M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 278
Submitted to ICCV95

'Video orbits': characterizing the coordinate transformation between two images using the projective group *

S. Mann and R. W. Picard
MIT Media Lab; 20 Ames Street; Cambridge, MA 02139
steve@media.mit.edu, picard@media.mit.edu

Abstract

Many applications in computer vision benefit from accurate, robust analysis of the coordinate transformation between two frames. Whether for image mosaicing, camera motion description, video stabilization, image enhancement, aligning digital photographs for modification (e.g. *ad-insertment*), or their comparison during retrieval, finding both an estimate of the coordinate transformation between two images, and the error in this estimate is important.

Perhaps the most frequently used coordinate transformation is based on the 6-parameter *affine* model; it is simple to implement and captures camera translation, zoom, and rotation. Higher order models, such as the 8-parameter *bilinear*, 8-parameter *pseudo-perspective*, or 12-parameter 'biquadratic', have also been proposed to approximately capture the two extra degrees of freedom that a camera has (pan, tilt) that are not captured by the affine model. However, none of these models exactly captures the eight parameters of camera motion. The desired parameters are those of elements in the *projective group*, which map the values at location $\mathbf{x}$ to those at location $\mathbf{x}' = (\mathbf{A}\mathbf{x}+\mathbf{b})/(\mathbf{c}^T\mathbf{x}+1)$, where the numerator contains the six affine parameters, and the denominator contains the two additional pan-tilt or "chirp" parameters, $\mathbf{c}$.

This paper presents a new method to estimate these eight parameters from two images. The method works without feature correspondences, and without the huge computation demanded by direct nonlinear optimization algorithms. The method yields the "exact" eight parameters for the two no-parallax cases: 1) a rigid planar patch, with arbitrary 3D camera translation, rotation, pan, tilt, and zoom; and 2) an arbitrary 3D scene, with arbitrary camera rotation, pan, tilt, and zoom about a fixed center of projection.

We demonstrate the proposed method on real image pairs and discuss new applications for facilitating logging and browsing of video databases.

Contents

1 Introduction

Many computer vision problems require finding the coordinate transformation between two images of the same scene or object. Whether trying to relate photographs taken from two different cameras, trying to align images for mosaicing and enhancement, trying to recover camera motion between video frames, or trying to stabilize video images, it is important to have both a precise description of the coordinate transformation between any given pair of images or video frames, and some indication as to how accurately this coordinate transformation accounts for the differences in the two images.

Much of the research to recover this coordinate transformation has been conducted within the framework of motion analysis of a static scene where the camera is assumed to have moved slightly between two frames. However, a solution to this "motion analysis" problem has broader applications, since the same class of coordinate transformation also exists between any two still pictures taken of the same scene using different cameras. Solutions to this coordinate transformation can therefore be used not just for motion, but also for applications such as recognition of scenes photographed from different angles, alignment and mosaicing of images taken with different cameras, or computation of depth with respect to a planar surface in a 3D scene.

1.1 Camera motion analysis: common assumptions and terminology

Two common assumptions are typically adhered to in solving for the coordinate transformation between two images. The first assumption is that the scene is constant – changes of scene content and lighting are small during the time that elapses between successive frames of the image sequence. The second assumption is that of an ideal pinhole camera[1] which has eight degrees of freedom in 3D space – translation (X, Y, Z), zoom (scale in each of the image coordinates x and y), and rotation (rotation about the optical axis, pan, and tilt. These two assumptions are also made in this paper.

The diverse communities addressing camera motion analysis use a variety of terminology. In this paper, an "uncalibrated camera" refers to one in which the principal point[2] is not necessarily at the center (origin) of the image and the scale is not necessarily isotropic[3]. We also suppose that the zoom is continually adjustable by the camera user, and that we do not know the zoom setting, or whether it changed between recording frames of the image sequence. We assume that all cameras (calibrated or not) map straight lines in 3D space to straight lines in the 2D image, and that each element in the camera sensor array returns a quantity that is linearly proportional to

*This work sponsored in part by Hewlett-Packard Research Labs and by BT, PLC.

[1] That is, we assume unlimited depth of field – everything is in focus (infinite resolution); and straight lines map to straight lines.

[2] The principal point is where the optical axis intersects the film.

[3] Isotropic scale means that magnification in the x direction is equal to magnification in the y direction.

the quantity of light received[4].

The concept of *parallax* is also relevant to this paper. To briefly review, when imaging a flat scene (like a whiteboard, or when the camera is far from the scene relative to the depth of the scene, as in aerial photography) then regardless of how the camera moves, there is no parallax (and consequently, no apparent occlusion). However, when imaging an arbitrary 3D scene, translation of the camera causes parallax[5]. The cases which are exactly solved in this paper do not include parallax, but their accurate solution is an important first step in compensating for parallax [3].

1.2 Coordinate transformations: terminology

Terms like *affine*, *perspective*, and *projective* have become confused as they are used differently by people from different backgrounds. For example, in the field of projective geometry [4], *perspective* is often used to imply a mapping to a lower dimension (e.g. 3D to 2D), while *projective* implies a mapping to the same dimension (e.g. 2D to 2D). Many others, however, use the term *perspective* to denote a projective mapping from 2D to 2D (e.g. Wolberg's [5] *perspective mapping*, which is so commonly used that it is incorporated into many software applications packages, such as Adobe Photoshop.) The projective coordinate transformation, for example, could also be described as *affine* or *Euclidean* in 3D, or linear in 4D (using homogeneous coordinates, writing $\mathbf{x}' = \mathbf{A}\mathbf{x}, \ \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^4$).

We find that adopting the point of view of the image coordinates, $\mathbf{x} = [x, y]^T$ simplifies matters; in this paper we describe coordinate transformations from $\mathbf{x}$ to a new set of coordinates, $\mathbf{x}' = [x', y']^T$. The coordinate transformations used in this paper are presented in Table 1.

An example of each of these coordinate transformations is shown in Fig. 1.

The 6-parameter *affine* model is frequently used as a simple description of the coordinate transformation between two pictures of the same scene, but it has too few degrees of freedom to account for the eight degrees of freedom of the desired exact *projective* coordinate transformation. However, because the desired projective parameters have traditionally been too mathematically and computationally intractable to estimate, a variety of approximations have been proposed. Later we will construct a linear system of equations involving an approximation to the projective coordinate transformation (although this will only be an intermediate step for solving for the exact projective parameters.) It is helpful to discuss a few of the commonly used approximate models before proceeding.

Going from first order (affine), to second order, gives the 12-parameter 'biquadratic' model. This model captures both the "chirping" and converging effects of the projective group, though despite its larger number of parameters, there is still considerable discrepancy between a projective coordinate transformation and the best-fit biquadratic coordinate transformation. Why stop at 2nd order? Why not use a 20-parameter 'triquadratic' model? While an increase in the number of model parameters will result in a better fit, there is a tradeoff, where the model begins to fit "noise." Since the physical camera model fits into the 8-parameter projective group, we know that "eight is enough." Hence, it is appealing to find an approximate model with only eight parameters. Let's briefly show how the bilinear and pseudo-perspective models can be obtained from the biquadratic model[6]

The bilinear model is perhaps the most widely-used [5] in the fields of image processing, medical imaging, remote sensing, and computer graphics. This model is easily obtained from the biquadratic by removing the four x^2 and y^2 terms. Although the resulting bilinear model captures the effect of converging lines, it completely fails to capture the effect of "chirping".

The 8-parameter *pseudo-perspective* model [6]) does, in fact, capture both the converging lines and the "chirping" of a projective coordinate transformation. This model may be thought of as first, removal of two of the quadratic terms ($q_{x'y^2} = q_{y'x^2} = 0$), which results in a ten parameter model (the 'q-chirp' of Mann [7]) and then constraining the four remaining quadratic parameters to have two degrees of freedom. This model may be thought of as first, removal of two of the quadratic terms ($q_{x'y^2} = q_{y'x^2} = 0$), and second, constraining the four remaining quadratic terms to have two degrees of freedom.

These constraints force the "chirping effect" (captured by $q_{x'x^2}$ and $q_{y'y^2}$) and the "converging effect" (captured by $q_{x'xy}$ and $q_{y'xy}$) to work together in the "right" way to match, as closely as possible, the effect of a projective coordinate transformation. By setting $q_\alpha = q_{x'x^2} = q_{y'xy}$, we force a chirping in the x-direction to correspond with the converging of parallel lines in the x-direction. A similar result holds for the y-direction. Therefore, of the 8-parameter approximations to true projective, we would expect the *pseudo-perspective* model to perform the best, and, indeed, this appears to follow from experiments.

1.3 Camera motions and the projective group

This paper will show a new way to get the parameters for the "exact" projective model. This model exactly captures the coordinate transformation resulting in camera motion for two interesting "no-parallax" cases outlined in Table 1.3

The second case is sometimes referred to as "computing the motion of a rigid planar patch." This paper is not the first to provide a means of calculating the motion of a rigid planar patch; Tsai and Huang [8] solved this problem when feature correspondences were available and stressed the significance of group theory. Faugeras and Lustman [9], Shashua and Navab [10], and Sawhney [11] have considered the problem of computing the motion of a rigid planar patch, as part of a larger problem of finding 3D motion and structure using parallax relative to an arbitrary plane in the scene. Kumar *et al.* [3] have also suggested registering frames of video by computing the flow along the *epipolar* lines, for which there is also an initial step of calculating the gross camera movement assuming zero parallax. Others have taken different approaches to the problem of image registration as well [12]. We propose a new approach to this problem of tracking large motions of a rigid planar patch (e.g. tracking the global image coordinate transformation with the zero-parallax assumption) without using explicit features, where one of the approximate models (such as *pseudo-perspective*) is used to update the exact model (projective) as part of an iterative (feedback) process.

