Measuring the Affine Transform - I: Recovering Scale and Rotation

R. Manmatha
J. Oliensis

COINS TR92-74

Measuring the Affine Transform - I: Recovering Scale and Rotation

R. Manmatha and John Oliensis *
Computer Science Department
University of Massachusetts at Amherst, Amherst, MA-01002
manmatha@cs.umass.edu

Abstract

Deformations due to relative motion between an observer and an object may be used to infer 3-D structure. Up to first order these deformations can be written in terms of an affine transform. This paper considers the problem of measuring the affine transform locally between two image patches using weighted moments of brightness. It is shown that the moments of image patches are related through functions of affine transforms. Finding the weighted moments is equivalent (for the purposes of measuring the affine transform) to filtering the images with gaussians and derivatives of gaussians. In the special case where the affine transform can be written as a scale change and an in-plane rotation, the zeroth and first moment equations are solved for the scale. Experimentally, the method seems to be robust.

*This research was supported by the National Science Foundation under grants CDA-8922572 and IRI-9113690.

1. Introduction

Changes in the relative orientation of a surface with respect to a camera cause deformations in the image of the surface. Up to first order, there are four kinds of image deformations - a scale change, rotation in the image plane and shear along the x and y axes. Deformations can be used to infer local surface geometry from motion [11, 13, 5, 8]. Since a repeating texture pattern can be thought of as a pattern in motion, shape from texture can also be derived from deformations [10, 14]. Constraints on the shape of the undeformed structure also allow the computation of shape from texture [3, 6].

The position of the surface's projection in the image also changes with motion. To first order, this image translation together with the deformation can be described using a six parameter affine transformation $\mathbf{A}$where

$$\mathbf{Ar} = \begin{vmatrix} u_0 \\ v_0 \end{vmatrix} + \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} \begin{vmatrix} r_x \\ r_y \end{vmatrix} \tag{1}$$

r_x and r_y are the image coordinates and u_0 and v_0 the image translation. Assuming local planarity, and the weak perspective projection, Kanade and Kender [10] showed that the image projections of two surface patches are related by an affine tranformation. Even in situations where the full-perspective projection must be used, it can be shown that if the change in relative orientation of the surface patches is small, the image projections can again be related by an affine transform [1]. The description in terms of the affine transform encodes useful 3-D information.

The recovery of 3-D structure requires robust *local* estimates of the affine parameters. Consider the case where an image patch F_1 is deformed into a patch F_2 (either in the same image or in another image) by an unknown affine transform. The problem of measuring the affine transform is to first find the corresponding patch F_2 given F_1 and second to recover the affine parameters from the two patches. Even if the centroids of the image patches are matched, the precise size and shape of F_2 is difficult to determine since it is a function of the unknown deformation. If this correspondence is not precisely done, the affine parameters will be determined incorrectly. Thus the problem is more difficult than in standard correspondence problems e.g. the determination of optical flow.

Existing methods using image patches usually ignore this problem. A number of techniques assume that the affine parameters are small and then linearize the brightness function or filtered versions of it with respect to the spatial coordinates to find the affine transform [9, 11, 4, 15]. Thus these methods are restricted to cases where the affine transform is small which in turn requires that the 3-D motion be small. Jones and Malik [8] do not assume that the affine transform is small. Their method, however, uses brute force search techniques and again ignores the determination of precise correspondence. A natural way to find correspondence is to use straight lines [13] or closed boundary contours [5] - the change in the size and shape of the enclosed area defining the affine tranform. These methods, however, fail when such structures are absent as in many richly textured scenes. Further, their use has only been demonstrated on homogeneous image regions with closed boundaries. For a detailed review of related work see section 1.1.

This paper presents a novel technique for reliably measuring affine transforms that correctly handles the difficulty of corresponding deformed image patches. The affine transform causes the shape and size of the the two patches be different. Using gaussians and derivatives of gaussians, this is handled naturally by deforming the filters appropriately. Measuring the deformation is therefore recast as a problem of finding the deformation parameters of a gaussian with which to filter. For example, let F_1 and F_2 be related by a scale change s. Then the output of F_1 filtered with a gaussian of σ will be equal to the output of F_2 filtered with a gaussian of $s\sigma$. Similar relationships hold for arbitrary affine transforms and filters described by derivatives of gaussians. These equations are exact for any arbitrary affine transform in arbitrary dimensions.

The second part of the paper focuses on solving for the affine transform when it can be written as the product of a scale change and a rotation (the solution for the general case will be considered in future papers). This particular situation arises frequently in many practical applications. For example, when a robot's motion is mostly translation, then the image projections of shallow structures (i.e. structures whose extent in depth is small compared to their distance from the camera) may be described in this way [13].

The equation can be solved by sampling the σ space. Rather than use a brute force search technique, the search space is sampled for a few different $\sigma's$ and one of the $\sigma's$ is picked as the operating point. The scale is recovered by linearizing the gaussian filter with respect to σ about this operating point using the diffusion equation. The choice of operating point depends upon the unknown scale change. Consistency is used to establish the correct operating point. By choosing different operating points, the technique is able to handle large scale changes - in principle arbitrary scale changes. The rotations can also be arbitrary. The scheme succeeds because *linearization is done with respect to* σ which is the correct way to handle the problem as opposed to the linearization with respect to the *image coordinates* done by other methods.

The gaussian (zeroth moment) equation gives an equation which is linear in the scale parameter. By sampling at several scales, an overconstrained linear system is obtained. This is solved for scale using singular value decomposition. Using the first moment an equation which is non-linear with respect to scale is obtained. Again, this may be sampled at multiple scales to provide an overconstrained system of equations. This non-linear system is solved using the Gauss-Newton technique. The first moment equation also allows the computation of the rotation. Note that although the in-plane rotation also causes deformations which must be taken into account in the computation of scale, the value of the rotation is not usually required by most applications. Both the formulation and the solution are done for arbitrary dimensions (not just 2). Experimental results are shown on both synthetic and real images attesting to the robustness and simplicity of the method.

1.1 Related Work

There are essentially three different approaches to finding deformations in the motion literature. The first approach linearizes the problem by assuming that the brightness can be written as follows:

$$E(\mathbf{r} + \mathbf{Ar}, t + \delta t) = E(\mathbf{r}, t) + \nabla E.\mathbf{Ar} \tag{2}$$

E($\mathbf{r}$,t) is the image irradiance at point $\mathbf{r}$ at time t. Given a sufficient number of points (≥ 6) this equation can be solved for the affine parameters A provided these are small (otherwise the Taylor approximation is not valid). The method is not robust except for computation of the image translation parameters [4]. Instead of the brightness function, higher order derivatives of brightness may also be used, so that there is one equation for each derivative of brightness. Alternatively, the brightness may be filtered with a set of basis functions, so that there is one equation for each basis function. Koenderink [11] and Werkhoven and Koenderinck [15] do this with a basis set of gaussians and derivatives of gaussians. A pyramid scheme [9] has also been used - this presumably improves the measurements for large image translations but does not address the problem of the robustness of the affine parameters. The linearization of these schemes implies that these methods are valid only when the affine transform is small.

