Factorization Methods for Projective Structure and Motion

Bill Triggs
LIFIA, INRIA Rhône-Alpes,
46, avenue Félix Viallet, 38031 Grenoble, France.
Bill.Triggs@imag.fr

Abstract

This paper describes a family of factorization-based techniques for the recovery of 3D scene structure and camera motion from multiple uncalibrated perspective images of 3D points and lines, up to an overall projective transformation. The methods can be viewed as generalizations of the Tomasi-Kanade algorithm from affine cameras to fully perspective ones, and from points to lines. They make no restrictive assumptions about scene or camera geometry, and unlike most existing reconstruction techniques they do not rely on 'privileged' points or images. All of the available image data is used, and each feature in each image is treated uniformly. The key to projective factorization is the recovery of a consistent set of **projective depths** (projective scale factors) for the image points. We show how this can be done using fundamental matrices and epipoles estimated from image measurements, and present a detailed study of the performance of the new reconstruction techniques as compared to several existing methods. We also describe an approximate structure/motion factorization technique that gives similar results to Singular Value Decomposition based factorization, but runs much more quickly for large problems.

Keywords: Projective Reconstruction, Multiple Images, Structure/Motion Decomposition, Matrix Factorization.

1 Introduction

There has been a considerable amount of progress on scene reconstruction from multiple images in the last few years, aimed at applications ranging from very precise industrial measurement systems with several fixed cameras, to approximate structure and motion from real time video for active robot navigation. It turns out that a profitable way to approach the problem is to ignore the issues of camera calibration and metric structure at first, reconstructing the scene only up to an unknown overall projective transformation in the first instance and only later adding metric information if this is really necessary for the task [5, 10, 1]. The key results are that projective reconstruction is the best that can be done without camera calibration or additional metric information about the scene, and that it is possible from at least two views of point-scenes or three views of line-scenes [2, 3, 8, 6].

However most current reconstruction methods either work *only* for the minimal number of views (typically two), or at very least single out a few such 'privileged' views for initialization before bootstrapping themselves to the multi-image case [5, 10, 9]. For robustness and accuracy, there is a need for methods that uniformly take into account all of the data in all of the images, without making restrictive special assumptions or relying on privileged images or features for bootstrapping.

The orthographic and paraperspective structure/motion factorization methods of Tomasi, Kanade and Poelman [15, 11] partially fulfill these requirements, but unfortunately they apply only when the camera projections are well approximated by affine mappings. This happens only for cameras viewing small, distant scenes, which is seldom the case in practice. A perspective generalization of the factorization method would be desirable, but it has not been clear how to achieve this. The problem is that the unknown projective scale factors of the image measurements must be recovered before factorization becomes possible. (In the affine case these are constant, so they can be directly eliminated from the problem).

As part of the current blossoming of interest in multi-image reconstruction, Shashua [13] recently extended the well-known two-image epipolar constraint to a trilinear constraint between matching points in three images. Hartley [6] showed that this constraint also applies to lines in three images, and Faugeras & Mourrain [4] and I [16, 17] completed that corner of the puzzle by systematically studying the constraints for lines and points in any number of images. A key aspect of the viewpoint presented in [16, 17] is that projective reconstruction is essentially a matter of recovering a coherent set of **projective depths** — projective scale factors that represent the depth information lost during image projection. These are exactly the missing factorization scale factors mentioned above. They satisfy a set of consistency conditions called 'joint image reconstruction equations' [16], that link them together via the corresponding image point coordinates and the various inter-image matching tensors.

In the MOVI group, we have recently been develop-

Submitted to *IEEE Conf. Computer Vision & Pattern Recognition*, San Francisco, June 1996.

ing a family of projective structure and motion algorithms based on this 'projective depth' picture. In the sense that these methods are based on the factorization paradigm, they can be viewed as generalizations of the Tomasi-Kanade method from affine to fully perspective projections. However they also require an additional projective depth recovery phase that is not present in the affine case.

The basic reconstruction method for point images was introduced in a paper by Peter Sturm and myself [14]. In the current paper, this method is extended significantly in several directions, and I also present a detailed assessment of the performance of the new methods as compared to existing techniques such as Tomasi-Kanade factorization and Levenberg-Marquardt nonlinear least squares. Perhaps the most significant result in the paper is the extension of the method to work for line structure as well as point structure, but I will also show how the factorization can be iteratively 'polished' (with results similar to nonlinear least squares iteration), and how any factorization-based method can be speeded up significantly for large problems, by using an approximate fixed-rank factorization technique in place of the Singular Value Decomposition.

The factorization paradigm has several attractive features that are only enhanced by moving from the affine to the projective case:

1) All of the data in all of the images is taken into account.

2) The data is treated uniformly: there is no need to single out 'privileged' features or images for special treatment.

3) Unlike iterative methods, no initialization is required and convergence is virtually guaranteed by the nature of the numerical methods used.

Of course, factorization also has some well known disadvantages:

1) Missing data is not handled gracefully. Every primitive must be visible in every image. This is unrealistic in practice given occlusion and extraction and tracking failures.

2) It is not possible to incorporate a full statistical error model for the image data, although some sort of implicit least-squares trade-off *is* made during factorization.

3) Factorization is an opaque 'black box'. If a point or image is added or any other minor change is made, the whole calculation must be restarted.

4) SVD-based factorization is rather slow for large problems.

Only the last problem will be considered in detail here. If there is missing data, the Tomasi-Kanade 'hallucination' process can be used to work around it [15], although — just as in the affine case — this greatly complicates the method and dilutes some of its principal benefits. There does not seem to be an easy solution to the error model problem, beyond using the factorization to initialize a nonlinear least squares routine (as is done in some of the experiments below). The 'black box' problem demands an incremental factorization update method. This is probably feasible, although there does not seem to be one in the standard numerical algebra literature.

Finally, the problem of the speed of factorization *will* be considered below. SVD-based factorization of a $k \times l$ data matrix takes time $\mathcal{O}(kl \min(k,l))$, which is $\mathcal{O}(\min(k,l))$ more than the size of the input data. However, the SVD is slow because it was designed for general, full rank matrices. For matrices of fixed low rank r (as here, where the rank is 3 for the affine method or 4 for the projective one), approximate factorizations can be computed in time $\mathcal{O}(mnr)$, *i.e.* directly proportional to the size of the input data.

The remainder of the paper outlines the theory of the projective factorization for points and then for lines, describes the final algorithm and implementation, reports on experimental results using synthetic and real data, and then concludes with a discussion. The full theory of projective depth recovery applies equally to the two, three and four image matching tensors, but throughout this paper I will eschew the tensorial notation and concentrate on the simplest two-image (fundamental matrix) case. The underlying theory for the higher dimensional cases can be found in [16].

2 Point Reconstruction

This section reviews the problem of recovering 3D structure (point locations) and motion (camera locations and projections) from a set of m uncalibrated perspective images of a scene containing n 3D points. We will work in homogeneous coordinates with respect to arbitrary projective coordinate frames. It is well known that without further information it is only possible to reconstruct the scene up to an overall projective transformation [2, 8], so the choice of frames is necessarily arbitrary.