[4] This condition can be enforced over a wide range of light intensity levels, by using the Wyckoff principle [1][2].

[5] For example, as you translate the camera to the left, you may see the left side of an object that previously you only saw the front of.

[6] In the same spirit, other interesting approximate models can also be derived, but space restricts us to those used in this paper.

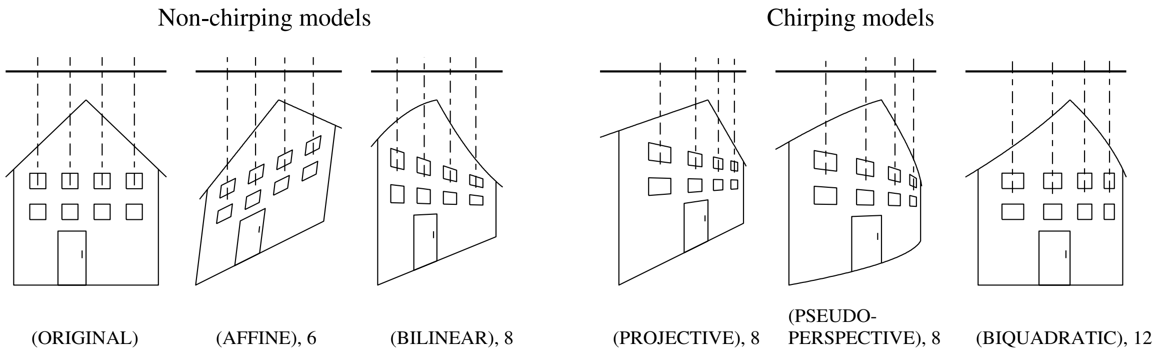


Figure 1: "(MODEL), number of parameters" is shown below each example of a planar coordinate transformation. (ORIGINAL) Original image (with no change in appearance under translation). (AFFINE) Result of affine coordinate transformation applied to original. Note parallel lines remain parallel, and periodicity remains unchanged (e.g. the 4 evenly spaced windows remain evenly spaced). (BILINEAR) Bilinear coordinate transformation captures the essence of converging lines but fails to capture the 'chirping effect' of perspective; the windows change in shape, but remain equally spaced. Straight lines that were parallel to the coordinate axes in the original image remain straight, but other lines become "warped". (PROJECTIVE) Projective coordinate transformation captures the increasing of spatial frequency toward the point where parallel lines meet (*vanishing point.*) Thus, periodic patterns are no longer periodic, but "chirped" – the windows get closer, but straight lines still map to straight lines. The eight parameters of the projective coordinate transform are exactly the eight desired camera parameters. (PSEUDO-PERSPECTIVE) The essence of both the chirping phenomenon and the converging lines are not only captured, but are also constrained to work together in the "right" way, with the "correct" number of parameters (eight). However, straight lines are not preserved. (BIQUADRATIC) The 'biquadratic' coordinate transformation has too many degrees of freedom for a rigid planar patch. It can independently model the 'chirping effect' and the converging of parallel lines. To show this independence, we have constructed an example with a high degree of chirping, and no convergence of parallel lines.

Model	Coordinate transformation from $\mathbf{x}$ to $\mathbf{x}'$	Parameters
Translation	$\mathbf{x}' = \mathbf{x} + \mathbf{b}$	$\mathbf{b} \in \mathbb{R}^2$
Affine	$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$	$\mathbf{A} \in \mathbb{R}^{2\times 2}, \mathbf{b} \in \mathbb{R}^2$
Bilinear	$x' = q_{x'xy}xy + q_{x'x}x + q_{x'y}y + q_{x'}$ $y' = q_{y'xy}xy + q_{y'x}x + q_{y'y}y + q_{y'}$	$q_* \in \mathbb{R}$
Projective	$\mathbf{x}' = \frac{\mathbf{A}\mathbf{x}+\mathbf{b}}{\mathbf{c}^T\mathbf{x}+1}$	$\mathbf{A} \in \mathbb{R}^{2\times 2}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^2$
Pseudoperspective	$x' = q_{x'x}x + q_{x'y}y + q_{x'} + q_\alpha x^2 + q_\beta xy$ $y' = q_{y'x}x + q_{y'y}y + q_{y'} + q_\alpha xy + q_\beta y^2$	$q_* \in \mathbb{R}$
Biquadratic	$x' = q_{x'x^2}x^2 + q_{x'xy}xy + q_{x'y^2}y^2 + q_{x'x}x + q_{x'y}y + q_{x'}$ $y' = q_{y'x^2}x^2 + q_{y'xy}xy + q_{y'y^2}y^2 + q_{y'x}x + q_{y'y}y + q_{y'}$	$q_* \in \mathbb{R}$

Table 1: Image coordinate transformations used in this paper

	Scene assumptions	Camera assumptions
Case 1:	arbitrary 3D	fixed center of projection (e.g. free to rotation about optical axis, pan, tilt, and zoom).
Case 2:	planar	free to move in any way: translate, rotate, zoom, pan, tilt.

Table 2: The two cases where there is no parallax

2 'Video Orbits' and group operations on images

2.1 Review: Groups, Operators, and Orbits

Three of the models presented so far, translation, affine, and projective, each form a *group*.[7] Thinking about the projective model as a group leads to some useful concepts which we exploit in the rest of this paper.

2.1.1 Definitions: group, group operation

A group [13] is a collection of objects upon which there is defined the structure:

1. *Closure*: a *law of composition* allowing us to combine two members of the group into a single object of the same kind. For example, if g, h are images, and p_1, p_2 are operators in the group, then under closure, applying p_1 and p_2 to g, $h = p_2 \circ p_1 \circ g$ implies that the new operator $p = p_2 \circ p_1$ is also in the group.
2. *Identity*: there is a member of the group which combines with any other element to leave the other element unchanged. For example, the 2-parameter translation group contains the identity operator $[0, 0]$.
3. *Inverse*: for every member of the group we can find some other member that combines with it to produce the identity. For example, the 2-parameter translation group containing operator $[b_x, b_y]$ also contains the inverse operator $[-b_x, -b_y]$.
4. *Associativity*: when 3 elements of the group are combined, only the ordering matters – it does not matter whether the first two are combined to produce some other element which is then combined with the third, or the first is combined with the result of combining the last two.[8]

When the elements of a group are operators, considered together with a set of operands, then a structure known as a *group operation*[9] results. In this paper, a group operation consists of a group of coordinate transformation operators together with a set of images. For simplicity, these images (operands) will be regarded as real-valued functions of real variables (e.g. discretization will be ignored).

2.1.2 Definition: orbits

Two operands of a group operation are said to be in the same *orbit* if and only if one can be made to take the place of the other by an operation of the group [13]. The orbit of a given operand is the set of all results that can be obtained allowing any operator from the group to act on the given operand[10].

2.2 'Video Orbits'

The 'video orbit' of a given image is defined to be the set of all images that can be produced by applying operators from the projective group to the given image. For example, a camera zooming, panning, and tilting while photographing a static 3D scene will produce many frames that lie in the same 8D orbit. All the possible coordinate transformations induced by the camera are described by the 8-parameter projective group.

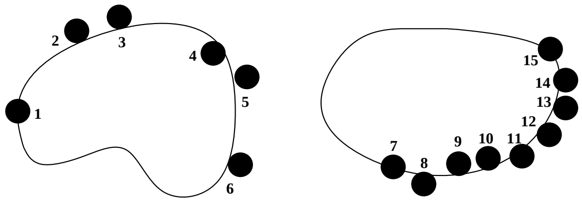


Figure 3: Video orbits form an 8D subspace of the ∞D space of the image (depicted here as 1D subspaces of the 2D page). In this example, frames 1 to 6 lie in orbits that are approximately the same (the orbit of frame 1 is depicted). Frames 7 and higher lie in a different orbit, corresponding to the second scene. The higher frames all lie close to the orbit of frame 7, the first frame after the scene change (the orbit of frame 7 is depicted).

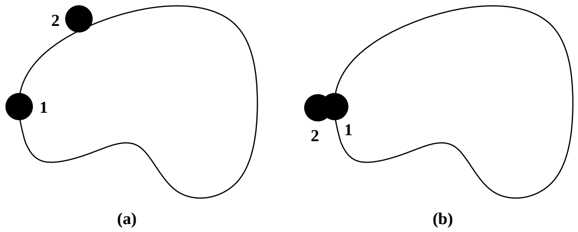


Figure 4: Video orbits. (a) The orbit of frame 1 is the set of all images that can be produced by acting on frame 1 with any element of the operator group. Assuming that frames 1 and 2 are from the same scene, frame 2 will be close to one of the possible projective coordinate transformations of frame 1. In other words, frame 2 lies near the orbit of frame 1. (b) By bringing frame 2 along its orbit (which is nearly the same orbit as the orbit of frame 1), we can determine how closely the two orbits come together at frame 1.