Jones and Malik [8, 7] proposed a method which makes no linear approximations. Their method encodes each image region using a set of basis filters - (gaussians and derivatives of gaussians at 7 different scales). If two patches are related by an affine transform, then the corresponding filter outputs are related as follows:

$$\mathbf{F_1} = \mathbf{PF_2} \tag{3}$$

where $\mathbf{F_1}$ and $\mathbf{F_2}$ are k dimensional vectors (each entry being the output of one of the filters on the image patch) and **P** is a k by k matrix whose entries are functions of the affine matrix.

For each set of affine parameters they precompute **P**. The parameters for which 3 is minimized in a least squares sense are picked as the correct ones. In general the method requries a brute force four-dimensional search and would therefore be very expensive. Jones and Malik, however, use the constraint that in stereo, the change in the vertical disparity terms can be neglected, so that they have to solve for only two parameters reducing the search space somewhat. Again, the deformation of the images is neglected.

Line methods The projection of the boundary of a planar surface depends on the surface orientation and distance of the planar surface. Thus under motion, the projection of the boundary deforms appropriately. Methods based on using lines/curves take advantage of this property.

Williams et al [16] defined straight line segments using line intersections. The ratios of lengths of these lines in a sequence of images was used to find depth under looming.

Sawhney et al [13] used straight lines to hypothesize shallow structures. Each pair of three straight lines was hypothesized as a shallow structure. If the resulting motion of the structure could be approximated by a four parameter affine transform (image translation + scale + 2-D rotation), the structure was declared to be shallow. One advantage of this method is the ability to check if a motion can be described by an affine transform.

Cipolla and Blake [5] use snakes to find affine transforms. They assume that the surface is of roughly homogeneous brightness with a well-defined boundary. Further, they assume that the surface is kept at the image center using tracking. They then initialize a snake at the image center and expand it until it reaches the boundary. Measurement of line integrals along the boundary enable them to compute the affine transform. Many surfaces do not satisfy the homogeneous brightness assumption so that the method is limited in its applicability.

While image region based methods use the complete brightness information in a region, line methods ignore valuable information present in the interior of the image. Further, satisfactory line based techniques do not exist for arbitrary shaped curves. Also, not every image has lines available. On the plus side, line based techniques do make use of the deformation of the lines.

Finally, mention should be made of shape from texture techniques which make use of the second moment to recover the orientation of textured planes. Two of these methods [3, 6] compare the texture of a patch with a model of an undeformed texture patch to find the orientation of planes. They assume that the undeformed texture is isotropic (i.e. its second moment is the identity matrix). Brown and Shyvaster [3] compute the second moment of the auto-correlation function of an image while Garding [6] does the same in the frequency domain. Super and Bovik [14] explicitly compare the second moments of two image patches under an affine deformation to find surface orientation.

2. Measuring Deformation

This discussion will assume that the image translation is zero, which is equivalent to assuming that the image translation is known. Methods for finding the image translation will be briefly discussed in section 5.1.4. Given that the image translation is zero, the affine transform has only the four deformation parameters. Two different methods for computing the image deformation are proposed. The first is based on the observation that, in general, the result of a filtering operation on the two image patches will be different unless the filter is also accordingly deformed for the second image patch - the deformation being a function of the affine transform. The second method shows that moments of the image patches are related by simple functions of the affine transforms, and exploits this to compute the affine transform.

It is assumed that shading and illumination effects can be ignored. These can, however, be taken care of multiplying the equations by an additional constant factor. Most of what follows is valid for arbitrary dimensions; the 2-D case is used as a specific example.

3. Deformation of Filters

Notation Vectors will be represented by lowercase letters in boldface while matrices will be represented by uppercase letters in boldface.

Consider two functions F_1 and F_2 related by an affine transform. Since the affine transform may be viewed as a transformation of the underlying coordinate system, this may be written as

$$F_1(\mathbf{r}) = F_2(\mathbf{Ar}) \tag{4}$$

The area under the functions over some finite interval are related by:

$$\int_{-\mathbf{a}}^{\mathbf{a}} F_1(\mathbf{r})d\mathbf{r} = \int_{-\mathbf{a}}^{\mathbf{a}} F_2(\mathbf{Ar})d\mathbf{r} = \int_{-\mathbf{Aa}}^{\mathbf{Aa}} F_2(\mathbf{Ar})d(\mathbf{Ar}) \times det\mathbf{A}^{-1} \tag{5}$$

expressed succinctly as

$$\nu_1 = \nu_2 \times det\mathbf{A}^{-1} \tag{6}$$

Let s_i be the scale change along the i^th dimension and n the number of dimensions. The fact that $\det(\mathbf{A}) = \Pi_1^n s_i$, the product of the scale changes s_i leads to an intuitive explanation for equation (6). Consider the 1-D case (n = 1), where the affine transform reduces to a scale change. Assume that the function F_1 is graphed on a rubber sheet. The graph of F_2 is obtained by stretching the sheet (assuming the sheet stretches linearly). Along with the function, the coordinate axes and the units marked off along them are also stretched. The determinant term is equal to the stretching undergone by the coordinates. Note that the area under a function may also be viewed as the zeroth moment of the function.

Equation (5) cannot directly be used because the limits on the right-hand side depend on the affine transform and are therefore unknown. One way out would be to take the limits from $-\infty$ to ∞. However, this will not preserve localization, which is a very desirable property. The solution to this problem is to weight the function by another function which decays rapidly - here the gaussian will be used. This crucial point has not been handled correctly before. The case where $\mathbf{A} = s\mathbf{R}$, i.e. the affine transform is composed of a scale change s and a rotation $\mathbf{R}$, will be discussed first; this is followed by a discussion of the more general case.

3.1 Case $\mathbf{A} = s\mathbf{R}$

Denote the un-normalized gaussian by

$$H(\mathbf{r}, \sigma^2) = exp(-\mathbf{r}^T\mathbf{r}/2\sigma^2) \tag{7}$$

Multiply both sides of equation (4) by $H(\mathbf{r}, \sigma^2)$ to obtain :

$$F_1(\mathbf{r})H(\mathbf{r}, \sigma^2) = F_2(\mathbf{Ar})H(\mathbf{r}, \sigma^2) \tag{8}$$

The following identity follows from the orthonormality of rotation matrices:

$$H(\mathbf{r}, \sigma^2) = exp(-\mathbf{r}^T\mathbf{r}/2\sigma^2) = exp(-s\mathbf{r}^T R^T Rs\mathbf{r}/2(s\sigma)^2) = H(s\mathbf{Rr}, (s\sigma)^2) = H(\mathbf{Ar}, (s\sigma)^2) \tag{9}$$

and allows equation (8) to be rewritten as

$$F_1(\mathbf{r})H(\mathbf{r}, \sigma^2) = F_2(\mathbf{Ar})H(\mathbf{Ar}, (s\sigma)^2) \tag{10}$$