Let $\mathbf{X}_\mathrm{p}$ be the unknown homogeneous coordinate vectors of the 3D points, $\mathbf{P}_\mathrm{i}$ the unknown 3×4 image projection matrices, and $\mathbf{x}_\mathrm{ip}$ the measured homogeneous coordinate vectors of the image points, where $p = 1, \ldots, n$ labels points and $i = 1, \ldots, m$ labels images. Each object is defined only up to an arbitrary nonzero rescaling, *e.g.* $\mathbf{X}_\mathrm{p} \sim \lambda \mathbf{X}_\mathrm{p}$ where '$\sim$' denotes equality up to scale.

The basic image projection equations say that the image points $\mathbf{x}_\mathrm{ip}$ are the projections of the world points $\mathbf{X}_\mathrm{p}$, up to an unknown set of scale factors λ_ip:

$$\lambda_\mathrm{ip}\, \mathbf{x}_\mathrm{ip} \;=\; \mathbf{P}_\mathrm{i}\, \mathbf{X}_\mathrm{p}$$

The scales chosen for the image points $\mathbf{x}_\mathrm{ip}$ are arbitrary, and the 3D structure can only be recovered up to: (*i*) an arbitrary individual rescaling of each world point $\mathbf{X}_\mathrm{p}$ and

each projection $\mathbf{P}_{\mathrm{i}}$; (*ii*) an overall 3D projective deformation, represented by an arbitrary nonsingular 4×4 transformation matrix $\mathbf{T}$ acting according to $\mathbf{X}_{\mathrm{p}} \rightarrow \mathbf{T}\,\mathbf{X}_{\mathrm{p}}$, $\mathbf{P}_{\mathrm{i}} \rightarrow \mathbf{P}_{\mathrm{i}}\,\mathbf{T}^{-1}$. Obviously, all of these transformations leave the image projection equations invariant, although they may change the unknown scale factors λ_{ip}.

The scale factors λ_{ip} will be called **projective depths**. With correctly normalized points and projections they become true optical depths, *i.e.* true orthogonal distances of the points from the focal planes of the cameras[1].

More generally, any individual projective depth can be set arbitrarily by choosing appropriate scale factors for the $\mathbf{P}_{\mathrm{i}}$, the $\mathbf{X}_{\mathrm{p}}$ and the $\mathbf{x}_{\mathrm{ip}}$. However, taken as a whole the complete set of projective depths has a strong internal coherence. Fixing once and for all the scales of the image points $\mathbf{x}_{\mathrm{ip}}$, the only remaining degrees of freedom in the $m \times n$ matrix of projective depths are the $m + n$ arbitrary overall scales of each row (the m projections $\mathbf{P}_{\mathrm{i}}$) and each column (the n world points $\mathbf{X}_{\mathrm{p}}$). The remaining $mn - (m + n)$ degrees of freedom contain real information that can be used for 3D reconstruction.

In fact, in [16, 17] I argue that just as the key to calibrated stereo reconstruction is the recovery of Euclidean depth, the essence of projective reconstruction is precisely the recovery of a coherent set of projective depths modulo overall projection and world point rescalings. Once this is done, reconstruction reduces to choosing a projective basis for a certain abstract three dimensional 'joint image' subspace, and reading off point coordinates with respect to it.

2.1 Factorization

One consequence of the above scale coherence can be seen by gathering the complete set of point projections into a single big $3m \times n$ matrix equation:

$$\begin{aligned}\mathbf{W} \;\equiv\; & \begin{pmatrix} \lambda_{11}\,\mathbf{x}_{11} & \lambda_{12}\,\mathbf{x}_{12} & \cdots & \lambda_{1\mathrm{n}}\,\mathbf{x}_{1\mathrm{n}} \\ \lambda_{21}\,\mathbf{x}_{21} & \lambda_{22}\,\mathbf{x}_{22} & \cdots & \lambda_{2\mathrm{n}}\,\mathbf{x}_{2\mathrm{n}} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{\mathrm{m}1}\,\mathbf{x}_{\mathrm{m}1} & \lambda_{\mathrm{m}2}\,\mathbf{x}_{\mathrm{m}2} & \cdots & \lambda_{\mathrm{mn}}\,\mathbf{x}_{\mathrm{mn}} \end{pmatrix} \\ = & \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_{\mathrm{m}} \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_{\mathrm{n}} \end{pmatrix}\end{aligned}$$

Notice that *with a consistent set of projective depths* λ_{ip}, the $3m \times n$ **rescaled measurement matrix** $\mathbf{W}$ has rank at most 4. Any such matrix can be factorized into a $3m \times 4$ matrix of 'projections' multiplying a $4 \times n$ matrix of 'points' as shown, and any such factorization corresponds to as a valid projective reconstruction: the freedom in factorization is exactly a 4×4 nonsingular linear transformation $\mathbf{P} \rightarrow \mathbf{P}\,\mathbf{T}^{-1}$, $\mathbf{X} \rightarrow \mathbf{T}\,\mathbf{X}$, which can be regarded as a projective transformation of the reconstructed 3D space.

Given a rank deficient matrix like the rescaled measurement matrix $\mathbf{W}$, one practical method of factorizing it is the Singular Value Decomposition [12]. This decomposes an arbitrary $k \times l$ matrix $\mathbf{W}_{\mathrm{k}\times\mathrm{l}}$ of rank r into a product $\mathbf{W}_{\mathrm{k}\times\mathrm{l}} = \mathbf{U}_{\mathrm{k}\times\mathrm{r}}\,\mathbf{D}_{\mathrm{r}\times\mathrm{r}}\,\mathbf{V}_{\mathrm{l}\times\mathrm{r}}^{\top}$, where the columns of $\mathbf{V}_{\mathrm{l}\times\mathrm{r}}$ and $\mathbf{U}_{\mathrm{k}\times\mathrm{r}}$ are orthonormal bases for the input (co-kernel) and output (range) spaces of $\mathbf{W}_{\mathrm{k}\times\mathrm{l}}$, and $\mathbf{D}_{\mathrm{r}\times\mathrm{r}}$ is a diagonal matrix of positive decreasing 'singular values'. The decomposition is unique when the singular values are distinct, and it can be computed very stably and reliably in time $\mathcal{O}(kl \min(k, l))$. The matrix of singular values can be absorbed into either $\mathbf{U}$ or $\mathbf{V}$ to give a decomposition of the projection/point form $\mathbf{P}\,\mathbf{X}$. (I absorb them into $\mathbf{U}$ to form $\mathbf{P}$, but absorbing them into $\mathbf{V}$ or splitting them between the two would be equally valid).

The SVD has been used by Tomasi, Kanade and Poelman [15, 11] for their affine (orthographic and paraperspective) reconstruction techniques. The current application can be viewed as a generalization of these methods to projective reconstruction. The projective case leads to slightly larger matrices ($3m \times n$ rank 4 as opposed to $2m \times n$ rank 3), but is actually simpler than the affine case as there is no need to subtract translation terms or apply nonlinear constraints to guarantee the orthogonality of the projection matrices.

