^T = \vec{r_1} = \mathbf{A}\vec{r}$. The left hand side of Equation (3.7) represents a convolution at point $\vec{r}$. If this is repeated at every point in the image, we have

$$LHS = I_1(\vec{r}) \otimes G(\vec{r}-\vec{l_i}, \sigma^2) \otimes P(\vec{r}-\vec{l_i}, R, \mathbf{A}^{-1})$$

We will assume that the unknowns $\mathbf{B}$ and R are very small. The function $P(\vec{r}, R, \mathbf{A}^{-1})$ can then be approximated by the Gaussian $G(\vec{r}, \frac{R^2}{2})$, to give

$$\begin{aligned} LHS &= I_1(\vec{r}) \otimes G(\vec{r}-\vec{l_i}, \sigma^2) \otimes G(\vec{r}, R^2/2) \\ &= I_1(\vec{r}) \otimes G(\vec{r}-\vec{l_i}, \sigma^2 + R^2/2) \qquad (3.8)\end{aligned}$$

The last step uses the result that the convolution of two Gaussians yields another Gaussian.

The right side of Equation (3.7) can be expressed as

$$\begin{aligned} RHS = \ & P(\mathbf{A}(\vec{r}-\vec{l_i}), R, \mathbf{I}) \otimes \Big[I_a(\vec{r_1}) \otimes G(\vec{r_1}-\vec{l_i}, \sigma^2) \\ & +(\mathbf{B}\vec{l_i})^T I_a(\vec{r_1}) \otimes G'(\vec{r_1}-\vec{l_i}, \sigma^2) \\ & +\sigma^2 b_{11}\ I_a(\vec{r_1}) \otimes G_{xx}(\vec{r_1}-\vec{l_i}, \sigma^2) \\ & +\sigma^2 b_{12}\ I_a(\vec{r_1}) \otimes G_{xy}(\vec{r_1}-\vec{l_i}, \sigma^2)\end{aligned}$$

$$
\begin{aligned}
& +\sigma^2 b_{21}\ I_a(\vec{r_1}) \otimes G_{yx}(\vec{r}_1 - \vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{22}\ I_a(\vec{r_1}) \otimes G_{yy}(\vec{r}_1 - \vec{l}_i, \sigma^2)\Big] \\
= \quad & I_2(\vec{r_1}) \otimes G(\vec{r}_1 - \vec{l}_i, \sigma^2) + (\mathbf{B}\vec{l}_i)^T I_2(\vec{r_1}) \otimes G'(\vec{r}_1 - \vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{11}\ I_2(\vec{r_1}) \otimes G_{xx}(\vec{r}_1 - \vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{12}\ I_2(\vec{r_1}) \otimes G_{xy}(\vec{r}_1 - \vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{21}\ I_2(\vec{r_1}) \otimes G_{yx}(\vec{r}_1 - \vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{22}\ I_2(\vec{r_1}) \otimes G_{yy}(\vec{r}_1 - \vec{l}_i, \sigma^2) \qquad (3.9)
\end{aligned}
$$

The last step follows because convolution is commutative, and because I_2 is obtained from I_a as a result of a convolution with a Pill-box function. Finally, equating eqns. (3.8) and (3.9) we get a relationship which is linear in the unknown $\mathbf{B}$, albeit still non-linear in R.

3.1.2 General Affine Transform Including Image Translation

We now consider the case when there is an arbitrary image translation $\delta\vec{T}$ along with the above affine transformation and blurring.

We see that convolution about a point $\vec{r}+\delta\vec{T}$ by a Gaussian $G(\vec{r}, .)$ is equivalent to convolving about the point $\vec{r}$ by the Gaussian $G(\vec{r} - \delta\vec{T}, .)$. Hence, once a rough estimate of the image translation $\vec{T_0}$ is known, the residual translation $\delta\vec{T}$ is computed by using the result

$$
I_1(\vec{r} + \vec{T_0} + \delta\vec{T}) \otimes G(\vec{r} - \vec{l}_i, \sigma^2 + R^2/2) = I_1(\vec{r} + \vec{T_0}) \otimes G(\vec{r} - \vec{l}_i - \delta\vec{T}, \sigma^2 + R^2/2)
$$

Substituting the above in Equations (3.8) and (3.9) we get

$$\begin{aligned}
I_1(\vec{r}+\vec{T}_0) \otimes G(\vec{r}-\vec{l}_i-\delta\vec{T}, \sigma^2+R^2/2) \quad \approx \quad & I_2(\vec{r}_1) \otimes G(\vec{r}_1-\vec{l}_i, \sigma^2) \\
& +(\mathbf{B}\vec{l}_i)^T I_2(\vec{r}_1) \otimes G'(\vec{r}_1-\vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{11}\ I_2(\vec{r}_1) \otimes G_{xx}(\vec{r}_1-\vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{12}\ I_2(\vec{r}_1) \otimes G_{xy}(\vec{r}_1-\vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{21}\ I_2(\vec{r}_1) \otimes G_{yx}(\vec{r}_1-\vec{l}_i, \sigma^2) \\
& +\sigma^2 b_{22}\ I_2(\vec{r}_1) \otimes G_{yy}(\vec{r}_1-\vec{l}_i, \sigma^2) \qquad (3.10)
\end{aligned}$$

Thus Equation (3.10) relates the affine unknowns, $\mathbf{B}$ and $\delta\vec{T}$ and the level of blur R. To solve this equation, we linearize in $\delta\vec{T}$ and subsequently in R.

Using the relation that

$$G_1(\vec{r}-\vec{l}_i-\delta\vec{T}, .) \approx G_1(\vec{r}-\vec{l}_i, .) - \delta\vec{T}^T G'(\vec{r}-\vec{l}_i, .)$$

(which is a Taylor's series approximation about the point $(\vec{r}-\vec{l}_i)$) we obtain Equation (3.10) in a linear form of the unknowns $\mathbf{B}$ and $\delta\vec{T}$.

Finally, below, we linearize it further in the unknown radius of blur, R. We first set $\sigma^2 = \eta$ and define

$$G_1(\vec{r}, \eta) = \frac{1}{2\pi\eta} exp\left(\frac{-\vec{r}^T\vec{r}}{2\eta}\right)$$

Let $\sigma^2 + \frac{R^2}{2} = \eta_2 = \eta_1 + \beta$ where η_1 is the current best estimate of η and β is the residual error in the estimation. $G_1(\vec{r}, \eta_2)$ can be evaluated by expanding by Taylor's series about the point η_1, to give

$$G_1(\vec{r}, \eta_2) = G_1(\vec{r}, \eta_1) + \beta \left. \frac{\partial G_1}{\partial \eta} \right|_{\eta=\eta_1}$$

Expanding the left hand side of Equation (3.10) about η_1,

$$\begin{aligned} &I_1(\vec{r}) \otimes G(\vec{r}, R^2 + \sigma^2) \\ &\qquad \approx I_1(\vec{r}) \otimes G_1(\vec{r}, \eta_1) + \beta I_1(\vec{r}) \otimes \left. \frac{\partial G_1}{\partial \eta} \right|_{\eta=\eta_1} \\ &\qquad = I_1(\vec{r}) \otimes G_1(\vec{r}, \eta_1) + 2\beta \; I_1(\vec{r}) \otimes \nabla^2 G_1 |_{\eta=\eta_1} \end{aligned}$$

The last step follows from $\frac{\partial G_1}{\partial \eta} = 2\nabla^2 G_1$.

The overall computation is embedded in a 2-level pyramid scheme. All parameters are estimated using the higher level first, and then are propagated as initial estimates for the detailed level. A translational component of $(T_x^s, T_y^s)^T$ in an image shrunk by a factor of s corresponds to $(sT_x^s, sT_y^s)^T$ in the original image. The pyramid is constructed by performing bilinear interpolation to shrink overlapping patches of the original images.

3.2 Solution Method

Using one or more values of σ (in our experiments, values of σ were chosen as 1.75, 2.5, 3.0, 3.5, 4.5) as well as different values of $\vec{l_i}$ an over-determined system is obtained to solve for the unknowns, the affine parameters $\mathbf{B}$, the residual translation $\delta\vec{T}$ and the radius of blur R.

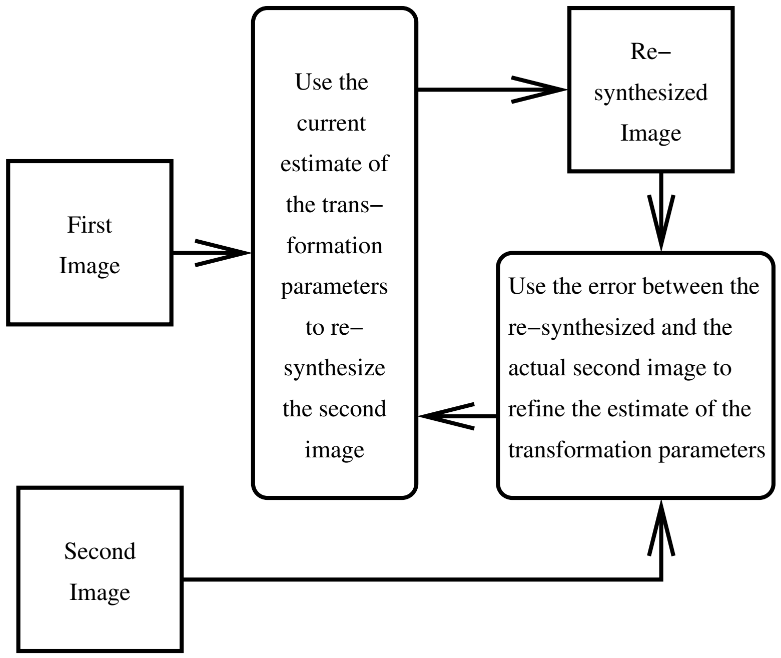

Use the current estimate of the trans– formation parameters to re– synthesize the second image
Re– synthesized Image
First Image
Use the error between the re–synthesized and the actual second image to refine the estimate of the transformation parameters
Second Image

Figure 3.3: Overview of the Iterative Parameter Estimation Process

In practice even if the initial estimate of the translation is three to four pixels in error, the above method is able to rightly identify the exact image translation to sub-pixel accuracy.

Since the linearization is an approximation which holds true for only small values of the unknowns, an iterative scheme was developed to handle large deformations (see Fig. 3.2). At the first iteration approximate values of the unknowns are obtained. An intermediate synthetic image is then obtained by transforming (via bilinear interpolation) the first image using the computed affine parameters and then blurring the latter using the Pill-box model of blur.

At the next iteration the difference between this intermediate image and the second image is computed and the overall affine parameters and degree of blur are calculated. The first image is then transformed using these *new* parameters to form the *next* intermediate image. This process is repeated until the residual of the linear system is below a predetermined threshold.

Since the linearization approximation is valid when the deformations are small and the method has to iteratively recover smaller and smaller values of the unknowns, the above method converges.

The parameters $\mathbf{B}$ and η_1 are initialized to $\mathbf{0}$ and some σ^2 respectively, corresponding to no affine transformation or blurring. At every iteration, $\mathbf{B}$ is updated as

$$\mathbf{A}_{new} \leftarrow \mathbf{A}_{residual} \cdot \mathbf{A}_{previous}$$

(matrix multiplication) and R is updated as

$$R \leftarrow \sqrt{2}\sqrt{\eta_1 + \beta - \sigma^2} \tag{3.11}$$

and a new value of η_1 is generated as $\eta_1 = \sigma^2 + \frac{R^2}{2}$.

A negative value for $\eta_1 + \beta - \sigma^2$ indicates that the second image has been sharpened instead of blurred. In this situation, we compute the level of blur between the second image and first image. This offers the capability of automatically segmenting an image into areas which are blurred, sharpened or focally unperturbed (i.e., $R = 0$).

3.3 Implementation and Results

We first describe results obtained when a real image was artificially deformed using large affine parameters and substantial levels of blur. The method was then implemented on several real image pairs, i.e., the second image in a pair was obtained by camera motion or zoom rather than artificially generated.

At every point a texture measure was computed. In a small patch (typically 20×20) around the pixel the difference between each pixel and its n^{th} neighbor (at its right and below it) was summed. When the average difference was less than 10 (grey-levels) the pixel was not considered to have enough texture and the recovery of affine parameters was not attempted there. The over-determined system was solved using a least mean squares algorithm. We used gradient descent. The residual of the least squares system is interpreted as a measure of confidence in the observed parameters. It was experimentally determined that the number and organization of the points l_i (see Equation 3.3) at which convolutions are performed depend upon both the level of affine deformation as well as the level of blur. For instance, when recovering relatively large values, such as a radius of blur of 3.5 pixels and a scaling factor of 1.2 as well as in-the-plane translation of $[2, 2]^T$, one needed 121 evenly spaced points within a 41×41 grid. However for smaller deformations such as a radius of blur of one pixel, a scale factor of 1.04, an in-the-plane rotation of 2^o, and an in-the-plane translation of $[1.0, 0.5]^T$ pixels, 25 evenly-placed points l_i were chosen within a 17×17 grid. If the approximate level of deformation is known *a priori*, the number and configuration of points l_i can be judiciously selected. For output purposes, optical flow is the translational component of the computed affine transformation.

3.3.1 Artificially Deformed Images : Experiments 1 and 2

We ran several experiments on a wide range of test images. We artificially deformed them by performing an affine transform using bilinear interpolation (expansion factors ranging from 0.7 to 1.4, rotations up to 30 degrees and image translations within 4 pixels) followed by a blurring operation. The program correctly recovered all parameters. The first image was then transformed according to the parameters recovered with the effect of blurring removed.

Results for two sets of images are shown in Figures 3.3.1.1 and 3.3.1.2.

3.3.1.1 Experiment 1: Brain

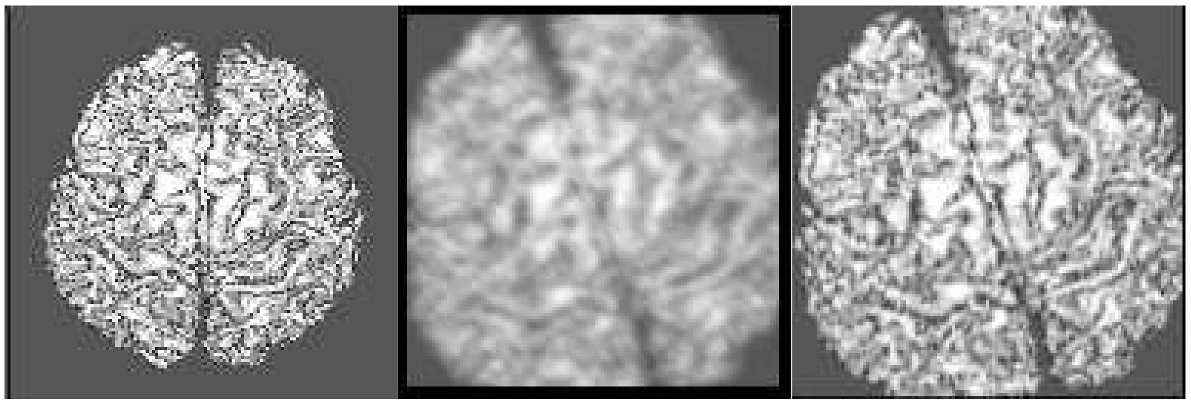

(a) Original Image (b) Artificially Deformed (c) Regenerated Image

Figure 3.4: Experiment 1 : Artificially Transformed Real Image (Brain)

An image of a brain (Fig. 3.3.1.1(a)) was transformed by $\begin{bmatrix} 1.2216 & -0.4446 \\ 0.4446 & 1.2216 \end{bmatrix} + \begin{bmatrix} -1.0 \\ 0.7 \end{bmatrix}$ (scaling of 1.3, rotation of 20^o, translation of $[-1, .7]^T$) and Pill-box blurred with radius R=3.5 pixels.

The artificially transformed image is shown in (Fig. 3.3.1.1(b)). The affine transformation recovered were $\begin{bmatrix} 1.2215 & -0.4446 \\ 0.4446 & 1.2218 \end{bmatrix} + \begin{bmatrix} -1.0 \\ 0.7 \end{bmatrix}$ and the radius of the blur was computed to be 3.5 pixels. Fig. 3.3.1.1(c) has been generated from the first image using the recovered affine parameters at the center of the image.

3.3.1.2 Experiment 2: Pineapple

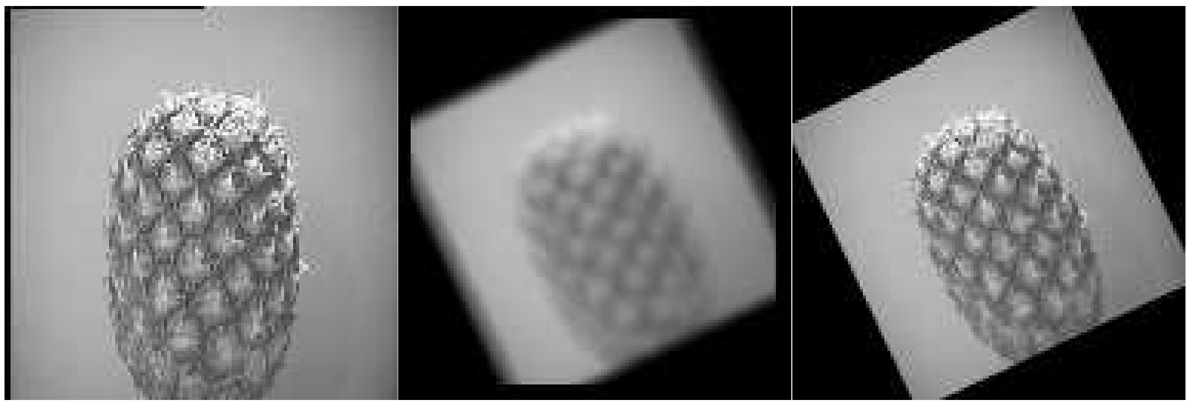

(a) Original Image (b) Artificially Deformed (c) Regenerated Image

Figure 3.5: Experiment 2 : Artificially Transformed Real Image (Pineapple)

Figures 3.3.1.2(a), (b) and (c) describe an experiment done with a photograph of a pineapple. The affine transformation involved is $\begin{bmatrix} 0.7250 & -0.3381 \\ 0.3381 & 0.7250 \end{bmatrix}$ and the blur radius is 4.5 pixels. The affine parameters were accurately recovered as $\begin{bmatrix} 0.7251 & -0.3380 \\ 0.3381 & 0.7252 \end{bmatrix}$ and the recovered radius of blur was 4.47 pixels.

3.3.2 Pairs of Real Images : Experiments 3 through 9

Experiments were then conducted on pairs of images of real scenery. In order to study the limits of the above theory, a series of experiments of increasing complexity was performed. In later chapters we run this algorithm on experiments of still further complexity.

In all these experiments the camera's aperture was deliberately set so that motion induced a large level of blur. This was done to demonstrate that the algorithm is able to handle large deformations. Also, in a depth-from-defocused-motion system it may be useful to set up a stop-and-shoot sequence which facilitates large blur.

In certain cases the blur level was uniformly large. We show that the flowfield can be recovered in spite of considerable blur. This flowfield can be used as input to any depth-from-motion algorithm.

However, in other cases, the camera parameters and size of translation resulted in varying degrees of blur. We show in the next chapter how the variation in blur information can be taken advantage of in order to obtain an initial estimate of depth.

3.3.2.1 Experiment 3: Flat Image of Bird

In the first experiment a photograph of a bird was imaged. The object was hence flat. The camera was moved closer to the object and to a slight angle to obtain a second image as can be seen in Figures 3.6 and 3.7.

For the sake of visual clarity, the flowfield and radius of blur/sharpening are highly subsampled in all the images that follow.

The flowfield is shown in Figures 3.8. The degree of blur is displayed in Figure 3.9 where the radius of the circle is proportional to the radius of blur. The average radius of blur 3.48 and the standard deviation is 0.43.

It can be seen that large displacements were accurately retrieved in spite of substantial defocus blur.

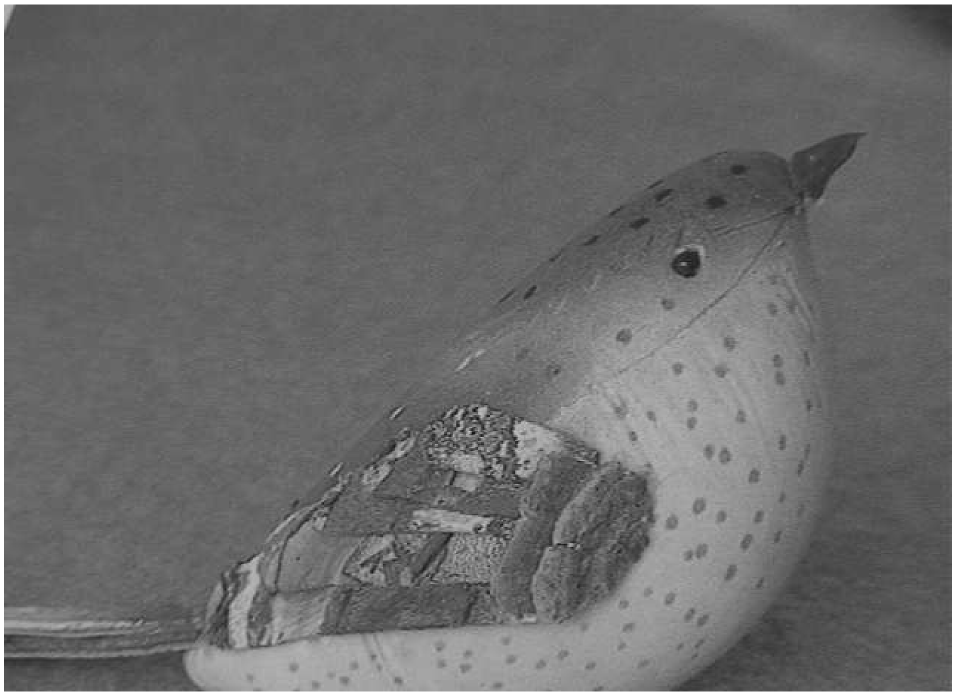


Figure 3.6: Experiment 3: Flat Bird (Image 1)

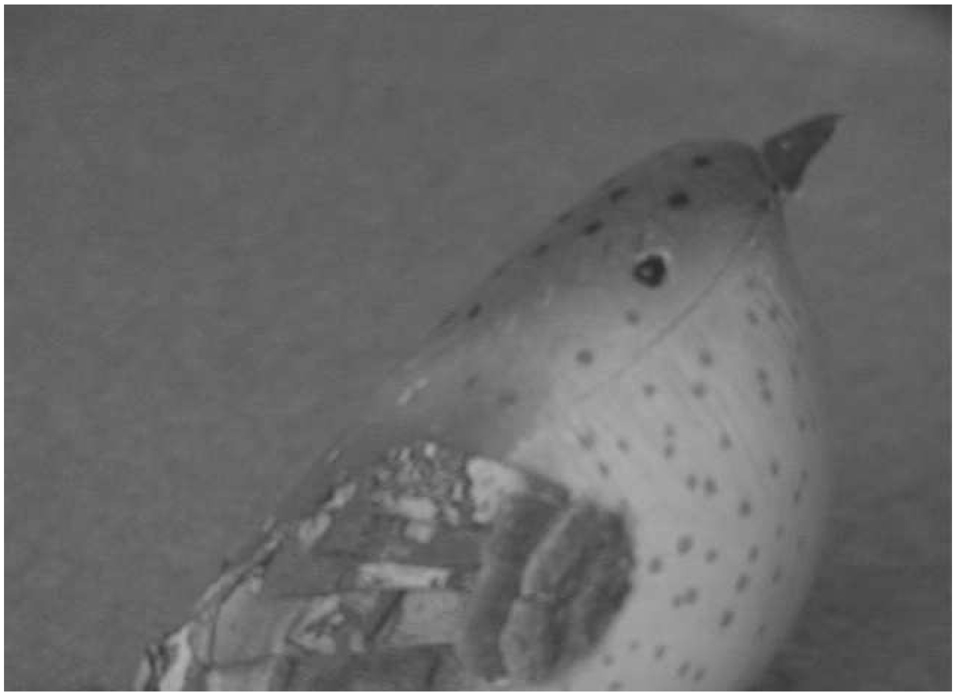


Figure 3.7: Experiment 3: Flat Bird (Image 2)

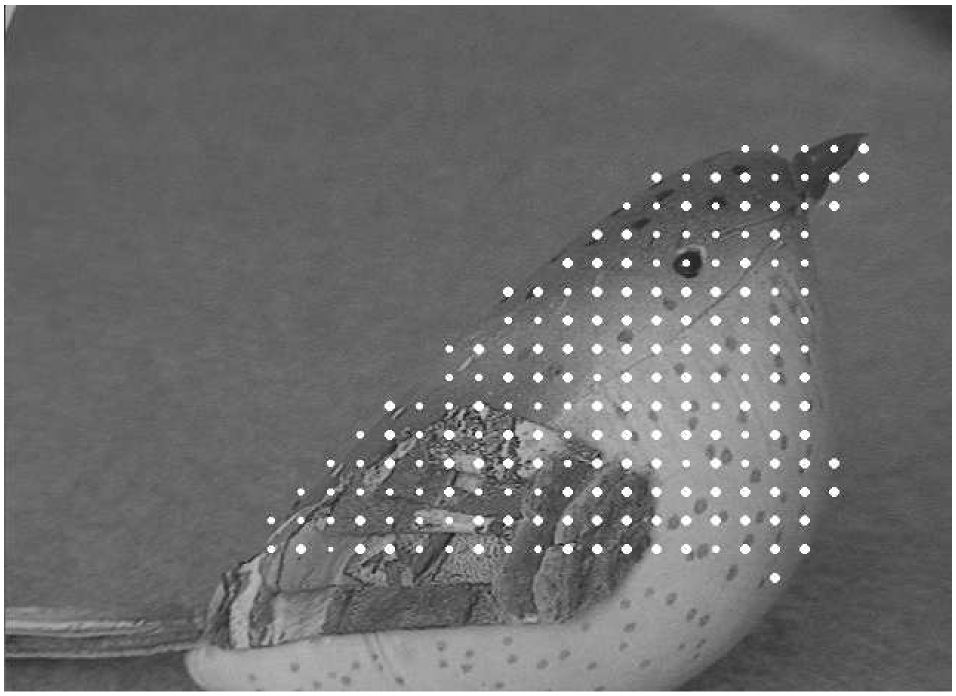


Figure 3.8: Experiment 3: Flat Bird (Subsampled Radius Map)

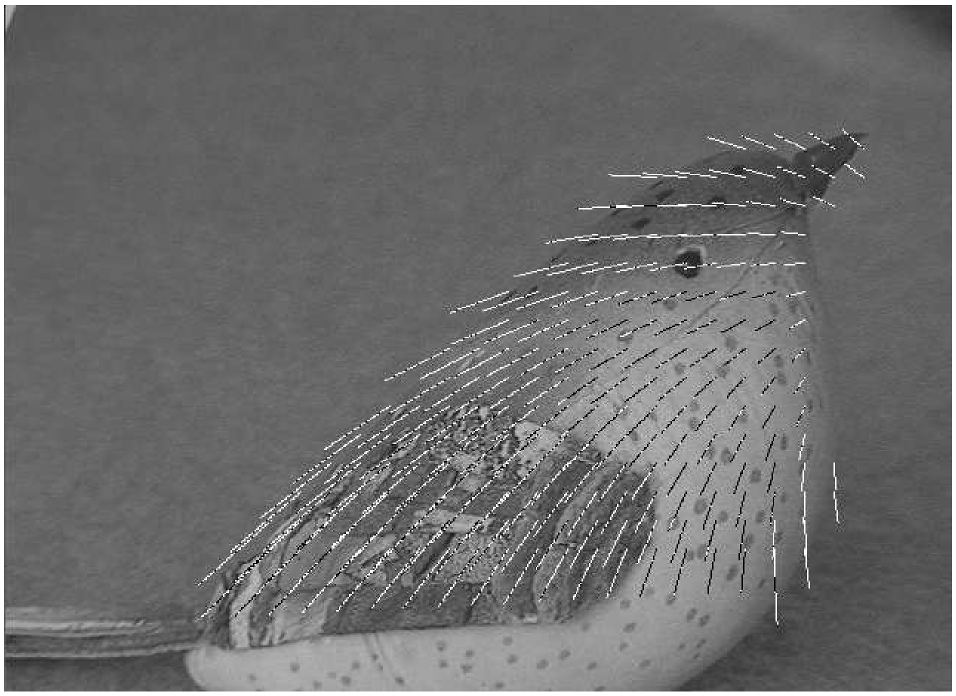


Figure 3.9: Experiment 3: Flat Bird (Subsampled Flow Field)

3.3.2.2 Experiment 4: Photograph of Garden (Zoom In)

In the next experiment the camera was held stationary and the zoom feature of the lens was employed. We first captured a photograph of a picture of a beautiful garden. Keeping the aperture fixed, we zoomed in to the picture and took another photograph. The second photograph was considerably blurred and enlarged. The results of the experiment are displayed in Figures 3.10 through 3.13.

