Factorization as a Rank 1 Problem

Pedro M. Q. Aguiar*

Instituto de Sistemas e Robótica
IST, Lisboa, Portugal
aguiar@isr.ist.utl.pt

José M. F. Moura

Carnegie Mellon University
Pittsburgh, PA, USA
moura@ece.cmu.edu

Abstract

Tomasi and Kanade [1] introduced the factorization method for recovering 3D structure from 2D video. In their formulation, the 3D shape and 3D motion are computed by using an SVD to approximate a matrix that is rank 3 in a noiseless situation. In this paper we reformulate the problem using the fact that the x and y coordinates of each feature are known from their projection onto the image plane in frame 1. We show how to compute the 3D shape, i.e., the relative depths z, and the 3D motion by a simple factorization of a matrix that is rank 1 in a noiseless situation. This allows the use of very fast algorithms even when using a large number of features and large number of frames. We also show how to accommodate confidence weights for the feature trajectories. This is done without additional computational cost by rewriting the problem as the factorization of a modified matrix.

1 Introduction

The recovery of the 3D structure (3D shape and 3D motion) from a 2D video sequence has been widely considered by the computer vision community. If no prior knowledge about the scene is available, the cue to estimating the 3D structure is the 2D motion of the brightness pattern in the image plane. For this reason, this problem is generally referred to as *structure from motion*. The two major steps in *structure from motion* are usually the following: compute the 2D motion in the image plane; and estimate the 3D shape and the 3D motion from the computed 2D motion. Early approaches to *structure from motion* processed a single pair of consecutive frames and provided existence and uniqueness results to the problem of estimating 3D motion and absolute depth from the 2D motion in the camera plane between two frames. Two-frame based algorithms are highly sensitive to image noise, and, when the object is far from the camera, i.e., at a large distance when compared to the object depth, they fail even at low level image noise. More recent research has been oriented towards the use of longer image sequences, leading to filtering algorithms that integrate along time a set of two-frame depth estimates.

Tomasi and Kanade [1] introduced an elegant method to recover rigid structure from an image sequence. Instead of representing the 3D positions of feature points by their image coordinates and their depths, they adopt Ullman's original formulation of the *structure from motion* problem [2]. In [1], as in [2], the 3D positions of the feature points are expressed in terms of cartesian coordinates in a world-centered coordinate system, and the images are modeled as orthographic projections. In [1], the 2D projection of each feature point is tracked along the image sequence. The 3D shape and motion are then estimated by factorizing a measurement matrix whose entries are the set of trajectories of the feature point projections. This work was later extended to the scaled-orthography and paraperspective projections [3, 4].

In this paper, we adopt the scenario of [1] and [2], i.e., we also consider a world-centered coordinate system and the orthographic projection model. In our formulation the unknowns are the 3D motion and the relative depths of the set of features, not their 3D positions as considered in [1]. The coordinates of the features along the camera plane are given by their image positions in the first frame. The knowledge of the x and y coordinates of the features enables us to solve the *structure from motion* problem by factorizing a rank 1 matrix instead of a rank 3 matrix as in [1]. This simplifies the decomposition and normalization stages involved in the factorization approach. Our approach is easily extended to the scaled-orthography and paraperspective projections.

We avoid the computation of the *Singular Value Decomposition* (SVD) by using a fast iterative method to compute the rank 1 matrix that best matches the data. Reference [5] introduced a recursive formulation for the original rank 3 factorization that also does not

*The work of P. Aguiar was partially supported by INVOTAN.

0-7695-0149-4/99 $10.00 © 1999 IEEE

use SVD. The method in [5] stores and updates a matrix that becomes very large with the increasing of the number of features. In our implementation, it is not necessary to store or compute any other matrix than the matrix to be factorized.

In the paper we also show how to include confidence weights for the feature trajectories, i.e., how to weight more the trajectory of a "sharp" feature than the trajectory of a "smooth" feature. Reference [3] proposes an iterative method to solve a more general problem, where the weights are time-variant. As reported in [3] the iterative method may fail to converge. We show that when the weights are time-invariant, the problem is rewritten as the non-weighted factorization of a modified matrix. Then, any method can be used to factorize this matrix.