One disadvantage of the SVD is the fact that it is relatively costly for large problems. One would like a method of order $\mathcal{O}(m\ n)$ (the size of the input data). The SVD is a factor of $\min(3m, n)$ slower than this, which can be significant when there are both many points and many images. For general matrices the order of the SVD is probably close to optimal, but for matrices of fixed low rank r one might hope to find a factorization in an 'output sensitive' time like $\mathcal{O}(mnr)$. I am not aware of an exact factorization method with this complexity, but it is relatively easy to find approximate methods that run this fast. I have experimented with one such $\mathcal{O}(mnr)$ approximate factorization technique, and find it to be almost as accurate as the SVD and significantly faster for large problems.

The idea of the method is very simple. In theory, a basis of row vectors that spans the matrix can be found by orthogonalization. The matrix is swept repeatedly, each time eliminating the largest row and orthogonalizing the others with respect to it. If the matrix really has rank r this will zero the remaining rows after r iterations. The eliminated rows form the '$\mathbf{V}$' (row) half of the factorization, and the '$\mathbf{U}$' (column) half can be found by dot-producting these with the rows of the original data matrix. The main practical problem with this is that although it chooses r significant directions, some small perturbation of these may be a better overall choice if the matrix is not

[1] For this, the world and image points should be given affine normalization ('weight' components equal to 1) and the projections should be normalized so that the normal vector of the focal plane (*i.e.* the vectorial part of the 'weight' component row) has norm 1.

exactly of rank r. Only a small fraction of the input rows (directions in row space) are actually sampled: if the remaining rows have a small but consistent bias along some direction, this will not be taken into account in the final factorization.

To counter this we do two things. Firstly, instead of just taking the largest remaining row, we find this and add all of the remaining rows to it with signs chosen to increase its magnitude. If there is a consistent bias in the remaining rows this tends to accumulate, and hence to be well represented in the final factorization. Secondly, we accumulate more than r rows — typically $2r$ in practice — and then find the r best overall ones by SVD of the resulting nearly-orthogonal row matrix.

Finally, in the reconstruction application it turns out that the results are slightly better if the data matrix is decomposed by columns (points) rather than rows (image coordinates), so we actually work with the transposed matrix $\mathbf{W}^\top$.

2.2 Projective Depth Recovery

The above projective factorization techniques can *only* be used if a self-consistent set of projective depths λ_{ip} can be found. The key technical advance that makes this work possible is a practical method for estimating projective depths using fundamental matrices and epipoles obtained from the image data. The full theory can be found in [16], which also describes how to use trivalent and quadrivalent matching tensors for depth recovery. Here I shall just give a brief outline of the fundamental matrix case.

The projection equation $\lambda_{\mathrm{ip}}\ \mathbf{x}_{\mathrm{ip}} = \mathbf{P}_{\mathrm{i}}\ \mathbf{X}_{\mathrm{p}}$ implies that the 6×5 matrix

$$\begin{aligned}\left(\begin{array}{c|c} \mathbf{P}_{\mathrm{i}} & \lambda_{\mathrm{ip}}\ \mathbf{x}_{\mathrm{ip}} \\ \mathbf{P}_{\mathrm{j}} & \lambda_{\mathrm{jp}}\ \mathbf{x}_{\mathrm{jp}} \end{array}\right) &= \left(\begin{array}{c|c} \mathbf{P}_{\mathrm{i}} & \mathbf{P}_{\mathrm{i}}\ \mathbf{X}_{\mathrm{p}} \\ \mathbf{P}_{\mathrm{j}} & \mathbf{P}_{\mathrm{j}}\ \mathbf{X}_{\mathrm{p}} \end{array}\right) \\ &= \left(\begin{array}{c} \mathbf{P}_{\mathrm{i}} \\ \mathbf{P}_{\mathrm{j}} \end{array}\right) \left(\begin{array}{c|c} \mathbf{I}_{4\times 4} & \mathbf{X}_{\mathrm{p}} \end{array}\right)\end{aligned}$$

has rank at most 4. Hence, all of its 5×5 minors vanish. Expanding by cofactors in the last column gives homogeneous linear equations in the components of $\lambda_{\mathrm{ip}}\ \mathbf{x}_{\mathrm{ip}}$ and $\lambda_{\mathrm{jp}}\ \mathbf{x}_{\mathrm{jp}}$, with coefficients that are 4×4 determinants of projection matrix rows. These turn out to be just the expressions for fundamental matrix and epipole components in terms of projection matrix components [17, 4]. In particular, if abc and $a'b'c'$ are even permutations of 123, $\mathbf{P}_{\mathrm{i}}^{a}$ denotes row a of $\mathbf{P}_{\mathrm{i}}$, and $\det(\cdot)$ denotes the determinant of four 4-component rows, we have:

$$\begin{aligned}\left[\mathbf{F}_{\mathrm{ij}}\right]_{a\,a'} &= \det\left(\mathbf{P}_{\mathrm{i}}^{b},\ \mathbf{P}_{\mathrm{i}}^{c},\ \mathbf{P}_{\mathrm{j}}^{b'},\ \mathbf{P}_{\mathrm{j}}^{c'}\right) \\ \left[\mathbf{e}_{\mathrm{ij}}\right]^{a} &= \det\left(\mathbf{P}_{\mathrm{i}}^{a},\ \mathbf{P}_{\mathrm{j}}^{1'},\ \mathbf{P}_{\mathrm{j}}^{2'},\ \mathbf{P}_{\mathrm{j}}^{3'}\right)\end{aligned}$$

Applying these relations to the three 5×5 determinants built from two rows of image i and three rows of image j gives the following basic **projective depth recovery equation**:

$$\left(\mathbf{F}_{\mathrm{ij}}\ \mathbf{x}_{\mathrm{jp}}\right)\lambda_{\mathrm{jp}}\ =\ \left(\mathbf{e}_{\mathrm{ij}} \wedge \mathbf{x}_{\mathrm{ip}}\right)\lambda_{\mathrm{ip}} \qquad (1)$$

This identity says two things: (*i*) The epipolar line of $\mathbf{x}_{\mathrm{jp}}$ in image i is the same as the line through the corresponding point $\mathbf{x}_{\mathrm{ip}}$ and epipole $\mathbf{e}_{\mathrm{ij}}$ (which is well known); (*ii*) *With the correct projective depths, the two terms have exactly the same size*. The equality is exact, not just up to scale. This is the new result that allows us to recover projective depths using fundamental matrices and epipoles. Analogous results based on higher order matching tensors can be found in [16]. Some intuitive motivation for this depth recovery equation is given in fig. 1.