The weighted zeroth moment can therefore be written as

$$\begin{aligned} \int F_1(\mathbf{r})H(\mathbf{r}, \sigma^2)d\mathbf{r} &= \int F_2(\mathbf{Ar})H(\mathbf{Ar}, (s\sigma)^2)d\mathbf{r} \\ &= \int F_2(\mathbf{Ar})H(\mathbf{Ar}, (s\sigma)^2)d(s\mathbf{Rr})s^{-n} \end{aligned} \tag{11}$$

where the limits are taken from $-\infty$ to ∞. The factor $s^{-n} = det\mathbf{A}^{-1}$ and can be eliminated by using normalized gaussians

$$G(\mathbf{r},\sigma^2) = \frac{1}{(2\pi)^{n/2}\sigma^n} H(\mathbf{r},\sigma^2) \tag{12}$$

in place of H. The moment equation then becomes:

$$\begin{aligned}
\int F_1(\mathbf{r})G(\mathbf{r},\sigma^2)d\mathbf{r} &= \frac{1}{(2\pi)^{n/2}\sigma^n}\int F_1(\mathbf{r})H(\mathbf{r},\sigma^2)d\mathbf{r} \\
&= \frac{1}{(2\pi)^{n/2}\sigma^n}\int F_2(\mathbf{Ar})H(\mathbf{Ar},(s\sigma)^2)d(\mathbf{Ar})s^{-n} \\
&= \int F_2(\mathbf{Ar})G(\mathbf{Ar},(s\sigma)^2)d(\mathbf{Ar}) \qquad (13)
\end{aligned}$$

The integral may be interpreted as the result of convolving the function with a gaussian at the origin. It may also be interpreted as the result of a filtering operation with a gaussian. To emphasize these similarities, equation (13) may be written as

$$F_1 * G(\mathbf{r},\sigma^2) = F_2 * G(\mathbf{r}_1,(s\sigma)^2) \tag{14}$$

where $\mathbf{r}_1 = \mathbf{Ar}$.

Equation (14) is exact and is valid for arbitrary dimensions. The problem of recovering the affine parameters has been reduced to finding the deformation of a known function, the gaussian, rather than the unknown brightness functions. The basic intuition is that by using gaussian filters, the outputs are equal provided the gaussian is affine-transformed in the same manner as the function. The equation is invariant to rotation which implies that scale can be recovered but not rotation (the rotation can be recovered by other means to be discussed later). Note that although the limits are infinite, since the gaussian is a rapidly decaying function, it suffices in practice to take limits from -4σ to 4σ (and correspondingly from $-4s\sigma$ to $4s\sigma$ on the right hand side), maintaining spatial localization.

3.2 General Case

A similar equation can be shown to hold for arbitrary affine transforms, provided generalized gaussians are used. Define a generalized gaussian as

$$G(\mathbf{r},\mathbf{M}) = \frac{1}{(2\pi)^{n/2}det(\mathbf{M})^{1/2}}exp(-\frac{\mathbf{r}^T\mathbf{M}^{-1}\mathbf{r}}{2}) \tag{15}$$

where $\mathbf{M}$ is a symmetric positive semi-definite matrix. Then

$$\begin{aligned} G(\mathbf{r},\sigma^2\mathbf{I}) &= \frac{1}{(2\pi)^{n/2}det(\sigma^2\mathbf{I})^{1/2}}exp(\frac{-\mathbf{r}^T\mathbf{r}}{2\sigma^2}) \\ &= \frac{1}{(2\pi)^{n/2}\sigma^n}exp(\frac{-\mathbf{r}^T\mathbf{A}^T(\mathbf{A}^T)^{-1}\mathbf{A}^{-1}\mathbf{A}\mathbf{r}}{2\sigma^2}) \\ &= \frac{1}{(2\pi)^{n/2}\sigma^n}exp(-\frac{(\mathbf{A}\mathbf{r})^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{r})}{2\sigma^2}) \\ &= det(\mathbf{A}\mathbf{A}^T)^{1/2}G(\mathbf{A}\mathbf{r},\sigma^2(\mathbf{A}\mathbf{A}^T)) \end{aligned} \tag{16}$$

The weighted moment equation may therefore be written

$$\begin{aligned} \int F_1(\mathbf{r})G(\mathbf{r},\sigma^2 I)d\mathbf{r} &= det(\mathbf{A})\int F_2(\mathbf{A}\mathbf{r})G(\mathbf{A}\mathbf{r},\sigma^2(\mathbf{A}\mathbf{A}^T))d\mathbf{r} \\ &= det(\mathbf{A})\int F_2(\mathbf{A}\mathbf{r})G(\mathbf{A}\mathbf{r},\sigma^2(\mathbf{A}\mathbf{A}^T))d(\mathbf{A}\mathbf{r})det(\mathbf{A}^{-1}) \\ &= \int F_2(\mathbf{A}\mathbf{r})G(\mathbf{A}\mathbf{r},\sigma^2(\mathbf{A}\mathbf{A}^T))d(\mathbf{A}\mathbf{r}) \end{aligned} \tag{17}$$

where the identity $det(\mathbf{A}\mathbf{A}^T)^{1/2} = det(\mathbf{A})$ has been used. The matrix $\mathbf{A}\mathbf{A}^T$ is a symmetric, positive semi-definite matrix and may therefore be written

$$\frac{1}{\sigma^2}\mathbf{A}\mathbf{A}^T = \mathbf{R}\mathbf{\Sigma}\mathbf{R}^T \tag{18}$$

where $\mathbf{R}$is a rotation matrix and $\mathbf{\Sigma}$ a diagonal matrix with entries $s_1\sigma^2, s_2\sigma^2 ... s_n\sigma^2$ $(s_i \geq 0)$. Thus

$$\int F_1(\mathbf{r})G(\mathbf{r},\sigma^2\mathbf{I})d\mathbf{r} = \int F_2(\mathbf{A}\mathbf{r})G(\mathbf{A}\mathbf{r},\mathbf{R}\mathbf{\Sigma}\mathbf{R}^T)d(\mathbf{A}\mathbf{r}) \tag{19}$$

Again, to show the connections to convolution and filtering, this may be written as

$$F_1 * G(\mathbf{r}, \sigma^2 \mathbf{I}) = F_2 * G(\mathbf{r}_1, \mathbf{R}\mathbf{\Sigma}\mathbf{R}^T) \tag{20}$$

The level contours of the generalized gaussian are ellipsoids rather than spheres. The tilt of the ellipsoid is given by the rotation matrix while its eccentricity is given by the matrix $\mathbf{\Sigma}$, which is actually a function of the scales along each dimension. The equation clearly shows that to recover affine transforms by filtering, one must deform the filter appropriately; a point ignored in previous work [9, 11, 4, 15, 8]. This equation alone does not permit the recovery of the complete affine matrix - only the scales and the tilt. To find the complete affine transform, higher order moments need to be considered. Using higher order moments also permits the use of more overconstrained equations.

3.3 Using other Weighting Functions

Weighting functions other than the gaussian may be used. The weighting function must have certain desirable properties. It should be maximum at the origin and then decay, so that points away from the origin are weighted less. Further, points equidistant from the origin should be weighted symmetrically. It should also be smooth. Functions of the form $f(\mathbf{r}/\sigma)$ are good candidates, although, these are not rotation invariant. If rotation invariance is desired, functions which have the form $f(\mathbf{r}^t\mathbf{r}/\sigma^2)$ may be used. For example one possible weighting function is

$$\begin{aligned} w(\mathbf{r}) &= cos(\mathbf{r}^T\mathbf{r}/\sigma^2)\ for - \pi/2 \leq |\mathbf{r}/\sigma| \leq \pi/2 && (21) \\ &= 0\ otherwise && (22) \end{aligned}$$

Note that this is not normalized (normalization is a desirable property because then weight functions at different scales can be directly compared). Other possibilities include Gabor functions. These will not be investigated further here.