The average radius of blur 3.70 and the standard deviation is 0.51.

We see that the optical flow was recoverable in spite of considerable blurring and change in size.

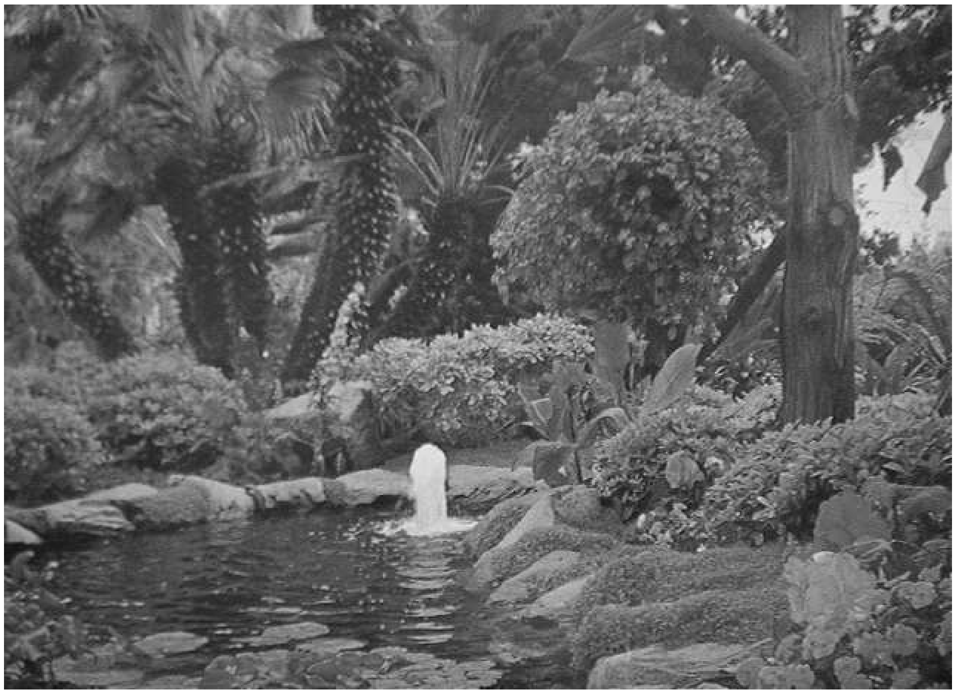


Figure 3.10: Experiment 4: Flat Garden (Image 1)

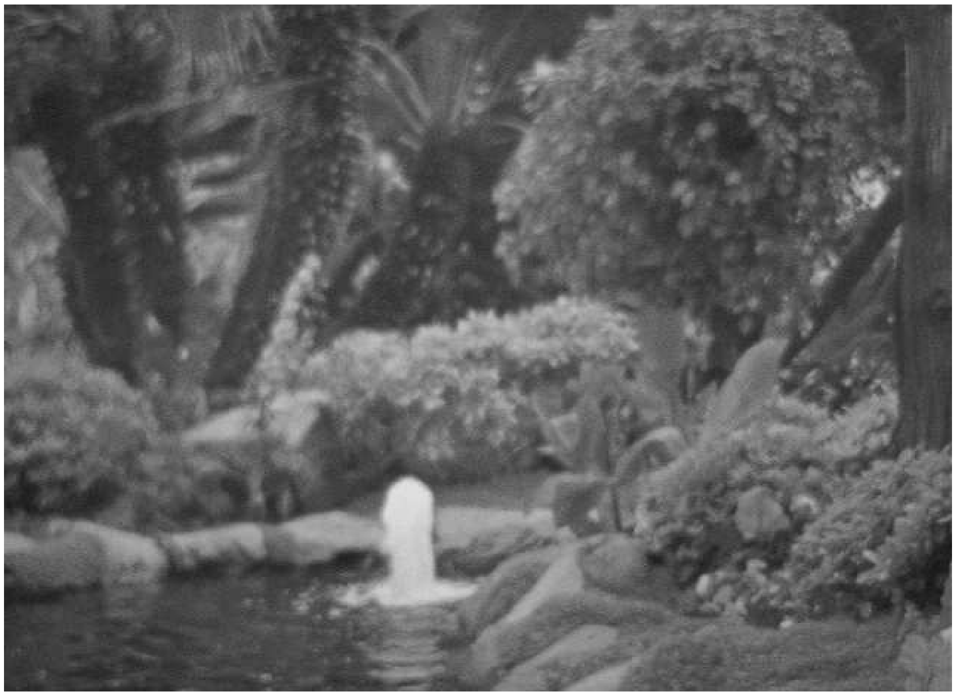


Figure 3.11: Experiment 4: Flat Garden (Image 2)

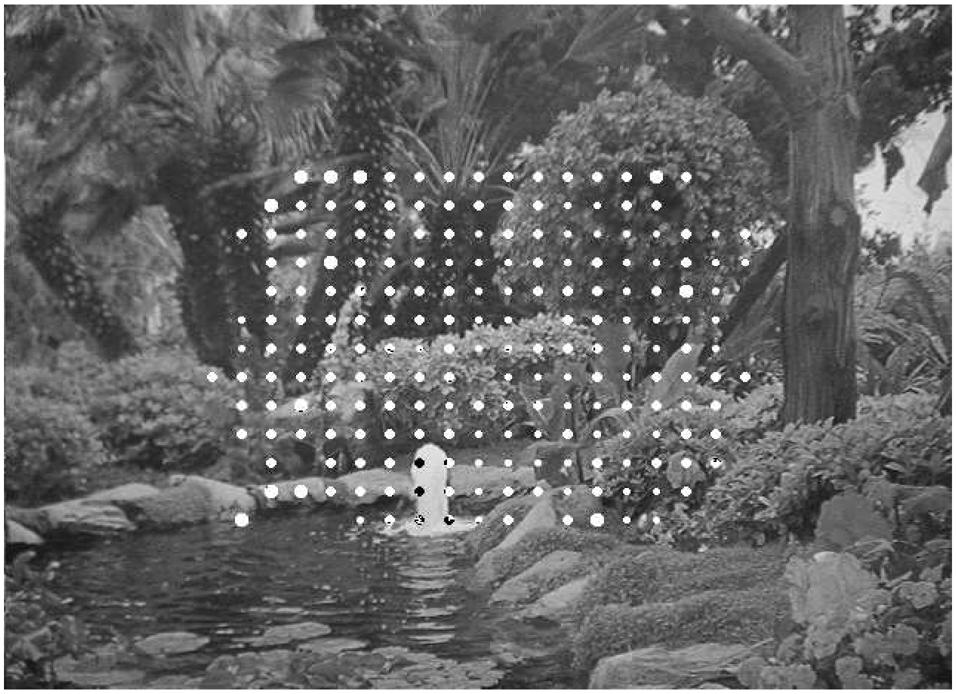


Figure 3.12: Experiment 4: Flat Garden (Radius Map)

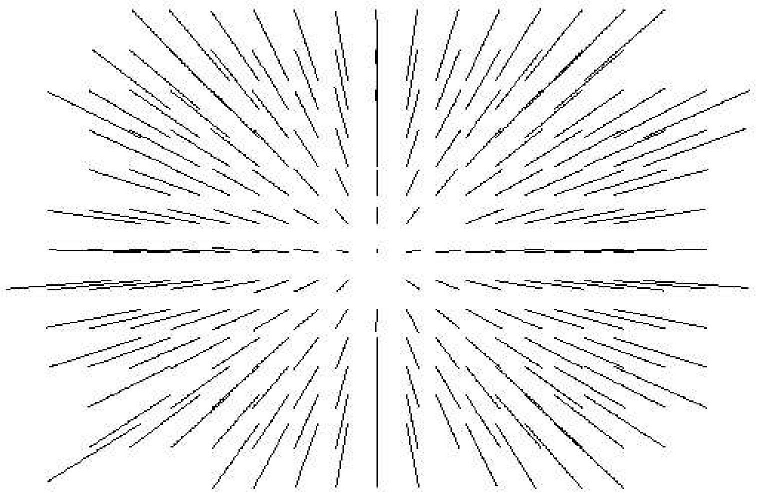


Figure 3.13: Experiment 4: Flat Garden (Flow Field)

3.3.2.3 Experiment 5: Slanted Geometric Pattern

In order to facilitate computing the correct affine parameters using point correspondences, geometric patterns were photographed with a Pulnix camera.

Results of the experiment are shown in Figures 3.14 through 3.17.

Figure 3.14 shows the first image of a slanted geometrical pattern and Fig. 3.15 shows the second image, which is obtained by keeping the camera stationary and using the zoom mechanism.

The affine parameters obtained at the center were $\begin{bmatrix} 0.9400 & 0.0146 \\ 0.0149 & 0.9327 \end{bmatrix}$ along with an image translation of $\begin{bmatrix} -0.12 \\ -0.05 \end{bmatrix}$ and the radius of pill-box blur of 4.4 pixels.

Since the degree of blur varies as one traverses the image of the slanted object from left to right, the affine parameters and the blur were obtained for a thin horizontal strip. Fig. 3.16(a) shows a strip from the first image and Fig. 3.16(b) shows the strip from the same position of the second. A map of the radii of blur in the above strip is shown in Fig. 3.16(c). The radius varied from 0.5 pixels obtained near the right edge to 7.0 pixels obtained near the left edge of the strip. Black patches in the bottom left corner represent points of no convergence. The flow field in a 41×41 pixel window centered on the image is shown in Fig. 3.16(d).

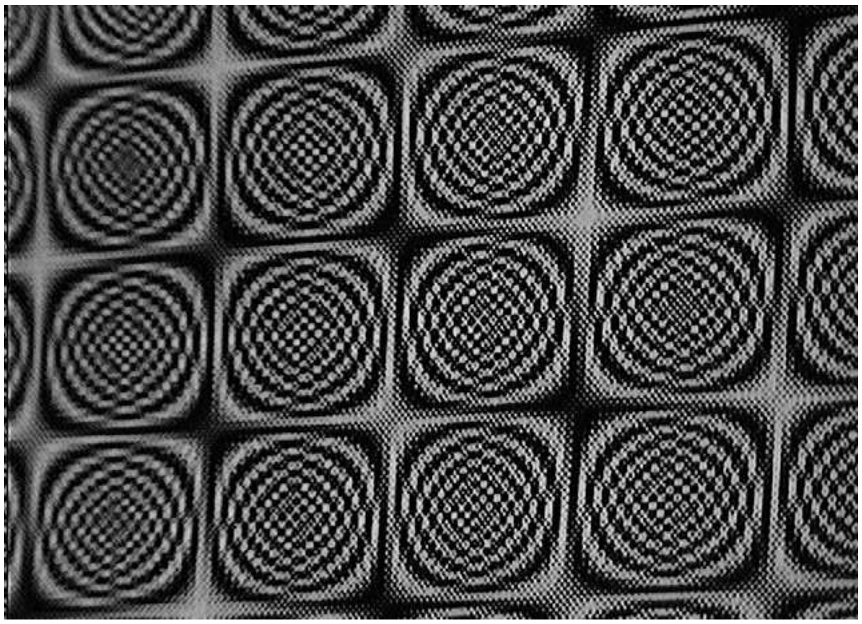


Figure 3.14: Experiment 5: Slanted Geometric Pattern (Image 1)

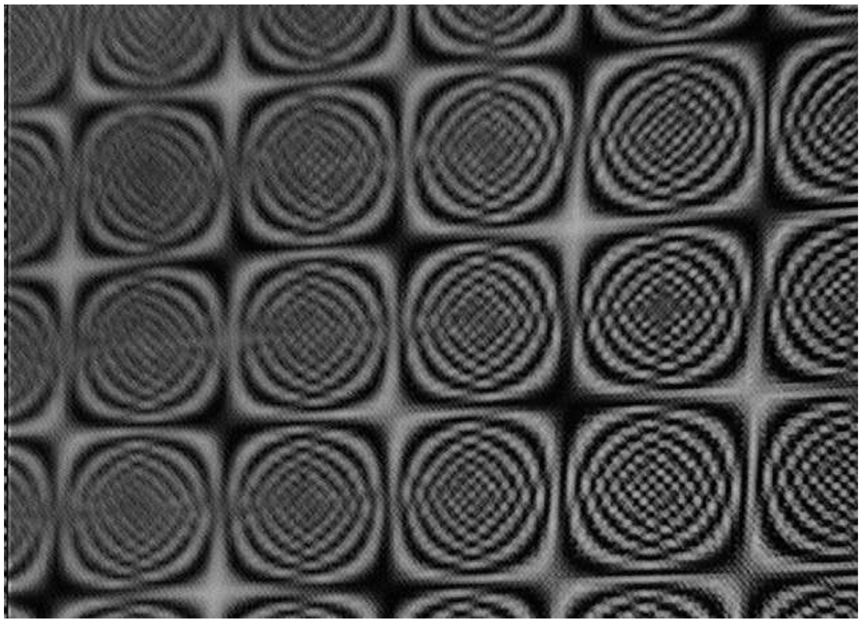


Figure 3.15: Experiment 5: Slanted Geometric Pattern (Image 2)

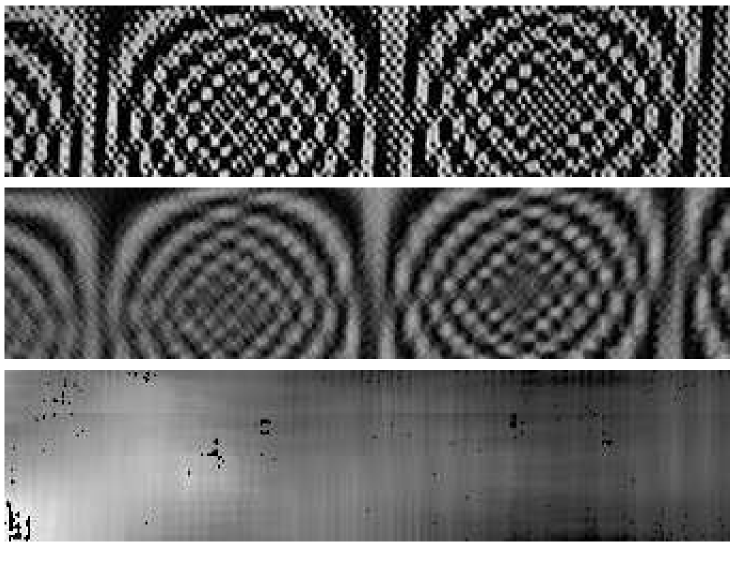

(a) Strip from first image
(b) Corr. strip of 2^{nd} image
(c) Radius map of above strip

Figure 3.16: Experiment 5: Strips from Both Images

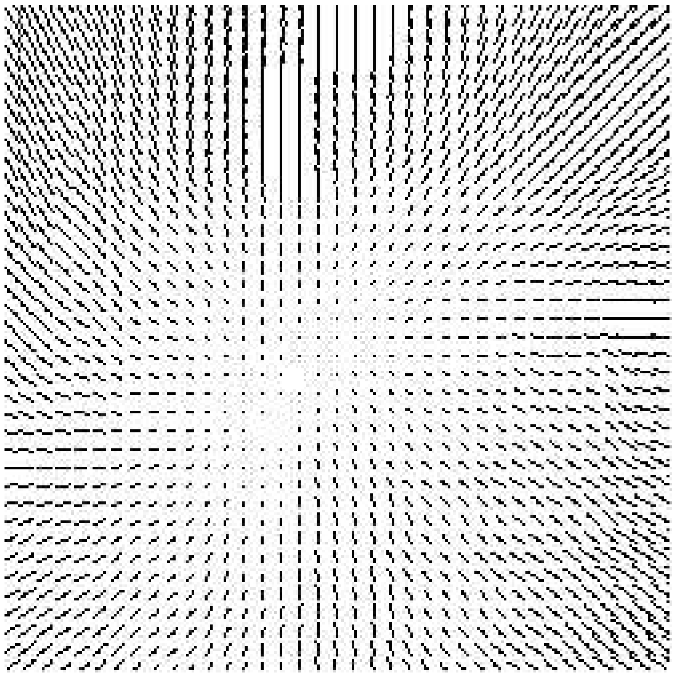


Figure 3.17: Experiment 5: Geometric Pattern (Flow Field in 41×41 Patch)

3.3.2.4 Experiment 6: Two Objects At Different Depths

Figures 3.18 through 3.22 demonstrate the result of the next experiment. In this experiment we imaged two objects. One was a box with an irregular design, the other was a slanted box with a picture of a woman. The translation component of camera motion caused the left object to get blurred and the right to get sharpened. The program correctly recognized the two situations.

Figures 3.20 and 3.21 show the sub-sampled radius map and flowfield respectively. The radii corresponding to points which got sharper/blurred are shown as open/filled-in circles. Since the flow is large, the flow image is highly sub-sampled. In areas corresponding to the woman's dress there is insufficient texture to compute the optical flow and the blur.

In order to assess the accuracy of the blur parameter, we considered a 60×60 patch (Figure 3.22(a)) centered on the position (205, 320). This patch was affine transformed and blurred using the recovered parameters. The modified patch (Figure 3.22(c)) was compared with the equivalent patch in the second image (Figure 3.22(b)). A difference image (Figure 3.22(d)) was computed. The average absolute error of the difference was 12 grey values and the maximum error was 67.

Once again, in this experiment the camera's aperture was deliberately set so that motion induced a large level of blur. This was done to demonstrate that the algorithm is able to handle large deformations.

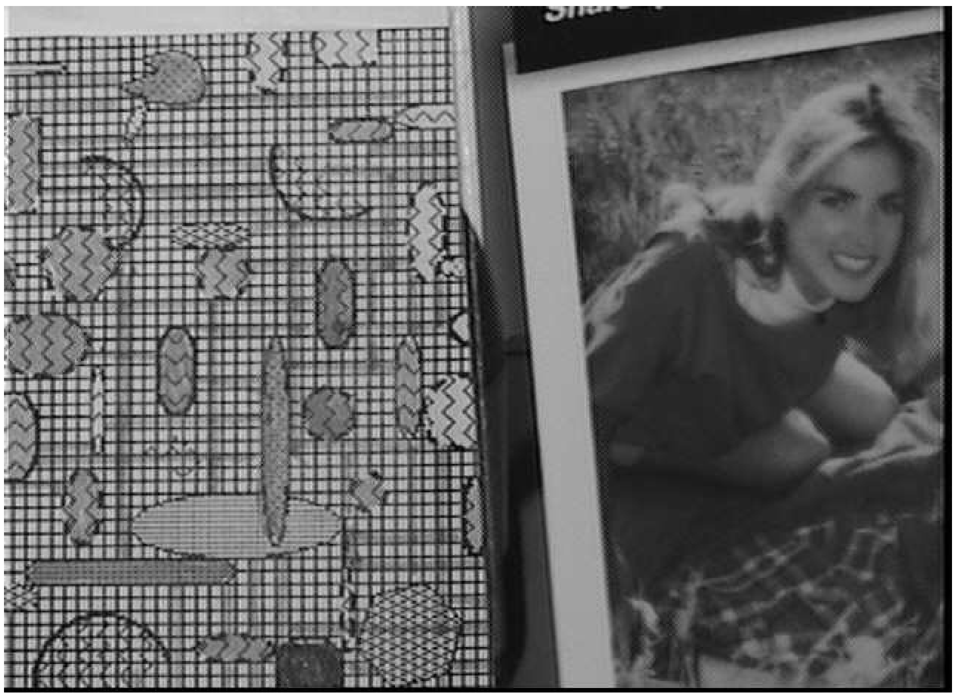


Figure 3.18: Experiment 6: Two Objects (Image 1)

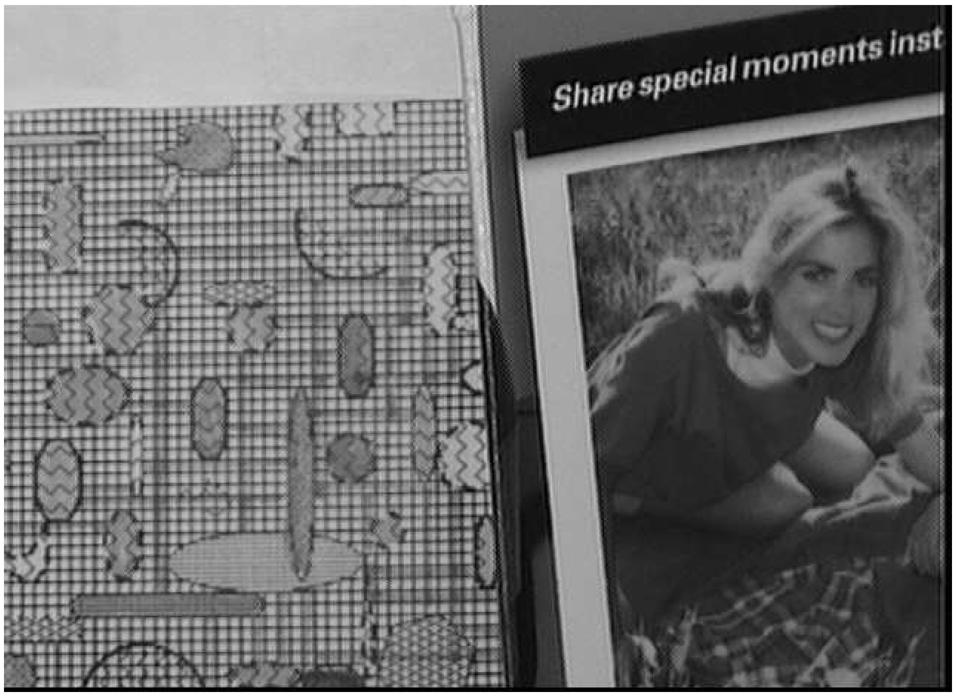


Figure 3.19: Experiment 6: Two Objects (Image 2)

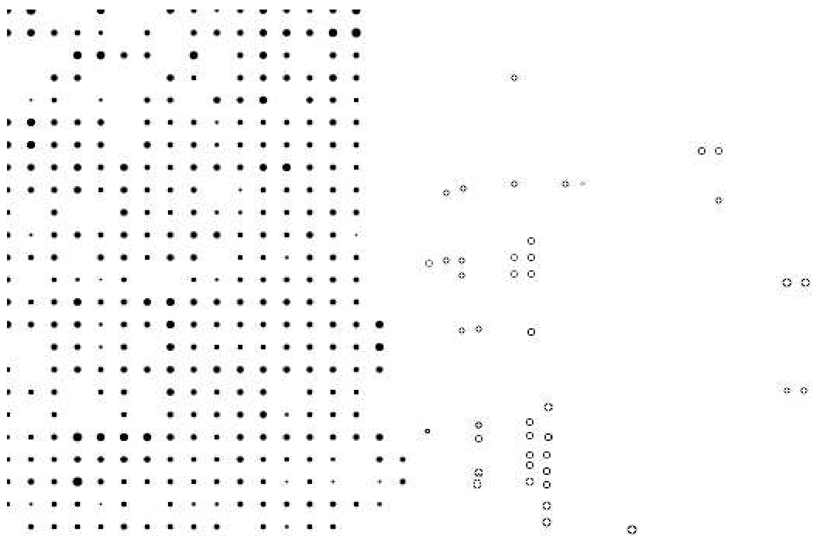


Figure 3.20: Experiment 6: Two Objects (Radius Map)

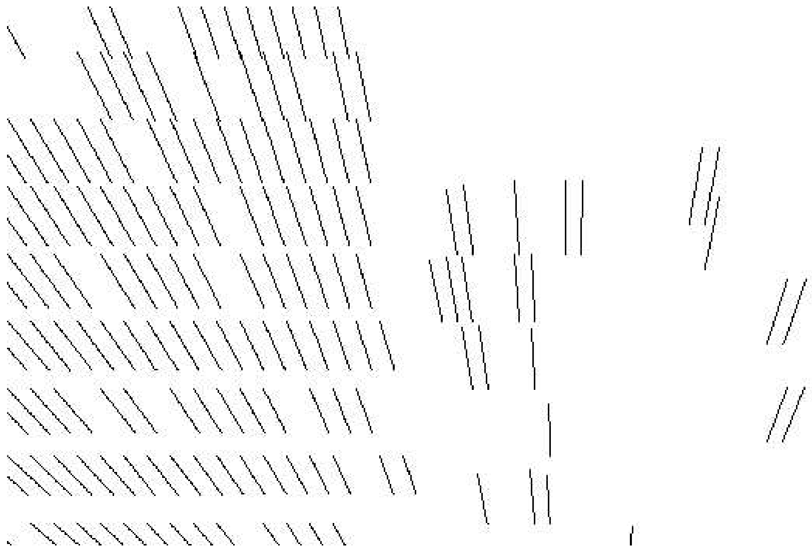
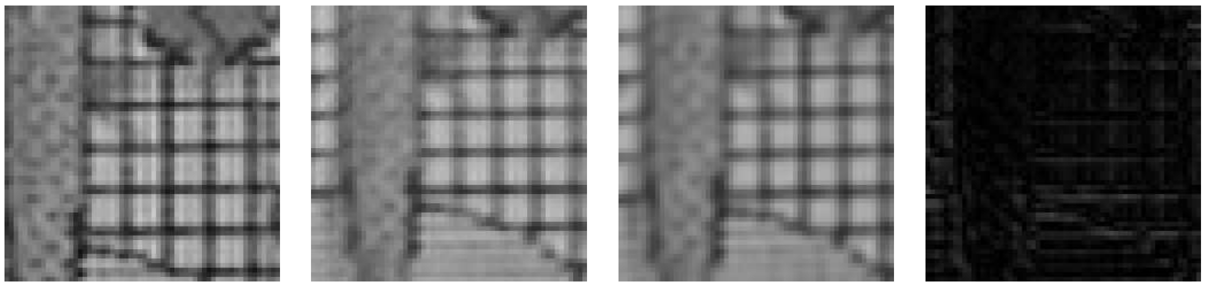


Figure 3.21: Experiment 6: Two Objects (Flow Field)



(a) Patch from 1st Image (b) Corres.Patch in 2nd Image (c) Regenerated Image (d) Difference Image

Figure 3.22: Experiment 6 (Analysis)

3.3.2.5 Experiment 7: Frontoparallel and Slanted Boxes

Figures 3.23 through 3.28 demonstrate the result of another experiment in which the object consisted of a scene containing two objects, a box with a grid design with one side placed fronto-parallel with respect to the camera and a printed box above it placed at a very steep angle. The first image (Figure 3.23) was taken from above. The box with the grid design was then rotated and the camera was moved downwards towards the objects and the second image (Figure 3.24) obtained. The box with the grid design is uniformly blurred (due to it being fronto-parallel). The printed box on the other hand, is in sharper focus in certain areas. An estimate of the translation for a point on the grid box was obtained and the residual transformation at a 31x31 patch centered at that point was computed. Figures 3.25(a) and (b) show the original patches in the two images. The affine transformation and blur were computed in a 31x31 area. The original patch was regenerated using computed affine parameters and this image is displayed in Figure 3.25(c) for visual comparison with Figure 3.25(b). Figures 3.26(a) and (b) show respectively the corresponding flow map (for the residual component) and the radius map computed within the patch.