A hypothetical video sequence is illustrated in Fig. 2, where the camera is zooming, tilting, and panning, and there is one scene change. Frames lie in the same orbit if the only frame-to-frame change is camera movement, and in different orbits if there is a scene change.

Fig. 3 illustrates the idea of the video orbit. Although one continuous image may be thought of as a single point in an infinite dimensional space[11], the eight parameters of the projective coordinate transformation acting on this image move it along a trajectory in 8D space. Since these dimensions are difficult to visualize, Fig. 3 shows only a 1D orbit in a 2D space.

Given a set of images that lie in the same orbit of the group, we may wish to find for each image pair, that operator in the group which takes one image to the other image (e.g. makes any one image look exactly like any other). For example, after finding the coordinate transformation between these two frames, we can begin to answer questions such as "did the camera move, and how?" or "did the scene change?". (Many other applications are also possible; a few will be addressed in Sec. 5.) Fig. 4 illustrates the operator p acting on frame 2, f_2, to move it nearest to frame 1, f_1: $f_2' = p \circ f2$. If the two frames are in the same orbit, then the mean-squared error (MSE) between f_1 and f_2' will be zero, indicating that the coordinate transformation described by p is an exact explanation of what happened between frames 1 and 2.

In practice, however, we find which element of the group

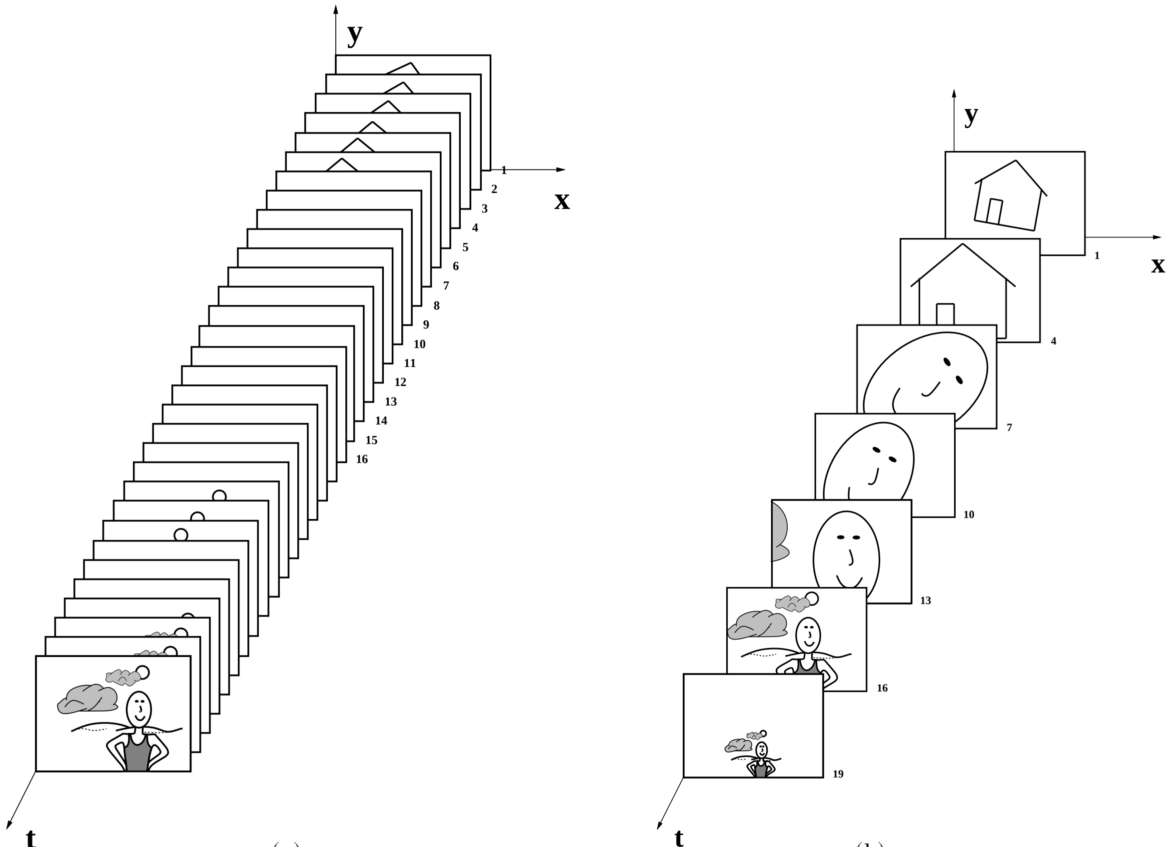
[7] It is interesting to note that of the models shown, only these three correspond exactly to "physical" situations arising from camera motions.

[8] Note that commutativity is also present for the translation group, but not for the affine or projective. Although commutativity is not required by the group definition, its presence greatly simplifies parameter estimation – see Sec. 3.4.

[9] also known as a *group action* or *G-set* [13].

[10] Two operands of a group operation are also said to be *congruent* if they lie in the same orbit [4].

[11] Similarly, a spatially discrete image of, for example, dimensions 480×640, may be thought of as a point in $\mathbb{R}^{480\times640} = \mathbb{R}^{307200}$.



(a) (b)

Figure 2: Hypothetical video sequence with a scene change from frame 6 to frame 7. (a) Original sequence. (b) Temporally subsampled sequence.

takes one image "nearest" the other, for there will be a certain amount of parallax, noise, interpolation error, edge effects, changes in lighting, depth of focus, etc. The problem of finding the 8-parameter operator that maps one image into another is now the problem of parameter estimation for the projective group.

3 Parameter estimation for coordinate transformations

In this section we discuss some existing methods of estimating the parameters for a coordinate transformation, and present some new results for the projective case.

We break the existing methods into two categories: feature-based, and featureless. Of the featureless methods, we consider two subcategories: 1) methods based on minimizing MSE (generalized correlation, direct nonlinear optimization) and 2) methods based on spatiotemporal derivatives and optical flow.

These categories of methods are briefly described in the rest of this section, in an effort to highlight some of their advantages and disadvantages, and to point out a couple of important similarities that do not seem to have been noticed before. More emphasis is placed on methods that provide background for the new algorithm. Note that variations such as *multi-scale* have been omitted from the list above; multiscale analysis can be applied to any of these methods. The new algorithm we develop in this paper (with final form given in Sec. 4) is featureless, based on spatiotemporal derivatives, and multiscale.

Some of the methods described below will be presented in a lower-dimensional space, with "1D images". Although 1D images are not of direct use for the applications that interest us, the corresponding equations sometimes yield a clearer presentation of the similarities and differences in the estimation methods. In a 2D world, the "camera" consists of a center of projection (COP) and a line ("film"). We will call this 2D world "Flatland"[12], and 1D images in this world, "flatland images" to distinguish them below. When the parameter estimation methods for the 1D and 2D cases are conceptually different, then we will note these differences.

3.1 Feature-based methods

Feature-based methods [15][16] assume that point correspondences in both images are available. Given these features we can derive the parameters of the coordinate transformation that gives the mapping from the coordinate x to the coordinate x' by using least squares. (in the affine case, this will amount to linear regression; in the projective case, to a fit to an offset rectangular hyperbola).

In the projective case, given at least three correspondences between point pairs in the two flatland images, we will find the element, $\mathbf{p} = \{a, b, c\} \in \mathbf{P}$ that maps the second image into the first. Let $x_k, k = 1, 2, 3, ...$ be the points in one image, and let x'_k be the corresponding points in the other image. Then: $x'_k = (ax_k+b)/(cx_k+1)$. Re-arranging yields $ax_k+b-x_k x'_k c = x'_k$, so that a, b, and c can be found by solving $k \geq 3$ linear equations in 3 unknowns:

$$\left[\begin{array}{ccc} x_k & 1 & -x'_k x_k \end{array}\right] \left[\begin{array}{ccc} a & b & c \end{array}\right]^T = \left[\begin{array}{c} x'_k \end{array}\right] \qquad (1)$$

using least squares if there are more than three correspondence points. The extension from flatland images to 2D images is conceptually identical; for the affine and projective models, the

[12] after the title of Abbott's classic book [14], which is a story about an alien culture living in a 2D world.

number of correspondence points needed are at least six and eight respectively.

The major problem with feature-based methods is, of course, finding the features. These features are often hand-selected, or computed, possibly with some degree of human intervention [17]. If the motion is to be tracked automatically (e.g. using features that are found computationally), then most of the computation involves feature selection. Once the features are found, and are known to lie on a rigid planar patch, the computation of the motion of that patch is straightforward [8].

A second problem with feature-based methods is their sensitivity to noise and occlusion. Even if reliable features exist between frames (e.g. line markings on a playing field in a football video, see Sec. 5.2) these features may be subject to signal noise and occlusion (e.g. running football players blocking a feature). In video applications (browsing, query, manipulation) of large general databases (movie archives, home video, news footage, etc.), one can not rely on the availability of robust features; hence, the emphasis in the rest of this paper will be on featureless methods for parameter estimation.

3.2 Featureless methods based on generalized cross-correlation

We present generalized cross-correlation in flatland first, and then briefly discuss its adaptation to 2D images.