4. Higher Order Moments

The first order moments of F_1 and F_2 are related by

$$\begin{aligned}\int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r}F_1(\mathbf{r})d\mathbf{r} &= \int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r}F_2(\mathbf{Ar})d\mathbf{r} \\ &= \mathbf{A}^{-1}\int_{-\mathbf{Aa}}^{\mathbf{Aa}} \mathbf{Ar}F_2(\mathbf{Ar})d(\mathbf{Ar})detA^{-1} \qquad (23)\end{aligned}$$

or

$$\boldsymbol{\mu_1} = \mathbf{A}^{-1}\boldsymbol{\mu_2} \times detA^{-1} \qquad (24)$$

The second order moments are given by

$$\begin{aligned}\int_{-a}^{a} \mathbf{r}\mathbf{r}^T F_1(\mathbf{r})d\mathbf{r} &= \int_{-\mathbf{a}}^{\mathbf{a}} \mathbf{r}\mathbf{r}^T F_2(\mathbf{Ar})d\mathbf{r} \\ &= \mathbf{A}^{-1}\int_{-\mathbf{Aa}}^{\mathbf{Aa}} \mathbf{Ar}(\mathbf{Ar})^T F(\mathbf{Ar})d(\mathbf{Ar})(\mathbf{A}^{-1})^T detA^{-1} \qquad (25)\end{aligned}$$

and this may be expressed as

$$\boldsymbol{\Gamma_1} = \mathbf{A}^{-1}\boldsymbol{\Gamma_2}(\mathbf{A}^{-1})^T det(\mathbf{A}^{-1}) \qquad (26)$$

Note that the zeroth moment equation is a scalar equation, the first moment a vector one, and the second moment is a matrix equation. As before, the moments are not directly measurable in the form given and need to be weighted. If gaussian weights are used, a simple derivation is available. This uses the fact that the derivatives of gaussians are closely related to the weighted moments using gaussians.

The effect of filtering with derivatives of gaussians (i.e. convolution with derivatives of gaussians) can be obtained by differentiating the gaussian (15)). First write $\mathbf{r}_1 = \mathbf{Ar}$ and then differentiate equation (15) to give

$$F_1(\mathbf{r}) * dG(\mathbf{r}, \sigma^2\mathbf{I})/d\mathbf{r} = \mathbf{A}^T F_2(\mathbf{r}_1) * dG(\mathbf{r}_1, \mathbf{AA}^T\sigma^2)/d\mathbf{r}_1 \qquad (27)$$

where

$$dG(\mathbf{r}, \sigma^2\mathbf{I})/d\mathbf{r} = -\frac{\mathbf{r}}{\sigma^2} G(\mathbf{r}, \sigma^2\mathbf{I}) \tag{28}$$

and

$$dG(\mathbf{r}_1, \mathbf{A}\mathbf{A}^T\sigma^2)/d\mathbf{r}_1 = -(\mathbf{A}\mathbf{A}^T\sigma^2)^{-1}\mathbf{r}_1 G(\mathbf{r}_1, \mathbf{A}\mathbf{A}^T\sigma^2) \tag{29}$$

This equation looks different from the first moment equation (24) because the first derivative of the gaussian has been normalized. Convolving with second derivatives of a gaussian gives

$$F_1(\mathbf{r}) * d^2G(\mathbf{r}, \sigma^2\mathbf{I})/d(\mathbf{r}\mathbf{r}^T) = \mathbf{A}^T F_2(\mathbf{r}_1) * d^2G(\mathbf{r}_1, \mathbf{A}\mathbf{A}^T\sigma^2)/d(\mathbf{r}_1{\mathbf{r}_1}^T)\mathbf{A} \tag{30}$$

where

$$d^2G(\mathbf{r}, \sigma^2\mathbf{I})/d(\mathbf{r}\mathbf{r}^T) = \frac{(\mathbf{r}\mathbf{r}^T)/\sigma^2 - \mathbf{I}}{\sigma^2} G(\mathbf{r}, \sigma^2\mathbf{I}) \tag{31}$$

and

$$d^2G(\mathbf{r}_1, \mathbf{A}\mathbf{A}^T\sigma^2)/d(\mathbf{r}_1{\mathbf{r}_1}^T) = [(\mathbf{A}^T\mathbf{A}\sigma^2)^{-1}\mathbf{r}_1\mathbf{r}_1^T(\mathbf{A}^T\mathbf{A}\sigma^2)^{-1} - (\mathbf{A}\mathbf{A}^T)^{-1}]G(\mathbf{r}_1, \mathbf{A}\mathbf{A}^T\sigma^2) \tag{32}$$

Equations (30) and (26) are seen to be closely related; the differences are the additional term due to the gaussian in equation (30) and due to normalization.

Since convolutions with gaussians and derivatives of gaussians are so closely related to the original weighted moment equations, they will often be referred to as moments in the rest of the paper.

4.1 Moments in Other Domains

Identical moment equations can be written using the auto-correlation of the functions F_1 and F_2 instead of the functions themselves; the auto-correlation of a function F is defined as

$$L(\mathbf{r}) = \int_{-\infty}^{\infty} F(\mathbf{r}_1)F(\mathbf{r} + \mathbf{r}_1)d\mathbf{r}_1 \tag{33}$$

The auto-correlation is radially symmetric. This implies that odd moments of the auto-correlation function are zero, while all even moments exist even for functions F which are odd. The usefulness of these symmetry properties is discussed in the next section.

Equations for the moments can also be written using the power spectral density. They look exactly like equations (6,24,26) except that they lack the determinant term.

4.2 The problem of even and odd functions F

If the value of the moments is zero, (or near zero in practice), the moment equations are ill-conditioned and cannot be solved. This can occur in two ways; either the signal strength is too low (i.e. the magnitude of F is small) or the function F is purely even or purely odd causing some of the moment equations to be zero. There is little that can be done in the first case. The latter case, however, provides insight into the number of moment equations required to solve for the affine parameters.

Consider first the 1-D case. The even moments of any odd function will be zero while the odd moments of any even function are zero (this is a standard result and can be proved easily). Since the zeroth moment is even, and the first moment is odd and only one parameter (the scale) needs to be determined, these two equations suffice to find it.

The situation is a little more complicated in higher dimensions. One way of stating the problem is to consider each dimension separately. Then if a function is odd along any dimension, its contribution to the even moment from that dimension will be zero and hence inferences along that dimension cannot be made. Similarly, if a function is even along any dimension, its contribution is zero to the odd moments along that dimension. First note that this is not a very common occurence. A function cannot be purely even or odd at every point at which it is defined. A good example is a sine-wave sin(x), which is odd at x $= 0$, and even when x $= \pi/2$ and has both even and odd components at other points In practice, one must also consider situations where the function is mostly even or mostly odd.

How many moments are required in 2-D to solve for the affine transform ? In general four affine parameters need to be determined. Straightforward equation counting seems to show that there are four even equations (1 from the zeroth moment + 3 from the second moment), and there are six odd equations (2 from the first moment + 4 if the third moment is used). Thus even if the function is purely even or odd, moments up to third order suffice to solve for the affine parameters.

The third moment is not really required, since the previous analysis ignores the information available from the deformation of a gaussian. Consider the zeroth moment when an arbitrary affine transform $\mathbf{A}$needs to be measured. In this case, the zeroth moment may be used to find the matrix $\mathbf{A}\mathbf{A}^T$ (i.e. 3 parameters may be computed). Accounting for the additional information available from the deformation of the gaussian, there are at least 6 even equations (3 from the zeroth moment + at least 3 from the second moment) and at least 4 odd equations (from the first moment).