In [6] we extended the factorization approach to recover 3D structure from a sequence of optical flow parameters. Instead of tracking pointwise features, we track regions where the optical flow is described by a single set of parameters. The reliability of the estimates of the optical flow parameters depends not only on the spatial variability of the brightness pattern but also on the size of the region. The weighted factorization method we describe below is general enough to be applied to that framework.

Paper overview In section 2 we formulate the *structure from motion* problem. Section 3 shows how to recover structure from motion by factorizing a rank 1 matrix. In section 4 we extend the approach to accommodate confidence weights. The algorithm to compute the best rank 1 approximation is described in section 5. Experimental results are in section 6. Section 7 concludes the paper.

2 Problem Formulation

We consider the same scenario of [1]. Attach a coordinate system to the object. The object coordinate system has axes labeled by (x, y, z). The shape of the object is described by the 3D position of a set of N feature points. The 3D position of feature n is expressed in terms of the object coordinate system by (x_n, y_n, z_n). To describe the 3D motion of the object, we attach a different coordinate system to the camera. The camera coordinate system has axes labeled by (u, v, w). The 3D motion of the object is described by the position of the object coordinate system with respect to the camera coordinate system.

A set of N feature points are tracked along an image sequence of F frames. Under orthography, the projection of feature n in frame f, denoted by (u_{fn}, v_{fn}), is

$$u_{fn} = i_{xf} x_n + i_{yf} y_n + i_{zf} z_n + t_{uf} \qquad (1)$$

$$v_{fn} = j_{xf} x_n + j_{yf} y_n + j_{zf} z_n + t_{vf} \qquad (2)$$

where $i_{xf}, i_{yf}, i_{zf}, j_{xf}, j_{yf}$, and j_{zf} are entries of the well known 3D rotation matrix, and t_{uf} and t_{vf} are the components of the object translation along the camera plane. We make the object coordinate system and camera coordinate system coincide in the first frame, so we have

$$u_{1n} = x_n \qquad \text{and} \qquad v_{1n} = y_n. \qquad (3)$$

The coordinates of the feature points along the camera plane $\{x_n, y_n, 1 \leq n \leq N\}$ are given by (3). We formulate the *structure from motion* problem as solving the overconstrained equation system (1,2) with respect to the following set of unknowns: the 3D positions of the object for $2 \leq f \leq F$, and the relative depths $\{z_n, 1 \leq n \leq N\}$.

By choosing the origin of the object coordinate system to coincide with the centroid of the set of feature points, we get the estimate for the translation as the centroid of the feature point projections

$$\widehat{t_{uf}} = \frac{1}{N} \sum_{n=1}^{N} u_{fn} \qquad \text{and} \qquad \widehat{t_{vf}} = \frac{1}{N} \sum_{n=1}^{N} v_{fn}. \quad (4)$$

Replacing the translation estimates in equation system (1,2), and defining

$$\tilde{u}_{fn} = u_{fn} - \frac{1}{N} \sum_{n=1}^{N} u_{fn}, \; \tilde{v}_{fn} = v_{fn} - \frac{1}{N} \sum_{n=1}^{N} v_{fn}, \quad (5)$$

$$\boldsymbol{R} = \begin{bmatrix} \tilde{u}_{21} & \tilde{u}_{22} & \cdots & \tilde{u}_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ \tilde{u}_{F1} & \tilde{u}_{F2} & \cdots & \tilde{u}_{FN} \\ \tilde{v}_{21} & \tilde{v}_{22} & \cdots & \tilde{v}_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ \tilde{v}_{F1} & \tilde{v}_{F2} & \cdots & \tilde{v}_{FN} \end{bmatrix}, \qquad (6)$$

$$\boldsymbol{M} = \begin{bmatrix} i_{x2} & \cdots & i_{xF} & j_{x2} & \cdots & j_{xF} \\ i_{y2} & \cdots & i_{yF} & j_{y2} & \cdots & j_{yF} \\ i_{z2} & \cdots & i_{zF} & j_{z2} & \cdots & j_{zF} \end{bmatrix}^T, \qquad (7)$$

$$\text{and} \qquad \boldsymbol{S}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \\ z_1 & z_2 & \cdots & z_N \end{bmatrix}, \qquad (8)$$

we rewrite (1,2) in matrix format as

$$\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T. \qquad (9)$$

Matrix $\boldsymbol{R}$ is $2(F-1) \times N$ but it is rank deficient. In a noiseless situation, $\boldsymbol{R}$ is rank 3 reflecting the high redundancy in the data, due to the 3D rigidicity of the object. The relation expressed in matrix format as in (9) was first introduced in [1]. In our formulation, the rows corresponding to frame 1 are not included in $\boldsymbol{R}$ and $\boldsymbol{M}$, and we use the fact that the first two rows of $\boldsymbol{S}^T$ are known.