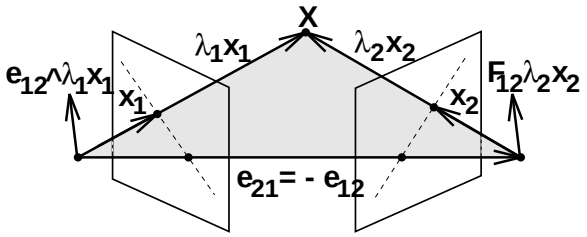

Figure 1: Motivation for the depth recovery equation. Think of image vectors as being embedded in 3D space. With the correct scalings, the epipoles $\mathbf{e}_{12} = -\mathbf{e}_{21}$ are just the vectors between the optical centres, and the rescaled image points $\lambda_1\mathbf{x}_1$ and $\lambda_2\mathbf{x}_2$ form a triangle of 3D vectors with apex $\mathbf{X}$. The normal to the epipolar plane is just $\mathbf{e}_{12} \wedge (\lambda_1\mathbf{x}_1) \sim \mathbf{e}_{21} \wedge (\lambda_2\mathbf{x}_2)$, and these two vectors are actually equal since both have length equal to twice the area of the epipolar triangle. On the other hand, the epipolar line $\mathbf{F}_{12}\,(\lambda_2\mathbf{x}_2)$ is the projection of the epipolar plane, so viewed as a 3D vector it is also normal to the epipolar plane. Given that it is linear in $\lambda_2\mathbf{x}_2$ and vanishes for $\mathbf{x}_2 \sim \mathbf{e}_{21}$, it turns out to be directly proportional to $\mathbf{e}_{21} \wedge (\lambda_2\mathbf{x}_2)$ and with the correct scaling for $\mathbf{F}_{12}$ the result follows.

It is straightforward to recover projective depths using the above equation. Each instance of it linearly relates the depths of a single 3D point in two images. By estimating a sufficient number of fundamental matrices and epipoles, we can amass a system of homogeneous linear equations that allows the complete set of depths for a given point to be found, up to an arbitrary overall scale factor.

At a minimum, this can be done by selecting any set of $m-1$ equations that link the m images into a single connected graph. With such a non-redundant set of equations the depths for each point p can be found trivially by chaining together the solutions for each image, starting from some arbitrary initial value such as $\lambda_{1\mathrm{p}} = 1$. Solving the depth recovery equation in least squares gives a simple recursion relation for λ_{ip} in terms of λ_{jp} :

$$\lambda_{\mathrm{ip}}\ :=\ \frac{\left(\mathbf{e}_{\mathrm{ij}} \wedge \mathbf{x}_{\mathrm{ip}}\right) \cdot \left(\mathbf{F}_{\mathrm{ij}}\ \mathbf{x}_{\mathrm{jp}}\right)}{\left\|\mathbf{e}_{\mathrm{ij}} \wedge \mathbf{x}_{\mathrm{ip}}\right\|^2}\ \lambda_{\mathrm{jp}}$$

If additional fundamental matrices are available, the system becomes redundant (and hence potentially more robust) and a least squares solution can be found. In the limit, all $m(m-1)$ equations could be used to find the m unknown depths for each point, but this would be computationally very expensive.

It is not yet clear how to choose the best trade-off between economy and robustness. In this paper we will restrict ourselves to two very simple non-redundant choices: the images are either taken pairwise in sequence, $\mathbf{F}_{21}, \mathbf{F}_{32}, \ldots, \mathbf{F}_{\mathrm{m\,m-1}}$, or all subsequent images are scaled in parallel from the first one, $\mathbf{F}_{21}, \mathbf{F}_{31}, \ldots, \mathbf{F}_{\mathrm{m1}}$. It might seem that chains of rescalings would prove numerically unstable, but in practice the depth recovery system seems to be surprisingly well conditioned. Both of the above systems work well despite their non-redundancy and chain length/reliance on a 'key' image. In fact the two systems give very similar results except when there are many ($>$50) images, when the shorter chains of the parallel system become more robust.

To use a redundant depth recovery system, one additional refinement is required. In practice, fundamental matrices and epipoles can only be recovered from the image data up to an unknown scale factor, so we do not actually know the scale factors in the depth recovery equation after all. With a non-redundant system this is not an issue because projective depths can only be recovered up to an unknown overall rescaling of each image in any case: the arbitrary relative scales of the estimated fundamental matrices and epipoles can be absorbed into the arbitrary image scalings. However, with a redundant system it is essential to choose a *self-consistent* set of normalizations for the fundamental matrices and epipoles. This can be done using the quadratic identities between matching tensors described in [16], but I will not describe the process in detail here.

2.3 Balancing the Scales

A further point is that with arbitrary choices of scale for the fundamental matrices and epipoles, the average size of the recovered depths might tend to increase or decrease exponentially during the solution-chaining process. Theoretically this is not a problem because the overall scales are arbitrary, but it could easily make the factorization phase of the reconstruction algorithm numerically ill-conditioned. To counter this the recovered matrix of projective depths must be balanced after it has been built, by judicious overall row and column rescalings.

The process is very simple. The image points are normalized on input, so ideally all of the scale factors λ_{ip} should be of more or less the same order of magnitude, $\mathcal{O}(1)$ say. For each point the depths are estimated as above, and then: (*i*) each row (image) of the estimated depth matrix is rescaled to have length $\sqrt{n}$; (*ii*) each column (point) of the resulting matrix is rescaled to length $\sqrt{m}$. This process is repeated until it roughly converges, which happens very quickly (within 2–3 iterations).

3 Line Reconstruction

Scenes containing 3D lines can also be reconstructed using the above techniques. We will only give a brief sketch of the theory here as a full discussion requires consideration of the trilinear three-image matching constraint for lines [6, 13, 4, 17, 16].

A 3D line $\mathbf{L}$ can be defined by any two distinct points lying on it, say $\mathbf{Y}$ and $\mathbf{Z}$. In each image i, $\mathbf{L}$ projects to some image line $\mathbf{l}_{\mathrm{i}}$ and $\mathbf{Y}$ and $\mathbf{Z}$ project to image points $\mathbf{y}_{\mathrm{i}}$ and $\mathbf{z}_{\mathrm{i}}$ lying on $\mathbf{l}_{\mathrm{i}}$: $\mathbf{l}_{\mathrm{i}} \cdot \mathbf{y}_{\mathrm{i}} = 0, \mathbf{l}_{\mathrm{i}} \cdot \mathbf{z}_{\mathrm{i}} = 0$. The points $\mathbf{y}_{\mathrm{i}}$, $i = 1, \ldots, m$ are in epipolar correspondence and hence can be used in the projective depth recovery equations (1), and similarly for the $\mathbf{z}$'s. Of course, this is not much use unless corresponding $\mathbf{y}$'s and $\mathbf{z}$'s can be recovered in the first place. This requires point-to-point correspondences between different images of the same line, which can be found by estimating an appropriate subset of the matching tensors between the various images. If the scene contains *only* lines, at least three images are required, but then the correspondences can be found very simply using one of the three trilinear matching constraints between the images [6, 16]. Here, we will simplify things further by assuming that fundamental matrices and epipoles are available between the images. These can be obtained directly from point matches if the scene contains points, or indirectly from trivalent tensors obtained using line matches.