The function F may also be transformed so that some of the moments are always non-zero. For example, if instead of the function F, the magnitude of its auto-correlation is used, the zeroth and the second moment are always non-zero. This follows from the radial symmetry of the auto-correlation function - which implies that the odd moments are all zero while the even moments are non-zero.

A different transformation uses certain algebraic tricks to convert any function to an odd or an even function thus ensuring that every moment equation is well-defined. For example, consider the 1-D case again. Every function F can be written as the sum of an even part EF and an odd part OF. The odd part OF can be converted into an odd function by taking its magnitude. The even part EF can be converted to an odd function by flipping one half of the function. The problem is somewhat more complicated in higher dimensions.

The 2-D case will be dealt with in the solution section.

5. Solving the Moment Equations

In the remainder of this paper, only the case where the affine transform $\mathbf{A}$= s $\mathbf{R}$(i.e. a scale change and a rotation) will be considered (see section 3.1); The general affine transform will be considered in a later paper. This case arises in a number of different practical situations. For example, when a robot's motion is mostly translational with small rotational components and the structures in the image are shallow (i.e. have small extent in depth compared to their actual depth), the deformation can be adequately explained by a scale change and a rotation [13]. The scale change is then interpreted as the depth of the corresponding structure.

The zeroth moment equation will be dealt with first followed by the first moment equation.

5.1 Solving the Zeroth Moment Equation

When the affine transform is described by $\mathbf{A}$= s $\mathbf{R}$, equation (14) can be written as

$$F_1(\mathbf{r}) * G(\mathbf{r}, \sigma^2) = F_2(\mathbf{r}_1) * G(\mathbf{r}_1, (\sigma^*)^2) \tag{34}$$

where $\sigma^* = s\sigma$. The important point here is that the rotation matrix does not figure in σ^*. The problem of finding the scale parameter has therefore been converted into the problem of finding the value of σ^*. Older methods [9, 11, 4, 15, 8]. have instead concentrated on the much more difficult problem of trying to correspond the functions F_1 and F_2. The present formulation is simpler because the problem of correpondence has been reduced to finding the appropriate affine parameters of a known function - the gaussian. Additional simplicity derives from the fact that the affine transform can be analytically interpolated.

The equation can be solved by sampling the space of possible values of σ^*, filtering for each sampled value and declaring the solution to be that value of σ^* for which the above equation has smallest residual error according to some norm. A more elegant approach uses the fact that the affine transform can be analytically interpolated. The idea is to sample over a small set of σ^* and then interpolate using a Taylor series approximation. Consider first a given σ^*. The Taylor series approximation to first order gives

$$\begin{aligned} G(\mathbf{r}_1, (s\sigma)^2) &\approx G(\mathbf{r}_1, \sigma^2) + \alpha\sigma \frac{\partial G(\mathbf{r}_1, \sigma^2)}{\partial \sigma} && (35) \\ &= G(\mathbf{r}_1, \sigma^2) + \alpha\sigma^2 \nabla^2 G(\mathbf{r}_1, \sigma^2) && (36) \\ & && (37) \end{aligned}$$

where $s = 1 + \alpha$. The last equality follows from the diffusion equation $\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$. This allows the convolution (34) to be written as

$$F_1 * G(\mathbf{r}, \sigma^2) \approx F_2 * G(\mathbf{r}_1, \sigma^2) + \alpha\sigma^2 F_2 * \nabla^2 G(\mathbf{r}_1, \sigma^2) \tag{38}$$

The above equation is linear in s. To find s, three filtering operations need to be performed, two gaussian filtering operations and one laplacian operation. Note that the above equation expresses the well-known result that a laplacian can be approximated by a difference of gaussians.

5.1.1 Issues of Scale

Information in an image is scale dependent. There may be information present at several different scales or at only one of them. A method which does not take this into account is not likely to be robust. Thus it is desirable to solve the above equation at several different scales (σ's). Let a set of σ_i's be chosen. For each such σ_i an equation of the form (38) may be written giving the following system of equations

$$F_1 * G(\mathbf{r}, \sigma_0^2) \approx F_2 * G(\mathbf{r}_1, \sigma_0^2) + \alpha\sigma_0^2 F_2 * \nabla^2 G(\mathbf{r}_1, \sigma_0^2)$$

$$\begin{aligned}
F_1 * G(\mathbf{r},\sigma_1^2) &\approx F_2 * G(\mathbf{r}_1,\sigma_1^2) + \alpha\sigma_1^2 F_2 * \nabla^2 G(\mathbf{r}_1,\sigma_1^2) \\
\ldots\ldots &\approx \ldots\ldots \\
F_1 * G(\mathbf{r},\sigma_i^2) &\approx F_2 * G(\mathbf{r}_1,\sigma_i^2) + \alpha\sigma_i^2 F_2 * \nabla^2 G(\mathbf{r}_1,\sigma_i^2) \\
\ldots\ldots &\approx \ldots\ldots \\
F_1 * G(\mathbf{r},\sigma_l^2) &\approx F_2 * G(\mathbf{r}_1,\sigma_l^2) + \alpha\sigma_l^2 F_2 * \nabla^2 G(\mathbf{r}_1,\sigma_l^2) \qquad (39)
\end{aligned}$$

This is an overconstrained set of equations in the unknown α. The redundancy offered by the over-constrained problem also makes it more robust with respect to noise.

The particular choice of $\sigma_i's$ is to some extent arbitrary although some general criteria may be specified. Too small a σ_i will make the system sensitive to noise while localization requires that σ_i not be too large. The actual values are not very crucial. In practice, a set of eight different $\sigma's$ were chosen. They were all spaced apart by half an octave (a factor of 1.4). The filter width $= 8\ \sigma$ (since the filters need to range from -4σ to 4σ). The widths actually chosen were (3,5,7,10,14,20,28,40) (see also Jones and Malik [7]).

The above system of equations was cast into the following linear least squares problem

$$\mathbf{\Sigma}_i \| F_1 * G(\mathbf{r},\sigma_i^2) - F_2 * G(\mathbf{r}_1,\sigma_i^2) + \alpha\sigma_i^2 F_2 * \nabla^2 G(\mathbf{r}_1,\sigma_i^2) \|^2 \qquad (40)$$

and was solved using Singular Value Decomposition (SVD). It was found that the lowest filters (widths = 3,5,7) were noisy and hence they were disregarded (one reason may be that the laplacian is noisy when the filter size is small). The scale was recovered fairly accurately using the other widths (see the experimental section for details). This set of filter widths worked better than another one where 8 filters were used with their $\sigma's$ spaced apart by a factor of 1.2; presumably because with a larger variation in scale, there is more information available at multiple scales.