Generalized cross-correlation is based on an inner-product formulation which establishes a similarity metric for two functions, say, g and h, where $h \approx p \circ g$ is a coordinate-transformed version of g (in practice, with some "noise", making the relationship approximate[13] rather than exact), but p is unknown. We can find, by exhaustive search (applying all possible operators, p, to h), the "best" p as the one which maximizes the quantity:

$$\int_{-\infty}^{\infty} g(x) \frac{p^{-1} \circ h(x)}{\int_{-\infty}^{\infty} p^{-1} \circ h(x) dx} dx \tag{2}$$

where we have normalized the energy of each coordinate-transformed h before making the comparison.

Equivalently, instead of maximizing a similarity metric, we can minimize some distance metric, such as MSE, given by $\int_{-\infty}^{\infty} (g(x) - p^{-1} \circ h(x))^2 dx$. Solving (2) has an advantage over finding MSE when one image could be an amplitude-scaled version, as well as a coordinate-transformed version of the other (as happens when there is an overall change in scene illumination, for example).

Just as the cross-correlation recovers translation, generalizations of cross-correlation can be used to recover both the affine and projective coordinate transformations.

In flatland, the affine model permits only stretching and translating. Given h, an affine coordinate-transformed version of image g, generalized correlation amounts to estimating the parameters for stretch, a and translation, b by exhaustive search, trying all possible dilations (1D zooms), and translations. The collection of all possible coordinate transformations, when applied to one of the images (say, h) serves to produce a family of templates to which the other image, g, can be compared. If we normalize the whole family of templates to have the same energy, by expressing the family as:

$$h_{a,b}(x) = \frac{1}{\sqrt{a}} h(ax + b) \tag{3}$$

[13] In the presence of additive white Gaussian noise, this method, also known as "matched filtering", leads to a maximum likelihood estimate of the parameters [18].

then the maximum likelihood estimate corresponds to selecting the member of the family that gives the largest inner product:

$$\langle g(x), h_{a,b}(x) \rangle = \sum_x g(x) h_{a,b}(x) \tag{4}$$

This result is known as a *cross-wavelet transform.* A computationally efficient algorithm for the cross-wavelet transform has recently been presented [19]. A good general review article dealing with the use of the wavelet transform for the estimation of affine coordinate transformation is presented in [20].

The cross-wavelet transform provides a mapping from a pair of 1D images to a parameter space that is a function of two variables – stretch and translation. This parameter space may be sampled on any desired 2D lattice of parameters. Rather than testing every possible a and b value, we may apply a coarse-to-fine search strategy – computing a coarse grid, observing which point on the grid has the highest 'correlation score', and then computing more points (a finer grid) in that vicinity. Alternatively, we may apply some form of gradient-based iterative procedure to maximize the inner product (or minimize a cost function such as the MSE.)

Note that in the full 2D cross-wavelet transform, there is also rotation and shear, in addition to stretch and translation, although some implementations of the 2D wavelet transform do not use all six affine parameters.

Just like the cross-correlation for the translation group, and the cross-wavelet for the affine group, the "cross-chirplet" can be used to find the parameters of a projective coordinate transformation in flatland, searching over a 3-parameter space. The cross-chirplet is based on a generalization of the wavelet [21] known as the "p-chirplet." A p-chirplet has the form:

$$h_{a,b,c} = h(\frac{ax + b}{cx + 1}) \tag{5}$$

where h is the 'mother chirplet', analogous to the *mother wavelet* of wavelet theory. Members of this family of functions are related to one another by projective coordinate transformations.

With 2D images, the search is over an 8-parameter space. A dense sampling of this volume is computationally prohibitive. Consequently, combinations of coarse-to-fine and iterative gradient-based search procedures are required. Adaptive variants of the chirplet transform have been previously reported in the literature [22].

3.3 Featureless methods based on spatiotemporal derivatives

This section addresses featureless methods which are motivated by the original optical flow equations of Horn and Schunk for the case of translational motion. After a brief review of that case, two cases for affine motion are discussed, and two corresponding new cases for projective motion are presented. A detailed new algorithm for applying the last case is given in Sec. 4.

3.3.1 Optical flow: translation (review)

When the change from one image to another is small, optical flow [23] may be used. In flatland, the traditional optical flow formulation assumes each point x in frame t is a translated version of the corresponding point in frame $t + \Delta t$, and that Δx and Δt are chosen in the ratio $\Delta x / \Delta t = u_f$. The image brightness $E(x, t)$ is described by:

$$E(x, t) = E(x + \Delta x, t + \Delta t), \;\; \forall (x, t), \tag{6}$$

where u_f is the translational flow velocity of the point in question. In the case of pure translation, u_f is constant across the entire image. More generally, though, a pair of 1D images are related by a quantity, $u_f(x)$ at each point in one of the images.

Expanding the right hand side of (6) in a Taylor series, and canceling 0th order terms gives the well-known optical flow equation: $u_f E_x + E_t + h.o.t. = 0$, where E_x and E_t are the spatial and temporal derivatives respectively, and $h.o.t.$ denotes higher order terms in the Taylor series representation. Typically, the higher order terms are neglected, giving the expression for the optical flow at each point in one of the two images:

$$u_f E_x + E_t \approx 0 \tag{7}$$

The derivation for translational optical flow in 2D follows directly [24], although its solution won't be addressed until Sec. 4.

3.3.2 "Affine fit" and "affine flow": a new relationship

We now address the following problem: given the optical flow between two images, g and h, we wish to find the coordinate transform to apply to h to make it look most like g. We describe two approaches: (1) finding the optical flow at every point, and applying linear regression to the optical flow field to estimate the affine motion ('affine fit'), and (2) estimating the motion directly, using a generalized optical flow model ('affine flow').

Wang and Adelson have proposed fitting an affine model to an optical flow field [25] of 2D images. We briefly examine their approach with flatland images; the analysis in 2D is the same but with more notation. Coordinates in the original image, g, are denoted by x, and those in the new image, h, are denoted by x'. Suppose that h is a stretched and translated version of g, so $x' = ax + b$ for every corresponding pair (x, x'). Equivalently, the affine model of velocity (normalizing $\Delta t = 1$), $u_m = x' - x$, is given by $u_m = (a-1)x + b$. We can expect a discrepancy between the flow velocity, u_f, and the model velocity, u_m, due to either errors in the flow calculation, or to errors in the model assumption (e.g. an image pair where h is not exactly a stretched and translateed version of g). However, we can apply linear regression, to get the best (in the least-squares sense) fit by minimizing the quantity:

$$\varepsilon_{fit} = \sum_x (u_m - u_f)^2 = \sum (u_m + E_t/E_x)^2 \tag{8}$$

The constants a and b that minimize ε_{fit} over the entire patch are found by differentiating (8), and setting the derivatives to zero. This results in equations for the model we call "affine fit:"

$$\begin{bmatrix} \sum_x x^2, \sum_x x \\ \sum_x x, \sum_x 1 \end{bmatrix} \begin{bmatrix} a-1 \\ b \end{bmatrix} = - \begin{bmatrix} \sum_x x E_t/E_x \\ \sum_x E_t/E_x \end{bmatrix} \tag{9}$$

Alternatively, the affine coordinate transformation may be directly incorporated into the brightness change constraint equation (6). Bergen *et al.* [26] have proposed this method, which we will call 'affine flow', to distinguish it from the 'affine fit' model of Wang and Adelson above. Let us show how 'affine flow' and 'affine fit' are related. The generalized brightness change constraint equation may be written:

$$E(x', t') = E(ax + b, t + \Delta t) = E(x, t) \tag{10}$$

where Δt has been normalized to 1. Expanding the left hand side in a Taylor series about the identity $(a = 1, b = 0, \Delta_t = 0)$ gives:

$$E(ax+b, t+\Delta t) = E(x,t) + ((a-1)x+b)E_x + \Delta_t E_t + h.o.t. \tag{11}$$

Combining (10) and (11), then canceling E(x,t) and ignoring higher order terms gives:

$$((a-1)x + b)E_x + E_t = u_m E_x + E_t \approx 0 \tag{12}$$

We could have also obtained (12) by substituting $u_m = (ax + b) - x$ directly into (7) in place of u_f.

Summing the squared error:

$$\varepsilon_{flow} = \sum_x (u_m E_x + E_t)^2 \tag{13}$$

over the whole image, differentiating, and equating the result to zero, gives a linear solution for both a and b:

$$\begin{bmatrix} \sum_x x^2 E_x^2, \sum_x x E_x^2 \\ \sum_x x E_x^2, \sum_x E_x^2 \end{bmatrix} \begin{bmatrix} a-1 \\ b \end{bmatrix} = - \begin{bmatrix} \sum_x x E_x E_t \\ \sum_x E_x E_t \end{bmatrix} \tag{14}$$

To see how this result compares to the 'affine fit' proposed by Wang and Adelson, we rewrite (8)

$$\varepsilon_{fit} = \sum_x \left(\frac{u_m E_x + E_t}{E_x}\right)^2 \tag{15}$$

and observe, comparing (13) and (15) that 'affine flow' is equivalent to a weighted least-squares fit, where the weighting is given by E_x^2. Thus the 'affine flow' method tends to put more emphasis on areas of the image that are spatially varying than does the 'affine fit' method. Of course, one is free to separately choose the weighting for each method in such a way that 'affine fit' and 'affine flow' methods both give the same result. Both our intuition and our practical experience tends to favor the 'affine flow' weighting, but, more generally, perhaps we should ask "What is the best weighting?" (e.g. maybe there is an even better answer than the choice among these two). Lucas and Kanade [27], among others, have considered weighting issues.