Given any point $\mathbf{y}_{\mathrm{j}}$ on a line $\mathbf{l}_{\mathrm{j}}$ in image j, the corresponding point $\mathbf{y}_{\mathrm{i}}$ on the corresponding line $\mathbf{l}_{\mathrm{i}}$ in image i can be found by intersecting the epipolar line $\mathbf{F}_{\mathrm{ij}}\,\mathbf{y}_{\mathrm{j}}$ with $\mathbf{l}_{\mathrm{i}}$: $\mathbf{y}_{\mathrm{i}} \sim \mathbf{l}_{\mathrm{i}} \wedge (\mathbf{F}_{\mathrm{ij}}\,\mathbf{y}_{\mathrm{j}})$. This is well known, but it apparently only provides equality up to scale which is not enough for our purposes. However, if we take the depth recovery equation (1), apply it to the rescaled image points $\lambda_{\mathrm{i}}\mathbf{y}_{\mathrm{i}}$ and $\lambda_{\mathrm{j}}\,\mathbf{y}_{\mathrm{j}}$, wedge-product the result with $\mathbf{l}_{\mathrm{i}}$, expand the double wedge-product, and simplify using the fact that $\mathbf{l}_{\mathrm{i}} \cdot \mathbf{y}_{\mathrm{i}} = 0$, we have:

$$\begin{aligned} \mathbf{l}_{\mathrm{i}} \wedge (\mathbf{F}_{\mathrm{ij}}\,\mathbf{y}_{\mathrm{j}})\;\lambda_{\mathrm{j}} &= \mathbf{l}_{\mathrm{i}} \wedge (\mathbf{e}_{\mathrm{ij}} \wedge \mathbf{y}_{\mathrm{i}})\;\lambda_{\mathrm{i}} \\ &= [-(\mathbf{l}_{\mathrm{i}} \cdot \mathbf{e}_{\mathrm{ij}})\,\mathbf{y}_{\mathrm{i}} + (\mathbf{l}_{\mathrm{i}} \cdot \mathbf{y}_{\mathrm{i}})\,\mathbf{e}_{\mathrm{ij}}]\;\lambda_{\mathrm{i}} \\ &= -(\mathbf{l}_{\mathrm{i}} \cdot \mathbf{e}_{\mathrm{ij}})\,\mathbf{y}_{\mathrm{i}}\,\lambda_{\mathrm{i}} \qquad (2) \end{aligned}$$

Once again we find that a well-known formula actually gives us a little more than was obvious at first sight. Not only can this equation be used to transfer points between lines, but when it does so *it automatically gives the correct relative projective depths for 3D reconstruction*[2].

It is straightforward to implement a factorization-based line reconstruction algorithm using this. For each line,

[2]For those familiar with trivalent tensors and tensorial notation, the equivalent trivalent tensor based relation is just $\left(\mathrm{l}_{\mathrm{B_k}}\,\mathbf{G}_{\mathrm{C_j}}{}^{\mathrm{A_i B_k}}\,\mathbf{y}^{\mathrm{C_j}}\right)\,\lambda_{\mathrm{j}} = \left(\mathrm{l}_{\mathrm{B_k}}\,\mathbf{e}_{\mathrm{j}}^{\mathrm{B_k}}\,\mathbf{y}^{\mathrm{A_i}}\right)\,\lambda_{\mathrm{i}}$.

two well-spread points are chosen in the first image. These are then transferred to the other images by chaining together instances of equation (2), being careful to preserve the correct relative scale factors. If desired, redundant equations can be included for robustness and a least squares solution can be found for the $3m$ coordinates of the $\mathbf{y}_i$ ($i = 1, \ldots, m$), and similarly for the $\mathbf{z}_i$. The two $3m$-coordinate vectors for each of the n_{lines} lines are combined into a big $3m \times 2n_{\text{lines}}$ rescaled measurement matrix, which is then factorized as above to give the reconstructed 3D points $\mathbf{Y}$ and $\mathbf{Z}$ that define each reconstructed line.

As in the point case, a self-consistent normalization for the fundamental matrices and epipoles is required if (and only if) redundant equations are used, but it is always advisable to balance the resulting depth estimates. Balancing is a little more involved for lines than for points. It works directly with the $3m \times 2n_{\text{lines}}$ rescaled measurement matrix, iteratively rescaling all coordinates of each image (triple of rows) and all coordinates of each line (pair of columns) until an approximate equilibrium is reached. The overall mean square size of each coordinate is $\mathcal{O}(1)$ in each case. To ensure that the two points chosen to represent each line are on average well separated, I also orthonormalize the two $3m$-component column vectors for each line with respect to one another. The depth recovery equations (2) are linear and hence invariant with respect to this, but it does of course change the 3D representatives $\mathbf{Y}$ and $\mathbf{Z}$ recovered for each line.

4 Implementation

In this section we summarize the complete algorithm for factorization-based 3D projective reconstruction from image points and lines, and discuss a few important implementation details and variant methods. The complete algorithm goes as follows:

0) Extract and match point and line features in all the images.

1) Standardize the coordinates in each image, as described below.

2) Use the point matches to estimate a set of fundamental matrices and epipoles sufficient to connect all of the images together.

3) For each point estimate the projective depths using equation (1). Build and balance the point-depth matrix λ_{ip}, and use it to build the rescaled point measurement matrix $\mathbf{W}$.

4) For each line choose two representative points and transfer them to the other images using the transfer equations (2). Build and balance the rescaled line measurement matrix.

5) Combine the line and point measurement matrices into a single $3m \times n$ data matrix ($n = n_{\text{points}} + 2n_{\text{lines}}$), and factorize it using either SVD or the approximate fixed-rank method.

6) Recover 3D projective structure (point and line-representative coordinates) and motion (projection matrices) from the factorization.

7) Un-standardize the projection matrices, as described below.

The asymptotic complexity of the algorithm is dominated by the $\mathcal{O}(mn \min(3m, n))$ SVD step if this is used, while if an approximate factorization is used it is proportional to the input data size $\mathcal{O}(mn)$.

4.1 Standardization of Image Coordinates

To get acceptable results from the above algorithm, it is *absolutely essential* to work in a well-adapted image coordinate system. Hartley [7] has pointed out the importance of this for fundamental matrix estimation, and I can only re-emphasize it here. I had originally advocated a 'geometry based' standardization scheme where pixel coordinates were standardized to be approximate 3D angles by dividing by a rough estimate of the focal length in pixels, but Hartley's 'numerically based' scheme where all coordinates are standardized to $\mathcal{O}(1)$ turns out to be somewhat better and easier to apply in practice.

In fact, a slight generalization of Hartley's scheme is used here. In each image a 'scatter matrix' (*i.e.* mean and covariance matrix) of the point coordinates is accumulated, and then an affine image transformation is applied that deforms the 'one standard deviation' covariance ellipse into a unit circle centred at the origin. This ensures that the standardized point coordinates are not only well-normalized, but also well spread out even if the input points happen to be clustered in a narrow belt across the image. Because of the standardization process, the recovered camera projections need to be un-standardized by multiplying by the inverse transformation before they are used.

4.2 Generalizations & Variants

We have implemented and experimented with a number of variants of the above algorithm, the more promising of which are featured in the experiments described below.

Iterative Factorization: The best set of projective depths depends on the 3D structure, which in turn derives from the depths. It is possible to improve the estimated reconstruction iteratively by re-estimating the depths from the factorization and then re-factorizing. For points this is simply a matter of finding the component of the reprojected 3D point vector along each image vector, while for lines the reprojected via point is perturbed orthogonally to lie on the corresponding image line.