5.1.2 Choosing a Different Operating Point

For large $\sigma's$ (say $\sigma's \geq 1.3$) the recovered scale sometimes tends to be poor. This is because the Taylor series approximation is good only for a small change in σ. The problem arises because in equation (38) the right-hand side is expanded around the same σ as on the left-hand side. A better approximation is obtained by expanding σ^* as close to the correct scale as possible. An example should clarify this point. Assume that the left-hand side uses $\sigma = \sigma_0$ and that the scale change s is 1.3, then it is better to expand the right-hand side around $\sigma_1 = 1.4\sigma_0$ (i.e. the half-octave step closest to the actual scale) rather than expanding at σ_0. In this case, equation (38) may therefore be modified to

$$F_1 * G(\mathbf{r}, \sigma_i^2) \approx F_2 * G(\mathbf{r}_1, \sigma_{i+1}^2) + \alpha' \sigma_{i+1}^2 F_2 * \nabla^2 G(\mathbf{r}_1, \sigma_{i+1}^2) \tag{41}$$

where $s = 1.4(1 + \alpha')$. Since filtering by a set of $\sigma's$ is already being performed, (for the overconstrained system), no additional filtering operations are required. Again, an overconstrained system may be implemented easily. For each value of σ_i on the left-hand side of equation (41), expand around σ_{i+1} on the right-hand side.

A similar scheme may be implemented if the scale s ≤ 0.8 (by expanding around a σ which is smaller by a factor of 1.4 (or $0.70 \approx 1/1.4$).

Apriori the σ around which the expansion should be done is not known since the value of the scale is not available. The solution is to expand around all three of them (i.e $\sigma, 1.4\sigma$ and $\sigma/1.4$) and then pick the correct answer to be the one which gives the closest σ). Again an example will clarify this point. Assume that the correct scale is again 1.3 and that the three different operating points return the following values of s (1.25, 1.32, 1.18). Consistency decides the correct answer here. 1.18 is inconsistent with expanding around $\sigma/1.4$ and can be rejected. The other answers are both between 1 and 1.4 and closer to 1.4. Therefore, the appropriate operating point to pick is $1.4\sigma_{i+1}$. Experimentally, this method seems to work fine (an alternative is to compare the residual error after SVD minimization; this does not seem to work as well, partly because the different errors are not really comparable - they have different numbers of equations). Another technique that has been tried is to make all the equations into a single overconstrained system and solve it using SVD - based on the answer obtained, some of the equations may be dropped and the system resolved.

In principle the same technique can be used to expand around other operating points σ if the scale gets even larger or much smaller. The range of scales to be expected depends on the application. For structure for motion, a scale change of more than 1.4 almost never happens in practice. In finding shape from texture, in principle any scale change can occur. If the surface is smooth, it is expected that there is likely to be a neighbouring texture patch whose scale change is less than 2.5. Thus in practice these can probably be limited to a factor of about 2.5 (in which case one should also expand around 2.0σ and 0.5σ in addition to expanding around $\sigma, 1.4\sigma$ and $1/1.4\sigma$). Very high scale changes are probably difficult to measure in any case because of the extreme foreshortening that this implies.

There are other methods which try to linearize the function F or the gaussian with respect to $\mathbf{r}$. They are therefore valid only when the affine transform is small. The method presented here *linearizes with respect to* σ *and not* $\mathbf{r}$ and provided it is expanded around the appropriate σ, the affine transform does not need to be small.

5.1.3 Ensuring that F_i is always even

The method does not work if the output of the gaussian convolution is zero (or close to zero in practice). This can happen either if the signal is weak (little can be done in this case), or if the signal shows odd symmetry along any dimension.

The 1-D case was dealt with in section 4.2. Here it is shown how a function in 2-D may always be converted into an even function. One cannot consider each dimension separately for this would destroy the rotational symmetry of the gaussian. Instead, the function is decomposed into parts which are radially even $E_r F_i$ and radially odd $O_r F_i$ where

$$E_r F_i = F_i(\mathbf{r}) + F_i(-\mathbf{r}) \tag{42}$$

$$O_r F_i = F_i(\mathbf{r}) - F_i(-\mathbf{r}) \tag{43}$$

The magnitude of both functions ($\mid E_r F_i \mid \mid O_r F_i \mid$) is then taken. The resulting functions are both even and the gaussian convolution is non-zero for both. The SVD is performed on each set of these functions separately and the one with the lower error is then used (this ensures that if either the even or odd components is really small, it is ignored.

5.1.4 Finding Image Translation

Before scale can be recovered, the two patches must be aligned by finding the image translation. These can be found either using some traditional optical flow or displacement schemes [2]. Alternatively, the residual of the SVD error can be used to localize the image translation to ± 0.5 pixels. This is done in the following manner, the first image patch is filtered with the set of gaussians The second patch is filtered with gaussians at every pixel in a small window centered at the first patch's location and the SVD computed. That pixel for which the SVD residual is minimized is declared to be the correct image translation. Experimentally, this method was found to work satisfactorily. Note that in general no additional filtering operations are required since the filtering operations are done at every point in the image anyway.

5.2 Experimental Results

Experiments were carried out both on synthetic images as well as a pair of real images. The first synthetic image (fig:1) shows a cosine wave generated by the equation $F_1(x, y) = 127cos(\omega_y y + \phi_y)$. The cosine was picked for the following interesting properties.

Table 1:

Actual	No Noise			Gaussian Noise $\sigma = 10$			Uniform Noise (-10,10)		
	scale s	$\frac{\delta s}{s} * 100$	$\frac{\delta s}{s-1} * 100$	scale s	$\frac{\delta s}{s} * 100$	$\frac{\delta s}{s-1} * 100$	scale s	$\frac{\delta s}{s} * 100$	$\frac{\delta s}{s-1} * 100$
1.05	1.050	0	0	1.040	1.0	20	1.040	1.0	20
1.10	1.101	0.09	0.01	1.082	1.6	18	1.098	0.2	2.0
1.15	1.158	0.7	5.3	1.152	0.1	1.3	1.148	0.1	1.3
1.20	1.213	1.1	6.5	1.198	0.2	1.0	1.192	0.7	4.0
1.40	1.408	0.6	2.0	1.415	1.1	3.8	1.427	1.9	6.8
1.60	1.639	2.4	6.5	1.630	1.9	5.0	1.591	0.6	1.5
1.80	1.855	3.1	6.9	1.838	2.1	4.8	1.816	0.9	2.0

The function can be made even or odd at any point depending on the value of ϕ_y. Further, there is no information along the y direction (the so-called aperture problem). Inspite of that the scale can be recovered.

For the first experiment, ϕ_y was chosen to be zero,so that the function was even. $\omega_y = 0.2$ was chosen. A second cosine function was generated using the following function $F_2(x, y) = 127cos(s\omega_y x + \phi_y)$ (fig:2) F_2 is rotated 90 deg. with respect to F_1 and also scaled by the factor s. For various values of s, the scale was recovered using the zeroth moment. The results are tabulated in table 1. The experiment was repeated with noise added. First, uniform noise ranging from -10 to 10 was added to F_2. Second, gaussian noise with a standard deviation of 10 was added to F_2. These results are also tabulated in table 1. Two operating points were used: σ and 1.4σ. The appropriate operating point was picked as discussed in the text.

Table 1 is to be read as follows. The first column in table 1 is the actual scale while column 2 shows the recovered scale in the noise-free case. Two different percentage errors are tabulated and they arise from the following considerations. Assume that an object is at a depth of z_0 and after a translation T_z in the z direction, its new depth is $z_1 = z_0 + T_z$. Then the percentage error in finding the quantity z_1/z_0 is given by $\frac{\delta s}{s} * 100$ and this is tabulated in column 3 for the noise-free case. On the other hand, the percentage error in finding the quantity T_z/z_0 is given by $\frac{\delta s}{s-1} * 100$) and this is tabulated for the noise-free case in column 4. Which of these quantities is more important? Since the depth z_0 is apriori unknown, the quantity of relevance at least in the motion case is T_z/z_0 and the corresponding percentage error is more significant. Similar values are tabulated when gaussian noise (columns 5,6 and 7) and uniform noise (columns 8,9, and 10) are added .