Given a choice of weightings that make the two methods equivalent, there is another important difference. The 'affine fit' method provides more flexibility in defining the boundaries of the patch because, in addition to simply fitting the optical flow field to the affine model, we can test each point individually to see how well it fits the model, and reject those points that do not fit the model. This gives us a means of segmenting the image into regions that have similar affine motion [25].

Another approach to the 'affine fit' of Wang and Adelson involves computation of the optical flow field using the hierarchical and iterative method of Lucas and Kanade, and *then* a fit to the affine model. An analogous variant of the 'affine flow' method involves iteration and hierarchy as well, but in this case the iteration and hierarchy are incorporated directly into the affine estimator [26]. When they are implemented hierarchically, the two methods differ in additional respects. In situations where the boundaries of the patch are explicitly known, our intuition and experience indicates that the direct hierarchical 'affine flow' method of Bergen *et al.* performs better than the 'affine fit' to the hierarchical flow. The hierarchical optical flow makes the assumption that small blocks of the image are moving with pure translational motion, and then, paradoxically, the affine fit refutes this assumption. However, when we wish to determine the boundary of the patch, the 'affine fit' method has a clear advantage, for it gives us, essentially free of computational cost, the ability to test each pixel, whereas the Bergen *et al.* method would require that we try every possible region shape, which is computationally prohibitive. Since there is an assumption of small motion involved in optical flow, there is generally only a small performance loss in using the hierarchical 'affine fit' as opposed to the 'affine flow', because when the motion is small, each block may be described reasonably by a *pure translation*, and still give accurate enough information to contribute to the overall estimate of the affine model. However, when the motion is a little larger, the blocks that are square or rectangular in one frame will be some other shape in the other, and will not likely match. Fitting an affine model to blocks that do not match is not as good an approach as finding the affine flow directly.

3.3.3 "Projective fit" and "projective flow"

For the affine coordinate transformation, the graph of the range coordinate as a function of the domain coordinate is a straight line; for the projective coordinate transformation, the graph of the range coordinate as a function of the domain coordinate is a rectangular hyperbola [28]. Thus the affine case used linear regression; in the projective case, we wish to use 'hyperbolic regression' to fit the optical flow field to a rectangular hyperbola by considering the flow velocity given by (7) and the model velocity:

$$u_m = x' - x = \frac{ax+b}{cx+1} - x \tag{16}$$

and minimizing the sum of the squared difference:

$$\varepsilon = \sum_x \left(\frac{ax+b}{cx+1} - x + \frac{E_t}{E_x}\right)^2 \tag{17}$$

Differentiating with respect to parameters a, b, and c, then setting derivatives to zero gives the system of equations:

$$\begin{aligned} \sum\left(\frac{ax+b}{cx+1} - x + \frac{E_t}{E_x}\right)\frac{x}{cx+1} &= 0 \\ \sum\left(\frac{ax+b}{cx+1} - x + \frac{E_t}{E_x}\right)\frac{1}{cx+1} &= 0 \\ \sum\left(\frac{ax+b}{cx+1} - x + \frac{E_t}{E_x}\right)\frac{(ax+b)x}{(cx+1)^2} &= 0 \end{aligned} \tag{18}$$

which may be solved using various iterative strategies.

A simpler, more direct solution is to expand u_m in its own multi-variate Taylor series about the identity, $a = 1, b = 0, c = 0$ (same result as univariate Taylor series about $x = 0$). This approach is justified because we know that we must be near the identity – nearness to the identity is the very assumption we made when we used optical flow to begin with.

$$u_m + x = b + (a - bc)x + (bc - a)cx^2 + (a - bc)c^2x^3 + \ldots \tag{19}$$

and then fit the optical flow field to the first three terms:

$$\varepsilon = \sum (b + (a - bc - 1)x + (bc - a)cx^2 + E_t/E_x)^2 \tag{20}$$

Differentiating with respect to $q_2 = (bc - a)c$, $q_1 = a - bc - 1$, and $q_0 = b$, (which, given the constraints of the camera motion, are independent near the identity) and setting to zero gives the equations for what we call "projective fit":

$$\begin{bmatrix} \sum x^4 E_x & \sum x^3 E_x & \sum x^2 E_x \\ \sum x^3 E_x & \sum x^2 E_x & \sum x E_x \\ \sum x^2 E_x & \sum x E_x & \sum E_x \end{bmatrix} \begin{bmatrix} q_2 \\ q_1 \\ q_0 \end{bmatrix} = - \begin{bmatrix} \sum x^2 E_t/E_x \\ \sum x E_t/E_x \\ \sum E_t/E_x \end{bmatrix} \tag{21}$$

Thus, given two functions (images), say, g and h, we can use (21) to find the parameters of the approximate **q**uadratic model, $\mathbf{q} = [q_1, q_2, q_3]^T$, and then relate the parameters of the quadratic model to the projective model, to obtain a, b, and c that indicate to what extent we need to stretch, translate, and "chirp" h to make it similar to g.[14]

In Sec. 4 we will extend the algorithm to 2D images and also present an important part of the algorithm that is omitted here, namely a procedure for finding the parameters, $\mathbf{p}$, of the exact projective model iteratively, by using a feedback system that has (21) in the feedforward loop.

As we did for affine flow, we can also apply the projective group model, $x' = (ax+b)/(cx+1)$, directly to the optical flow equation (7), and obtain a set of nonlinear equations:

$$\begin{aligned} \sum_x \left(\frac{ax+b}{cx+1}E_x - xE_x + E_t\right)\frac{x}{cx+1} &= 0 \\ \sum_x \left(\frac{ax+b}{cx+1}E_x - xE_x + E_t\right)\frac{1}{cx+1} &= 0 \\ \sum_x \left(\frac{ax+b}{cx+1}E_x - xE_x + E_t\right)\frac{(ax+b)x}{(cx+1)^2} &= 0 \end{aligned} \tag{22}$$

which differ from (18) only in the weighting. Szeliski and Coughlan [29] suggest a good framework for solving this nonlinear optimization problem, which they applied to image mosaicing [30]. (Their method could also be applied to the "Projective fit" equations in (18)).

The approach presented here, however, is different. Rather than solving (22) using nonlinear optimization, we may, again (as we did for affine flow) use the Taylor series of u_m (19) and again use the first 3 terms, obtaining enough degrees of freedom to account for the 3 parameters being estimated. Letting $\epsilon = \sum(-h.o.t.)^2 = \sum((b + (a - bc - 1)x + (bc - a)cx^2)E_x + E_t)^2$, and differentiating with respect to each of the 3 parameters of $\mathbf{q}$, setting the derivatives equal to zero, and verifying with the second derivatives, gives the linear system of equations for "projective flow":

$$\begin{bmatrix} \sum x^4 E_x^2 & \sum x^3 E_x^2 & \sum x^2 E_x^2 \\ \sum x^3 E_x^2 & \sum x^2 E_x^2 & \sum x E_x^2 \\ \sum x^2 E_x^2 & \sum x E_x^2 & \sum E_x^2 \end{bmatrix} \begin{bmatrix} q_2 \\ q_1 \\ q_0 \end{bmatrix} = - \begin{bmatrix} \sum x^2 E_x E_t \\ \sum x E_x E_t \\ \sum E_x E_t \end{bmatrix} \tag{23}$$

In Sec. 4 we will extend this derivation to 2D images and show how an iterative approach may be used to compute the parameters, $\mathbf{p}$, of the exact model, by using a feedback system where the feedforward loop involves computation of the approximate parameters, $\mathbf{q}$ in the extension of (23) to 2D.

As with the affine case, 'projective fit' (21) and 'projective flow' (23) differ only in the weighting assumed, and so 'projective fit' provides the added advantage of enabling the boundaries of a moving patch to be easily found, although we will see that in a multiscale implementation, it is better to use the projective flow, if we know *a-priori* the boundary of the patch (e.g. when it is the whole image).

3.4 Exploiting commutativity for parameter estimation

There is a fundamental uncertainty [31] involved in the simultaneous estimation of parameters of a noncommutative group of coordinate transformations, which is akin to the Heisenberg uncertainty relation of quantum mechanics. On the other

[14] Note that in 1D the approximate quadratic model and the desired projective model both have 3 parameters; in 2D, there will be several possible approximate models that can be used to get the 8 projective parameters.

hand, for a commutative[15] group of coordinate transformations (in the absence of noise), we can obtain the exact coordinate transformation. Estimating the parameters of a commutative group of coordinate transformations is computationally efficient, through the use of Fourier cross-spectra [32]. We describe an algorithm which exploits this commutativity for estimating the parameters of the non-commutative 2D projective group in Sec. 4.3 and propose it as an optional pre-processing step.

4 Estimating the parameters of the projective group in 2D

The brightness constancy constraint equation for 2D images [23] gives us the flow velocity components in each of the x and y directions (analogous to (7)):

$$u_f E_x + v_f E_y + E_t \approx 0 \tag{24}$$

As with the 1D case, we may derive two variants of the generalized optical flow: the 'projective fit' to the optical flow and the direct 'projective flow'. However, as is well-known [23], the optical flow field in 2D is underconstrained[16]. The model of *pure translation* at every point has two parameters, but there is only one equation (24) to solve. To deal with the 2D translation problem, it is common practice to compute the optical flow over some neighborhood, which must be at least two pixels, but is generally taken larger than this minimum size. The block is analogous to the *window* of signal processing (e.g. Fourier theory). Typically blocks are square neighborhoods, 3×3, 5×5, or sometimes larger. In order to obtain a better localization tradeoff between spatial domain certainty and velocity certainty, a somewhat Gaussian-shaped (e.g. separable binomial) window is often used instead of a rectangular window.