3 Rank 1 Factorization

The factorization approach [1] finds a suboptimal solution to the problem

$$\min_{\boldsymbol{M},\boldsymbol{S}} \left\| \boldsymbol{R} - \boldsymbol{M}\boldsymbol{S}^T \right\|_F \tag{10}$$

where the solution space is constrained by the orthonormality of the rows of the matrix $\boldsymbol{M}$ (7). $\|.\|_F$ denotes the Frobenius norm. In [1], the nonlinear minimization above was solved in two stages. The first stage, *decomposition stage*, solves $\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T$ in the *Least Squares* (LS) sense by computing the SVD of the matrix $\boldsymbol{R}$ and selecting the 3 largest singular values. From $\boldsymbol{R} \simeq \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, the solution is $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{A}$ and $\boldsymbol{S}^T = \boldsymbol{A}^{-1}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{V}^T$ where $\boldsymbol{A}$ is a non-singular 3×3 matrix. The second stage, *normalization stage*, computes $\boldsymbol{A}$ by approximating the constrains imposed by the structure of the matrix $\boldsymbol{M}$.

Our formulation takes advantage of the fact that the first two rows of $\boldsymbol{S}$ are known ($\{x_n, y_n\}$ are known from the position of the features in the first frame). The problem is now reduced to, given $\boldsymbol{R}$, compute $\boldsymbol{M}$ and $\{z_n\}$. This problem, although nonlinear, has a specific structure: it is a bilinear constrained LS problem. The bilinear relation comes from (9) and the constrains are imposed by the orthonormality of the rows of the matrix $\boldsymbol{M}$ (7). We also perform in sequence the decomposition stage (that solves the unconstrained bilinear problem) and the normalization stage.

Decomposition Because the first two rows of $\boldsymbol{S}^T$ are known, we show that the unconstrained bilinear problem $\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T$ is solved by the factorization of a rank 1 matrix, rather than a rank 3 matrix like in [1]. Define $\boldsymbol{M} = [\boldsymbol{M}_0, \boldsymbol{m}_3]$ and $\boldsymbol{S} = [\boldsymbol{S}_0, \boldsymbol{z}]$. $\boldsymbol{M}_0$ and $\boldsymbol{S}_0$ contain the first two columns of $\boldsymbol{M}$ and $\boldsymbol{S}$, respectively, $\boldsymbol{m}_3$ is the third column of $\boldsymbol{M}$, and $\boldsymbol{z}$ is the third column of $\boldsymbol{S}$. We decompose the relative depth vector $\boldsymbol{z}$ into the component that belongs to the space spanned by the columns of $\boldsymbol{S}_0$ and the component orthogonal to this space as

$$\boldsymbol{z} = \boldsymbol{S}_0\boldsymbol{b} + \boldsymbol{a}, \qquad \text{with } \boldsymbol{a}^T\boldsymbol{S}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}. \tag{11}$$

We rewrite $\boldsymbol{R}$ by inserting (11) in (9), as

$$\boldsymbol{R} = \boldsymbol{M}_0\boldsymbol{S}_0^T + \boldsymbol{m}_3\boldsymbol{b}^T\boldsymbol{S}_0^T + \boldsymbol{m}_3\boldsymbol{a}^T. \tag{12}$$

The decomposition stage is formulated as

$$\min_{\boldsymbol{M}_0,\boldsymbol{m}_3,\boldsymbol{b},\boldsymbol{a}} \left\| \boldsymbol{R} - \boldsymbol{M}_0\boldsymbol{S}_0^T - \boldsymbol{m}_3\boldsymbol{b}^T\boldsymbol{S}_0^T - \boldsymbol{m}_3\boldsymbol{a}^T \right\|_F. \tag{13}$$