With SVD-based factorization and standardized image coordinates the iteration turns out to be extremely stable. In fact a very simple reconstruction method for points is simply to start the SVD at some arbitrary initial depths (say the affine ones $\lambda_{\mathrm{ip}} = 1$) and iterate to convergence. This requires no fundamental matrices or depth recovery equations, and proves very reliable in practice. Its only real problem is speed: started far from the correct solution, the iteration can sometimes take as many as a 50–100 iterations to converge. However, if started from depths estimated using fundamental matrices, the iteration usually converges within a few (say 2–5) cycles and always improves the recovered structure somewhat.

By contrast, an iteration based on the approximate fixed-rank factorization method turns out to be less reliable, although it still sometimes improves the reconstruction slightly.

Nonlinear Least Squares: The 'linear' factorization-based projective reconstruction methods described above are a suitable starting point for more refined nonlinear least-squares estimation. This can take account of image point error models, camera calibrations, or Euclidean constraints, as in the work of Hartley [5] and Mohr, Boufama and Brand [10]. The standard workhorse for such problems is the Levenberg-Marquardt method [12], so for comparison with the linear methods I have implemented a simple Levenberg-Marquardt based projective point reconstruction algorithm. This turns out to work well in combination with either the fixed-rank or the SVD-based factorization techniques, and often produces a useful amount of 'polishing' at a moderate cost. If run time is not a consideration, iterative SVD followed by Levenberg-Marquardt is probably the most accurate and reliable method of all.

Affine Factorization: I have also implemented SVD-based 'projective' point reconstruction based on an affine projection model, as in the original Tomasi-Kanade-Poelman work [15, 11][3]. This is simpler and slightly faster than the projective SVD-based method, but in most of the below experiments it gives rather poor results because the perspective distortion is usually quite large.

5 Experiments

To quantify the performance of the various algorithms discussed, I have run a variety of simulations using artificial data, and also tested the algorithms on several sequences derived from real images. The results and conclusions for lines are subject to change as the current line implementation is very preliminary. A more thorough study will be reported in the final version of this paper.

[3] The method is 'projective' only in the sense that projective rather than affine alignment is used to estimate 3D reconstruction errors.

5.1 Simulations

The simulations are based on trial scenes consisting of random 3D points and lines in the unit cube $[-1,1] \times [-1,1] \times [-1,1]$, viewed by identical perspective cameras at uniformly spaced intervals along one of a number of trajectories defined with respect to the scene. The spacing of the viewpoints is reduced as cameras are added, so that the total range of viewpoints is constant. The camera projection is chosen so that the scene almost fills the nearest camera. Unless otherwise stated this is at 2 units from the centre of the scene, so the focal lengths used are quite short and perspective distortion is significant.

Uniformly distributed noise is added to the image measurements to model quantization error and localization uncertainty. For each experiment the mean-square and worst-case 2D reprojection error and relative 3D reconstruction error are accumulated over 100 trials. The errors in the estimated 3D structure are found after linear least-squares projective alignment with the true Euclidean structure. Default values of 10 cameras, 50 points or lines, ± 1 pixel noise for points and ± 0.2 pixel noise for lines should be assumed unless otherwise stated.

Fundamental matrices and epipoles are estimated using the linear least squares ('8 point') method with all the available point matches, followed by a supplementary SVD to project the fundamental matrices to rank 2 and find the epipoles. In standardized coordinates this method performs very well [7], and it has not proved necessary to refine the results with a nonlinear method. Unless otherwise noted, the projective depths of points are recovered by chaining sequentially through the images (*i.e.* $\mathbf{F}_{12}, \mathbf{F}_{23}, \ldots, \mathbf{F}_{\mathrm{m}-1\,\mathrm{m}}$), but a parallel method (*e.g.* $\mathbf{F}_{12}, \mathbf{F}_{13}, \ldots, \mathbf{F}_{1\,\mathrm{m}}$) usually gives similar results. For lines the choice is more sensitive and a parallel method has been used.

Most of the point-reconstruction experiments below are based on a 90° arc of cameras in the equatorial plane of the scene and looking directly at it, but we have also experimented with a sideways-looking camera driving past the scene, a forward-looking camera driving directly towards it, and a circle of cameras looking at the scene along a 45° cone. The passing trajectory gives similar results to the arc, but slightly larger 3D errors. The approach trajectory always gives very poor results, as should be expected: there is no baseline so the depths of points lying on or near the line of motion can not be recovered. Disembodied points and lines are not sufficient to reconstruct such scenes: continuity assumptions such as disparity gradient limit or surface element models are needed.