Our task is not to deal with the 2D translation case, but with the 2D projective case, estimating the eight parameters in the coordinate transformation:

$$x' = \frac{a_{x'x}x + a_{x'y}y + b_{x'}}{c_x x + c_y y + 1}, \qquad y' = \frac{a_{y'x}x + a_{y'y}y + b_{y'}}{c_x x + c_y y + 1}$$

or, in matrix form,

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{\mathbf{A}[x,y]^T + \mathbf{b}}{\mathbf{c}^T[x,y]^T + 1} = \frac{\mathbf{A}\mathbf{x} + \mathbf{b}}{\mathbf{c}^T\mathbf{x} + 1} \tag{25}$$

The desired eight parameters are denoted by $\mathbf{p} = [\mathbf{A}, \mathbf{b}; \mathbf{c}, 1]$, $\mathbf{A} \in \mathbb{R}^{2\times 2}$, $\mathbf{b} \in \mathbb{R}^{2\times 1}$, and $\mathbf{c} \in \mathbb{R}^{2\times 1}$.

Suppose we compute the optical flow field at every pixel, using a 5×5 pixel binomial window, obtaining two parameters at every pixel location: u_f and v_f. We wish to fit this flow field to the projective group model given in (25), establishing the best "projective fit" by minimizing the error:

$$\varepsilon = \sum_{\mathbf{x}} \left(\frac{\mathbf{A}\mathbf{x} + \mathbf{b}}{\mathbf{c}\mathbf{x} + 1} - \mathbf{x} - \mathbf{u_f}\right)^2 \tag{26}$$

This minimization results in a set of eight scalar equations in eight scalar unknowns, similar to (18). Since we have made assumptions about the extent of the motion in use of optical flow (by the Taylor series underlying optical flow itself), it is reasonable to make similar assumptions in expanding (25) in its own Taylor series, analogous to (19). If we take the Taylor series up to 2nd order terms, we obtain the biquadratic model that we have already presented.

As mentioned in Sec. 1.2 several approximate models arise by constraining the twelve parameters of the biquadratic model. In the derivations below we use the bilinear model as the "approximate model" in an intermediate step to get the true projective group parameters.[17] First we will incorporate the approximate model directly into the generalized fit or generalized flow. The Taylor series for the bilinear case gives:

$$\begin{aligned} u_m + x &= q_{x'xy}xy + q_{x'x}x + q_{x'y}y + q_{x'} \\ v_m + y &= q_{y'xy}xy + q_{y'x}x + q_{y'y}y + q_{y'} \end{aligned} \tag{27}$$

Incorporating these into the flow criteria yields a simple set of eight linear equations in eight unknowns, for "bilinear flow", appearing in (28). where the summations are over the entire image (over all x and y) if we are computing global motion, or over a windowed patch if we are computing local motion. This equation looks similar to the 6×6 matrix equation presented in Bergen *et al.* [26].

In order to see how well the model describes the coordinate transformation between 2 images, say, g and h, one might *warp*[18] h to g, using the estimated motion model, and then compute some quantity that indicates how different the resampled version of h is from g. The MSE between the reference image and the warped comparison image might serve as a good measure of similarity. However, since we are really interested in how the *exact model* describes the coordinate transformation, we assess the goodness of fit by first relating the parameters of the approximate model to the exact model, and then find the MSE between the reference image and the comparison image after applying the coordinate transformation of the exact model. A method of finding the parameters of the exact model, given the approximate model, is presented in Sec 4.1.

4.1 'Four point method' for relating approximate model to exact model

Any of the approximations to the projective model described above, after being related to the exact model, will most likely be good in the neighborhood of the identity, $\mathbf{A} = \mathbf{I}, \mathbf{b} = \mathbf{0}, \mathbf{c} = \mathbf{0}$. In Flatland, we explicitly expanded the model Taylor series about the identity; here, although we do not explicitly do this, we shall assume that the terms of the Taylor series of the model correspond to those taken about the identity. In the flatland case (18) we solved the 3 linear equations in 3 unknowns to estimate the parameters of the approximate motion model, and then related the terms in this Taylor series to the exact parameters, a, b, and c (which involved solving another set of 3 equations in 3 unknowns, the second set being nonlinear, although very easy to solve).

In the extension to 2D, the estimate step is straightforward, but the relate step is more difficult, because we now have eight nonlinear equations in eight unknowns, relating the terms in the Taylor series of the model to the desired exact model parameters. Instead of solving these equations, we now propose a simple procedure for relating the parameters of the approximate model to those of the exact model, which we call the 'four point method':

[15] A commutative (or *Abelian*) group is one in which it does not matter what order two elements of the group are applied, for example, translation along the x-axis commutes with translation along the y-axis, so the 2D translation group is commutative.

[16] Optical flow in 1D did not suffer from this problem.

[17] Note that use of an approximate model that doesn't capture chirping or preserve straight lines can still give us the true projective parameters as long as the approximate model captures all eight degrees of freedom.

[18] Here the term *warp* is most appropriate, as the approximate model "warps" straight lines into curves, as seen in Fig 1.

Equation 28

$$
\begin{bmatrix}
\sum x^2y^2E_x^2, & \sum x^2yE_x^2, & \sum xy^2E_x^2, & \sum xyE_x, & \sum x^2y^2E_yE_x, & \sum x^2yE_yE_x, & \sum xy^2E_yE_x, & \sum E_yxyE_x \\
\sum x^2yE_x^2, & \sum x^2E_x^2, & \sum xyE_x^2, & \sum xE_x^2, & \sum x^2yE_yE_x, & \sum x^2E_yE_x, & \sum xyE_yE_x, & \sum E_yxE_x \\
\sum xy^2E_x^2, & \sum xyE_x^2, & \sum y^2E_x^2, & \sum yE_x^2, & \sum xy^2E_yE_x, & \sum xyE_yE_x, & \sum y^2E_yE_x, & \sum E_yyE_x \\
\sum xyE_x^2, & \sum xE_x^2, & \sum yE_x^2, & \sum E_x^2, & \sum xyE_yE_x, & \sum xE_yE_x, & \sum yE_yE_x, & \sum E_yE_x \\
\sum x^2y^2E_xE_y, & \sum x^2yE_xE_y, & \sum xy^2E_xE_y, & \sum E_xxyE_y, & \sum x^2y^2E_y^2, & \sum x^2yE_y^2, & \sum xy^2E_y^2, & \sum xyE_y^2 \\
\sum x^2yE_xE_y, & \sum x^2E_xE_y, & \sum xyE_xE_y, & \sum E_xxE_y, & \sum x^2yE_y^2, & \sum x^2E_y^2, & \sum xyE_y^2, & \sum xE_y^2 \\
\sum xy^2E_xE_y, & \sum xyE_xE_y, & \sum y^2E_xE_y, & \sum E_xyE_y, & \sum xy^2E_y^2, & \sum xyE_y^2, & \sum y^2E_y^2, & \sum yE_y^2 \\
\sum xyE_xE_y, & \sum xE_xE_y, & \sum yE_xE_y, & \sum E_xE_y, & \sum xyE_y^2, & \sum xE_y^2, & \sum yE_y^2, & \sum E_y^2
\end{bmatrix}
\begin{bmatrix} q_{x'xy} \\ q_{x'x} \\ q_{x'y} \\ q_{x'} \\ q_{y'xy} \\ q_{y'x} \\ q_{y'y} \\ q_{y'} \end{bmatrix}
= - \begin{bmatrix} \sum E_txyE_x \\ \sum E_txE_x \\ \sum E_tyE_x \\ \sum E_tE_x \\ \sum E_txyE_y \\ \sum E_txE_y \\ \sum E_tyE_y \\ \sum E_tE_y \end{bmatrix}
$$

1. Select four ordered pairs (e.g. the four corners of the bounding box containing the region under analysis, or the four corners of the image if the whole image is under analysis). Here suppose, for simplicity, that these points are the corners of the unit square: $\mathbf{s} = [s_1, s_2, s_3, s_4] = [(0,0)^T, (0,1)^T, (1,0)^T, (1,1)^T]$.
2. Apply the coordinate transformation using the Taylor series for the approximate model (e.g. (27) for the bilinear choice) to these points: $\mathbf{r} = \mathbf{u}_m(\mathbf{s})$.
3. Finally, the correspondences between $\mathbf{r}$ and $\mathbf{s}$ are treated just like features. This results in four easy to solve linear equations:
$$\begin{bmatrix} x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} x_k, y_k, 1, 0, 0, 0, -x_k x'_k, -y_k x'_k \\ 0, 0, 0, x_k, y_k, 1, -x_k y'_k, -y_k y'_k \end{bmatrix} \left[a_{x'x}, a_{x'y}, b_{x'}, a_{y'x}, a_{y'y}, b_{y'}, c_x, c_y\right]^T \tag{29}$$
where $1 \leq k \leq 4$. This results in the exact eight parameters, $\mathbf{p}$.