The affine transformation at the center of the patch was computed to be

$$\begin{bmatrix} 1.0633 & -0.1337 \\ 0.1375 & 1.0598 \end{bmatrix}$$

and the radius of the blur was 2.7 pixels. The computed radius values in the map varied very slightly from 2.6 pixels to 2.8 pixels, as can be expected for a fronto-parallel plane.

While Figures 3.25 and Figures 3.26 describe results for the frontoparallel grid box, results for the slanted printed box part of the scene are shown in Figures 3.27 and Figures 3.28. For the slanted box, corresponding patches were extracted. For this box object, the second image is in sharper focus than the first.

Results of the inverse transformation for this printed box are shown in Figure 3.27. On close examination of the two original images it was seen that the level of blur in the upper part of the printed box is the same. However, in the lower part of the printed box, the second image is in sharper focus. This relative change in blur level can be noted in Figure 3.28. White regions in the radius map represent points of non-convergence. In the above experiment, since we do not have ground truth, we can only note qualitatively that the blur radius maps and the affine transformations appear correct.

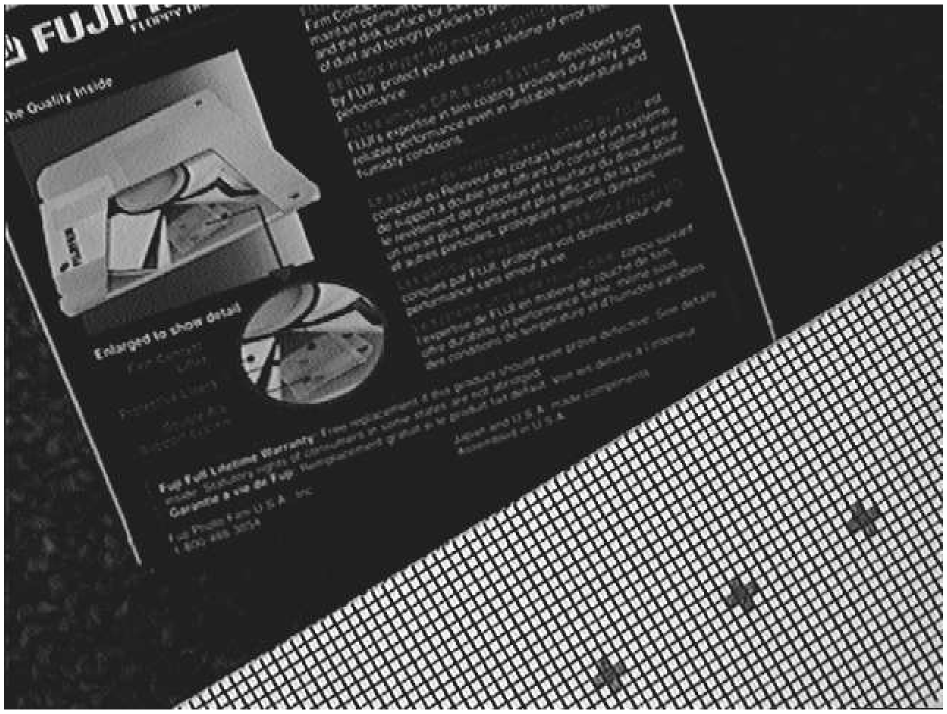


Figure 3.23: Experiment 7: Frontoparallel and Slanted Boxes (Image 1)

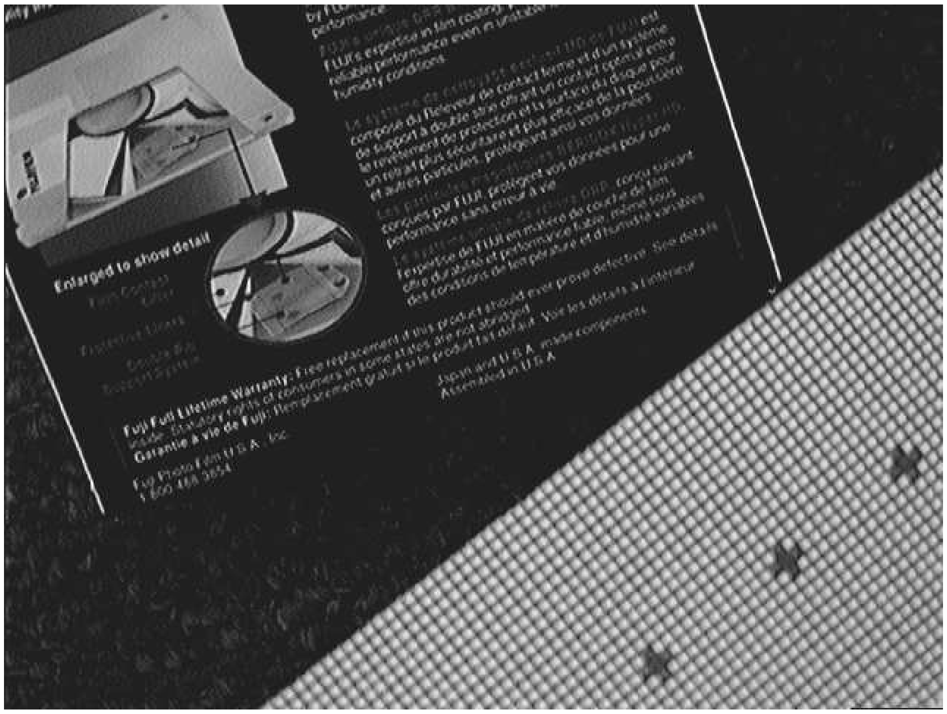


Figure 3.24: Experiment 7: Frontoparallel and Slanted Boxes (Image 2)

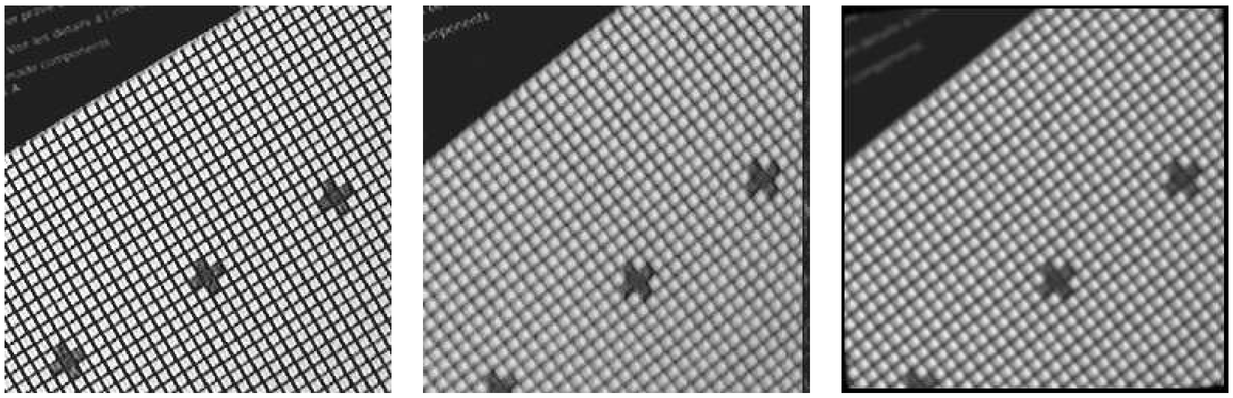

(a) Patch of first image (b) Corresponding patch of second (c) First regenerated using all recovered parameters

Figure 3.25: Experiment 7: Frontoparallel Box (Patches)

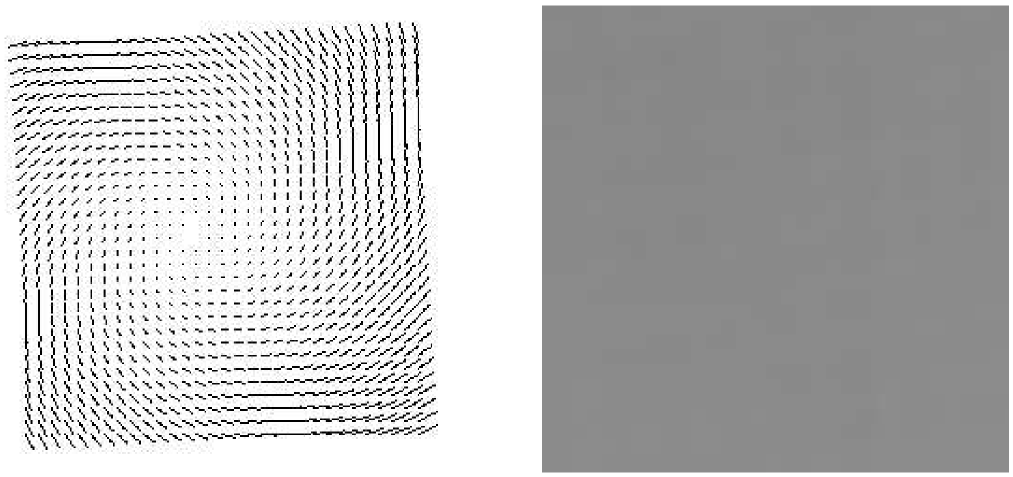
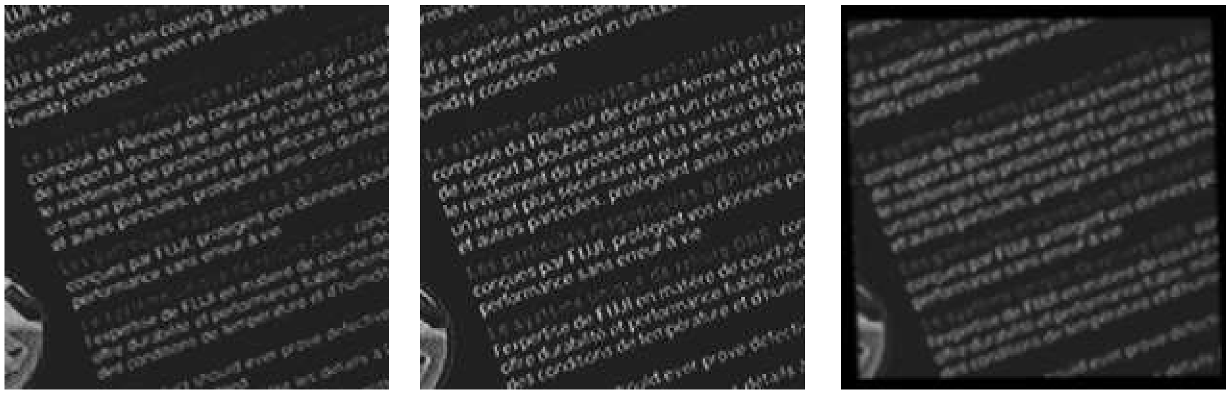
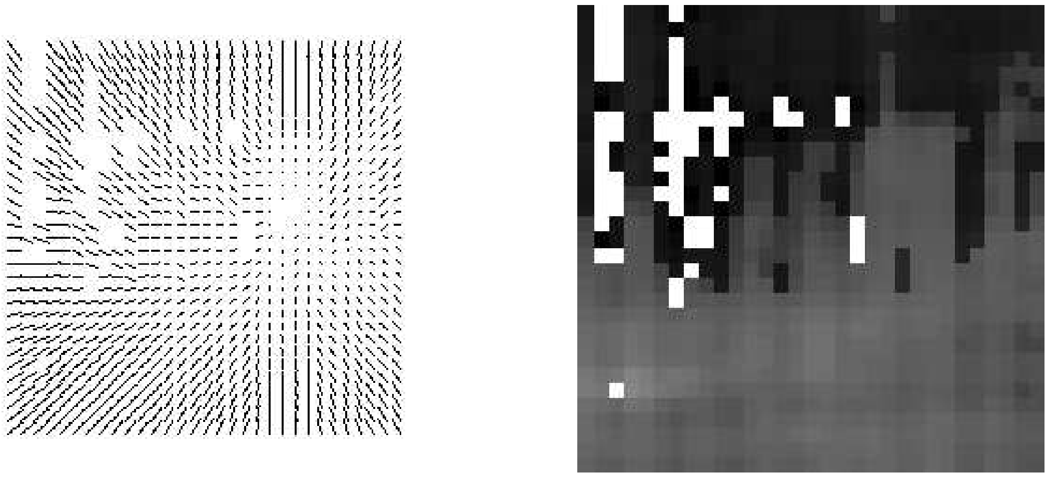

(a) Flow Field(31 x 31 pixel window) (b) Radius Map(31 x 31 pixel window)

Figure 3.26: Experiment 7: Frontoparallel Box (Flow Field and Radius Map)



(a) Patch of second image (b) Corresponding patch of first image (c) Second regenerated using all recovered parameters

Figure 3.27: Experiment 7: Slanted Box (Patches)



(a) Flow Field(31 x 31 pixel window) (b) Radius Map(31 x 31 pixel window) (White region represents points of no convergence)

Figure 3.28: Experiment 7: Slanted Box (Flow Field and Radius Map)

3.3.2.6 Experiment 8: Slanted Stretch of Cloth

In the next experiment two photographs were taken of a textured piece of cloth which was slanted so that the left bottom of the cloth was closest to the camera. In the first image the points closest to the camera were about to get blurred. The camera was moved slightly towards the cloth, so that in the second image the bottom left hand side of the cloth was blurred although the top right hand side was still within depth of field and remained fairly sharp. The images are shown in Figures 3.29 and 3.30.

The recovered degree of blur difference (subsampled every twentieth pixel) is displayed in Figure 3.31. One can observe the gradual increase in the radius of blur while moving from bottom left to the top right of the image.

The recovered flowfield (subsampled) is shown in Figure 3.32.

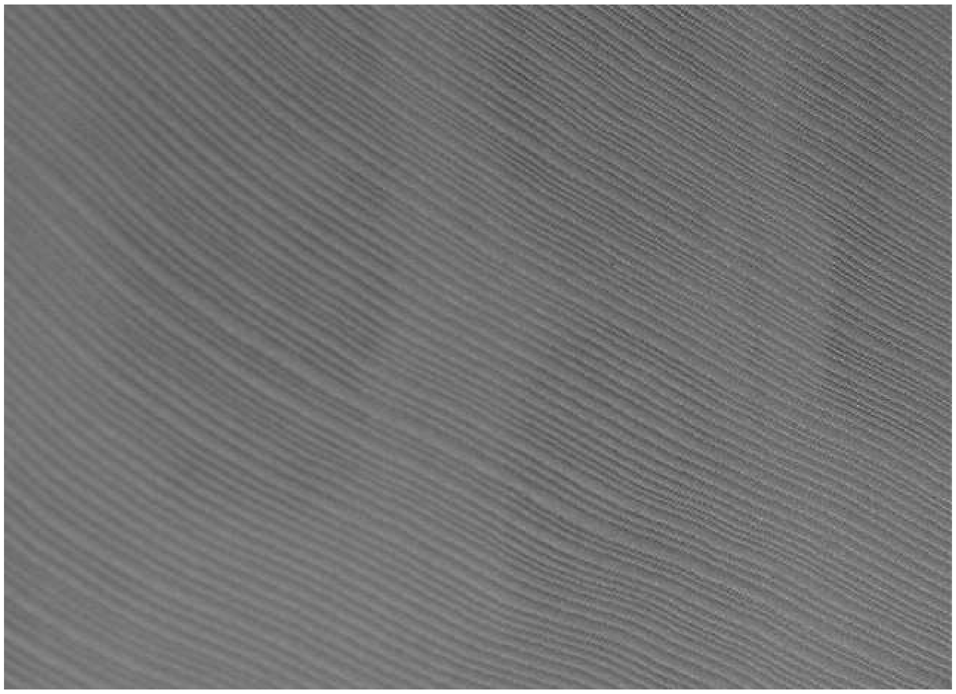


Figure 3.29: Experiment 8: Slanted Cloth (Image 1)

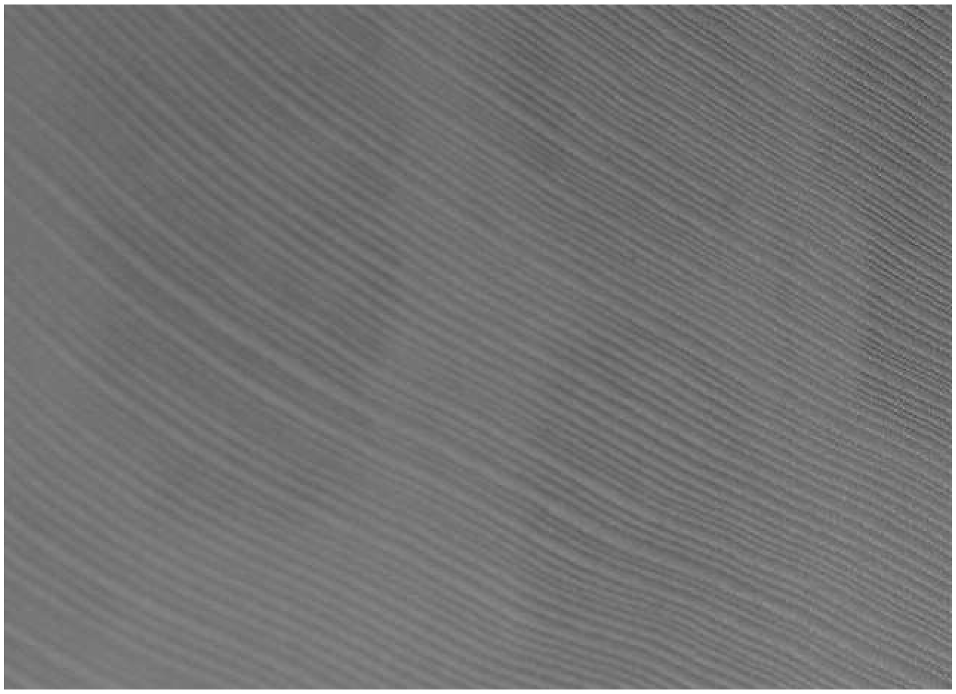


Figure 3.30: Experiment 8: Slanted Cloth (Image 2)

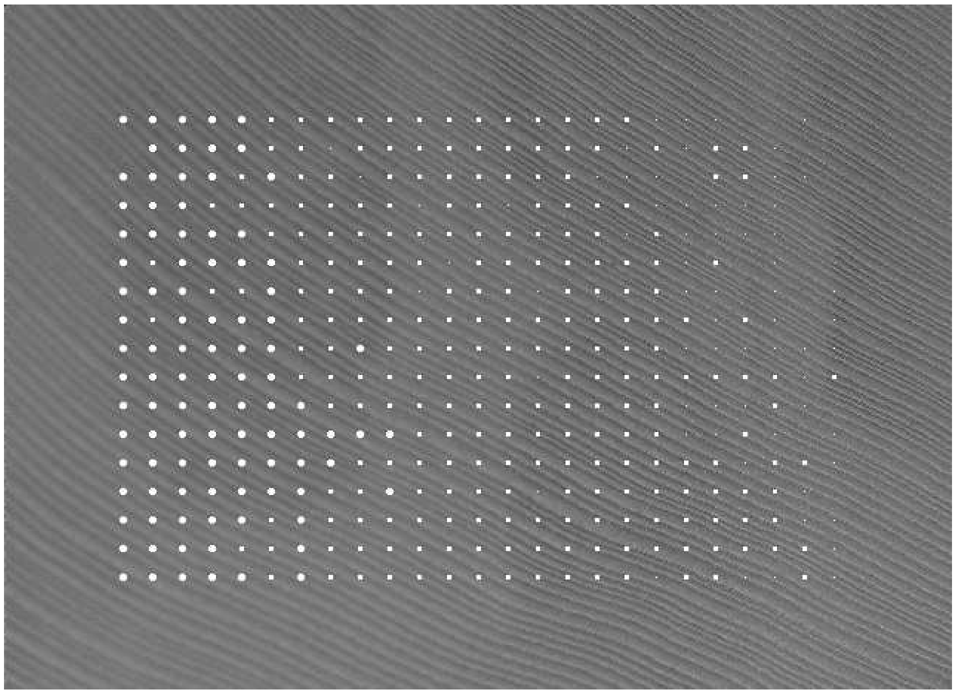


Figure 3.31: Experiment 8: Slanted Cloth (Radius Map)

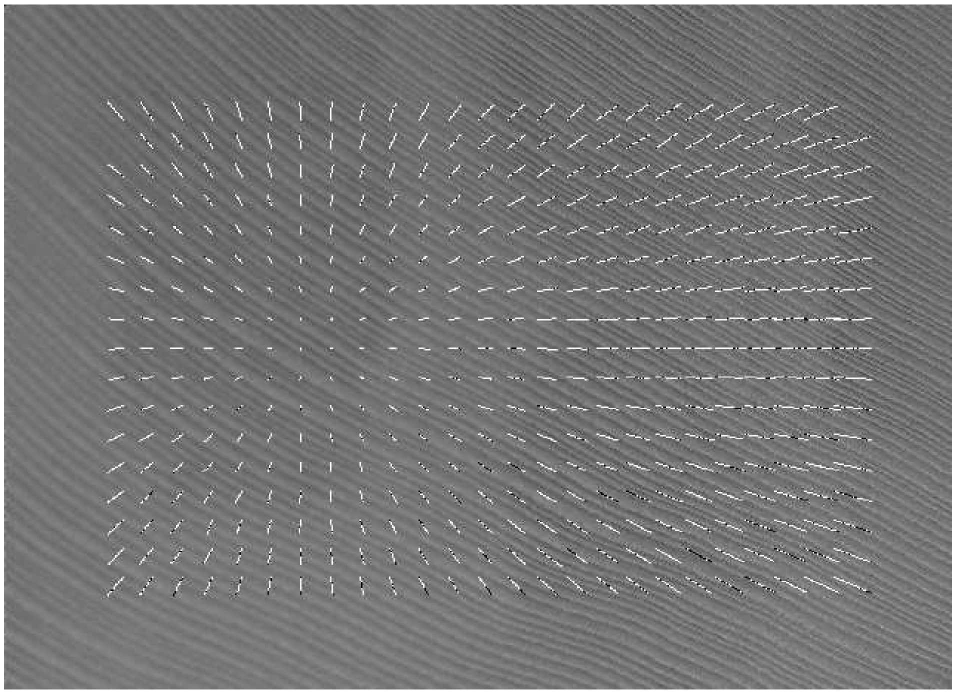


Figure 3.32: Experiment 8: Slanted Cloth (Flow Field)

3.3.2.7 Experiment 9: Cylindrical Object

In this next experiment two photographs were taken of a cylindrical box as shown in Figures 3.33 and 3.34. The recovered degree of blur is displayed in Figure 3.35. It can be seen that at depth discontinuities (left and right edges of the cylinder) the recovered degree of blur is slightly larger. The average radius of blur is 2.76 and the standard deviation is 0.50. The flowfield is correctly recovered as shown in Figure 3.36. No values were recoverable in areas of the cylinder with low texture.