Since we know $\boldsymbol{S}_0$, we eliminate the dependence of the expression above on $\boldsymbol{M}_0$ by solving the linear LS for $\boldsymbol{M}_0$ in terms of the other variables. We get

$$\widehat{\boldsymbol{M}}_0 = \boldsymbol{R}\boldsymbol{S}_0\left(\boldsymbol{S}_0^T\boldsymbol{S}_0\right)^{-1} - \boldsymbol{m}_3\boldsymbol{b}^T \tag{14}$$

where we used the Moore-Penrose pseudoinverse and the orthogonality between $\boldsymbol{a}$ and $\boldsymbol{S}_0$. By replacing $\widehat{\boldsymbol{M}}_0$ in (13), we get

$$\min_{\boldsymbol{m}_3,\boldsymbol{a}} \left\| \widetilde{\boldsymbol{R}} - \boldsymbol{m}_3\boldsymbol{a}^T \right\|_F, \tag{15}$$

$$\text{where} \qquad \widetilde{\boldsymbol{R}} = \boldsymbol{R}\left[\boldsymbol{I} - \boldsymbol{S}_0\left(\boldsymbol{S}_0^T\boldsymbol{S}_0\right)^{-1}\boldsymbol{S}_0^T\right]. \tag{16}$$

We see that the decomposition stage does not determine the vector $\boldsymbol{b}$. This is because the component of $\boldsymbol{z}$ that lives in the space spanned by the columns of $\boldsymbol{S}_0$ does not affect the space spanned by the columns of the entire matrix $\boldsymbol{S}$ and the decomposition stage restricts only this last space.

The solution for $\boldsymbol{m}_3$ and $\boldsymbol{a}$ is given by the rank 1 matrix that best approximates $\widetilde{\boldsymbol{R}}$. In a noiseless situation, $\widetilde{\boldsymbol{R}}$ is rank 1 (we get $\widetilde{\boldsymbol{R}} = \boldsymbol{m}_3\boldsymbol{a}^T$ by replacing (12) in (16)). By computing the largest singular value of $\widetilde{\boldsymbol{R}}$ and the associated singular vectors, we get

$$\widetilde{\boldsymbol{R}} \simeq \boldsymbol{u}\sigma\boldsymbol{v}^T, \qquad \widehat{\boldsymbol{m}}_3 = \alpha\boldsymbol{u}, \qquad \hat{\boldsymbol{a}}^T = \frac{\sigma}{\alpha}\boldsymbol{v}^T \tag{17}$$

where α is a normalizing scalar different from 0. To compute $\boldsymbol{u}$, σ, and $\boldsymbol{v}$ we could an SVD, but the rank deficiency of $\widetilde{\boldsymbol{R}}$ enables the use of less expensive algorithms to compute $\boldsymbol{u}$, σ, and $\boldsymbol{v}$, as detailed in section 5. This makes our decomposition stage simpler than the one in [1]. In fact, $\widetilde{\boldsymbol{R}}$ in (16) is $\boldsymbol{R}$ multiplied by the orthogonal projector onto the orthogonal complement of the space spanned by the columns of $\boldsymbol{S}_0$. This projection reduces the rank of the problem from 3 (matrix $\boldsymbol{R}$) to 1 (matrix $\widetilde{\boldsymbol{R}}$).

Normalization In this stage, we compute α and $\boldsymbol{b}$ by imposing the constrains that come from the structure of $\boldsymbol{M}$. By replacing $\widehat{\boldsymbol{m}}_3$ in (14), we get

$$\widehat{\boldsymbol{M}} = \begin{bmatrix} \widehat{\boldsymbol{M}}_0 & \widehat{\boldsymbol{m}}_3 \end{bmatrix} = \boldsymbol{N}\begin{bmatrix} \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 1} \\ -\alpha\boldsymbol{b}^T & \alpha \end{bmatrix}, \tag{18}$$

$$\text{where} \qquad \boldsymbol{N} = \begin{bmatrix} \boldsymbol{R}\boldsymbol{S}_0\left(\boldsymbol{S}_0^T\boldsymbol{S}_0\right)^{-1} & \boldsymbol{u} \end{bmatrix}. \tag{19}$$