The results are excellent in the noise-free case. The percentage errors in column 3 are all less than about 3% while even in column 4 the percentage errors do not exceed 7% (the average errors are of course much lower). Note that the method recovers scale accurately inspite of the large rotation.

With noise added, the results are as good except for the lowest scales (1.05 and 1.10). One possible method of improving the results for the lower scales is to use filters separated by ratios smaller than 1.4.

The experiment was repeated using $\phi_y = \pi/2$ for both images. The method failed because the function now becomes an odd function at the origin and thus the result of gaussian filtering it is zero. However, if the function is transformed into an even function using the methods discussed in the text, the zeroth moment can once again be applied and the results are similar.

The experiment was repeated with random dot images. A random dot image of size 64 by 64 was generated (fig:3). The image was then affine transformed and smoothed using a cubic interpolation scheme (fig:4). For various values of the scale factor s, the scale was recovered using the zeroth moment method. The results are tabulated in table 2. The highest error in column 4 is less than 9% if the smallest scale (1.05) is ignored. Again the average error is much lower. The error is somewhat larger in this case because the program which affine transforms the image does interpolation which tends to destroy image structure. This is more serious at the lower scales.

Table 2:

Actual	Recovered Scale s	$\frac{\delta s}{s} * 100$	$\frac{\delta s}{s-1} * 100$
1.05	1.059	0.9	18
1.10	1.104	0.4	4.0
1.15	1.138	0.8	8.0
1.20	1.180	1.7	10
1.40	1.398	0.1	0.5
1.60	1.569	1.99	5.2
1.80	1.722	4.3	9.8

Finally the algorithm was tested on a pair of real images taken from a sequence [13]. The images were taken with a Sony ccd camera using a robot moving straight ahead. The robot moved about 1.4 ft between frames. Since the original images were taken with the intent of using a line based algorithm, most of the objects have little intensity variation in their interior. However, the posters on the back wall show some intensity variation on the same plane and can therefore be used. Points 1 and 2 were picked by hand in the first image (fig:5). The corresponding points in the second image (fig:6) were determined using the SVD residual error. For point 1, the recovered scale was 1.07 which corresponds to a distance of 1.4/(1.07 - 1) = 20 ft. For point 2 the recovered scale was 1.06 which corresponds to a distance of 1.4/(1.065 - 1) = 21.5 ft. The measured distance to the back wall is 20.3 ft. The accuracy is thus within 6%.

5.3 Solving the First Moment Equation

Again for the case where the affine transform is described by **A**= s **R**, the first moment equation (27) may be written

$$F_1(\mathbf{r}) * dG(\mathbf{r}, \sigma^2)/d\mathbf{r} = s\mathbf{R}F_2(\mathbf{r}_1) * dG(\mathbf{r}_1, (s\sigma)^2)/d\mathbf{r}_1 \tag{44}$$

The diffusion equation applied to the derivatives of the gaussian gives the following identity

$$\partial G_{r_j}/\partial\sigma = \nabla^2 G_{r_j} \tag{45}$$

where G_{r_j} denotes the derivative of the gaussian with respect to the r_j^{th} coordinate. Using this identity, the right-hand side of (44) is expanded around σ and rewritten as

$$F_1(\mathbf{r}) * G_{r_j}(\mathbf{r}, \sigma^2) \approx s\mathbf{R}_j F_2(\mathbf{r}_1) * [G_{r_j}(\mathbf{r}_1, \sigma^2) + \alpha\sigma^2\nabla^2 G_{r_j}(\mathbf{r}_1, \sigma^2)] \tag{46}$$

where $\mathbf{R}_j$ is the j^{th} row of $\mathbf{R}$and $s = 1 + \alpha$. Note that there are 2 such equations. This may be more conveniently written as

$$\mu_1(\sigma) = (1 + \alpha)\mathbf{R}[\mu_2(\sigma) + \alpha\sigma^2\xi(\sigma)] \tag{47}$$

where

$$\xi(\sigma) = F_2(\mathbf{r}_1) * d(\nabla^2 G(\mathbf{r}_1, \sigma^2))/d\mathbf{r}_1 \tag{48}$$

and

$$\mu_\mathbf{l}(\sigma) = F_l(\mathbf{r}) * dG(\mathbf{r}_l, \sigma^2)/d\mathbf{r}_l \tag{49}$$

The rotation matrix can be eliminated by taking the dot-product (i.e. the magnitude) of both sides of equation (47) and equating them. This gives

$$\boldsymbol{\mu_1}^T\boldsymbol{\mu_1} = (1 + \alpha)^2[\boldsymbol{\mu_2}^T\boldsymbol{\mu_2} + 2\alpha\sigma^2\boldsymbol{\mu_2}^T\xi + \alpha^2\sigma^4\xi^T\xi] \tag{50}$$

This is a polynomial equation in the unknown α. As before several different scales σ_i are used to give an overconstrained system. The resulting system can be solved using the Gauss-Newton technique [12]. The Gauss-Newton procedure works by linearizing the system around the current estimate of the solution reducing the problem to a linear least-squares problem. Define a vector function $\mathbf{c}(\alpha)$ where the i^{th} component is given by

$$c_i(\alpha) = \boldsymbol{\mu_{1i}}^T\boldsymbol{\mu_{1i}} - (1 + \alpha)^2[\boldsymbol{\mu_{2i}}^T\boldsymbol{\mu_{2i}} + 2\alpha\sigma_i^2\boldsymbol{\mu_{2i}}^T\boldsymbol{\xi_i} + \alpha^2\sigma_i^4\boldsymbol{\xi_i}^T\boldsymbol{\xi_i}] \tag{51}$$

Then if α_k is the current estimate of α, then $\alpha_{k+1} = \alpha_k + p_k$ where p_k is the solution of the linear least squares problem

$$||\mathbf{J}_k p + \mathbf{c}_k||_2^2 \tag{52}$$

where a quantity subscripted by k denotes that quantity evaluated at α_k and $\mathbf{J}(\alpha)$ is the Jacobian matrix of $\mathbf{c}(\alpha)$.

The least squares problem was solved using SVD and convergence was found to be rapid - within a couple of iterations. The method was tested on a sine-wave pattern.

In two dimensions, the rotation may be computed in the following manner. Consider equation (47) again. This may be rewritten as

$$\mu_1(\sigma) = \mathbf{R}\mathbf{b} \tag{53}$$

where

$$\mathbf{b} = (1+\alpha)[\mu_2(\sigma) + \alpha\sigma^2\xi(\sigma)] \tag{54}$$

is a known quantity (since α is now known). Let $\mathbf{b} = (b_1, b_2)^T$. Then equation (53) can be transformed into the following form

$$\mu_1(\sigma) = \mathbf{B}\omega \tag{55}$$

where

$$\mathbf{B} = \begin{vmatrix} b_1 & b_2 \\ b_2 & -b_1 \end{vmatrix} \tag{56}$$

and $\omega = (cos\theta, sin\theta)$ and θ is the rotation angle. Using the identities $cos\theta = \sqrt{1 - sin^2\theta}$, and $\mathbf{B}^{-1} = -\mathbf{B}/(det\mathbf{B})$ equation (55) can be transformed into the following pair of equations each linear in the unknown $sin\theta$.

$$\sqrt{1 - [(b_1\epsilon_1 + b_2\epsilon_2)/(det\mathbf{B})]^2} = sin\theta \tag{57}$$

$$(-b_2\epsilon_1 + b_1\epsilon_2)(det\mathbf{B}) = sin\theta \tag{58}$$

where $\mu_1 = (\epsilon_1, \epsilon_2)^T$. Such pairs of equations can be written for every σ_i and the resulting linear system of overconstrained equations can be solved using SVD for the rotation angle θ.