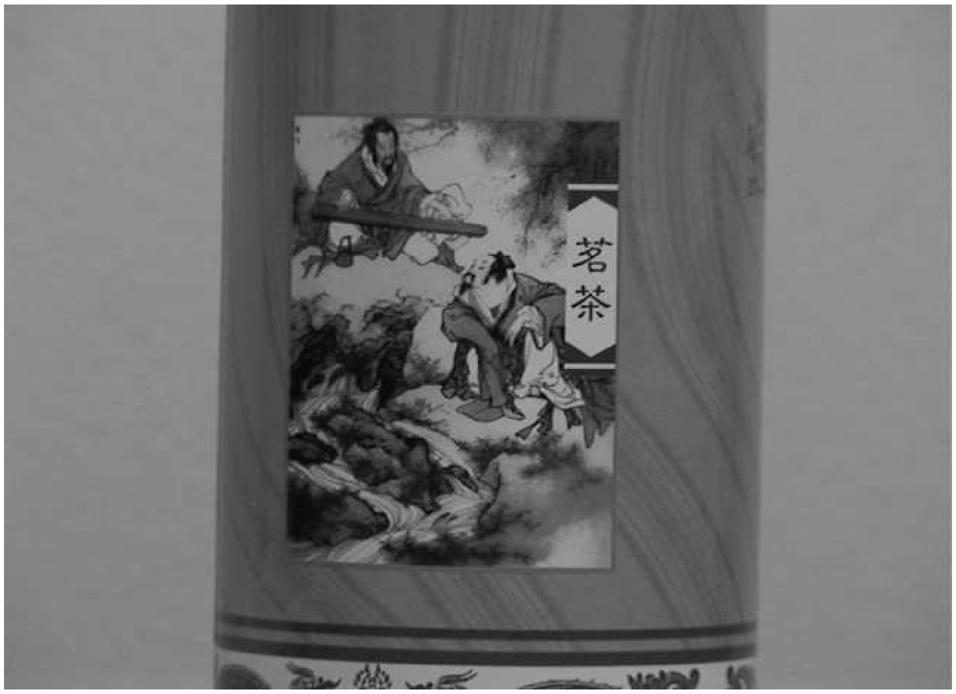


Figure 3.33: Experiment 9: Cylindrical Box (Image 1)

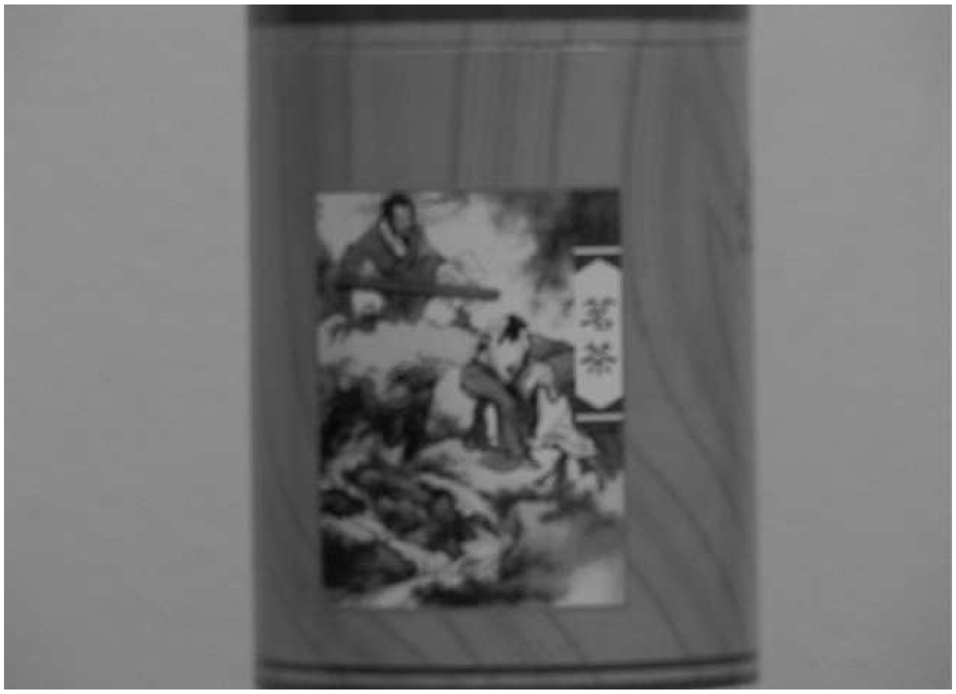


Figure 3.34: Experiment 9: Cylindrical Box (Image 2)

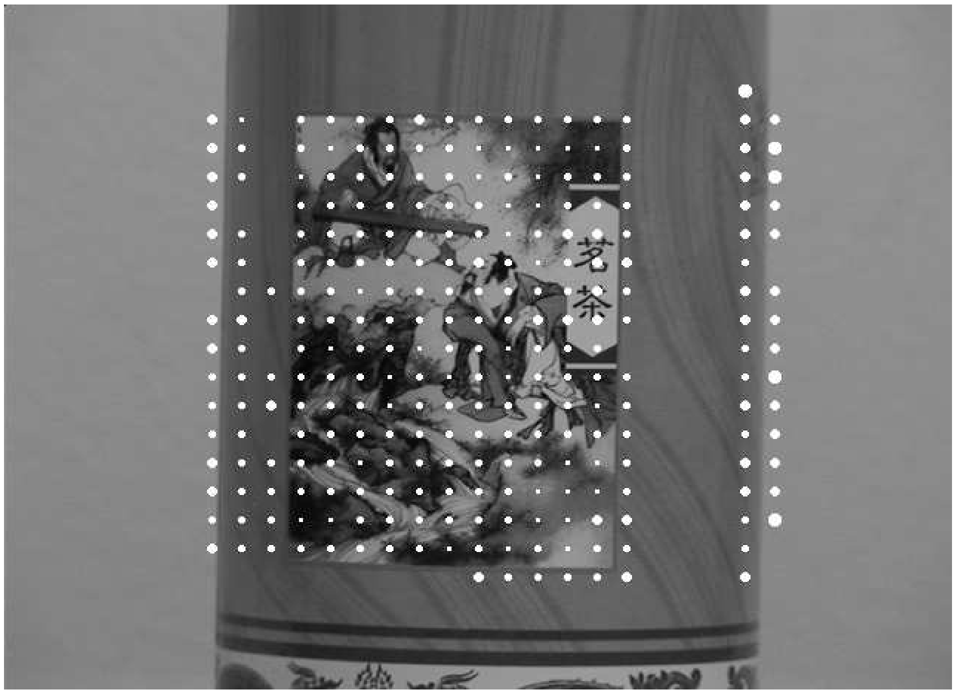


Figure 3.35: Experiment 9: Cylindrical Box (Radius Map)

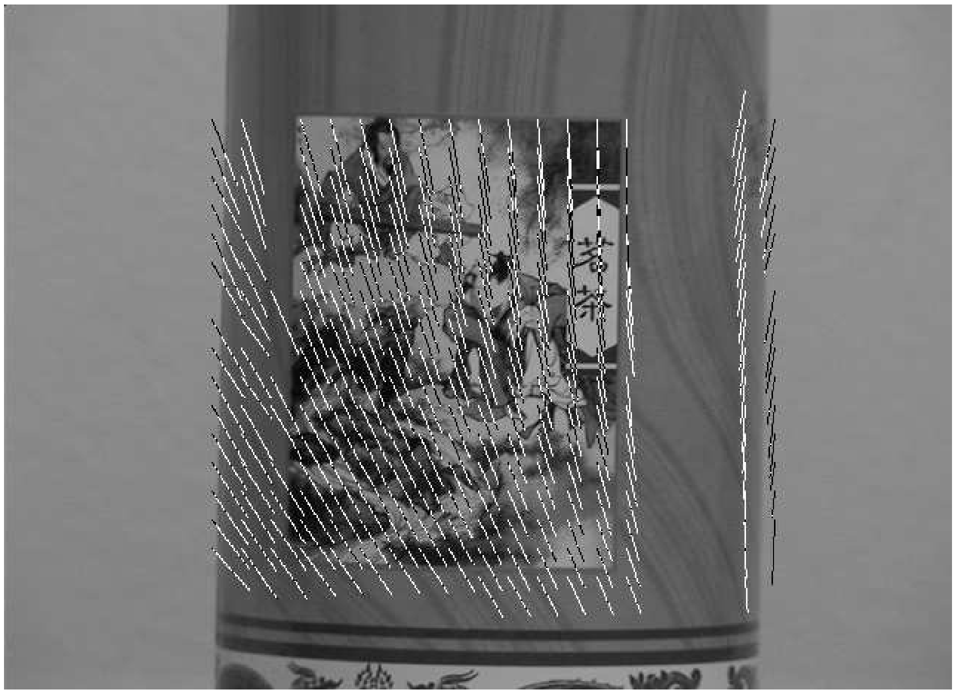


Figure 3.36: Experiment 9: Cylindrical Box (Flow Field)

3.4 Additional Tests and Discussion

We conducted additional experiments to study the stability of the blur estimates with respect to noise. To achieve this, a series of experiments was performed where the second image was deformed with increasing levels of affine deformation. In each experiment both images were subjected to increasing levels of Gaussian noise. In all the experiments the radius of blur in the second image was set to 3 pixels. The percentage error of the computed radius is plotted in Figure 3.4.

An artificially generated image of a sine pattern was employed. In Experiment 1 (represented in Figure 3.4 by a dotted line) the second image was rotated by 15^o, scaled by a factor of 1.2 and translated by [1,0] pixels. In Experiment 2 (represented by a dot-dash line) the second image was rotated by 20^o, scaled by a factor of 1.4 and translated by [2,2] pixels. In the third experiment (represented by a line and circles), the image was rotated 25^o, scaled by a factor of 1.5 and translated [3,3] pixels.

In all the experiments zero-mean Gaussian noise was added to each pixel of both images. The standard deviation of the Gaussian was varied from 0 to 225 in steps of 25.

An error of 100% in the figure implies that convergence did not take place. It was seen that the performance deteriorated gradually as the level of additive noise was increased and as the degree of deformation of the second image was increased. Thus we conclude that the computation is robust to noise.

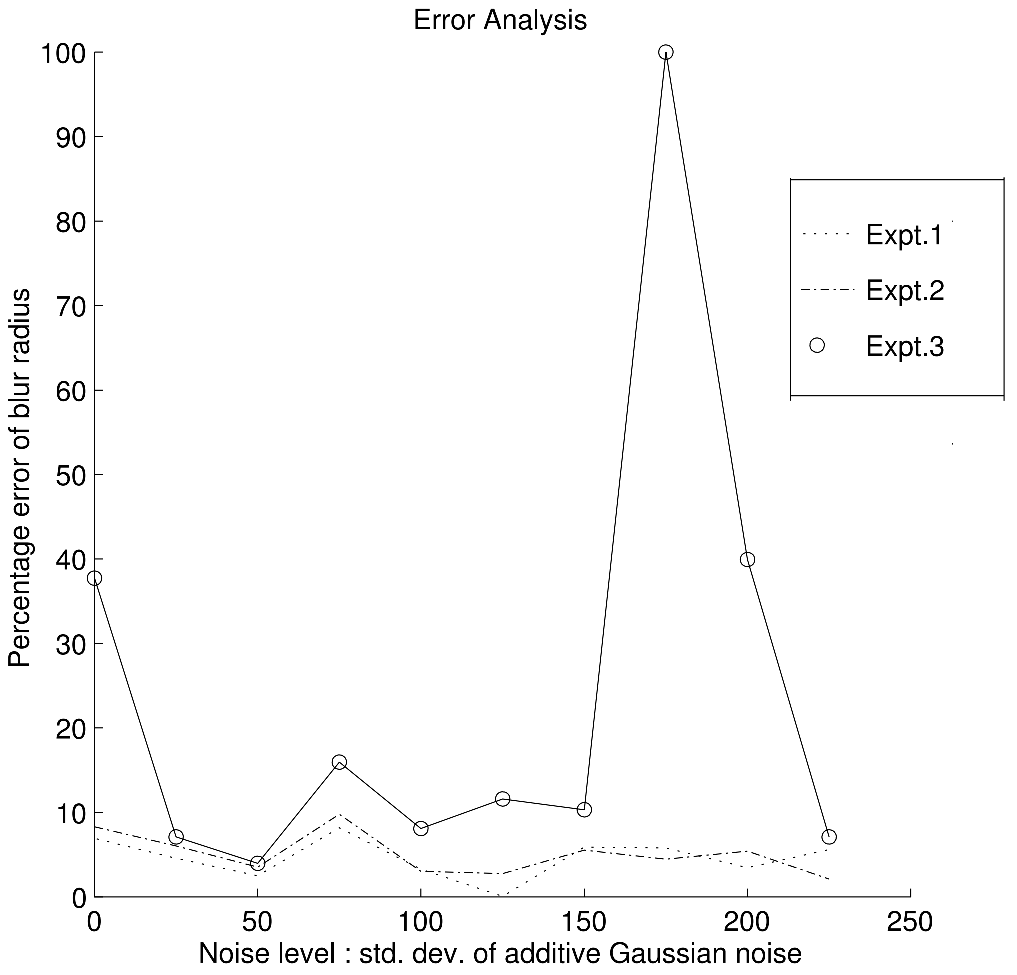


Figure 3.37: Percentage of Error in Blur Radius versus Gaussian Noise Level

This chapter introduced a novel method to measure affine motion and the defocus blur simultaneously. We have experimentally demonstrated the validity of our model using real image pairs. In the next chapter we use the recovered blur to get initial estimates of depth in an iterative computation of shape from defocused-motion.

Recent work [30] handles the displacement of image points due to blur using a telecentric lens. Our method obviates the need for an additional lens by including the deformation in a more comprehensive computational model. In addition, we have shown how the method can be used to segment an image into regions which have blurred, sharpened or remained focally unperturbed during the affine motion.

Our method's primary drawback is that it requires the existence of fairly large planar patches.

CHAPTER 4
COMPUTATION OF INITIAL ESTIMATE OF DEPTH

In this chapter we first derive the theory which relates the radius of blur, camera parameters and depth. We then show how the relationships described above are used as input to an Extended Kalman Filter in order to obtain the relative depth at every point. The theory assumes that the camera motion has no out-of-plane rotational component. However, we were not able to guarantee that in our experiments because we did not have the required precision hardware. In all our experiments, camera motion was performed manually. We show that we are still able to achieve reliable results, thus demonstrating the stability of our formulation.

The rest of this chapter is organized as follows. Section 4.1 develops a novel theory. Section 4.2 proposes a solution method. Section 4.3 demonstrates the validity of the theory by showing the results of experiments conducted on real images. Finally Section 4.4 concludes the chapter with a discussion.

The algorithm is summarized below.

1. With the camera parameters (focal length f, aperture size D and zoom) being fixed, obtain a set of 3 or more images with the camera positioned as shown in Figure 1.

2. Compute blur and image flow information between the first and second images and again between the first and third images using the method elaborated in Chapter 3.

3. For the pair consisting of the first and second images, identify whether the camera motion was towards the object or away from it by computing the average determinant of the matrix of affine transforms and checking whether this average is greater or less than 1.0.

4. Repeat this step for the pair consisting of the first and third images and identify the direction of camera motion between them.

5. For each point p_i in the first image consult section 4.1.3 to select two equations (one equation per image pair) which relate the two radii of blur/sharpening obtained with the unknowns C, E_1, E_2 and Z_i. Here Z_i is the depth at point p_i with respect to the first image. Note that we replace E in section 4.1.3 with E_1 if referring to the image pair consisting of the first and second image and with E_2 if referring to the image pair consisting of the first and third images.

6. If there are N points we get a system of $2N$ equations. This over-determined system of equations can be used to obtain relative depths Z_i's at all the points under consideration. Several non-linear techniques can be employed to achieve a solution. We select the Extended Kalman filter in which the $2N$ equations are input as the measurement equations.

4.1 Theoretical Formulation

We now proceed to derive the relationship between the radius of blur, camera parameters and depth. Consider a camera-centered coordinate system. Let P_a be the coordinate of an object point in the first frame. Now let either the object or the camera be moved. Let P_b be the position of the same object in the next frame. We consider two situations. In the first situation the object is blurred when it is closer as in Figure 4.1 and in the second case the object is sharper when it is closer as in Figure 4.2. We further subdivide these two cases to distinguish between the cases when the camera is moving towards the object and when the camera is moving away from the object. Hence there are four distinct situations. In the following two sections we discuss these four situations and derive four corresponding equations relating the relative depth at a point with the radius of blur, camera translation and fixed camera parameters. In each case we are interested in deriving the depth of the object with respect to the first image.

We first note that since the camera focal length is fixed in all the images, the thin lens law as applied to positions P_a and P_b yields the equations:

$$\frac{1}{f} = \frac{1}{U_a} + \frac{1}{V_a} \tag{4.1}$$

$$\frac{1}{f} = \frac{1}{U_b} + \frac{1}{V_b} \tag{4.2}$$

	U_a, U_b	: distance from the object to camera lens
where	V_a, V_b	: corresponding distance from focused image to camera lens, and
	f	: focal length of camera.

4.1.1 Object Blurred When Closer To Camera

Let us first consider the case where an object point is sharp at position P_a and blurred at position P_b as in Figure 4.1.

We see that blurring is caused by the image plane being in between the lens and the position of perfect focus. By observing similar triangles, we can derive the relationship between the camera aperture and the radius of blur as

$$\frac{D}{V_b} = \frac{2R}{V_b - S} \tag{4.3}$$

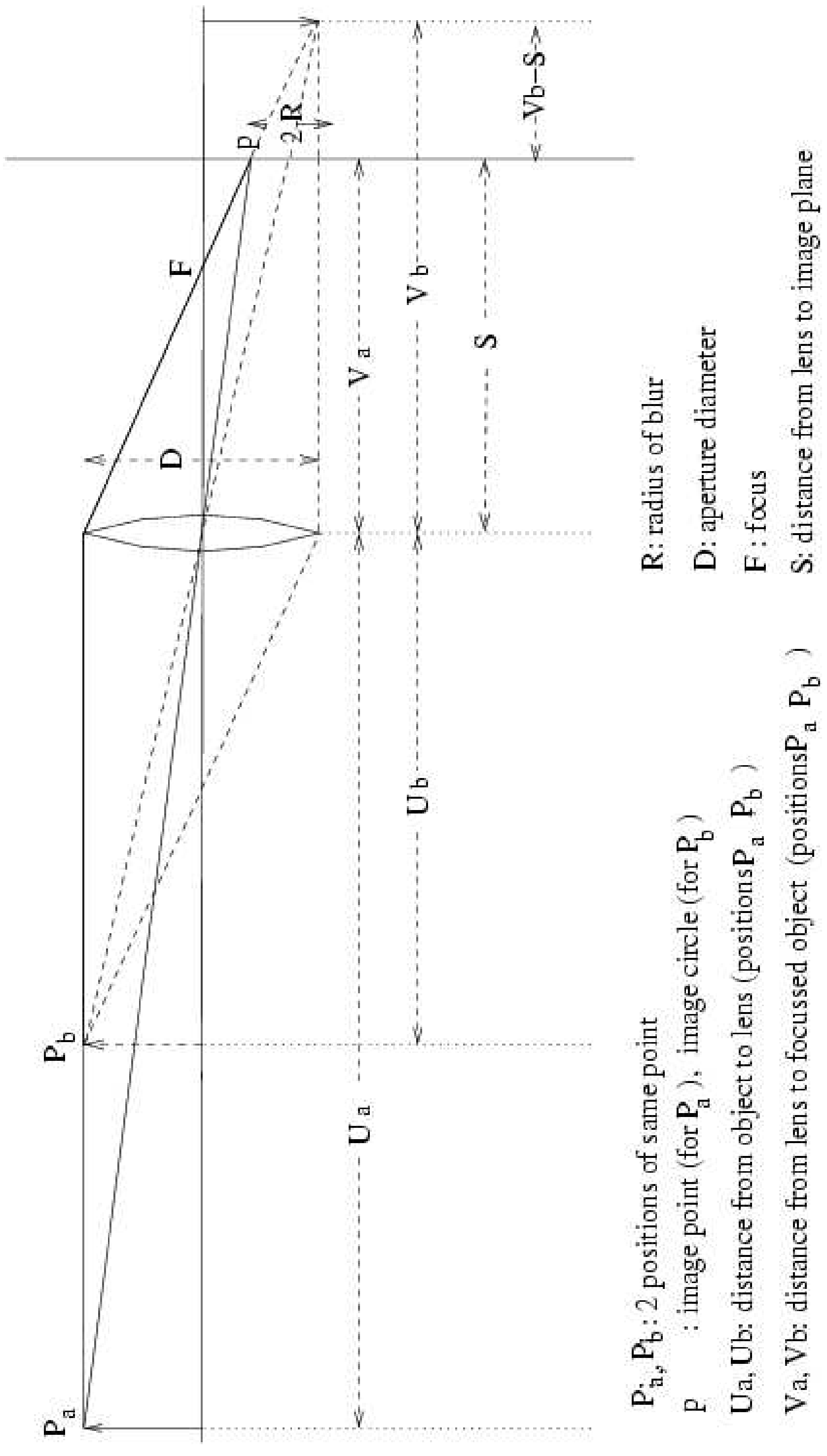
where

S	:	the distance between lens and image plane
D	:	the diameter of the aperture, and
R	:	the radius of the blur circle.

V_b depends upon U_b and hence the Equation (4.3) relates the radius of blur at a point with fixed camera parameters and the depth of that point.

4.1.1.1 Camera Motion Towards Object

Let us first assume that the camera has moved towards the object. Hence P_a is the position of the object with respect to the first image and U_a is the depth that we are interested in studying. Let the component of translation along the Z axis be $T_{z(a,b)}$. We observe that the image point has been *blurred* as a result of the translation. Thus, U_a, the depth of the object



P_a
P_b
P
2 R
D
F
V a
V b
S
U b
U a
Vb−S
P_a, P_b: 2 positions of same point
P : image point (for P_a), image circle (for P_b)
Ua, Ub: distance from object to lens (positions P_a P_b)
Va, Vb: distance from lens to focussed object (positions P_a P_b)
R: radius of blur
D: aperture diameter
F : focus
S: distance from lens to image plane

Figure 4.1: Camera Model - Object Blurred When Closer To Camera

from the camera with respect to the first image is given by

$$U_b = U_a + T_{z(a,b)} \tag{4.4}$$

This step assumes no out-of-plane rotation. In our experiments we moved the camera by hand and hence there was unavoidably some out-of-plane rotation. Yet we were able to attain satisfactory results, thus proving the robustness of our formulation. We note that in-plane rotations and image magnification/contraction can be accommodated in the computation of affine parameters at the first step.

We first eliminate V_b from Equations (4.2), (4.3) to yield

$$\frac{fU_b}{U_b - f} = \frac{DS}{D - 2R} \tag{4.5}$$

and this can be rewritten as

$$U_b = \frac{fDS}{DS - fD + 2fR} \tag{4.6}$$

Finally, using Equations (4.4) and (4.6) we express the depth U_a as

$$\begin{aligned} U_a &= U_b - T_z \\ &= \frac{fSD}{(DS - fD) + 2fR} - T_{z(a,b)} \end{aligned} \tag{4.7}$$

Equation (4.7) is important because it relates (a) real world depth to (b) a measured variable that is dependent upon depth and (c) intrinsic and extrinsic camera parameters. If the intrinsic camera parameters (f, S and D) and camera translation ($T_{z(a,b)}$) are known, the computation of absolute depth is trivial. Alternately one could estimate all these parameters along with depth U_a in a complicated formulation. Here we show that is unnecessary if all we need to obtain is relative depth. We achieve this in the following way. Equation (4.7) is rewritten as

$$Z = \frac{1}{(C + R)} - E \tag{4.8}$$

where

$$\begin{aligned} Z &= \left(\frac{2}{SD}\right) U_a \\ E &= \left(\frac{2}{SD}\right) T_{z(a,b)} \\ C &= \frac{D}{2f}(S - f) \end{aligned}$$

Here Z is not the true depth but a scaled depth, E is a scaled version of the component of translation along the Z axis and C depends solely on the fixed intrinsic camera parameters D, f and S.

The advantage of Equation (4.8) over Equation (4.7) is that there are fewer unknowns and they are related in a less tightly coupled fashion, thus facilitating an easier solution process.

Equation (4.8) is now rewritten to express the radius of blur R as

$$R = \frac{1}{(Z + E)} - C \tag{4.9}$$

A significant advantage of this equation is that it is linear in C and hence the solution of this unknown is often obtained in the first iteration of any non-linear solution method.

Equation (4.9) expresses the relative radius of blur R in terms on the unknowns in the case when the camera is moving towards the object and the closer object is more blurred.

4.1.1.2 Camera Motion Away From Object

Let us now consider the situation where the camera is moving away from the object. In this case the image point has been *sharpened* as a result of camera translation. The depth of the object from the camera with respect to the first image is U_b and Equation (4.6) represents the relationship between absolute depth and the unknowns f, S and D. In this case, there is no dependence upon T_z, the component of translation along the Z axis. We define R as the *radius of sharpening* of the second image with respect to the first. We compute it by finding the radius of blur of the first image with respect to the second. This time we rewrite Equation (4.6) to express the relative depth in terms of the R and the unknowns Z and C as

$$Z = \frac{1}{C + R} \tag{4.10}$$

where Z is now related to depth U_b as

$$Z = \left(\frac{2}{SD}\right) U_b$$

We rewrite Equation (4.10) to express the radius of sharpening as

$$R = \frac{1}{Z} - C \tag{4.11}$$

We will later show how Equations (4.9) and (4.11) can be used as measurement equations in an Extended Kalman filter to yield the relative depth Z.

4.1.2 Object Sharper When Closer To Camera

Let us now consider the situation when the object yields a sharper image when it is closer to the camera as in Figure 4.2. From the diagram it can be observed that the position of perfect focus is between the lens and the image plane.

Once again, by observing similar triangles we see that

$$\frac{D}{V_a} = \frac{2R}{S - V_a} \tag{4.12}$$

where D, V_a and R are defined as before. We derive two separate equations relating relative blur R with relative depth and unknowns dependent upon camera intrinsic and extrinsic parameters. One equation corresponds to the situation when the camera moves towards the object and the other equation corresponds to the situation when the camera moves away from the object.

4.1.2.1 Camera Motion Towards Object

Let us first assume that the camera is moving towards the object. Hence the depth of the object with respect to the first image is U_a and the object point has been *sharpened* as a result of camera translation. By eliminating V_a from Equations (4.1) and (4.12) we get the relation between absolute depth, radius of sharpening R and intrinsic camera parameters as

$$\frac{fU_a}{U_a - f} = \frac{DS}{2R + D} \tag{4.13}$$

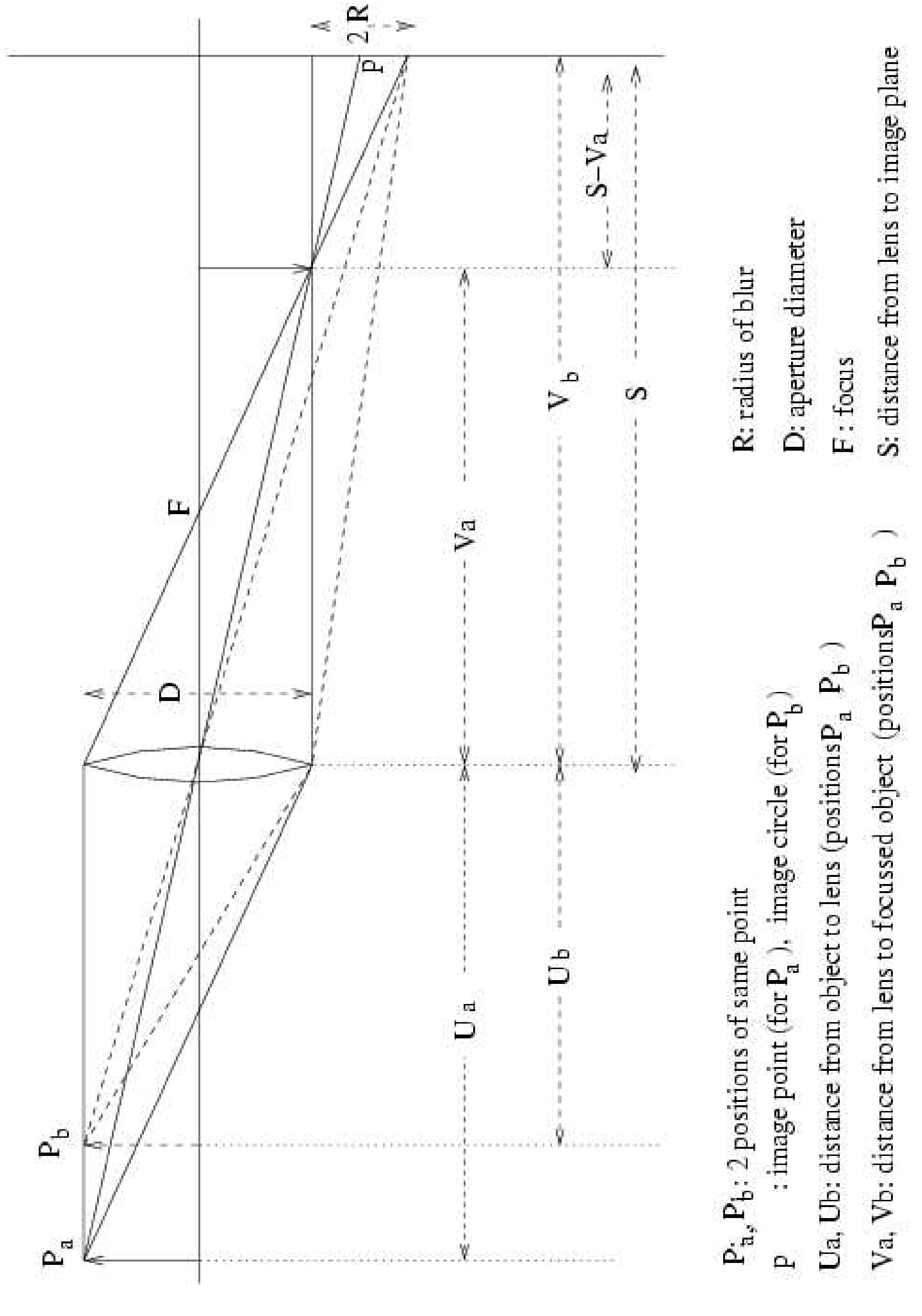

P_a
P_b
F
D
P
2R
U a
U b
Va
V b
S
S−Va
P_a, P_b: 2 positions of same point
P : image point (for P_a), image circle (for P_b)
Ua, Ub: distance from object to lens (positions P_a P_b)
Va, Vb: distance from lens to focussed object (positions P_a P_b)
R: radius of blur
D: aperture diameter
F : focus
S: distance from lens to image plane

Figure 4.2: Camera Model - Object Sharper When Closer To Camera

We rewrite Equation (4.13) to get an equation expressing absolute depth U_a as

$$U_a = \frac{DSf}{DS - fD - 2fR} \tag{4.14}$$

Once again, we express the complex equation above in terms of simpler unknowns Z and C as below:

$$Z = \frac{1}{C - R} \tag{4.15}$$

where the relative depth Z is related to the absolute depth U_a as

$$Z = \left(\frac{2}{SD}\right) U_a$$

and C is as defined earlier. Rewriting Equation (4.15), the radius of sharpening R is expressed as

$$R = C - \frac{1}{Z} \tag{4.16}$$

4.1.2.2 Camera Motion Away From Object

Finally, we derive an equation for R when for the situation where the camera is moving away from the object. In this case the object point has been *blurred* as a result of camera translation. The required depth (with respect to the first image) is U_b. U_b can be expressed in terms of camera translation $T_{z(b,a)}$ along the Z axis as

$$U_a = U_b + T_{z(b,a)} \tag{4.17}$$

From Equations (4.14) and (4.17) we express absolute depth U_b as

$$U_b \;=\; U_a + T_{z(b,a)}$$

$$= \frac{DSf}{DS - fD - 2fR} + T_{z(b,a)} \tag{4.18}$$

As before, we rewrite Equation (4.18) in terms of the simpler unknowns Z, C and E as

$$Z = \frac{1}{(C - R)} - E \tag{4.19}$$

where the relative depth Z is defined as

$$Z = \left(\frac{2}{DS}\right) U_b$$

and C and E are as defined earlier. Equation (4.19) can be rewritten to express the radius of blur as

$$R = C - \frac{1}{(Z + E)} \tag{4.20}$$

4.1.3 Summary: The Blur-Depth Relationship

Thus we have derived four different equations (Equations (4.9), (4.11), (4.16) and (4.20)) to express R in terms of unknowns C, E and Z. We summarize the results as follows.