The constrains imposed by the structure of $\boldsymbol{M}$ are the unit norm of each row and the orthogonality between row j and row $j + F - 1$. In terms of $\boldsymbol{N}$, α, and $\boldsymbol{b}$, the constraints are

$$\boldsymbol{n}_i^T\begin{bmatrix} \boldsymbol{I}_{2\times 2} & -\alpha\boldsymbol{b} \\ -\alpha\boldsymbol{b}^T & \alpha^2(1 + \boldsymbol{b}^T\boldsymbol{b}) \end{bmatrix}\boldsymbol{n}_i = 1, \tag{20}$$

$$\boldsymbol{n}_j^T \begin{bmatrix} \boldsymbol{I}_{2\times 2} & -\alpha\boldsymbol{b} \\ -\alpha\boldsymbol{b}^T & \alpha^2(1+\boldsymbol{b}^T\boldsymbol{b}) \end{bmatrix} \boldsymbol{n}_{j+F-1} = 0 \qquad (21)$$

where $\boldsymbol{n}_i^T$ denotes the row i of matrix $\boldsymbol{N}$. We compute α and $\boldsymbol{b}$ from the linear LS solution of the system above in a similar way to the one described in [1]. The normalization stage is also simpler than the one in [1] because the number of unknowns is 3 (α and $\boldsymbol{b} = [b_1, b_2]^T$) as opposed to the 9 entries of a generic 3×3 normalization matrix.

4 Weighted Factorization

Usually, feature tracking is achieved by matching the intensity pattern of a small block around each feature point across the frame sequence. The accuracy of the matching depends on the spatial variability of the intensity pattern around each feature. The model in the previous section, as well as the model in reference [1], weights equally the contribution of each feature to the final shape and motion estimates. A more robust estimate would weight more a trajectory corresponding to a "sharp" feature than a trajectory corresponding to a feature with more smooth texture. In this section, we introduce such a model and show that it leads to the factorization of a modified matrix. The computation of the weighted estimates is then done by using the method of the previous section, i.e., without additional computational cost.

The derivations in this section are valid for a more general factorization framework like the one we presented in [6]. In that paper, we parameterize the 3D shape and recover the 3D shape parameters and 3D motion parameters from the induced optical flow field. This is done by using the factorization of a matrix that collects optical flow parameters over a set of frames. The reliability of the estimates of the optical flow parameters depends on the size of the image region of analysis and on the spatial variability of the brightness pattern. The weighted factorization method we describe below is general enough to be applied to that framework.

We consider the model in expressions (1,2). Each component of the trajectory of the feature n is observed with additive gaussian white noise of variance σ_n^2. By choosing the origin of the object coordinate system in such a way that $\sum_n \frac{x_n}{\sigma_n^2} = \sum_n \frac{y_n}{\sigma_n^2} = \sum_n \frac{z_n}{\sigma_n^2} = 0$, we get the estimates

$$\widehat{t}_{uf} = \frac{\sum_{n=1}^N \frac{u_{fn}}{\sigma_n^2}}{\sum_{n=1}^N \frac{1}{\sigma_n^2}} \qquad \text{and} \qquad \widehat{t}_{vf} = \frac{\sum_{n=1}^N \frac{v_{fn}}{\sigma_n^2}}{\sum_{n=1}^N \frac{1}{\sigma_n^2}} \qquad (22)$$

for the translation along the camera plane. Expressions (22) generalize (4). Replacing the translation estimates in equation system (1,2), and defining

$$\tilde{u}_{fn} = u_{fn} - \frac{\sum_{n=1}^N \frac{u_{fn}}{\sigma_n^2}}{\sum_{n=1}^N \frac{1}{\sigma_n^2}}, \quad \tilde{v}_{fn} = v_{fn} - \frac{\sum_{n=1}^N \frac{v_{fn}}{\sigma_n^2}}{\sum_{n=1}^N \frac{1}{\sigma_n^2}} \qquad (23)$$

and the matrices $\boldsymbol{R}$, $\boldsymbol{M}$, and $\boldsymbol{S}$ as in (6,7,8), we obtain, as before, $\boldsymbol{R} = \boldsymbol{M}\boldsymbol{S}^T$.