Note that the 'four point method' will work with any of the approximate models having eight or more parameters. For example, with the biquadratic model, twelve parameters, $\mathbf{q}$ are estimated from a pair of images, and then the twelve parameter model is allowed to act on the corner points, causing them to move to some new location. The parameters of the exact model are then chosen as the eight parameters, $\mathbf{p}$, which explain this change in position of the four control points. In almost all situations, however, we have found that the pseudo-perspective model is the best of the approximate models. The biquadratic model, for example, in addition to requiring more computational effort, tends to be unstable, responding to "noise" by "overfitting" the true projective coordinate transformation.

We remind the reader that the four corners are **not** feature correspondences as used in the feature-based methods of Sec. 3.1, but, rather, are used so that the two featureless models (approximate and exact) can be related to one another.

It is important to realize the full benefit of finding the exact parameters. While the "approximate model" is sufficient for small deviations from the identity, it is not adequate to describe large changes in perspective. However, if we use it to track small changes incrementally, and each time relate these small changes to the exact model (25), then we can accumulate these small changes using the *law of composition* afforded by the group structure. This is an especially favorable contribution of the group framework. For example, with a video sequence, we can accommodate very large accumulated changes in perspective in this manner. The problems with cumulative error can be eliminated, for the most part, by constantly propagating forward the true values, computing the residual using the approximate model, and each time relating this to the exact model to obtain a goodness-of-fit estimate. We now describe this iterative strategy.

4.2 Overview of new algorithm for 'projective flow'

The algorithm is summarized in this section, with details on each of the steps given in subsequent sections.

Frames from an image sequence are compared pairwise to test whether or not they lie in the same orbit:

1. A Gaussian pyramid of three or four levels is constructed from each of the two frames from the sequence.
2. The parameters $\mathbf{p}$ are estimated at the top of the pyramid, between the two lowest-resolution images, using the iterative method depicted in Fig. 5.
3. The estimated $\mathbf{p}$ is applied to the next higher-resolution (finer) image in the pyramid, $\mathbf{p} \circ g$, to make the two images at that level of the pyramid nearly congruent before estimating the $\mathbf{p}$ between them.
4. The process continues down the pyramid until the highest-resolution image in the pyramid is reached.

4.2.1 Multiscale iterative implementation

Optical flow is based on a Taylor-series formulation, which implicitly assumes smoothness. However, typical images have many sharp edges and contours which violate this assumption. Therefore, the performance is improved if the images are blurry. Since we desire sharp images in general, we blur the images as part of the estimation process, but then use the original (unblurred) images when applying the final coordinate transformation.

Bergen *et al.* suggest a coarse-to-fine strategy, implemented by first constructing a Laplacian pyramid. An image motion

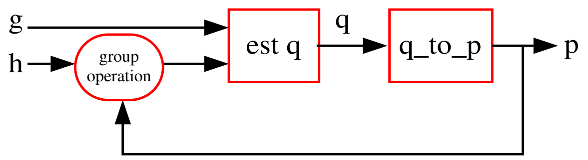


Figure 5: Method of computation of eight parameters $\mathbf{p}$ between two images from the same pyramid level, g and h. The approximate model parameters $\mathbf{q}$ are related to the exact model parameters $\mathbf{p}$ in a feedback system.

of several pixels at the lower (finer) levels of the pyramid may amount to a sub-pixel motion at the higher (small-image) levels. We use a similar strategy, but have found that the performance is actually quite a bit better if we don't downsample the images after blurring. This method involves more computation than the fully downsampled pyramid scheme, so we often draw a compromise which involves partial downsampling at each layer of the pyramid.

4.2.2 Iterative calculation of 'projective flow'

The strategy we present differs from the iterative (affine) strategy of Bergen *et al.* in one important respect beyond simply an increase from six to eight parameters. The difference is the fact that we have two motion models, the 'exact motion model' (25) and the 'approximate motion model', namely the Taylor series approximation to the motion model itself. The approximate motion model is used to iteratively converge to the exact motion model, using the algebraic *law of composition* afforded by the exact model. In this strategy, the exact parameters are determined at each level of the pyramid, and passed to the next level. The steps involved are summarized schematically in Fig. 5, and described below:

1. Initialize: Set $h_0 = h$ and set $\mathbf{p}_{0,0}$ to the identity operator.
2. Iterate ($k = 1 \ldots K$):
   (a) **ESTIMATE:** Estimate the 8 or more terms of the approximate model between two image frames, g and h_{k-1}. This results in approximate model parameters $\mathbf{q}_k$.
   (b) **RELATE:** Relate the approximate parameters $\mathbf{q}_k$ to the exact parameters using the 'four point method'. The resulting exact parameters are $\mathbf{p}_k$.
   (c) **RESAMPLE:** Apply the *law of composition* to accumulate the effect of the $\mathbf{p}_k$'s. Denote these composite parameters by $\mathbf{p}_{0,k} = \mathbf{p}_k \circ \mathbf{p}_{0,k-1}$. Then set $h_k = \mathbf{p}_{0,k} \circ h$. (This should have nearly the same effect as applying $\mathbf{p}_k$ to h_{k-1}, except that it will avoid additional interpolation and anti-aliasing errors you would get by resampling an already resampled image [5]).[19].

Repeat until either the error between h_k and g falls below a threshold, or until some maximum number of iterations is achieved. We find that two or three iterations are usually sufficient for frames from nearly the same orbit. After the first iteration, the parameters $\mathbf{q}_2$ tend to be near the identity since they account for the residual between the "perspective-corrected" image h_1 and the "true" image g. The algorithm (in Matlab on an HP 735) takes about six seconds per iteration for a pair of 320x240 images[20].

[19] The general resampling process is sometimes called "warping" but we refrain from that terminology, as the resampling here takes straight lines to straight lines – lines do not end up curved as might be implied by "warped".

4.3 Optional first step: commutative initialization

As per the discussion in Sec. 3.4), we can achieve better performance with parameter estimation by first estimating the parameters that commute. For example, we find better performance if we first estimate the two parameters of translation, then correct for the translation, and then proceed to estimate the eight projective parameters. Similarly, if image h is merely a rotated version of g, then if we apply a log-polar coordinate transformation to both g, and h, then they become related by a translation in the plane. Thus we can simultaneously estimate isotropic-zoom and rotation about the optical axis by applying a log-polar coordinate transformation followed by a translation estimator. Alternatively, this process may be achieved by a direct application of the Fourier-Mellin transform [33]. Similarly, if the only difference between g and h is a camera pan, then the pan may be estimated through a coordinate transformation to cylindrical coordinates, followed by a translation estimator.

In practice, we have found that it is computationally beneficial to run through the following 'commutative initialization' before estimating the parameters of the projective group of coordinate transformations:

1. Assume that h is merely a translated version of g.
   (a) Estimate this translation using the method of Girod [32].
   (b) Shift h by the amount indicated by this estimate.
   (c) Compute the *MSE* between the shifted h and g, and compare to the original MSE before shifting.
   (d) If an improvement has resulted, use the shifted h from now on.
2. Assume that h is merely a rotated and isotropically zoomed version of g.
   (a) Estimate the two parameters of this coordinate transformation.
   (b) Apply these parameters to h.
   (c) If an improvement has resulted, use the coordinate-transformed (rotated and scaled) h from now on.
3. Assume that h is merely an "x-chirped" (panned) version of g, and, similarly, 'x-dechirp' h. If an improvement results, use the 'x-dechirped' h from now on.
4. Assume that h is merely a "y-chirped" (tilted) version of g, and 'y-dechirp' h. If an improvement results, use the 'y-dechirped' h from now on.

It is quite likely that accounting for the rotation will cause a change in the translation that would best match the two images. Thus it might seem desirable to run through the commutative estimates iteratively. However, our experience on real video indicates that a single pass usually suffices, and, in particular, will catch (as is often the case) situations where there is a pure zoom, a pure pan, a pure tilt, etc, both saving the rest of the algorithm considerable computational effort, as well as accounting for simple coordinate transformations such as when one image is an upside-down version of the other. (Any of these pure cases corresponds to a single parameter group, which is always commutative.)

[20] This is not optimized software yet.

It is interesting to note that without the 'commutative initialization' step, the upside-down image would likely get caught in a local optimum and never converge to the corresponding right-side-up image. A similar argument can be made when one image is an extreme zoom of the other — the algorithm would simply not work without the 'commutative initialization' step.

5 Applications and experimental results

We now suggest some of the many possible applications of the proposed new 'projective flow' algorithm.

5.1 'Video orbits' and scene change detection

Scene change detection and motion description are important problems in "video understanding" for efficient logging and retrieval. Scene change detection can be thought of as a "temporal edge detection" problem, with early approaches based on statistical differences and thresholded filter outputs [34] [35]. Although the current statistics and filter-based methods tend to be low in computation and work 80-90% of the time, they tend to fail when there is lots of camera motion such as pan.

We make the assumption that for most video sequences, there is a large portion of static background included in the images, and that while the camera may pan, tilt, rotate about its optical axis, or zoom substantially between successive frames, that it will not translate substantially between two successive frames (Case 1 in Sec. 1.3). This assumption is particularly valid for cameras on tripods or cameras that are hand-held[21]. Our research defines a *shot boundary* or *scene change* in terms of a distance between *orbits* of the *projective group*. Therefore, if the two frames in question belong to the same scene, they will lie in the same orbit (or nearly so.)