1. Camera Motion Towards Object:

$$R = \frac{1}{(Z + E)} - C \quad \textit{image point is blurred} \tag{4.21}$$

$$R = C - \frac{1}{Z} \quad \textit{image point is sharpened} \tag{4.22}$$

2. Camera Motion Away From Object:

$$R = \frac{1}{Z} - C \quad \textit{image point is sharpened} \tag{4.23}$$

$$R = C - \frac{1}{(Z + E)} \quad \textit{image point is blurred} \tag{4.24}$$

where C, E and Z are functions of D (camera aperture diameter), f (focal length), S (distance between lens and image plane), T_z (translation along the optical axis) and U (depth of object point) as defined below:

$$\begin{aligned} C &= \frac{D}{2f}(S-f) \\ E &= \left(\frac{2}{SD}\right)T_z \\ Z &= \left(\frac{2}{DS}\right)U \end{aligned}$$

We now show how these results can be used in a further formulation in order to facilitate a solution process.

4.1.4 *Sequence of Three Images*

Let p_i be an image point and let R_i be the radius of blur/sharpening for point p_i between frames one and two. Let Z_i be the depth of p_i with respect to the first frame.

We first compute the affine transformation and defocus blur between frames one and two as elaborated in Chapter 3. As shown earlier, at each point we can automatically recognize whether blur or sharpening has taken place. We can also automatically deduce whether between frames one and two the camera has moved towards or away from the object. This is done by examining the matrix $\mathbf{A}$ of the affine transform described in Equation (3.1). $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ encodes the enlargement/contraction, rotation and shearing component of this affine transform. In particular, the determinant of $\mathbf{A}$ reveals whether the image has been diminished or enlarged. If the camera moves towards the object, the image is magnified and the determinant of $\mathbf{A}$ is greater than one. If the camera moves away from the object, the corresponding image is diminished and the determinant of $\mathbf{A}$ is less than one. By taking the average value of the determinants of the affine transforms at all the points in the first image, a computer program can recognize the direction of motion.

Once we know the direction of camera motion and whether blurring or sharpening has taken place we can identify the equation from section (4.1.3) which represents the relationship between R and unknowns C, E and Z_i.

We note that for any pair of images the equations points in the first image either satisfy Equations (4.21, 4.22) or Equations (4.23, 4.24) depending upon the direction of camera motion.

The equation selected has two or three unknowns (C, E, Z_i), depending upon whether or not E is included in the equation.

On studying the unknowns more closely, we note that C remains fixed if the camera intrinsic parameters of (1) D, the diameter of aperture, (2) f, the focal length, and (3) S, the distance between lens and sensor do not change. Moreover, the value of E depends upon the component of translation along the Z axis and hence is fixed for every point in a specific image pair. However, every additional point p_i in the image introduces one more unknown depth z_i.

Hence the equations corresponding to all N points in a single image pair always has $(N+2)$ or $(N+3)$ unknowns and is under-determined. In order to solve for the unknowns we need a sequence of three images. This would produce two image pairs, the first image pair consists of frame 1 and frame 2 and the second image pair consists of frame 1 and frame 3. We note that both pairs have a common first image.

A sample configuration of the three images is shown in Figure 1. It can be easily seen that the sequence could have been reversed, i.e., the images could have been captured by moving the camera away from the object instead of towards it.

With three consecutive images, two values of radii are computed for every point p_i. One is the radius of blur/sharpening between frame one and two and the other is the radius of blur/sharpening between frame one and three. Hence each point generates two equations. Since E is unique for every pair of images, we now have the unknowns E_1 and E_2, which depend upon the appropriate translation values as follows:

$$\begin{aligned} E_1 &= \left(\frac{2}{SD}\right) T_{z(1,2)} \\ E_2 &= \left(\frac{2}{SD}\right) T_{z(1,3)} \end{aligned}$$

In order to solve for the unknowns C, E_1, E_2 and Z_i we need at least three points and a sequence of three images. By computing the radii of blur at all the points in the image an overdetermined set of equations is generated.

4.2 Solution Method

The overdetermined set of equations described above can be solved using a variety of mathematical techniques. We employ the Extended Kalman Filter since it is optimal in the sense that it factors in the uncertainties associated with the model as well as the uncertainties associated with the measurements in such a way that the solution ensures that the error is minimized statistically.

We implemented the Kalman Filter in a manner similar to [48] since it affords an efficient way to handle a large number of points.

For the sake of completeness the Extended Kalman Filter is defined in section 4.2.1 and its variation as derived in [48] is elaborated in section 4.2.2 below.

4.2.1 Extended Kalman Filter

Let the parameters in a physical system be modeled as

$$\vec{w} = h(\vec{x})$$

where $\vec{w}$ represents a vector of *measurements* and the vector $\vec{x}$ represents the *state* of the system at any given instant. In our system, $\vec{w}$ consists of the radii of blur and $\vec{x}$ consists of the depths z_i of object points and the values C, E_1 and E_2.

However, this model is not complete because of the following factors [25]:

- *A mathematical model cannot completely define a system*

  In our case, the effect on blurring due to factors such as irregularities in the lens, imperfections in the sensor, etc. have not been included in the model.

- *The measurements are not precise*

  The values of radii computed earlier are imprecise because of several factors which include discretization and the establishment of convergence criteria to ensure computation in a finite amount of time.

Hence Kalman introduced the more precise *measurement equation*

$$\vec{w} = h(\vec{x}) + \vec{n}$$

where $\vec{n}$ represents zero-mean measurement error with covariance $\mathbf{R}$.

Kalman then derived an iterative technique to refine the estimate of the state in an iterative manner.

If $\vec{x}_-$ and $\mathbf{P}_-$ be the *a priori* estimates of the state vector and its covariance, then the *a posteriori* estimates of the state vector $\vec{x}_+$ and its covariance $\mathbf{P}_+$ are given by

$$\begin{aligned} \vec{x}_+ &= \vec{x}_- + \mathbf{K}\,(\vec{w} - h(\vec{x}_-)) \\ \mathbf{P}_+ &= (\mathbf{I} - \mathbf{KH})\,\mathbf{P}_- \end{aligned}$$

where $\mathbf{K}$ is the Kalman gain matrix, defined as

$$\mathbf{K} = \mathbf{P}_-\mathbf{H}^{\mathbf{T}}(\mathbf{H}\mathbf{P}_-\mathbf{H}^{\mathbf{T}} + \mathbf{R})^{-\mathbf{1}} \tag{4.25}$$

and $\mathbf{H}$ is the Jacobian matrix of the measurement equations. That is,

$$\mathbf{H} = \left. \frac{\partial h(\vec{x})}{\partial \vec{x}} \right|_{\vec{x}=\vec{x}_-} \tag{4.26}$$

4.2.2 Xiong and Shaefer

Traditional EKF techniques deal with $N \times N$ matrices where N is proportional to the number of features tracked. In the case of dense optical flow measurements in a typical image of size 640×480 the value of N would be 307200. This would make a solution based on the traditional Kalman filter infeasible in both time and space.

In [48] the Kalman filter equations were enhanced to take advantage of the special properties inherent in the uncertainty matrices $\mathbf{P}$ and $\mathbf{R}$ and the Jacobian $\mathbf{H}$ of the measurement equations to develop a robust and efficient solution process where the computation is $O(N)$.

In [48] the measurement equations were the instantaneous motion equations, viz. Equations (2.2). We have applied the same efficiency steps introduced [48] but replaced the measurement equation/s selected from (4.21), (4.22), (4.23) or (4.24) as shown in section (4.1.3).

Our measurements $\vec{w}$ are the blur/sharpening values and our state vector $\vec{x}$ is of size $N+3$, where N is the number of image points. For the sake of convenience, we decompose $\vec{x}$ into 2 vectors, viz. $\vec{M}$ and $\vec{Z}$ where

$$\vec{M} = [C,\ E_1,\ E_2]^T$$

and

$$\vec{Z} = [z_1,\ z_2,\ \dots\ ,\ z_N]^T$$

That is, $\vec{Z}$ consists of the depth value at every point under consideration and $\vec{M}$ consists of all the other unknowns.

The uncertainty of measurement is represented by the $2N \times 2N$ matrix $\mathbf{R}$

$$\mathbf{R} = \begin{pmatrix} u_1 & & & \\ & u_2 & & \\ & & \ddots & \\ & & & u_{2N} \end{pmatrix} \tag{4.27}$$

where u_i is a measure of the degree of confidence in the value of the radius r_i computed in the corresponding equation.

In addition, the Jacobian matrix $\mathbf{H}$ of the measurement equations defined as

$$\mathbf{H} = \begin{pmatrix} \frac{\partial r_1}{\partial \vec{\mathbf{M}}} & | & \frac{\partial r_1}{\partial z_1} & & & \\ \frac{\partial r_2}{\partial \vec{\mathbf{M}}} & | & & \frac{\partial r_2}{\partial z_2} & & \\ \vdots & | & & & \ddots & \\ \frac{\partial r_N}{\partial \vec{\mathbf{M}}} & | & & & & \frac{\partial r_N}{\partial z_N} \end{pmatrix} \tag{4.28}$$

is of the form $(\mathbf{A} \quad \mathbf{S})$ where $\mathbf{A}$ is a $2N \times 3$ matrix and $\mathbf{S}$ is an $N \times N$ block diagonal matrix with each block a 2×1 matrix.

Moreover, the correlated uncertainty of the depth values caused by a single uncertain motion is an $N \times N$ matrix with rank of only k, (k a constant, typically 3) corresponding to the 3 additional unknown parameters.

Hence the covariance matrix $\mathbf{P}$ can be decomposed as follows:

$$\mathbf{P} = \begin{pmatrix} \mathbf{C}_m & \mathbf{C}_p^T \\ \mathbf{C}_p & (\mathbf{C}_s + \mathbf{U}\mathbf{V}^T) \end{pmatrix} \tag{4.29}$$

Here $\mathbf{C}_m$ is a 3×3 matrix which represents the covariance of the parameters $\vec{M} = [C,\ E_1,\ E_2]^T$. $\mathbf{C}_p$ is an $N \times 3$ matrix which represents the correlation between $\vec{M}$ and structure. The covariance of depth parameters is a matrix of rank k and hence can be expressed as $(\mathbf{C}_s + \mathbf{U}\mathbf{V}^T)$ where $\mathbf{C}_s$ is an $N \times N$ diagonal matrix representing the *independent* uncertainty of depth value at each pixel, and $\mathbf{U}$ and $\mathbf{V}$ are N $\times$ k matrices representing the correlated uncertainty of the depth values.

Due to the special properties of the matrix $\mathbf{P}$, its memory requirement is of $O(N)$ instead of $O(N^2)$. We will see later that since $\mathbf{R}$ and $\mathbf{H}$ are special matrices, the format of $\mathbf{P}$ remains the same at every iteration. Hence $\mathbf{P}$ never needs to be represented as an $(N+3) \times (N+3)$ matrix.

The steps in the solution of the Kalman equations given a sequence of three image are listed below. These steps can be iterated through additional frames.

1. *Change Coordinates Corresponding to Parameters in* $\vec{M}$

The parameters in $\vec{M}$ can have varying uncertainty. In order to ensure stability in future calculations we first redefine these parameters to ensure that their uncertainty values are equal.

If $\vec{M}_{orig}$ are the original estimates of $\vec{M}$ and $\mathbf{C}_{m(orig)}$ is the original covariance, we reparameterize the unknowns so that their covariance is $\mathbf{I}$.

Since $\mathbf{C}_{m(orig)}$ is the covariance $\vec{M}_{orig}$, we have

$$(\vec{M}_{orig} - \bar{\vec{M}}_{orig})\ (\vec{M}_{orig} - \bar{\vec{M}}_{orig})^T = \mathbf{C}_{m(orig)}$$

where $\bar{\vec{M}}_{orig}$ represents the mean of $\vec{M}$.

Let the singular value decomposition of the positive symmetric matrix $\mathbf{C}_m$ be

$$\mathbf{C}_{m(orig)} = \mathbf{U}\ \mathbf{W}\ \mathbf{U}^{\mathbf{T}}$$

Then it can be seen that the new variables $\vec{M}_{new}$ given by

$$\vec{M}_{new} = (\mathbf{U}\ \mathbf{W}^{\frac{1}{2}})^{-1}\ \vec{M}_{orig}$$

has variance $\mathbf{I}$.

For the sake of convenience let us define

$$\mathbf{C}_x = (\mathbf{UW})^{\frac{1}{2}}$$

The above change requires the component $\mathbf{A}$ of the matrix $\mathbf{H}$ to be redefined as

$$\mathbf{A}_{new} = \mathbf{A}_{orig}(\mathbf{UW}^{\frac{1}{2}}) = \mathbf{A}_{orig}\mathbf{C}_x$$

2. *Compute* $\mathbf{HP\_H}^T + \mathbf{R}$

Making the assumption that motion is discontinuous, we set $\mathbf{C}_p^T = \mathbf{0}$.

Hence

$$\mathbf{HP\_H}^T + \mathbf{R} = \mathbf{C}_1 + \mathbf{U}_1\mathbf{V}_1^T \tag{4.30}$$

where
$$\begin{array}{lll} \mathbf{C}_1 & : \text{N} \times \text{N block diagonal} & = (\mathbf{SC}_s\mathbf{S}^T + U) \\ \mathbf{U}_1 & : 2\text{ N} \times (3 + \text{k}) & = (\mathbf{AC}_m \quad \mathbf{SU}) \\ \mathbf{V}_1 & : 2\text{ N} \times (3 + \text{k}) & = (\mathbf{A} \quad \mathbf{SV}) \end{array}$$

3. *Compute* $(\mathbf{HP\_H^T} + \mathbf{R})^{-1}$

We apply the Sherman-Morrison-Woodbury formula [13] (page 50) to obtain the inverse of the matrix $(\mathbf{HP\_H}^T + \mathbf{R})$. This formula reduces the problem of inverting the $N \times N$ matrix to one of inverting a $k \times k$ matrix where $k << N$.

Hence

$$(\mathbf{HP\_H}^T + \mathbf{R})^{-1} = \mathbf{C}_2 + \mathbf{U}_2\mathbf{V}_2$$

where

$$\begin{aligned} \mathbf{C}_2 &= \mathbf{C}^{-1} \\ \mathbf{U}_2 &= -\mathbf{C}^{-1}\mathbf{U}(\mathbf{I}_k + \mathbf{V}^T\mathbf{C}^{-1}\mathbf{U})^{-1} \\ \mathbf{V}_2 &= \mathbf{C}^T\mathbf{V} \end{aligned}$$

4. *Compute Kalman Gain*

The Kalman gain can now be expressed as

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_m \\ \\ \mathbf{C}_3 + \mathbf{U}_3\mathbf{V}_3 \end{pmatrix}$$

where

$$\begin{aligned}
\mathbf{K}_m &= \mathbf{C}_m\mathbf{A}^T\mathbf{C}_2 + \mathbf{C}_m\mathbf{A}^T\mathbf{U}_2\mathbf{V}_2^T \\
\mathbf{C}_3 &= \mathbf{C}_s\mathbf{S}^T\mathbf{C}_2 \\
\mathbf{U}_3 &= \begin{pmatrix} \mathbf{U} & \mathbf{C}_s\mathbf{S}^T\mathbf{U}_2 & \mathbf{U} \end{pmatrix} \\
\mathbf{V}_3 &= \begin{pmatrix} \mathbf{C}_2\mathbf{S}\mathbf{V} & \mathbf{V}_2 & \mathbf{V}_2(\mathbf{U}_2^T\mathbf{S}\mathbf{V}) \end{pmatrix}
\end{aligned}$$

5. *Compute the Updated State Covariance Matrix* $\boldsymbol{P}_+$

The updated matrix representing the covariance of the state vector $\mathbf{P}_+$ is of the form

$$\mathbf{P}_+ = \begin{pmatrix} \mathbf{C}_{mp} & \mathbf{C}_{pp}^T \\ \\ \mathbf{C}_{pp} & (\mathbf{C}_4 + \mathbf{U}_4{\mathbf{V}_4}^T) \end{pmatrix} \tag{4.31}$$

where

- The *posteriori* covariance matrix of the vector $\vec{M}$ is a 3×3 matrix C_{mp} given by

$$\mathbf{C}_{mp} = \mathbf{C}_m - \ \mathbf{K}_m\mathbf{A}\mathbf{C}_m$$

- The *posteriori* uncertainty correlation between the $\vec{Z}$ and $\vec{M}$ is an $N \times 3$ matrix $\mathbf{C}_{pp}$ given by

$$\mathbf{C}_{pp} = -(\mathbf{C}_s + \mathbf{U}\mathbf{V}^T)\mathbf{S}^T\mathbf{K}_m^T$$

- The independent uncertainty in structure estimation $\mathbf{C}_4$ is an $N \times N$ diagonal matrix given by

$$\mathbf{C}_4 = \mathbf{C}_s - \mathbf{C}_3 \mathbf{S} \mathbf{C}_s$$

- The correlated uncertainty in structure is represented as the outer product of $\mathbf{U}_4$ and $\mathbf{V}_4$, each of size $l = N \times (3k+3)$, where

$$\begin{aligned} \mathbf{U}_4 &= \begin{pmatrix} \mathbf{U} & -\mathbf{U}_3 & -\mathbf{C}_3\mathbf{S}\mathbf{U} & \mathbf{U}_3\mathbf{V}_3^T\mathbf{S}\mathbf{U} \end{pmatrix} \\ \mathbf{V}_4 &= \begin{pmatrix} \mathbf{V} & -\mathbf{C}_s\mathbf{S}^T\mathbf{V}_3 & \mathbf{V} & \mathbf{V} \end{pmatrix} \end{aligned}$$

It was seen that convergence was speeded up when a weight was added to the diagonal elements of $\mathbf{P}_+$. This weight was set proportional to the difference between two successive values of the corresponding unknown.

6. *Reduce the Dimension of $\boldsymbol{U}_4$ and $\boldsymbol{V}_4$*

Since k can increase linearly after each frame, the above algorithm is converted to O(MN) where M is the number of frames. In order to maintain an $O(N)$ algorithm, a weighted principle component technique is employed. Let

$$\mathbf{U}_4 V_4^T = (\vec{e_1}\ \vec{e_2}\ \ldots\ \vec{e_l}) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_l \end{pmatrix} \begin{pmatrix} \vec{e_1} \\ \vec{e_2} \\ \vdots \\ \vec{e_l} \end{pmatrix} \qquad (4.32)$$

where the λ are the eigenvalues of the matrix $\mathbf{U}_2\mathbf{V}_2$ and e_i are their corresponding eigenvectors representing eigenimages.

An eigen image represents a pattern of the depth uncertainty and its eigenvalue represents the corresponding magnitude of the uncertainty. Since the eigenvalues of Equation (4.32) are in decreasing sequence, the first k $(k = 3)$ eigenimages store the bulk of the (uncertainty) information. i.e.,

$$\mathbf{U}_4\mathbf{V}_4^T \approx (\vec{e_1}\ \vec{e_2}\ \ldots\ \vec{e_k}) \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_k \end{pmatrix} \begin{pmatrix} \vec{e}_1 \\ \vec{e}_2 \\ \vdots \\ \vec{e}_k \end{pmatrix} \tag{4.33}$$

However, since the independent uncertainty (represented by $\mathbf{C}_2$) is not uniform the truncation of the eigenvalues of $\mathbf{U}_2\mathbf{V}_2$ may not be appropriate. The independent uncertainty may be large and small in different areas of the image and it is useful to have a measure which shows the *combined* effect of both independent and correlated uncertainties. Hence a technique is proposed based on *weighted principle component analysis*. Since the independent uncertainty $\mathbf{C}_2$ is positive and diagonal, it can be decomposed as

$$\mathbf{C}_4 = \begin{pmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_N \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{c_1} & & & \\ & \sqrt{c_2} & & \\ & & \ddots & \\ & & & \sqrt{c_N} \end{pmatrix} \begin{pmatrix} \sqrt{c_1} & & & \\ & \sqrt{c_2} & & \\ & & \ddots & \\ & & & \sqrt{c_N} \end{pmatrix} = \mathbf{Q}\mathbf{Q}^T$$

Hence the overall uncertainties can be represented by

$$\mathbf{C}_4 + \mathbf{U}_4\mathbf{V}_4^T = \mathbf{Q}(\mathbf{I} + \mathbf{Q}^{-1}\mathbf{U}_4(\mathbf{Q}^{-1}\mathbf{V}_4)^T)\mathbf{Q}^T \tag{4.34}$$

In other words, the correlated uncertainty represented by $\mathbf{U}_4$ and $\mathbf{V}_4$ is weighted by the independent uncertainty $\mathbf{Q}^{-1}$. Hence the k largest eigenvalues of

$\mathbf{Q}^{-1}\mathbf{U}_4(\mathbf{Q}^{-1}\mathbf{V}_4)^T$

are used to regenerate the uncertainty of the depth information.

7. *Update The State Vector* $\boldsymbol{X}$

The state vector $\mathbf{X}$ (comprised of $\mathbf{M}$ and $\mathbf{Z}$) is now updated as

$$\begin{aligned} \mathbf{M} &= \mathbf{M} + \mathbf{C_x}\mathbf{K_m}\,(\vec{\mathbf{w}} - \mathbf{h}(\vec{\mathbf{x}})) \\ \mathbf{Z} &= \mathbf{Z} + (\mathbf{C_3} + \mathbf{U_3}\mathbf{V_3^T})\,(\vec{\mathbf{w}} - \mathbf{h}(\vec{\mathbf{x}})) \end{aligned}$$

Steps 1 through 7 above are iterated until the difference in values of the unknowns between two successive iterations is less than a predefined threshold.

4.3 Implementation and Results

We ran several experiments on real data captured using a Sony DCR-VX2000 camera. We fixed the camera parameters f, D and S and took a set of 3 images as shown in Figure 1.

We computed the optical flow and defocus blur using the technique described in chapter 3. The radius map was smoothed and this was used to compute depth using the method elaborated in this chapter.

We assumed that the depth was initially planar and were able to demonstrate that we could get a good depth estimates with only three images. This can be used in an iterative technique where the depth estimates are refined over a sequence of several images.

When computing the *a posteriori* covariance P we found that better convergence was obtained if a weight was added to the diagonal elements. This weight was proportional to the change of value of the corresponding unknowns in 2 successive iterations.

In conclusion, our measurement Equations (4.9), (4.11), (4.16) or (4.20) have fewer unknowns and are less complex than the motion/optical flow equations which were employed in [48]. Hence we found that our solution converges faster, is more stable and is more forgiving if the initial state estimates are far from the actual values.

In the next few experiments we show the result of implementing the above algorithms on several image sequences. We first show the result on an object consisting of a fairly planar patch (upholstery on the back of a chair). We then run the experiment on a sequence of images of a toy hat. Finally we run the experiment on two different sequences of a toy bird.

4.3.1 Experiment 10: Plain Upholstery

The first experiment was performed on three images of the upholstery at the back of a chair. The results are displayed in Figures 4.3 through 4.10. The surface of the chair was slightly convex and the chair was slanted so that the left end was closer than the right end. The camera was positioned so that the right bottom was farthest from it.

The first image was taken with the center in perfect focus. Subsequent images were taken by moving the camera closer to the object so that the level of blur in the left and center increases with time while the level of blur in the right end decreases with time since it gets closer to the depth of field. In all the images we follow the convention of displaying blur as a filled circle and sharpening as a plain circle where the radius of the circle indicates the level blurring/sharpening.

We display the radius of blur/sharpening and optical flow between the first and second images in Figures 4.5 and 4.6 respectively. The maximum radius of sharpening is 2.89 and the maximum radius of blur is 3.83. The average radius is 2.28 and the standard deviation is 1.94.

Figures 4.8 and 4.9 display the radius of blur and optical flow respectively between the first and third images. This time the maximum radius of sharpening is 3.03 and the maximum radius of blur is 5.00. The average radius is 3.02 and the standard deviation is 2.31.

Figure 4.10 displays the computed relative depth. It can be seen that the top left of the chair is closest while the bottom right of the chair is farthest from the camera.