To take into account the different variances $\{\sigma_n^2\}$, we define the $2(F-1) \times N$ matrix $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^{-1} & \sigma_2^{-1} & \cdots & \sigma_N^{-1} \\ \sigma_1^{-1} & \sigma_2^{-1} & \cdots & \sigma_N^{-1} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_1^{-1} & \sigma_2^{-1} & \cdots & \sigma_N^{-1} \end{bmatrix} \qquad (24)$$

where each entry of $\boldsymbol{\Sigma}$ represents the weight to be given to each entry of matrix $\boldsymbol{R}$. We formulate the weighted factorization by writing the *Maximum Likelihood* estimate. This estimate generalizes (10) as

$$\min_{\boldsymbol{M},\boldsymbol{S}} \left\| \left(\boldsymbol{R} - \boldsymbol{M}\boldsymbol{S}^T\right) \odot \boldsymbol{\Sigma} \right\|_F \qquad (25)$$

where $\odot$ denotes the Hadamard product. Due to the structure of $\boldsymbol{\Sigma}$, we rewrite (25) as the factorization of a modified matrix $\boldsymbol{R}_w$,

$$\min_{\boldsymbol{M},\boldsymbol{S}_w} \left\| \boldsymbol{R}_w - \boldsymbol{M}\boldsymbol{S}_w^T \right\|_F \qquad (26)$$

$$\text{where} \qquad \boldsymbol{R}_w = \boldsymbol{R}\boldsymbol{W}, \qquad \boldsymbol{S}_w = \boldsymbol{W}\boldsymbol{S}, \qquad (27)$$

$$\boldsymbol{W} = \text{diag}(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_N}) = [w_{ij}], \; w_{ij} = \frac{1}{\sigma_i}\delta_{i-j}. \qquad (28)$$

The factorization in (26) is similar to the one studied in the previous section. Note that $\boldsymbol{R}_w$ and the first two rows of $\boldsymbol{S}_w^T$ are known from (27). We minimize (26) by using the procedure described in the previous section to compute the estimates $\widehat{\boldsymbol{M}}$ and $\widehat{\boldsymbol{S}_w}$. The estimate $\widehat{\boldsymbol{S}}$ is obtained from $\widehat{\boldsymbol{S}_w}$ by $\widehat{\boldsymbol{S}} = \boldsymbol{W}^{-1}\widehat{\boldsymbol{S}_w}$.

Reference [3] also considers the reliability weights in estimating $\boldsymbol{M}$ and $\boldsymbol{S}$. In [3], the author allows the matrix $\boldsymbol{\Sigma}$ to have a general structure, i.e., one can give each feature a weight that varies along the time. The solution is found by an iterative process that, as reported in [3], may fail to converge. In our formulation, this is not the case because we restrict the weight matrix $\boldsymbol{\Sigma}$ to have the structure of (24). For a general matrix $\boldsymbol{\Sigma}$, it is not possible to write the minimization (25) in the form of a factorization (26). In fact, the unconstrained bilinear weighted LS problem (25) has a single global minimum, up to a scale factor, when $\boldsymbol{\Sigma}$ has the rows all equal or the columns all equal. It can be shown that this is not true for a generic matrix $\boldsymbol{\Sigma}$. In this case, the existence local minima makes nontrivial the using of iterative numerical techniques.

5 Singular Value Computations

This section describes how we compute the rank 1 approximation of a given matrix $\widetilde{\boldsymbol{R}}$. The problem is

$$\min_{\boldsymbol{u},\boldsymbol{v}} \left\| \widetilde{\boldsymbol{R}} - \boldsymbol{u}\boldsymbol{v}^T \right\|_F \tag{29}$$

which comes from expression (17) by including the scaling factor σ into the vector $\boldsymbol{v}$. The solution to (29) is known to be given by the SVD of $\widetilde{\boldsymbol{R}}$ after selecting the largest singular value.

The rank deficiency of $\widetilde{\boldsymbol{R}}$ enables the use of a less expansive iterative technique to compute the decomposition. It is based on the fact that only the singular vector $\boldsymbol{v}$ that corresponds to the largest singular value has to be computed. Since the vector $\boldsymbol{v}$ is the eigenvector of $\widetilde{\boldsymbol{R}}^T\widetilde{\boldsymbol{R}}$ that corresponds to the largest eigenvalue, we start with a random choice $\boldsymbol{v}_0$ and iterate,