6. Future Extensions

Future work includes the solution for the case of the general affine transform as well as the use of the second moment equation. Other possibilities include the automation of the process over the entire image and the detection of occlusions. Finally, the use of the affine transform to find surface orientation from both texture and motion cues will be explored.

Acknowledgements We wish to thank Al Hanson for his useful comments on early drafts of this paper. The first author also wishes to thank Harpreet Sawhney for many fruitful discussions and for his constant encouragement.

[1] G. Adiv. Determining 3D motion and structure from optical flows generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[3] L. Brown and H. Shyvaster. Surface orientation from projective foreshortening of isotropic texture autocorrelation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):584–588, 1990.

[4] M. Campani and A. Verri. Motion analysis from optical flow. *Computer Vision Graphics and Image Processing:Image Understanding*, 56(12):90–107, 1992.

[5] R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *Proc. 2nd European Conference on Computer Vision*, pages 187–202, 1992.

[6] J. Garding. *Shape from Surface Markings.* PhD thesis, KTH, Stockholm, 1990.

[7] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. 2nd European Conference on Computer Vision*, pages 395–410, 1992.

[8] D. G. Jones and J. Malik. Determining three-dimensional shape from orientation and spatial frequency disparities. In *Proc. 2nd European Conference on Computer Vision*, pages 661–669, 1992.

[9] K. J. Hanna J.R. Bergen, P. Anandan and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. 2nd European Conference on Computer Vision*, pages 237–252, 1992.

[10] T. Kanade and J. R. Kender. Mapping image properties into shape constraints: Skewed symmetry, affine–transformable patterns, and the shape–from–texture paradigm. In J. Beck et al, editor, *Human and Machine Vision*, pages 237–257. Academic Press, NY, 1983.

[11] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[12] W. Murray P. E. Gill and M. H. Wright. *Practical Optimization.* Academic Press, 1981.

[13] H. S. Sawhney and A. R. Hanson. Identification and 3D description of 'shallow' environmental structure in a sequence of images. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 179–186, 1991.

[14] B. J. Super and A.C. Bovik. Solution to shape-from-texture by wavelet-based measurement of local spectral moments. In *Proc. Computer Vision and Pattern Recognition Conference*, 1992.

[15] P. Werkhoven and J. J. Koenderinck. Extraction of motion parallax structure in the visual system 1. *Biological Cybernetics*, 1990.

[16] L. R. Williams and A. R. Hanson. Translating optical flow into token matches and depth from looming. In *Proc. 2nd Intl. Conf. on Computer Vision*, pages 441–448, 1988.

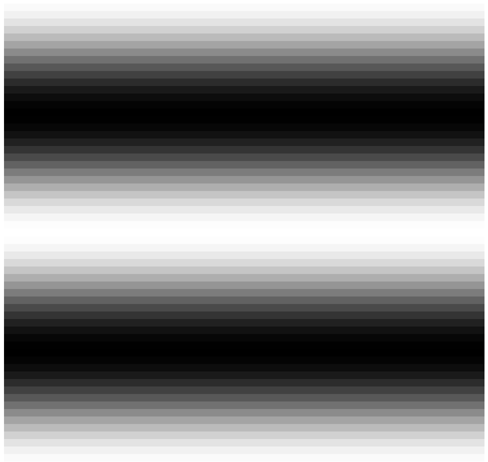

Figure 1: Cosine Image

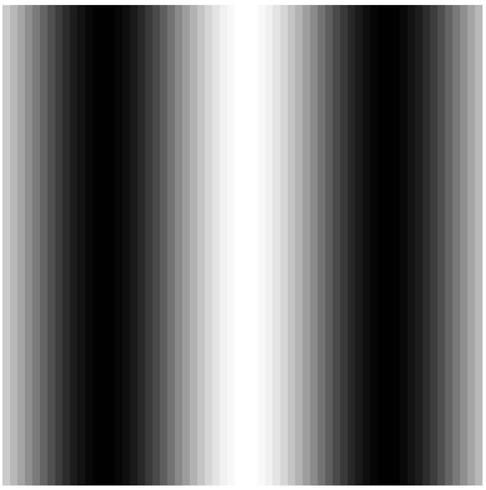

Figure 2: Scaled and Rotated Cosine Image

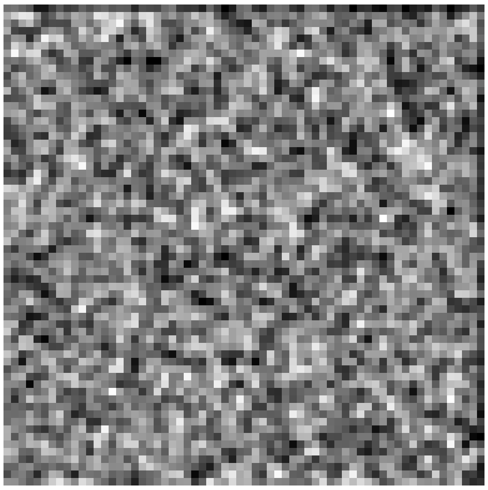

Figure 3: Random Dot Image

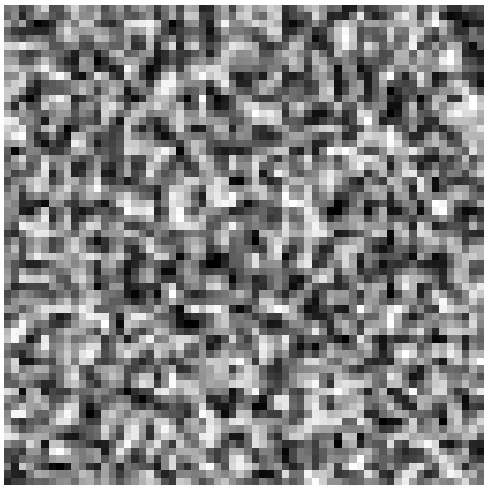

Figure 4: Scaled Version of Random Dot Image

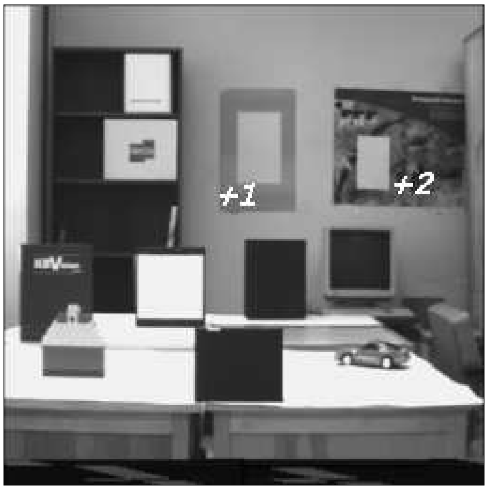

Figure 5: Real Image 1

Figure 6: Real Image 2