As previously mentioned, in practice, even if there is only a pure pan or zoom across a static scene, all of the frames from a given scene will not lie in exactly the same orbit, because of imperfections such as sensor noise and quantization noise (Fig. 3). Therefore our task is one of, say, finding the element of the projective group that takes the second frame *nearest* the first, along, say, the orbital path of the first (Fig. 4). If this element is found quickly (usually 2-3 iterations until the MSE is sufficiently small) then the algorithm moves along to the next pair of frames. If after ten iterations the MSE is not sufficiently small, then the algorithm terminates and concludes that a scene change has occurred. This idea is illustrated in Fig. 6. In most video sequences, there are far more frame pairs belonging to the same scene than there are transition pairs (scene cuts), so the computational savings in terminating early have a profound improvement on the average computation time.

The initial results on small amounts of real video (subsampled temporally at 3 frames/second) are very accurate for this method. Some example frame pairs demonstrating the robustness of the new algorithm under a variety of conditions are shown in Fig. 7.

5.2 Generation of video *"summary frames"*

How does one "browse" video? One way to speed up the process is to have the computer pull out similar shots, where again, we define a shot as a collection of frames lying in the same orbit. Although typically one might extract a "keyframe" as the first, middle, or last frame from a shot, we propose instead building up a "summary frame" from a shot. The summary frame includes not only the content of (at least) one keyframe, but also captures the nature of the motion in the shot. By glancing at a single summary shot, we can see if the camera dwelled in one spot (summary frame has small spatial extent, high resolution in region of main interest), panned left-to-right (summary frame is horizontally elongated) and so forth. Building a summary frame by mosaicing images using the projective group parameters preserves or enhances the resolution in the composited frames (in contrast, continuous addition of many affine-transformed frames will begin to blur the mosaic, as the affine model can not undo all of the physical camera motion.)

[21] In 1/30 second, most people can turn their head (pan/tilt) further than they can run (translate).

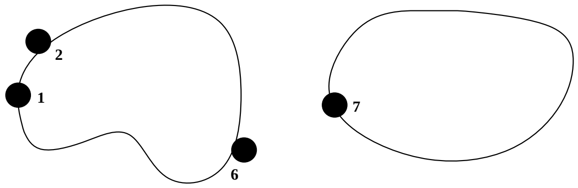


Figure 6: Terminating iteration prematurely. Here we show a frame pair from the same orbit (frames 1 and 2) where the iteration is terminated prematurely after the distance falls below the threshold, and a frame pair from different orbits (frames 6 and 7), where iteration has gone to completion, and the distance remains above the threshold. Since frame 7 is in a different orbit than frame 6, there is no plane projective coordinate transformation that will make frame 7 look like frame 6, and so the algorithm reports frames 6 and 7 as crossing a scene change boundary.

This idea of mosaicing from sequences of images has been done before with the affine model [36] for making "salient stills" and [37] for resolution enhancement, and with the projective model [7][29][30] for mosaicing and enhancement. The use of the new projective flow algorithm brings new accuracy and speed to this application.

Many games such as football or soccer, are played on a flat field so that the playing field forms a rigid planar patch over which the analysis may be conducted. When all of the frames are 'dechirped' with respect to the first frame, then when playing back the sequence, only the players and the image boundaries move around. Markings on the field (such as numbers and lines) remain at a fixed location, which makes subsequent analysis and summary of the video content easier. Despite the players moving in the video, applying the above scene change detector to the output of a broadcast television receiver will classify all of the shots of the football field as belonging to the same 'video orbit'. Furthermore, images that lie in the same orbit of the projective group can be mapped into a single high-resolution image mosaic of the entire playing field, because, even if the entire field might not have been visible in any one image, collectively, the image sequence will likely reveal every square yard of playing surface at one time or another. An example of running the new algorithm on noisy football video is shown in Fig. 7.

6 Acknowledgement

The authors would like to express their thanks to many individuals for their suggestions, encouragement, etc. In particular, thanks to Shawn Becker, Ujjaval Desai, Nassir Navab, John Wang, Walter Bender, Chris Graczyk, Fang Liu, and Constantine Sapuntzakis of MIT, and Alex Drukarev and Jeanne Wiseman at HP Labs, Palo Alto. Some of the software to implement the 'p-chirp' models was developed in collaboration with Shawn Becker at MIT.

References

[1] Charles W. Wyckoff. An experimental extended response film. *S.P.I.E. NEWSLETTER*, JUNE-JULY 1962.

[2] S. Mann and R.W. Picard. Extending dymanic range by combining differently exposed pictures. Technical Report 289, M.I.T. Media Lab Perceptual Computing Section, Boston, Massachusetts, August 1994.

[3] R. Kumar, P. Anandan, and K. Hanna. Shape recovery from multiple views: a parallax based approach. *ARPA image understanding workshop*, Nov 1984.

[4] Semple and Kneebone. *Algebraic Projective Geometry*. Oxford Science Publications, 1979.

[5] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 10662 Los Vaqueros Circle, Los Alamitos, CA, 1990. IEEE Computer Society Press Monograph.

[6] G. Adiv. Determining 3D Motion and structure from optical flow generated by several moving objects. *IEEE Trans. Pattern Anal. Machine Intell.*, pages 304–401, July 1985.

[7] Steve Mann and Nassir Navab. Recovery of relative affine structure using the motion flow field of a rigid planar patch. *Mustererkennung 1994, Tagungsband.*, 1994.

[8] R. Y. Tsai and T. S. Huang. Estimating three-dimensional motion parameters of a rigid planar patch. *tassp*, ASSP-29(9):1147–1152, Dec. 1981.

[9] O. D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[10] Amnon Shashua and Nassir Navab. Relative Affine: Theory and Application to 3D Reconstruction From Perspective Views. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 1994. 1994.

[11] H.S. Sawhney. Simplifying motion and structure analysis using planar parallax and image warping. *CVPR*, 1994.

[12] Qinfen Zheng and Rama Chellappa. A Computational Vision Approach to Image Registration . *IEEE Image Processing*, July 1993. pages 311-325.

[13] M. Artin. *Algebra*. Prentice Hall, 1991.

[14] Edwin A. Abbott. *Flatland*. Signet Classic, June 1984.

[15] R. Y. Tsai and T. S. Huang. Multiframe image restoration and registration. *ACM*, 1984.

[16] T. S. Huang and A.N. Netravali. Motion and structure from feature correspondences: a review. *Proc. IEEE*, Feb 1984.

[17] Nassir Navab and Amnon Shashua. Algebraic Description of Relative Affine Structure: Connections to Euclidean, Affine and Projective Structure. *MIT Media Lab Memo No. 270*, 1994.

[18] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory (Part I)*. John Wiley and Sons, 1968.

[19] R.K. Young. Wavelet theory and its applications. 1993.

[20] Lora G. Weiss. Wavelets and wideband correlation processing. *IEEE Signal Processing Magazine*, pages 13–32, 1993.

[21] Steve Mann and Simon Haykin. The chirplet transform — a generalization of Gabor's logon transform. *Vision Interface '91*, June 3-7 1991.

[22] Steve Mann and Simon Haykin. Adaptive "Chirplet" Transform: an adaptive generalization of the wavelet transform. *Optical Engineering*, 31(6):1243–1256, June 1992.

[23] B. Horn and B. Schunk. Determining Optical Flow. *Artificial Intelligence*, 1981.

[24] B. K. P. Horn. *Robot Vision*. The M.I.T. Press, Cambridge, Massachusetts, 1986.

[25] John Y.A. Wang and Edward H. Adelson. Spatio-Temopral Segmentation of Video Data . In *SPIE Image and Video Processing II*, pages 120–128, San Jose, California, February 7-9 1994.

[26] J. Bergen, P. Burt, R. Hingorini, and S. Peleg. Computing two motions from three frames. In *Proc. Third Int'l Conf. Comput. Vision*, pages 27–32, Osaka, Japan, December 1990.

[27] Lucas and Kanade. An iterative image-registration technique with an application to stereo vision . In *Image Understanding Workshop*, pages 121–130, 1981.

[28] S. Mann and R.W. Picard. Virtual bellows: constructing high-quality images from video. In *Proceedings of the IEEE first international conference on image processing*, Austin, Texas, Nov. 13-16 1994.

[29] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. *CVPR*, 1994.

[30] R. Szeliski. Image mosaicing for tele-reality applications. May 1994.

[31] Roland Wilson and Goesta H. Granlund. The Uncertainty Principle in Image Processing . *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1984.

[32] Bernd Girod and David Kuo. Direct Estimation of Displacement Histograms . *OSA Meeting on IMAGE UNDERSTANDING AND MACHINE VISION*, June 1989.

[33] Yunlong Sheng, Claude Lejeune, and Henri H. Arsenault. Frequency-domain Fourier-Mellin descriptors for invariant pattern recognition. *Optical Engineering*, May 1988.

[34] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. *Visual Database Systems*, II, 1992. editor E. Knuth and L. Wegner.

[35] K. Otsuji and K. Tonomura. Projection-detectiong filter for video cut detection. *Multimedia Systems*, 1:205–210, 1994.

[36] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. *Proc. ACM Multimedia Conf.*, 1993.

[37] M. Irani and S. Peleg. Improving Resolution by Image Registration. *Graphical Models and Img. Proc.*, May 1991.

Figure 7: See attached pages