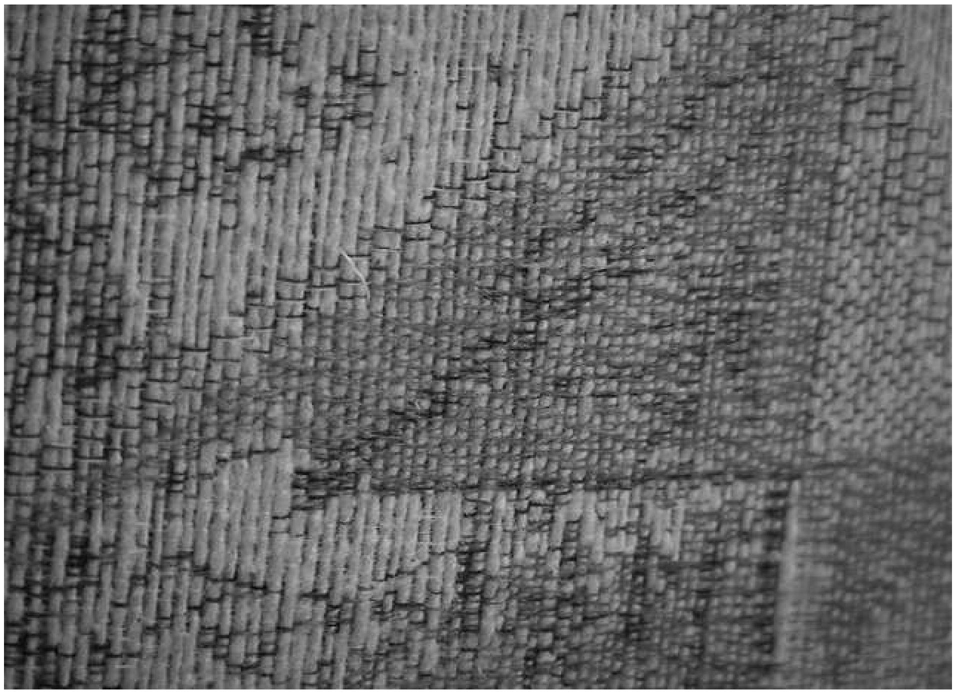


Figure 4.3: Experiment 10: Plain Upholstery (Image 1)

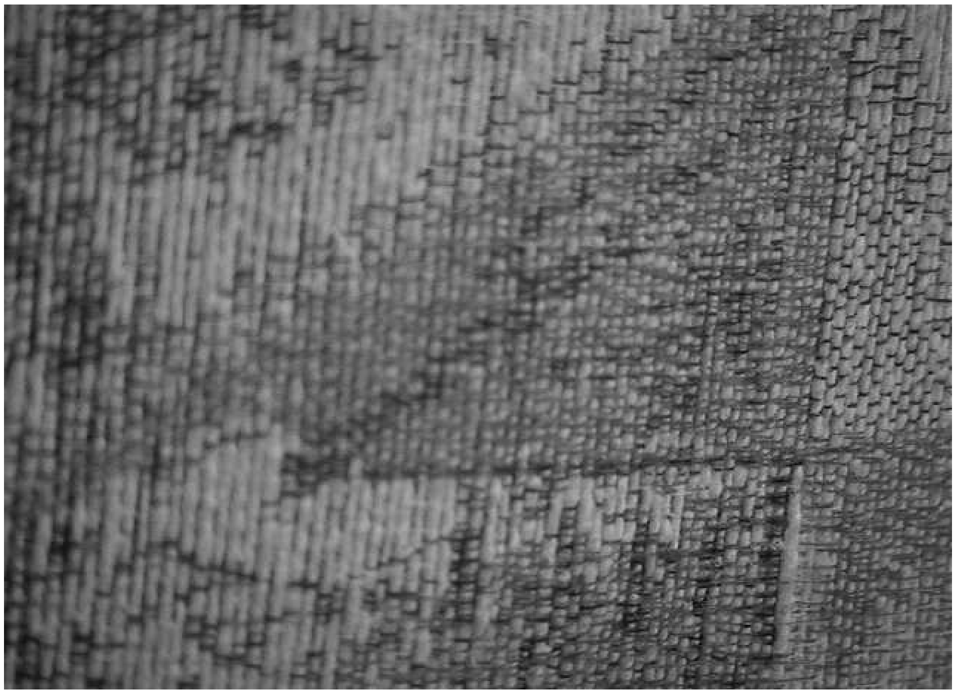


Figure 4.4: Experiment 10: Plain Upholstery (Image 2)

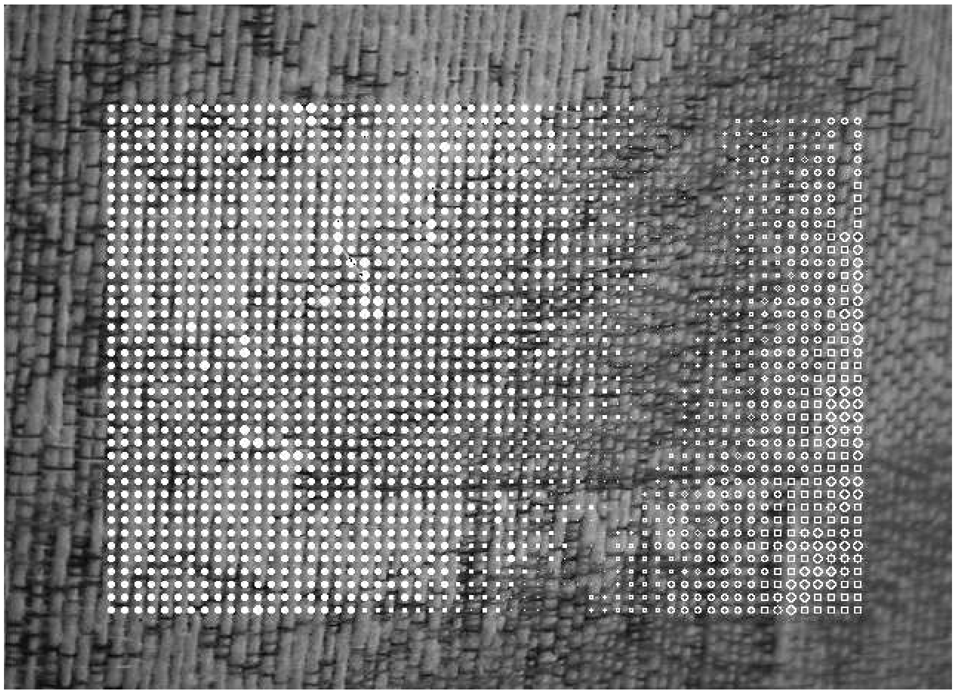


Figure 4.5: Experiment 10: Plain Upholstery (Radius Map From Image 1 to 2)

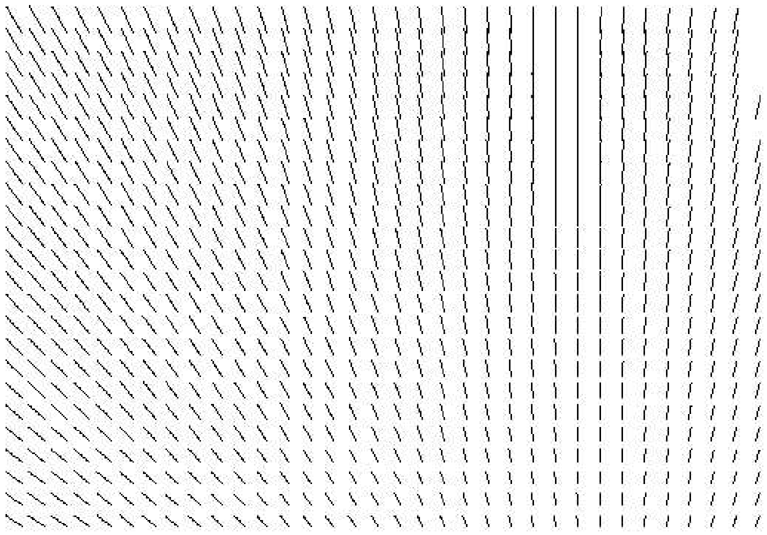


Figure 4.6: Experiment 10: Plain Upholstery (Flow Field From Image 1 to 2)

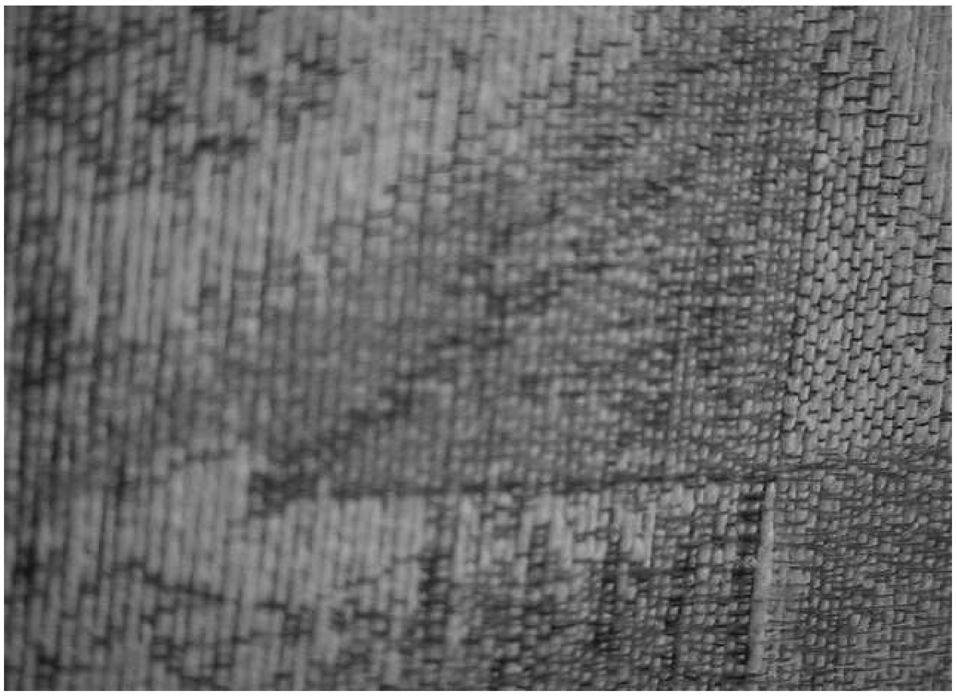


Figure 4.7: Experiment 10: Plain Upholstery (Image 3)

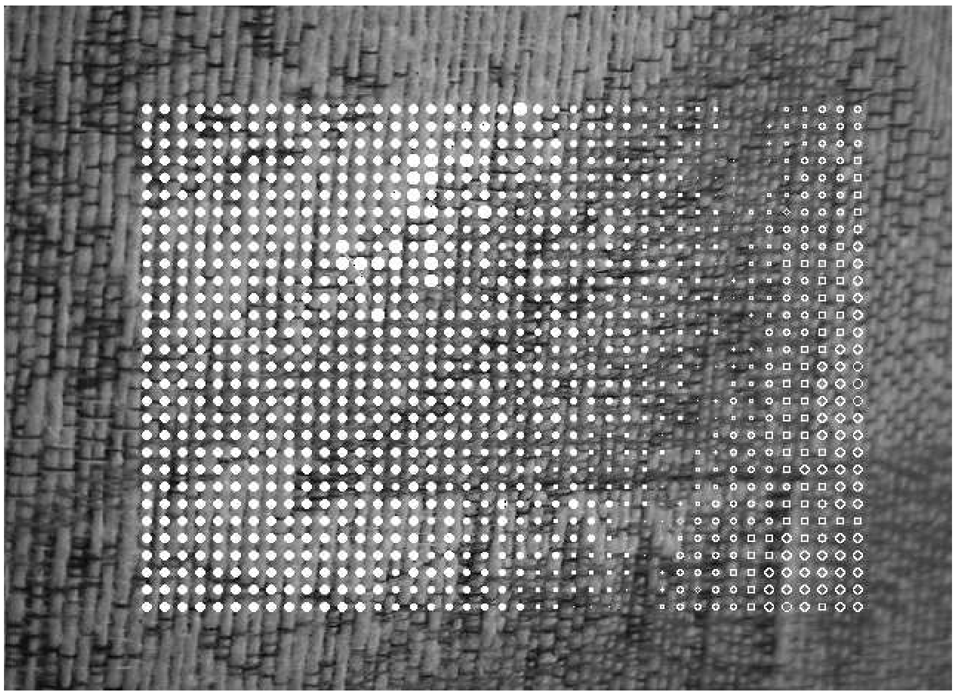


Figure 4.8: Experiment 10: Plain Upholstery (Radius Map From Image 1 to 3)

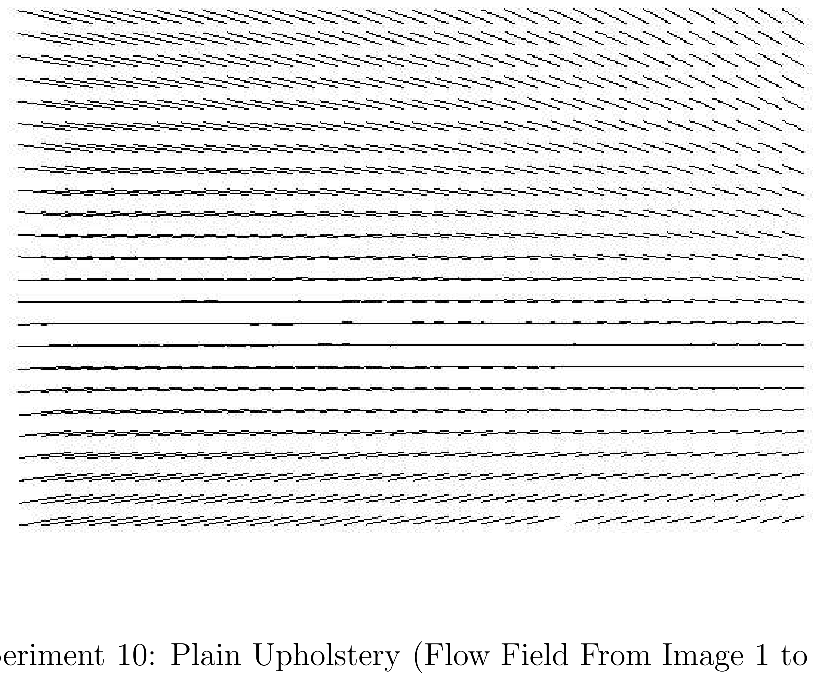


Figure 4.9: Experiment 10: Plain Upholstery (Flow Field From Image 1 to 3)

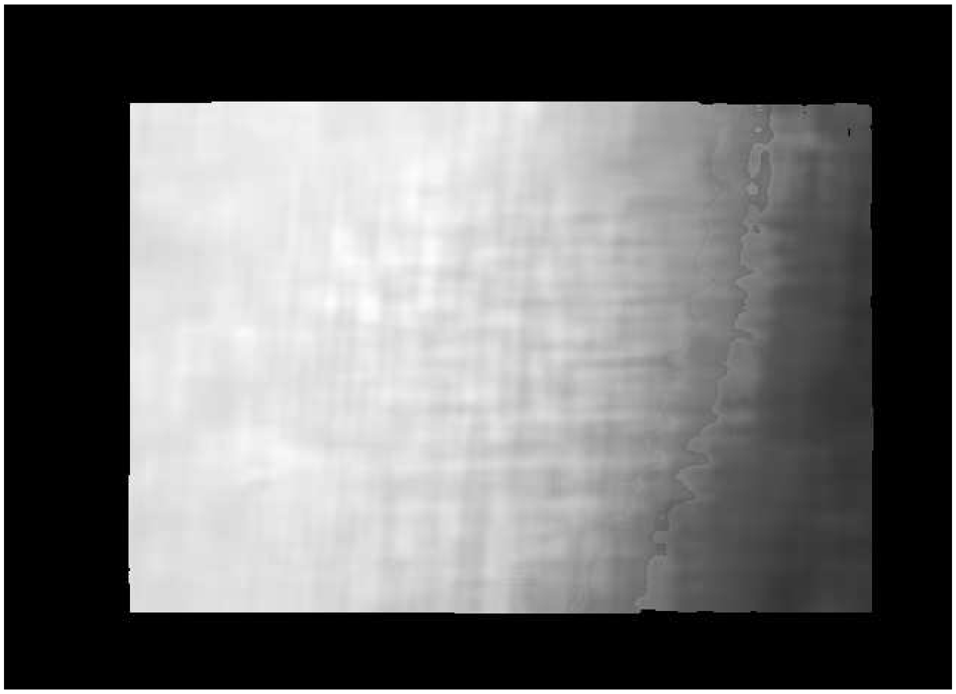


Figure 4.10: Experiment 10: Plain Upholstery (Depth Map Stage 1)

4.3.2 Experiment 11: Toy Hat

In the next experiment, three images of a toy hat were captured. The first image was taken with most points in perfect focus. Subsequent images were taken by moving the camera closer to the object so that the level of blur increased with time. The object was slanted so that the bottom left of the object was closer to the camera than the top right. Figures 4.11 through 4.18 display the results of this experiment. It can be seen that the radius of blur (Figures 4.13 and 4.16) is largest at points on the top of the hat since they are closest to the camera. On the other hand, points at the top right are sharpened in Figure 4.13 as they move closer to the depth of field of the camera. The radius of blur computed from image 1 to image 3 is greater than the radius of blur computed from image 1 to image 2. Once again, we follow the convention of displaying blur as a filled circle and sharpening as a plain circle with the radius of the circle indicating the level of blurring/sharpening.

Between the first and second images the maximum radius of sharpening is 1.75 and the maximum radius of blur is 2.92. The average radius is 0.95 and the standard deviation is 0.82.

Between the first and third images the maximum radius of sharpening is 1.83 and the maximum radius of blur is 4.09. The average radius is 1.98 and the standard deviation is 0.88.

Figure 4.18 displays the computed depth map after running the Kalman filter. It can be seen that the top of the hat is closest to the camera and the top right portion of the rim is farthest from the camera.

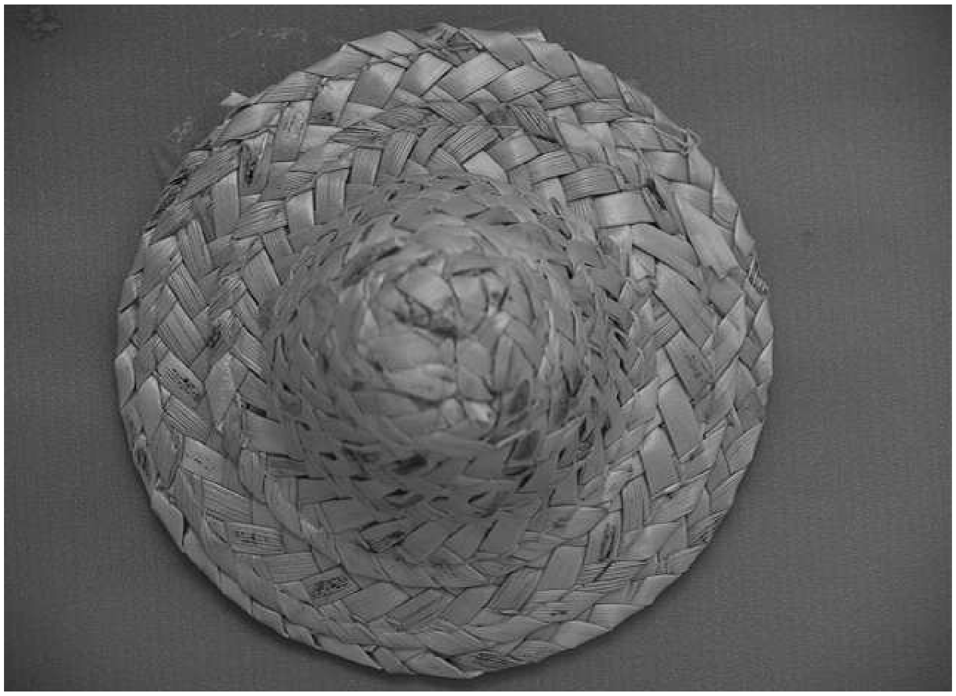


Figure 4.11: Experiment 11: Toy Hat (Image 1)

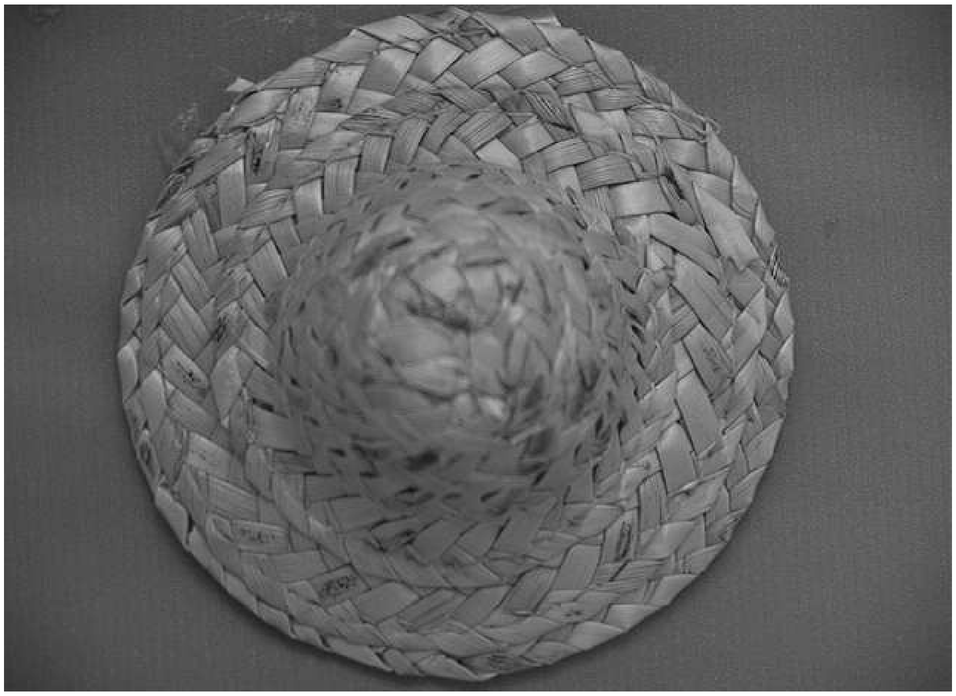


Figure 4.12: Experiment 11: Toy Hat (Image 2)

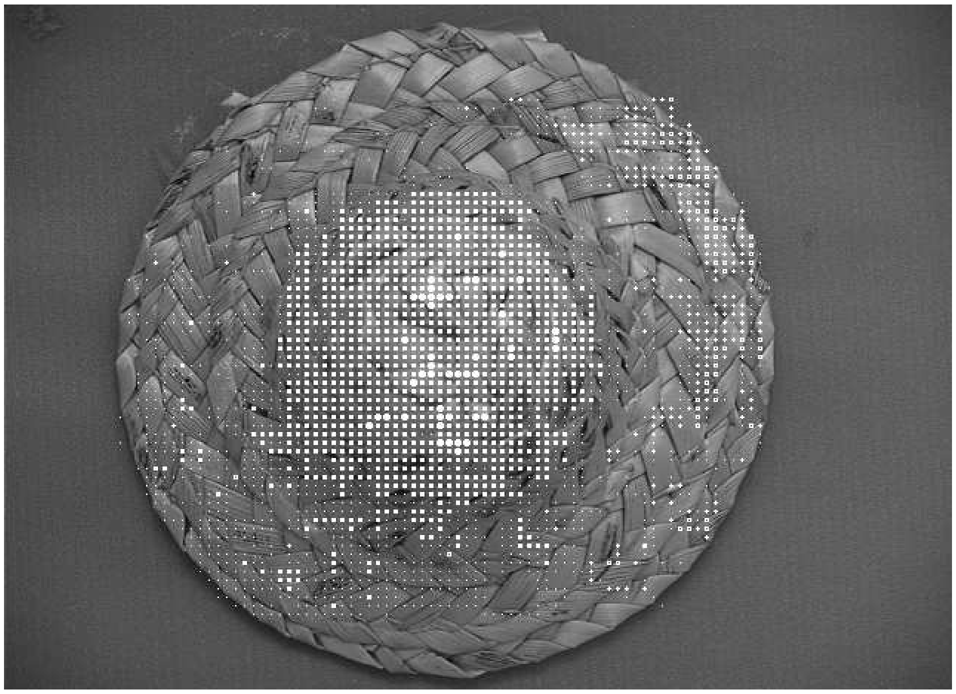


Figure 4.13: Experiment 11: Toy Hat (Radius Map From Image 1 to 2)

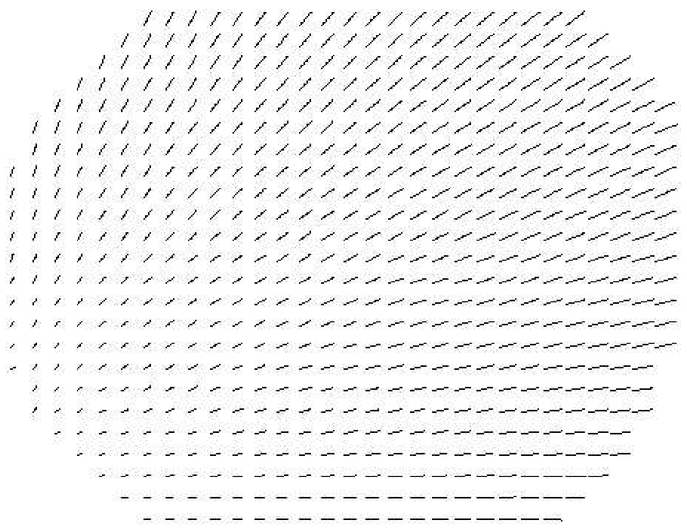


Figure 4.14: Experiment 11: Toy Hat (Flow Field From Image 1 to 2)

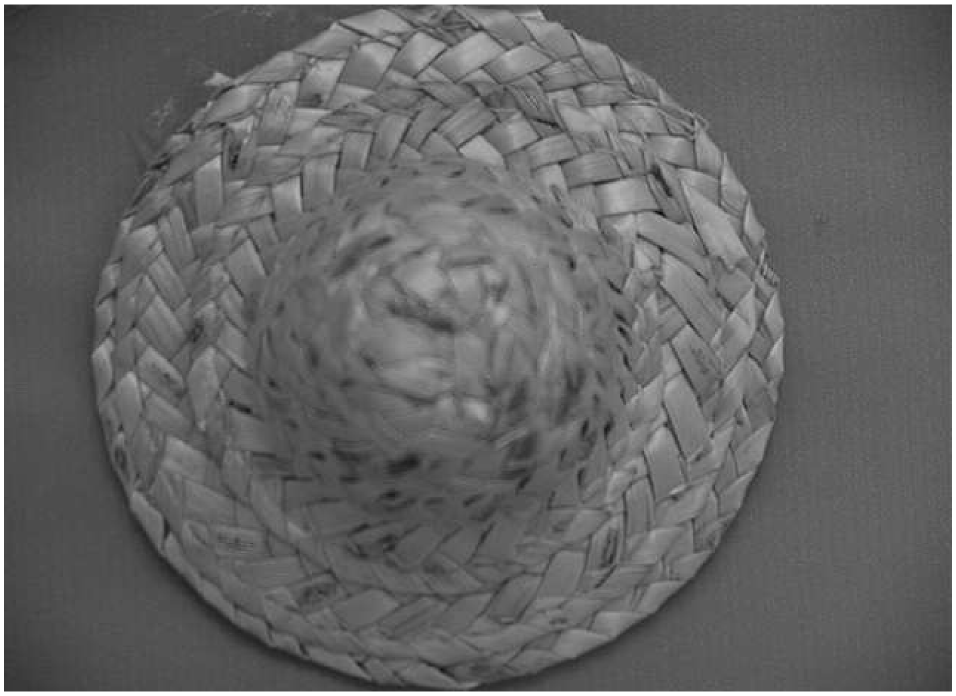


Figure 4.15: Experiment 11: Toy Hat (Image 3)

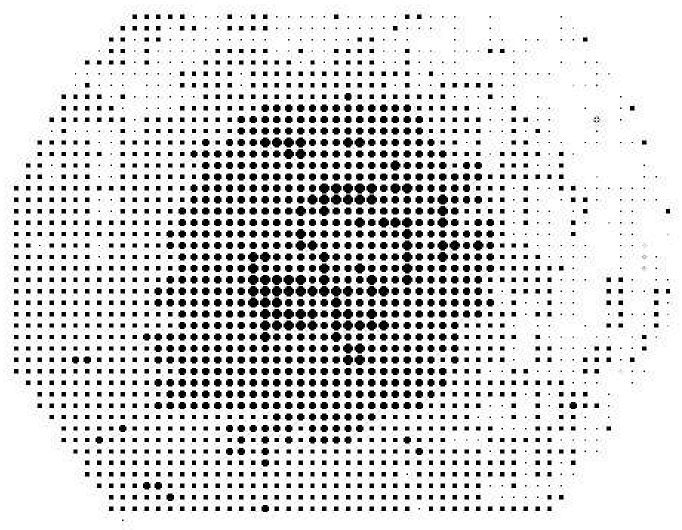


Figure 4.16: Experiment 11: Toy Hat (Radius Map From Image 1 to 3)

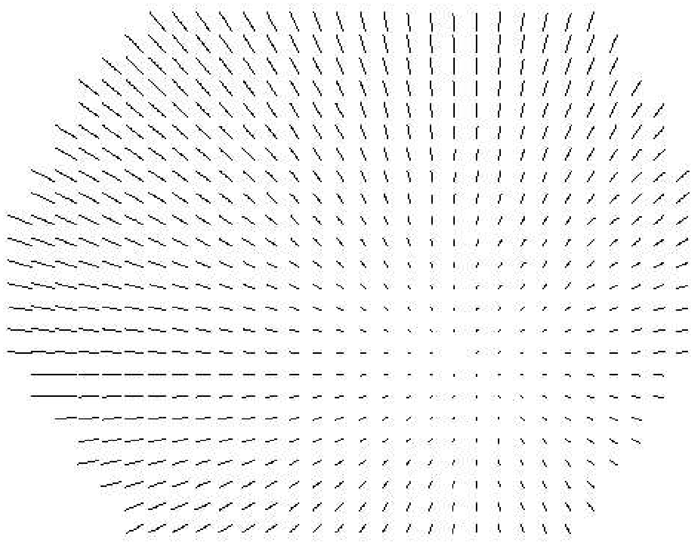


Figure 4.17: Experiment 11: Toy Hat (Flow Field From Image 1 to 3)

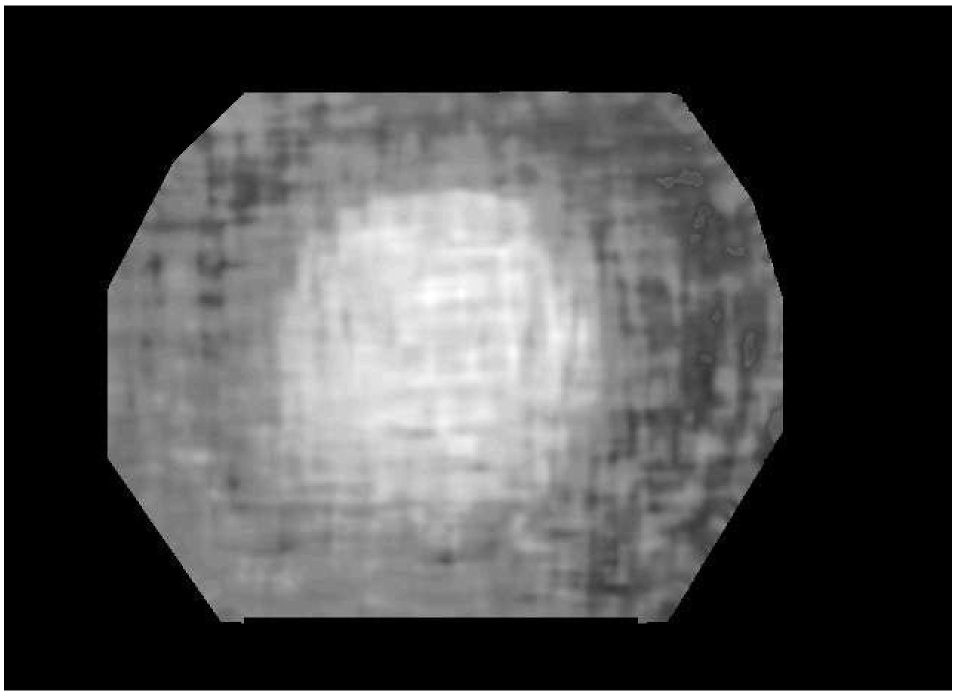


Figure 4.18: Experiment 11: Toy Hat (Depth Map Stage 1)

4.3.3 Experiment 12: Toy Bird 1 in 3-D

A toy bird was mounted on a stand and an image was captured of a side view such that the head was closer to the camera than the tail. Two more images were subsequently captured with the camera moving closer to the object so that the level of blur increased with time. Figures 4.19 through 4.26 display the results of this experiment.