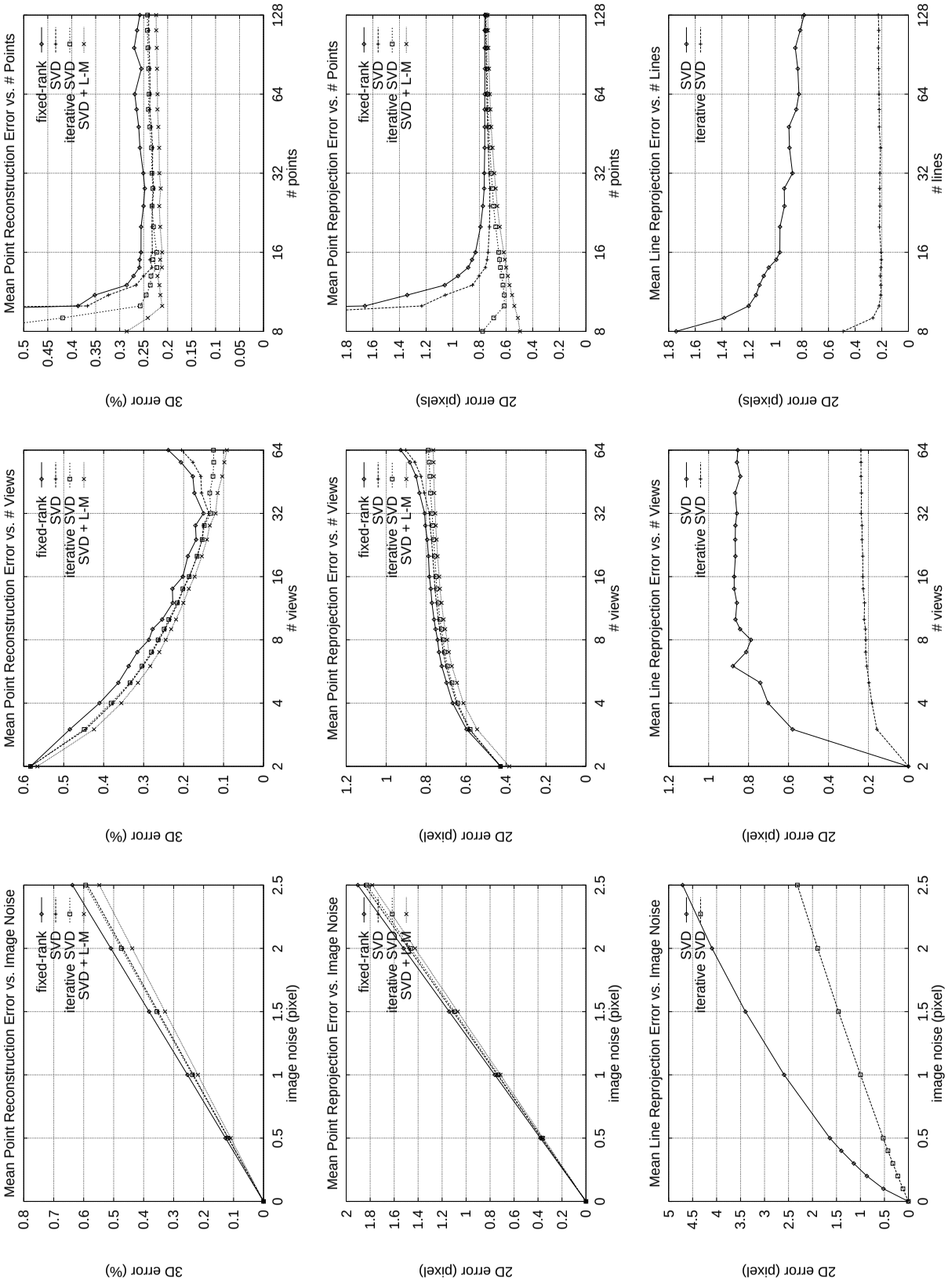

Figure 2: Mean 3D reconstruction error for points, 2D reprojection error for points and 2D reprojection error for lines, *vs.* image noise, number of views, and number of scene primitives. Default noise values are ± 1 pixel for points and ± 0.2 pixels for lines.

For line reconstruction we have used a cone of cameras rather than an arc. Reconstruction is intrinsically ill-conditioned for lines lying in the plane of the camera centres, and if this plane cuts the scene the resulting outliers disturb the experimental results significantly. The straight line trajectories are even more unsuitable for line reconstruction, because the epipolar planes of every camera pair coincide. Any line lying close to one of these can not be reconstructed accurately.

In general, line reconstruction seems much less well conditioned than point reconstruction. Even with a viewing geometry carefully chosen to avoid global singularities, the images of lines that happen to pass close to the optical centre of the camera are very sensitive to small perturbations, so the reprojection errors in these images tend to be large even if the 3D reconstruction is quite accurate. In practice, I found it necessary to perform outlier suppression on the line reprojection error measures to give a less biased idea of the performance of the method on "typical" lines.

Fig. 2 shows the sensitivity of various point and line reconstruction methods to image noise, number of views, and number of scene primitives (points or lines).

For points the errors vary linearly with noise and are similar for all methods. The iterative methods (iterative SVD and SVD followed by Levenberg-Marquardt) improve the recovered structure slightly, but the improvement is not usually significant unless there are very few points. The rise in error for the SVD and fixed-rank methods with more than 30 views is a direct consequence of chaining too many depth recovery equations together, and disappears if a parallel set of equations is used (*e.g.*, based on the fundamental matrices relating the first image to each of the others).

For lines the error still varies roughly linearly with noise, but the factorization is rather less stable. It is essential to use well-placed cameras and relatively short chains of depth recovery equations, and even in this case there is always a small proportion of outliers in the reprojection data (*i.e.* the reprojection error is often large for lines that pass close to the optical centre of a camera). Iteration improves the line reconstruction significantly. It is likely that lines that happen to lie close to an epipolar plane of one of the image pairs used for transfer are a significant component of the overall error, as transfer can be very ill-conditioned for such lines. If this is the case, the situation could be much improved by using trivalent tensor based transfer, or by passing through a less singular sequence of fundamental matrix based transfers.

To illustrate the importance of image coordinate standardization, consider fig. 3. In these experiments the point coordinates are standardized to have typical size $\mathcal{O}(\text{scale})$ rather than $\mathcal{O}(1)$, and then reconstruction is performed as usual. Clearly, standardization has an enormous effect on the quality of the results. Raw pixel coordinates correspond to a scale of 100–200, and give errors hundreds or thousands of times worse than well-standardized coordinates. The rapid increase in error at scales below 0.1 is caused by increasing truncation error.

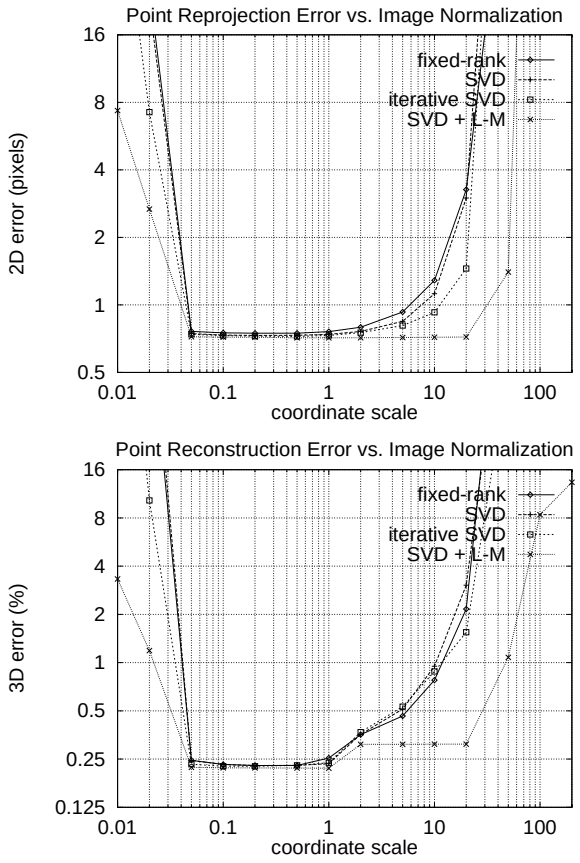

Figure 3: The effect of image coordinate de-standardization on point reconstruction error.

Fig. 4 illustrates the advantages of using a projective reconstruction method rather than an affine one. Clearly, for a camera driving around the scene, the affine approximation introduces a significant amount of systematic error even for quite distant scenes. Given this, it may come as a surprise that the affine model is extremely accurate one for a sideways-looking camera driving straight past the scene, no matter how close. This is because we are using *projective* alignment: whenever all of the cameras share the same focal plane (as here), this plane can be mapped to the plane at infinity by a suitable projective transformation, so that all of the cameras become effectively affine and the affine approximation is projectively exact[4]. Note that all of the projective methods are stable and accurate even for very distant scenes. Even in these cases, the only real advantage of the affine factorization model is the fact

[4]More generally, for any set of cameras whose optical centres lie in a plane (*e.g. any* three cameras, a robot with a fixed height head moving on a plane, an aircraft at constant altitude,…), there is some set of world and image homographies that makes the viewing geometry affine.