It can be seen that the radius of blur (Figures 4.21 and 4.24) is largest in that section of the body of the bird which is closest to the camera. The tail portion of the bird slopes away from the camera and there is increasing level of sharpening as one moves towards the stand on which the bird is mounted. There is also a level of sharpening at the left edge of the bird and near the beak since they are farther from the camera than the rest of the bird.

Between the first and second images the maximum radius of sharpening is 2.34 and the maximum radius of blur is 3.99. The average radius is 1.82 and the standard deviation is 1.09.

Between the first and third images the maximum radius of sharpening is 2.48 and the maximum radius of blur is 4.47. The average radius is 2.54 and the standard deviation is 1.19.

The depthmap in Figure 4.26 displays relative depth.

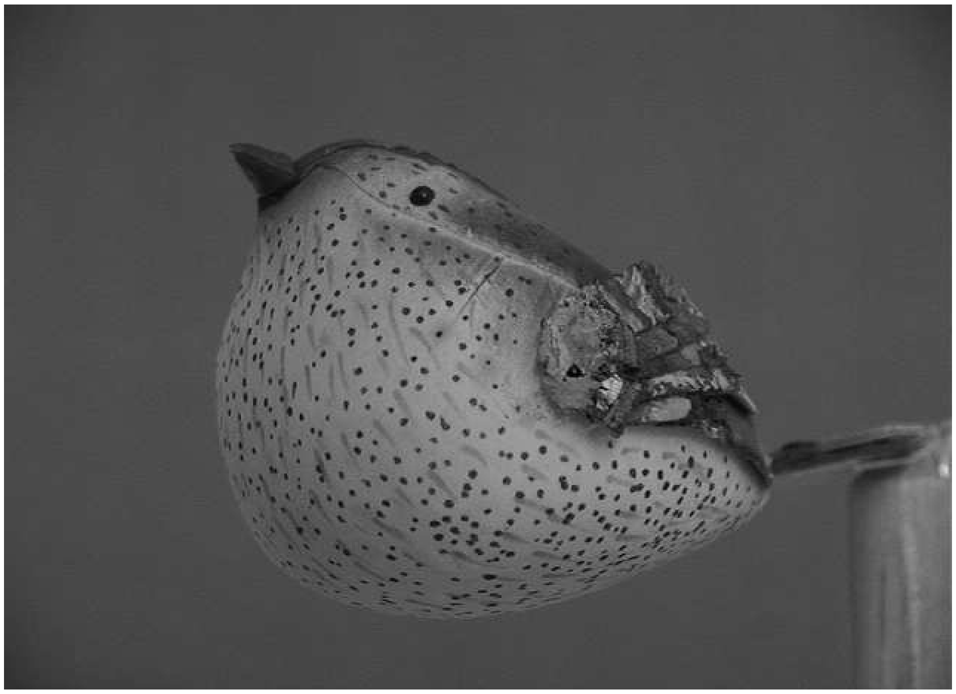


Figure 4.19: Experiment 12: Toy Bird 1 in 3-D (Image 1)

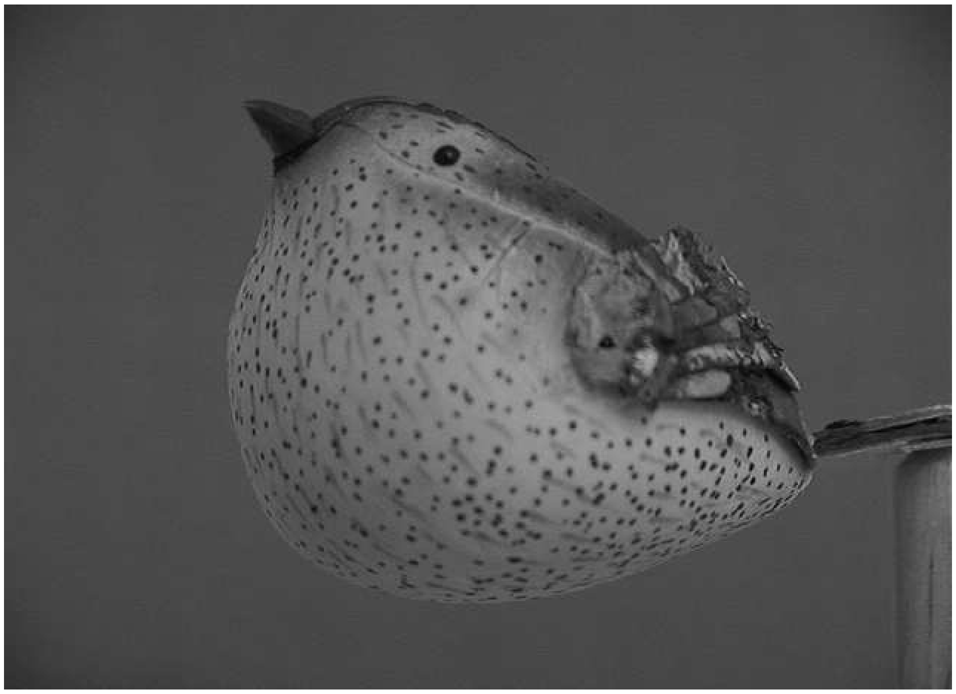


Figure 4.20: Experiment 12: Toy Bird 1 in 3-D (Image 2)

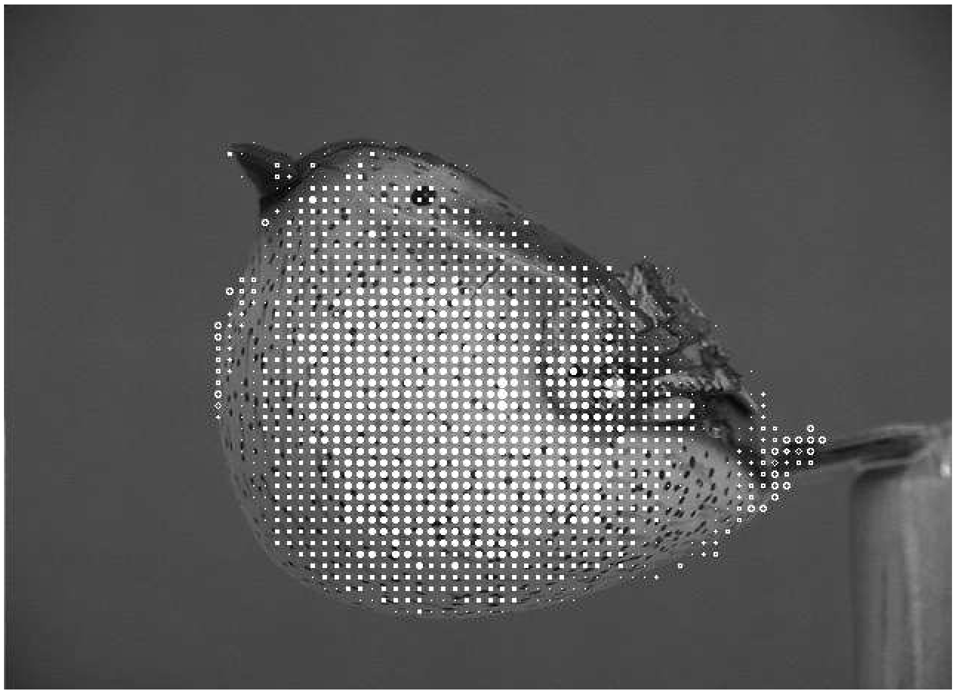


Figure 4.21: Experiment 12: Toy Bird 1 (Radius Map From Image 1 to 2)

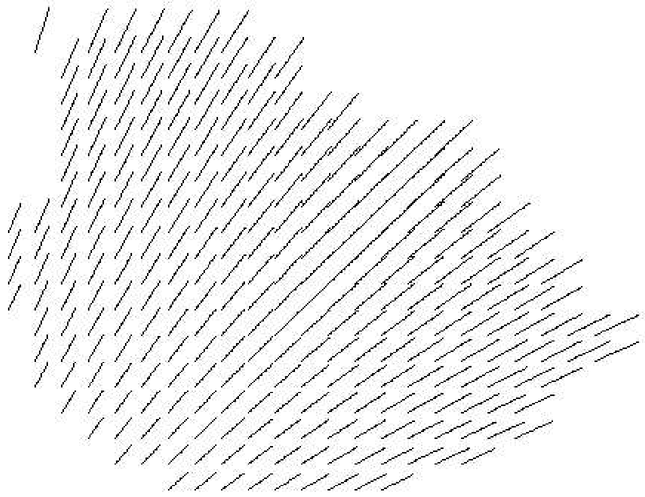


Figure 4.22: Experiment 12: Toy Bird 1 (Flow Field From Image 1 to 2)

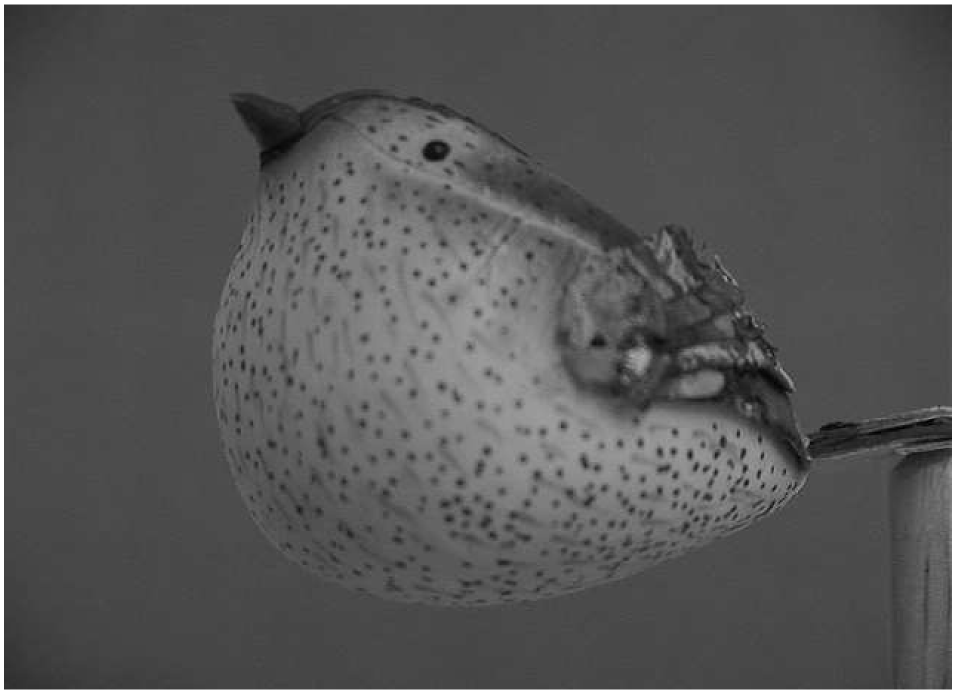


Figure 4.23: Experiment 12: Toy Bird 1 in 3-D (Image 3)

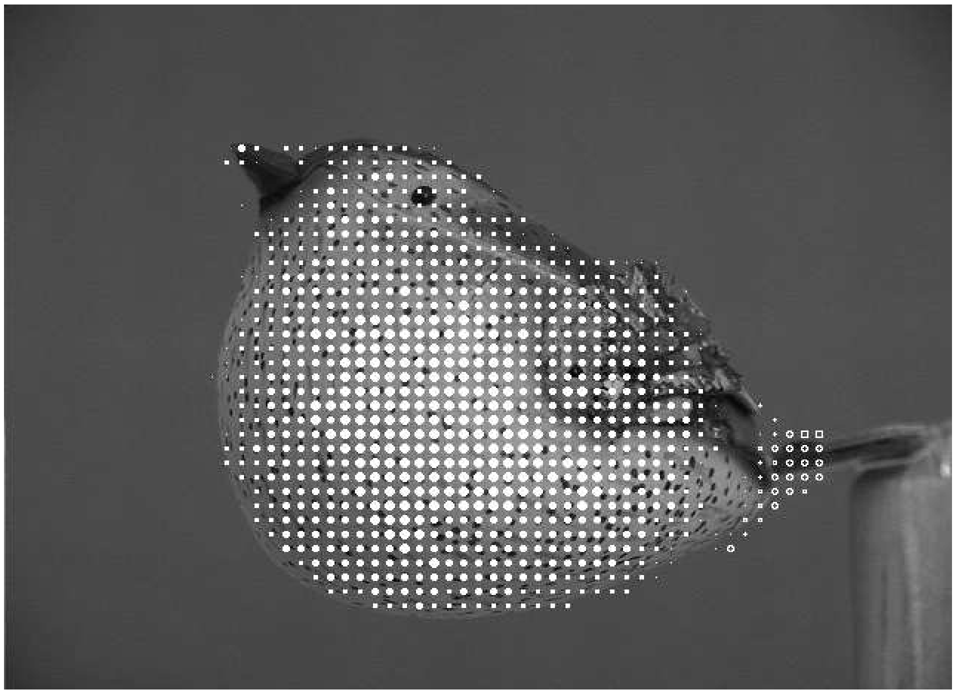


Figure 4.24: Experiment 12: Toy Bird 1 (Radius Map From Image 1 to 3)

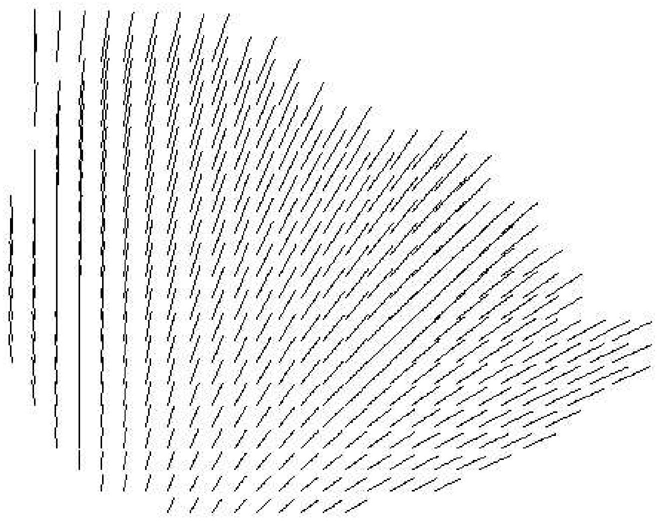


Figure 4.25: Experiment 12: Toy Bird 1 (Flow Field From Image 1 to 3)

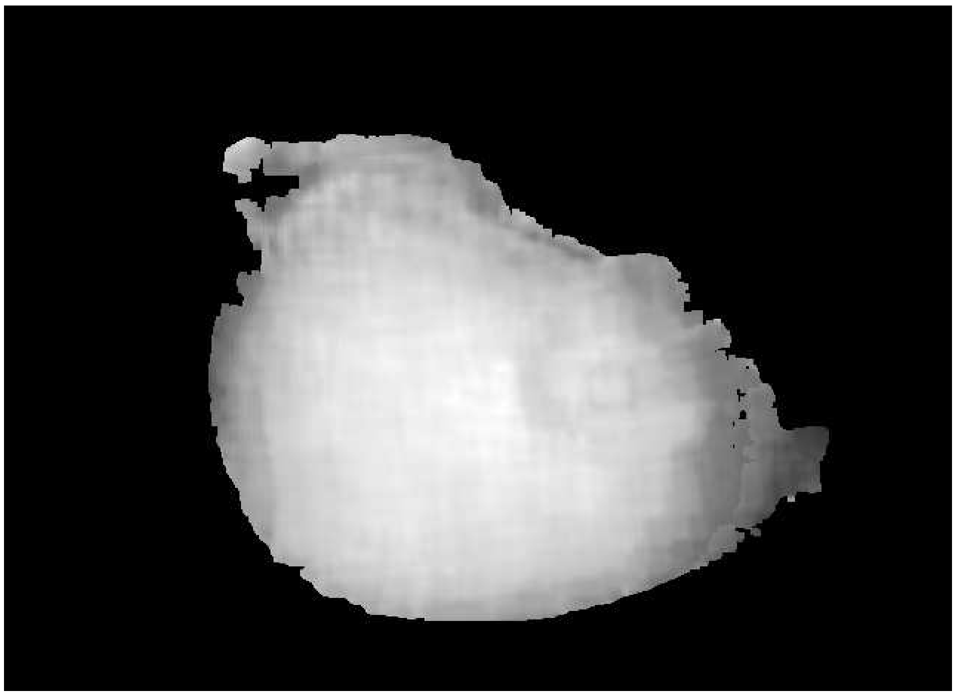


Figure 4.26: Experiment 12: Toy Bird 1 (Depth Map)

4.3.4 Experiment 13: Toy Bird 2 In 3-D

We then captured a three-quarters pose of the same toy bird. The results are displayed in Figures 4.27 through 4.34.

We display the radius of blur/sharpening and optical flow between the first and second images in Figures 4.29 and 4.30 respectively. The maximum radius of sharpening is 3.68 and the maximum radius of blur is 3.77. The average radius is 1.52 and the standard deviation is 1.14.

Figures 4.32 and 4.33 display the radius of blur and optical flow respectively between the first and third images. This time the maximum radius of sharpening is 3.99 and the maximum radius of blur is 4.48. The average radius is 2.61 and the standard deviation is 1.55.

The depth map generated (Figure 4.34 demonstrates that the tail of the bird and the stand it is mounted on are farthest away from the camera. The head of the bird and the edges of the object are also farther away than the central portion of the object.

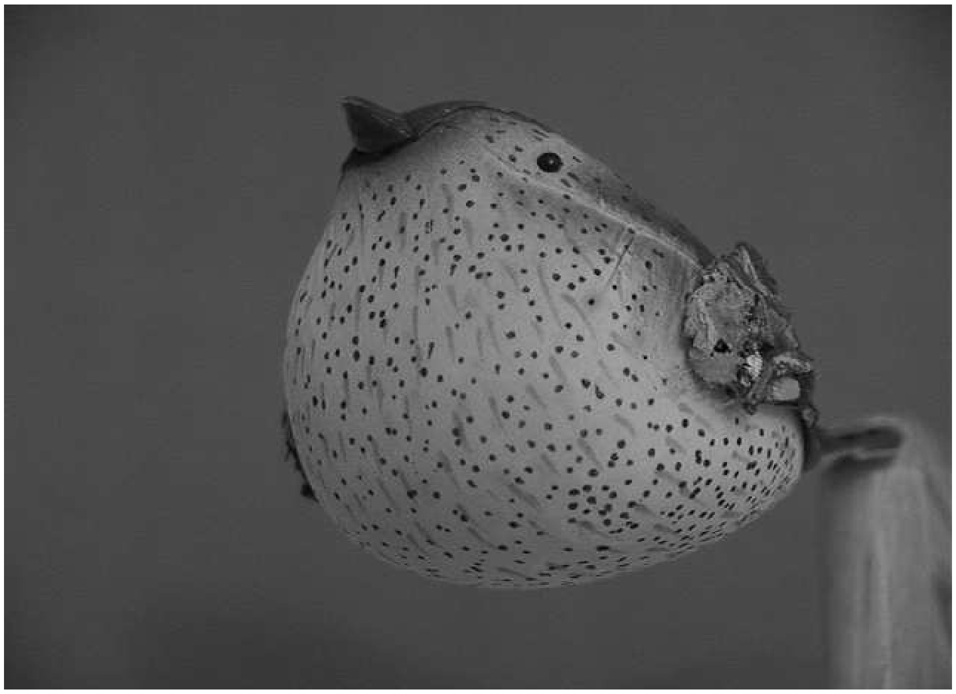


Figure 4.27: Experiment 13: Toy Bird 2 in 3-D (Image 1)

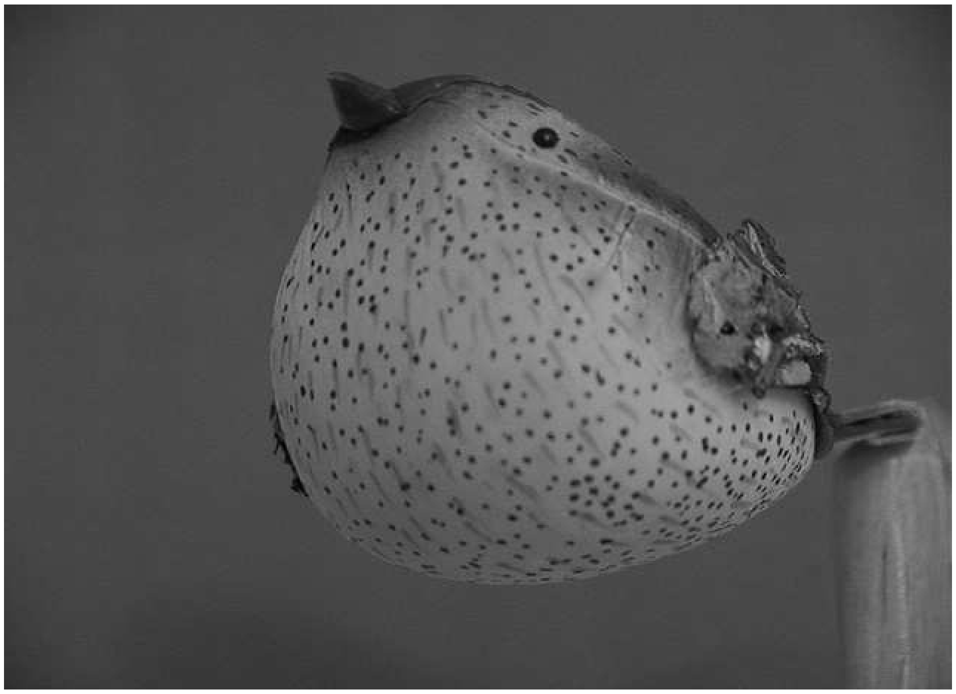


Figure 4.28: Experiment 13: Toy Bird 2 in 3-D (Image 2)

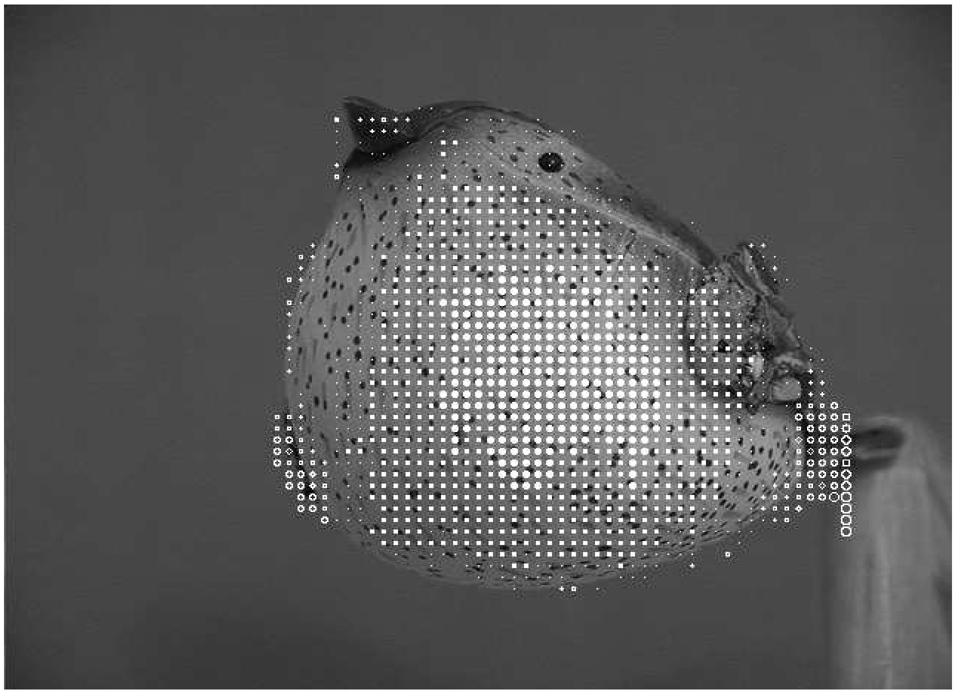


Figure 4.29: Experiment 13: Toy Bird 2 (Radius Map From Image 1 to 2)

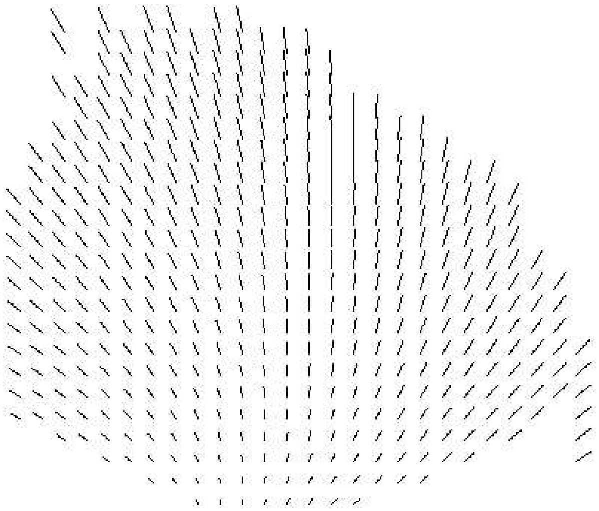


Figure 4.30: Experiment 13: Toy Bird 2 (Flow Field From Image 1 to 2)

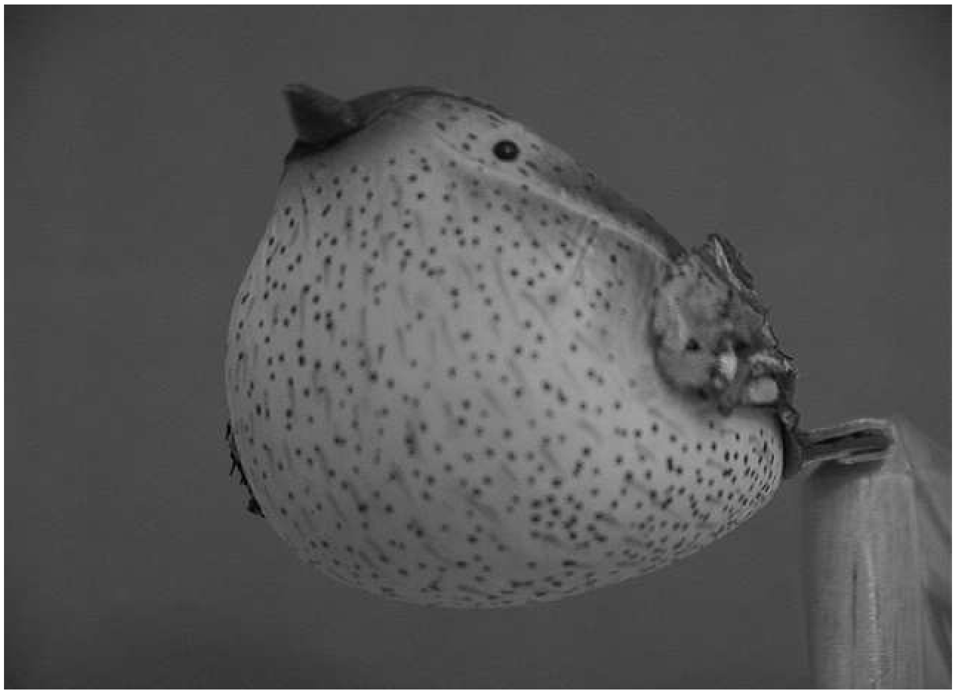


Figure 4.31: Experiment 13: Toy Bird 2 in 3-D (Image 3)

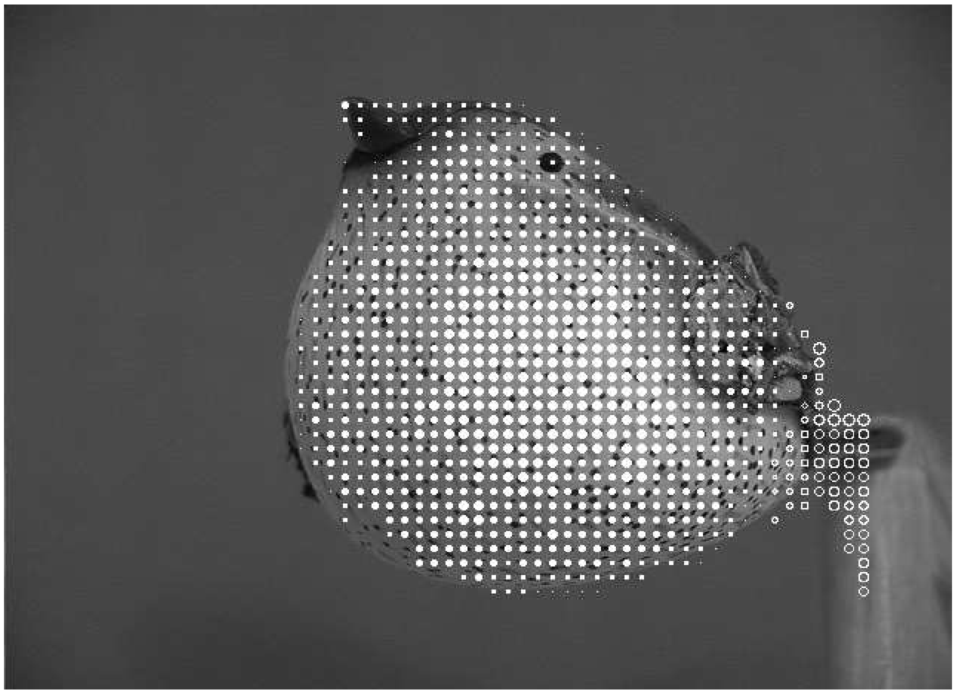


Figure 4.32: Experiment 13: Toy Bird 2 (Radius Map From Image 1 to 3)

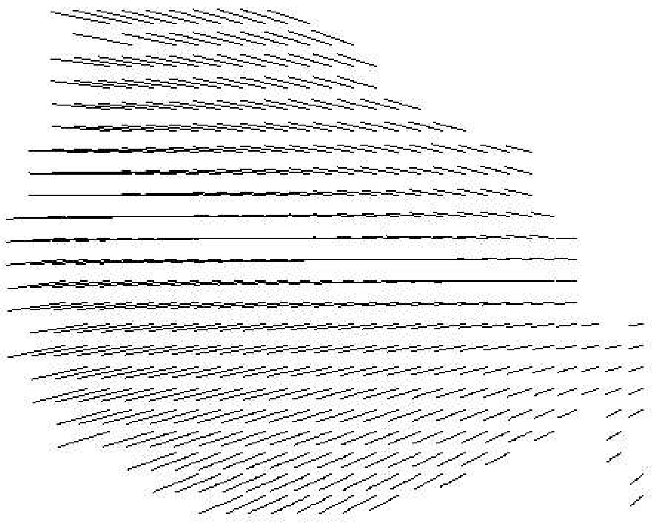


Figure 4.33: Experiment 13: Toy Bird 2 (Flow Field From Image 1 to 3)

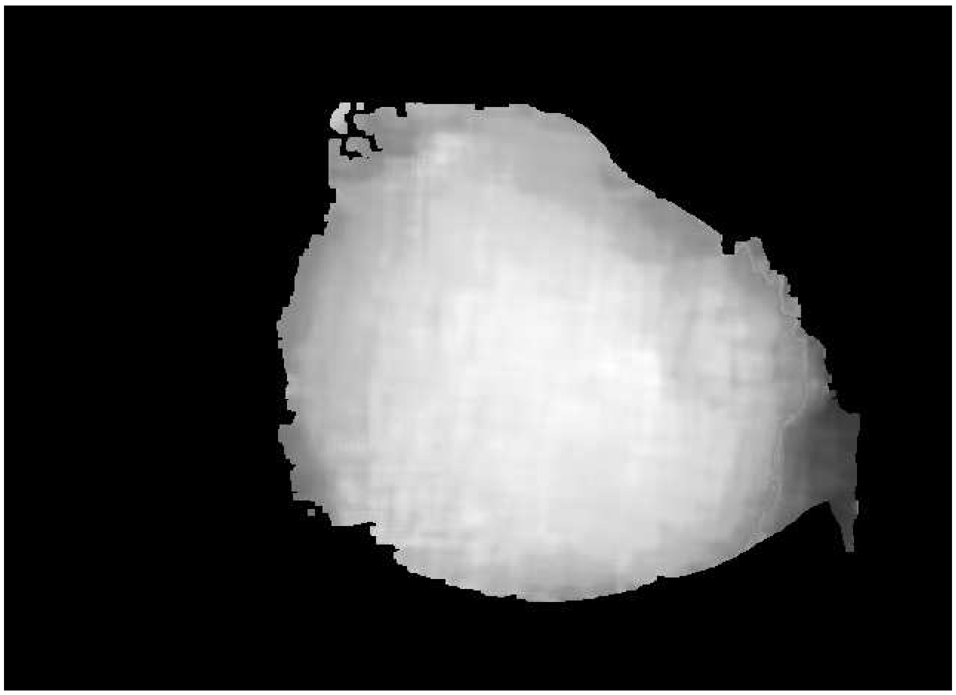


Figure 4.34: Experiment 13: Toy Bird 2 (Depth Map)

4.4 Discussion

The above method can be extended over a sequence of images in order to refine depth.

However, the approach in this chapter is only feasible if the following two conditions exist:

- The camera parameters and camera motion ensure that there is a variation in the level of blur. This variation reflects the change in depth and can hence be used to compute relative depth.
- The motion of the camera has very little out-of-plane rotation.

In cases where the blur level is so high (such as in the experiments in sections 3.3.2.1 and 3.3.2.2) that the blur is fairly uniform, the blur values do not provide information which can be used to compute depth. We studied the ratio between the standard deviation and average values of the radius of blur/sharpening on the results of our experiments. For all the experiments in which the initial estimate of depth was computed, the ratio was at least 0.4:1. The experiments in which we did not compute the initial value of depth the ratio was less than 0.15:1.

However, since our algorithm in chapter 3 can compute the optical flow in spite of considerable blur, we can still compute depth by using the optical flow values in the motion equations. In the following chapter we show how the optical flow values can be incorporated in an algorithm which computes relative depth in a depth-from-defocused-motion algorithm.

If relative depth can be inferred from blur values as shown in this chapter, these depth values can be incorporated to form a good initial estimate and thus ensure speedier convergence. Even if the radius values obtained through the algorithm in chapter 3 do not have sufficient variance to allow us to compute a reliable initial estimate of depth, the fact that the algorithm can recover large values of optical flow make it superior over other optical flow algorithms. Large values of optical flow and translation along the z axis are useful as they disambiguate camera motion and yield more reliable depth values.

CHAPTER 5

REFINEMENT OF DEPTH AND 3-D MOTION

Once an initial estimation of depth is obtained these values were input into the instantaneous motion Equations (2.2) in order to compute an initial estimate of the motion parameters.

These parameters (depth values, camera motion parameters) and their uncertainties are now input into another Kalman filter as described in chapter 4.2.2. However, this time the measurement equations are the discrete motion Equations 2.2. Hence C_m is a 6×6 matrix and the number of eigen values of the matrices $\mathbf{U}_4$ and $\mathbf{V}_4$ is six instead of three.

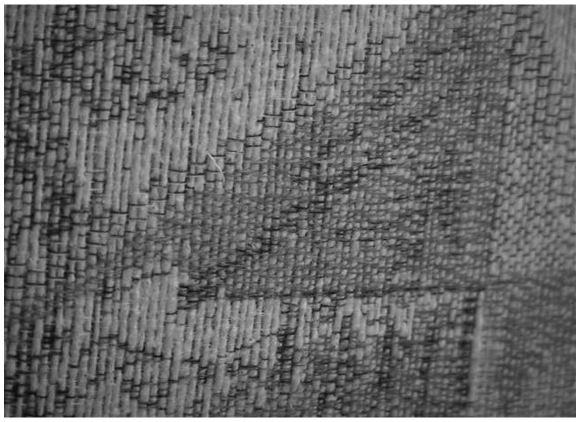
We show below, the results of some experiments. In these experiments we first computed an initial estimate of relative depth using the technique elaborated in Chapter 4. We then input these initial values of depth to a Kalman filter incorporating the motion equations to obtain better depth estimates. The depth values computed at both the phases are displayed. It can be seen that fairly good depth values are obtained even with only three frames in a sequence of images. This technique can be iterated over several frames in order to refine depth.

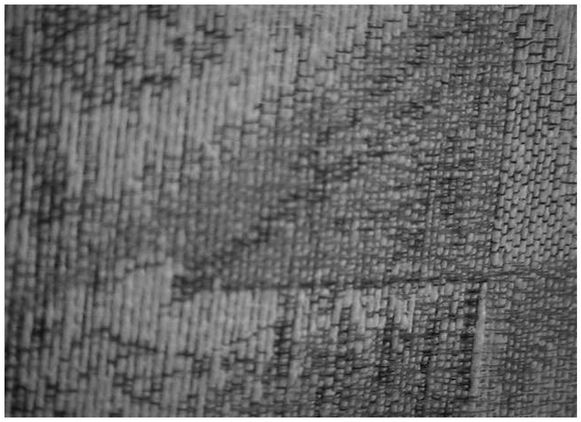
5.1 Implementation and Results

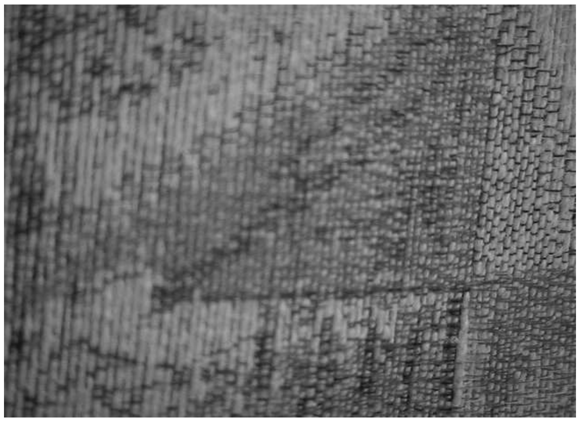
5.1.1 Experiment 14: Plain Upholstery

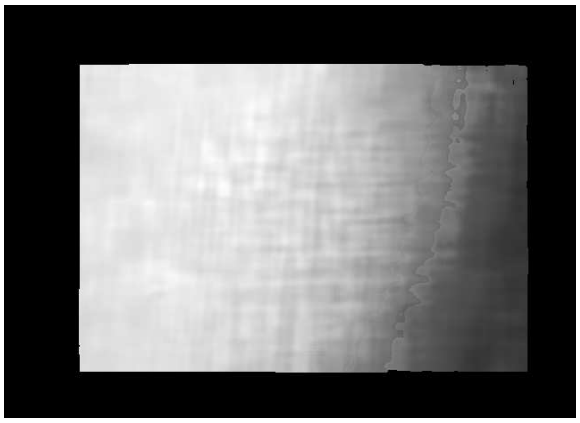
We ran the second Extended Kalman Filter on the upholstery sequence described earlier in Section (4.3.1).

Figure 5.1(d) displays the computed depth map at the end of the first stage (Chapter 4) and Figure 5.1(e) displays the computed depth map obtained after the second Kalman filter computation which incorporates the motion equations. It can be seen that the depth map is smoother and hence more accurate.


(a) Image 1


(b) Image 2


(c) Image 3

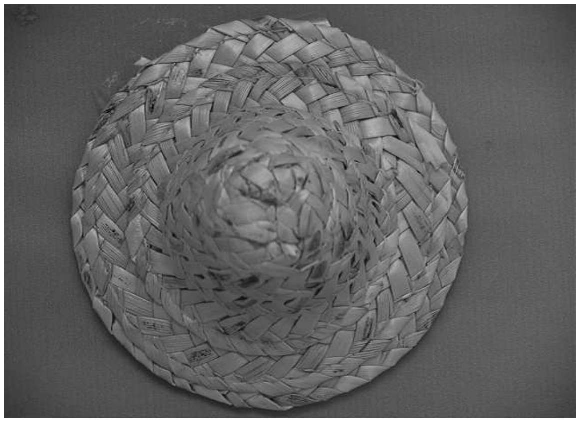

(d) Depth Map (Stage 1)

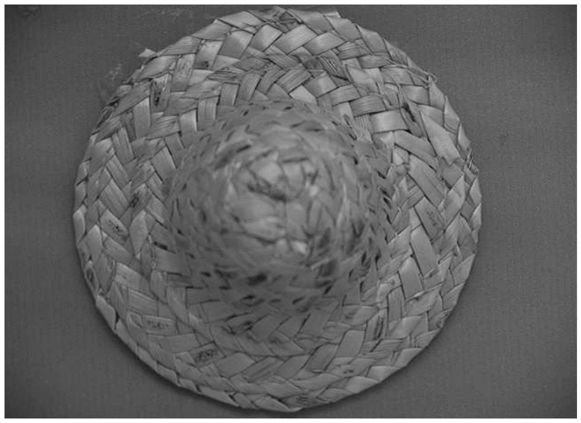
(e) Depth Map (Stage 2)

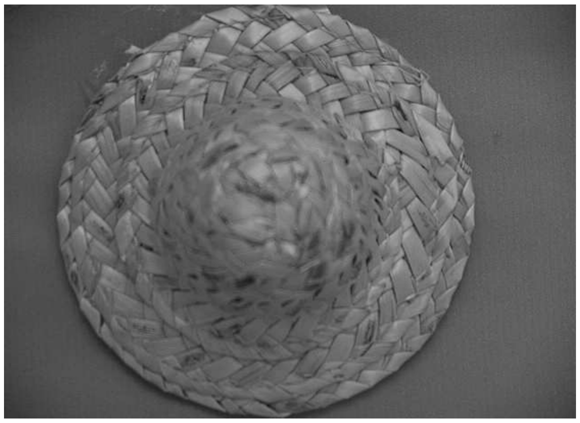
Figure 5.1: Experiment 14: Plain Upholstery - Refined Depth Map

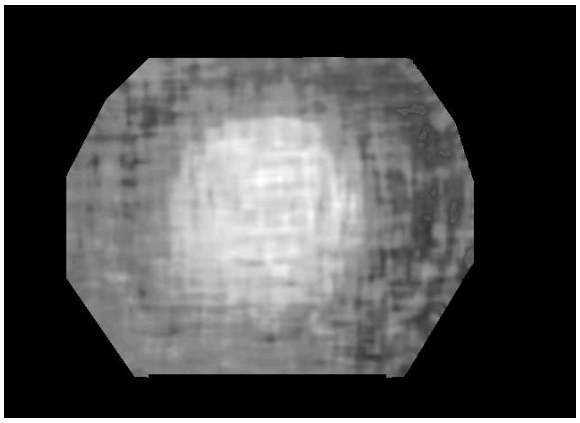
5.1.2 Experiment 15: Toy Hat

Next we ran the second Extended Kalman Filter on the toy hat sequence described earlier in Section (4.3.2). Figure 5.2(d) displays the computed depth map after the running the first Kalman filter and Figure 5.2(e) displays the computed depth map obtained after the second Kalman filter computation which incorporates the motion equations. Once again it can be seen that the depth map is smoother and hence more accurate.


(a) Image 1


(b) Image 2


(c) Image 3


(d) Depth Map (Stage 1)

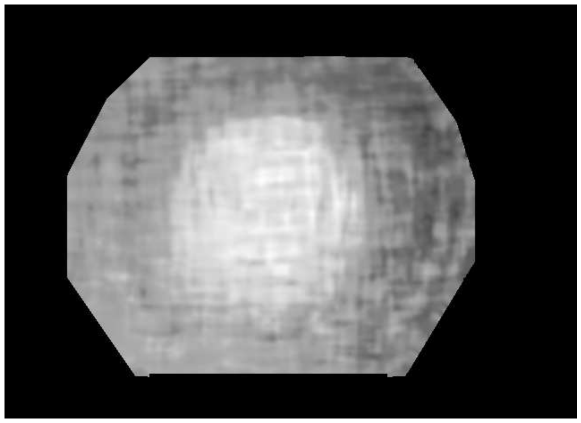

(e) Depth Map (Stage 2)

Figure 5.2: Experiment 15: Toy Hat - Refined Depth Map

CHAPTER 6

CONCLUSION

We have developed a novel technique which computes depth from defocused motion. This consists of several self-contained components each of which can be incorporated independently within other algorithms.

The first novel component involves the simultaneous computation of optical flow and defocus blur. It has been seen to be stable even for large optical flow and defocus blur values and on a variety of object shapes. The recovered values can be used as input in several different depth-from-motion or depth-from-blur algorithms.

The next component then uses the blur values obtained in a sequence of three images to derive a relationship between the radius of blur, object depth and parameters based on intrinsic camera parameters and camera motion.

Our formulation also handles the case when there is sharpening instead of blurring as well as the case when there is no blurring. More significantly, it is able to disambiguate the two situations corresponding to when the point of perfect focus is in front of the camera sensor and when it is behind the camera sensor. Our equations are superior to the motion equations in that there are fewer unknowns. Moreover the unknowns are related in a less complex fashion and one of them is linear, thus facilitating a more dependable solution process.

The *blur-depth relationship* is used as the measurement equation to an Extended Kalman Filter in order to obtain relative depth from the blur values. The depth values can be refined by being input into a second Kalman filter whose measurement equations are the discrete motion equations. This set of cascaded Kalman filters can be repeated over a longer sequence of images to obtain depth from defocused motion.

We have demonstrated the result of inputting our blur and optical flow values into two cascaded Kalman Filters. The results are shown for a sequence of three images and it can be easily extended over a longer sequence of images.

The algorithm which computes blur and optical flow (Chapter 3) assumes that there are large planar patches. However, our *blur-depth relationship* derived in Chapter 4 does not require large planar patches and can be used with blur values obtained through any other technique.

We have demonstrated how well our algorithms works for a sequence of just three images. The depth values can be further refined by using the Extended Kalman Filter to iterate over a longer sequence of images.

LIST OF REFERENCES

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *IJCV*, 2:283–310, 1989.

[2] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure, and focal length. *PAMI*, 17(6):562–575, June 1995.

[3] S.T. Barnard and W.B. Thompson. Disparity analysis of images. *T-PAMI-2*, 4:333–340, 1980.

[4] M.J. Black. Recursive non-linear estimation of discontinuous flow fields. In *ECCV*, volume A, pages 138–145, 1994.

[5] R. Bogen. *Macsyma Reference Manual*. Cambridge, Mass., Symbolics Incorporated, 1983.

[6] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *AeroSys*, 26(4):639–656, July 1990.

[7] C. Delherm, J.M. Lavest, M. Dhome, and J.T. La Preste. Dense reconstruction by zooming. In *ECCV96*, pages II:427–438, 1996.

[8] F. Deschenes, D. Ziou, and P. Fuchs. Simultaneous computation of defocus blur and apparent shifts in spatial domain. In *Proceedings of Vision Interface*, pages 236–243, 2002.

[9] J. Ens and P. Lawrence. A matrix method for determining depth from focus. In *CVPR*, pages 600–606, 1991.

[10] D.J. Fleet and A.D. Jepson. Computation of component image velocity from local phase information. In *IJCV*, volume 5, pages 77–104, 1990.

[11] J. Flusser and T. Suk. Image feature imvariant with respect to blur. *Pattern Recognition*, 28:1723–1732, 1995.

[12] J. Flusser and T. Suk. Degraded image analysis: An invariant approach. *IEEE PAMI*, 20:590–603, 1998.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins, 1996.

[14] D.J. Heeger and A.D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *IJCV*, 7:95–117, 1992.

[15] B.K.P. Horn. *Robot Vision*. McGraw-Hill, New York, 1986.

[16] B.K.P. Horn and B.G. Schunck. Determining optical flow. *AI*, 17:185–204, 1981.

[17] T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: A review. *PIEEE*, 82(2):252–268, February 1994.

[18] S.H. Lai, C.W. Fu, and S. Chang. A generalized depth estimation algorithm with a single image. *T-PAMI*, 14:405–411, 1992.

[19] J.M. Lavest, C. Delherm, B. Peuchot, and N. Daucher. Implicit reconstruction by zooming. *CVIU*, 66(3):301–315, June 1997.

[20] J.M. Lavest, G. Rives, and M. Dhome. Three-dimensional reconstruction by zooming. *IEEE Trans. on Robotics and Automation*, 9(2):196–207, April 1993.

[21] H. Liu, T.H. Hong, M. Herman, and R. Chellappa. A generalized motion model for estimating optical flow using 3-d hermite polynomials. *ICPR-A*, pages 361–366, 1994.

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

[23] Jun Ma and S.I. Olsen. Depth from zooming. *J. Optical Society of America*, 7:no.4:1883–1890, 1990.

[24] R. Manmatha. A framework for recovering affine transforms using points, lines, or image brightness. In *CVPR*, pages 141–146, 1994.

[25] Peter S. Maybeck. *Stochastic Models, Estimation and Control.* Academic Press, 1979.

[26] Bijan G. Mobasseri and Sivakumar Doraiswamy. Virtual motion: 3-d scene recovery using focal length-induced optic flow. *IEEE International Conference on Image Processing*, 3:78–82, 1994.

[27] U. Mudenagudi and S. Choudhury. Depth estimation using defocussed stereo image pairs. In *ICCV*, pages 483–488, 1999.

[28] Z. Myles and Lobo N. Recovering affine motion and defocus blur simultaneously. *T-PAMI*, 20:652–658, 1998(early version appeared in CVPR-96).

[29] H.H. Nagel. On a constraint equation for the estimation of displacement rates in image sequences. *T-PAMI*, 11:13–30, 1989.

[30] Shree Nayar, Masahiro Watanabe, and Minori Noguchi. Real-time focus range sensor. In *ICCV*, pages 995–1001, 1995.

[31] S. Negahdaripour and C.H. Yu. A generalized brightness change model for computing optical flow. In *ICCV*, pages 2–11, 1993.

[32] A.P. Pentland. A new sense for depth of field. *T-PAMI*, 9:522–531, 1987.

[33] A.P. Pentland, T. Darrell, M. Turk, and W. Huang. A simple real-time range camera. In *CVPR-89*, 1989.

[34] A.P. Pentland, S. Scherock, T. Darrell, and B. Girod. Simple range cameras based on focal error. In *Symposium on Advanced Image-Acquisition Technology*, pages 106–115, 1991.

[35] A. Singh. Optic flow computation: a unified perspective. *IEEE Computer Society Press*, 1992.

[36] W.J. Smith. *Modern Optical Engineering*. McGraw-Hill, 2nd Edition, 1990.

[37] M. Subbarao. Focussed image recovery from two defocussed images recorded with different camera settings. Technical Report 93.10.21, State Univ. of New York, 1993.

[38] M. Subbarao and N. Gurumurthy. Depth recovery from blurred edges. In *CVPR*, pages 498–503, 1988.

[39] M. Subbarao and T.C. Wei. Depth from defocus and rapid autofocusing: A practical approach. In *CVPR*, pages 773–776, 1992.

[40] G. Surya and M. Subbarao. Depth from defocus by changing camera aperture: A spatial domain approach. In *CVPR*, pages 61–67, 1993.

[41] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *CVPR*, pages 194–201, 1994.

[42] R. Szeliski and S.B. Kang. Recovering 3d shape and motion from image streams using non-linear least squares. In *CVPR*, pages 752–753, 1993.

[43] Y. Tian and M. Shah. Motion estimation and segmentation. *Machine Vision and Applications*, 9:32–42, 1196.

[44] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, November 1992.

[45] S. Uras, F. Girosi, and V. Verri, A.and Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–97, 1988.

[46] A.M. Waxman, J. Wu, and Bergholm F. Convected activation profiles and the measurement of visual motion. In *CVPR*, pages 717–723, 1988.

[47] J. Weber and J. Malik. Robust computation of optic flow in a multi-scale differential framework. In *ICCV*, pages 12–20, 1993.

[48] Y. Xiong and S.A. Shafer. Dense structure from a dense optical flow sequence. In *SCV95*, pages 1–6, 1995.

[49] Y. Zhang, C. Wen, and Y. Zhang. Estimation of motion parameters from blurred images. *Pattern Recognition Letters*, 21(5):425–433, 2000.

[50] Y. Zhang, C. Wen, and Y. Zhang. Simultaneously recovering affine motion and defocus blur using moments. In *Proceedings of the 15th International Conference on Pattern Recognition(ICPR'00)*, pages 881–884, 2000.