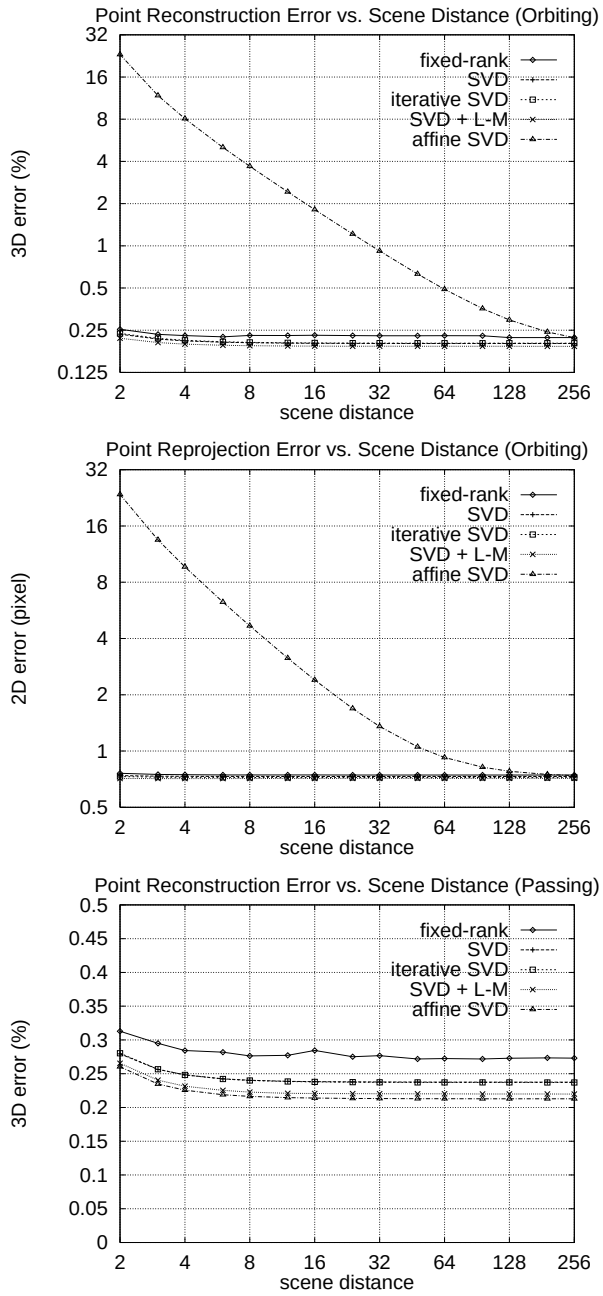

Figure 4: Point reconstruction error *vs.* scene distance for affine and projective factorization methods. For a camera driving around the scene the affine model introduces a significant amount of distortion out to about 50–100 times the scene radius. However for a sideways-looking camera driving straight past the scene the affine model is *projectively* exact even for very close approaches.

that it is 2–4 times faster to run.

We have also run the point-based algorithms on several point sequences extracted from real images. Without the ground truth it is hard to be precise, but the final aligned reconstructions seem qualitatively accurate and in good agreement with the results obtained using synthetic data.

6 Discussion & Conclusions

Within the limitations of the factorization paradigm, the factorization-based projective reconstruction technique for points seems very successful. The methods studied have proved accurate, stable, simple to implement, and fairly quick to run.

For lines the situation is less clear. For non-singular viewing geometries the reprojection error for the majority of lines is small, but there are often a few outliers in the data. However, as these always turn out to be lines with nearly singular reprojections that are by nature very sensitive to small perturbations, the 3D reconstructions are probably better than the 2D data indicates. In any event, the current implementation for lines is very preliminary and more work is required to fully assess the quality of the reconstructions and reduce the effects of the outliers.

The fixed-rank factorization method works well, although (as might be expected) the SVD always produces slightly more accurate results. Practically, the savings in run time over the SVD are modest for small problems, however for larger problems (say more than 50 images and 100 points) they can become very significant indeed.

As far as future work is concerned, the immediate priority is to improve the line reconstruction algorithm. Past experience suggests that trivalent tensor based transfer will probably prove significantly more robust than the current fundamental matrix based method. Point reconstruction methods based on redundant fundamental matrices and on higher order matching tensors also need to be investigated, although the relatively small distance between the results of the current point-based factorization method and those of the ‘optimal’ nonlinear least squares solution suggests that there is not much scope for improvement here.

Something akin to Tomasi and Kanade’s ‘data hallucination’ is needed to allow the method to be tested on real-world problems, although in the longer term it would be preferable to find some less-hallucinatory factorization method for sparse data. To produce a complete 3D reconstruction system, a visual front end (feature extraction and matching, robust matching tensor estimation,…) and a back end capable of handling constraints on camera calibration (*e.g.* identical cameras) and scene structure (metric information, alignment) would also need to be added. Currently, the preferred way to handle such constraints is just to pile everything into a big nonlinear least squares optimization [5, 10].

Ultimately, more general projective depth based reconstruction methods are required, that allow images or points to be added incrementally and that incorporate full statistical error models of the image data. However at present it seems unlikely that these will be explicitly factorization based.

In summary, projective structure and motion can be re-

covered from multiple perspective images of a scene consisting of points and lines, by estimating fundamental matrices and epipoles from the image data, using these to rescale the image measurements, and then factorizing the resulting rescaled measurement matrix using either SVD or a fast approximate factorization algorithm.

Acknowledgments

I would like to thank the European Community for financial support through Esprit programs HCM and SECOND, and Peter Sturm and Richard Hartley for enlightening discussions.

References

[1] P. Beardsley, I. Reid, A. Zisserman, and D. Murray. Active visual navigation using non-metric structure. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 58–64, Cambridge, MA, June 1995.

[2] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *European Conf. Computer Vision*, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[3] O. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera self calibration: Theory and experiments. In G. Sandini, editor, *European Conf. Computer Vision*, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[4] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 951–6, Cambridge, MA, June 1995.

[5] R. Hartley. Euclidean reconstruction from multiple views. In 2^{nd} *Europe-U.S. Workshop on Invariance*, pages 237–56, Ponta Delgada, Azores, October 1993.

[6] R. Hartley. Lines and points in three views – an integrated approach. In *Image Understanding Workshop*, Monterey, California, November 1994.

[7] R. Hartley. In defence of the 8-point algorithm. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 1064–70, Cambridge, MA, June 1995.

[8] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *IEEE Conf. Computer Vision & Pattern Recognition*, pages 761–4, Urbana-Champaign, Illinois, 1992.

[9] P. F. McLauchlan and D. W. Murray. A unifying framework for structure and motion recovery from image sequences. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 314–20, Cambridge, MA, June 1995.

[10] R. Mohr, B. Boufama, and P. Brand. Accurate projective reconstruction. In 2^{nd} *Europe-U.S. Workshop on Invariance*, page 257, Ponta Delgada, Azores, October 1993.

[11] C. J. Poelman and T. Kanade. A parapersective factorization method for shape and motion recovery. In J-O. Eklundh, editor, *European Conf. Computer Vision*, pages 97–108, Stockholm, 1994. Springer-Verlag.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. 2^{nd} edition.

[13] A. Shashua. Algebraic functions for recognition. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 1995.

[14] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. Submitted to 1996 *European Conf. Computer Vision*, Cambridge, England.

[15] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Computer Vision*, 9(2):137–54, 1992.

[16] B. Triggs. The geometry of projective reconstruction I: Matching constraints and the joint image. Submitted to *Int. J. Computer Vision*, March 1995.

[17] B. Triggs. Matching constraints and the joint image. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 338–43, Cambridge, MA, June 1995.