$$\boldsymbol{v}_{i+1} = \frac{\boldsymbol{v}_i^T\boldsymbol{v}_i}{\boldsymbol{v}_i^T\widetilde{\boldsymbol{R}}^T\widetilde{\boldsymbol{R}}\boldsymbol{v}_i}\widetilde{\boldsymbol{R}}^T\widetilde{\boldsymbol{R}}\boldsymbol{v}_i \tag{30}$$

until convergence, see [7]. In each iteration, the component of $\boldsymbol{v}_i$ along the vector $\boldsymbol{v}$ is more magnified that the components along the other eigenvectors of $\widetilde{\boldsymbol{R}}^T\widetilde{\boldsymbol{R}}$. The fraction in (30) is a normalizing factor. The vector $\boldsymbol{u}$ is then computed by solving (29) with $\boldsymbol{v}$ given by the final value of the iterative process above,

$$\boldsymbol{u} = \frac{\widetilde{\boldsymbol{R}}\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}}. \tag{31}$$

This iterative procedure can be generalized to compute the best rank r approximation of a given matrix, for $r > 1$. This was done in reference [5] where the authors propose a recursive formulation to the original rank 3 factorization. Their method stores and updates the $N \times N$ matrix $\boldsymbol{R}^T\boldsymbol{R}$. Then, they use this matrix to compute, iteratively, the best rank 3 approximation of $\boldsymbol{R}$. When the number of features is large, the matrix $\boldsymbol{R}^T\boldsymbol{R}$ becomes very large. In our implementation of the decomposition stage, instead of computing $\widetilde{\boldsymbol{R}}^T\widetilde{\boldsymbol{R}}$, we split the computation of each iteration (30) by first computing $\widetilde{\boldsymbol{R}}\boldsymbol{v}_i$,

$$\tilde{\boldsymbol{v}}_i = \widetilde{\boldsymbol{R}}\boldsymbol{v}_i, \qquad \boldsymbol{v}_{i+1} = \frac{\boldsymbol{v}_i^T\boldsymbol{v}_i}{\tilde{\boldsymbol{v}}_i^T\tilde{\boldsymbol{v}}_i}\widetilde{\boldsymbol{R}}^T\tilde{\boldsymbol{v}}_i. \tag{32}$$

6 Experiments

We describe three experiments. The first experiment uses synthetic data to illustrate the properties of the rank 1 matrix $\widetilde{\boldsymbol{R}}$. The second experiment compares the computational cost of our approach with the one of the original factorization approach. In the third experiment, we recover the 3D shape and 3D motion from a real video sequence.

Experiment 1 This experiment illustrates the properties of the rank 1 matrix $\widetilde{\boldsymbol{R}}$ by using synthetic data. We generated a set of 10 feature points randomly located inside a cube. The 3D rotational motion was simulated by synthesizing a smooth time evolution for the Euler angles. We used the perspective projection model to project the features onto the image plane. The lens focal length parameter was set to a value high enough such that the orthographic projection can be considered a valid approximation. Figure 1 shows the feature trajectories on the image plane obtained for a set of 50 frames, after adding noise. For each trajectory, the initial position is marked with 'o' and the final position is marked with '*'. The trajectories in figure 1 are a representation of the columns of matrix $\boldsymbol{R}$. The trajectory for feature n shows the the n^{th} column of $\boldsymbol{R}$, i.e., it is the evolution of the image point $(\boldsymbol{R}_{f,n}, \boldsymbol{R}_{F-1+f,n})$ across the frame index f, see expression (6). The challenge in the *structure from motion* problem comes from the fact that the 3D shape and the 3D motion are observed in a coupled way through the 2D motion on the image plane (the trajectories in figure 1).

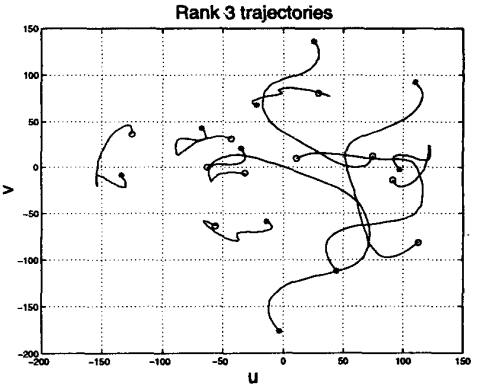

Figure 1: Feature trajectories on image plane.

From the data in matrix $\boldsymbol{R}$, we computed matrix $\widetilde{\boldsymbol{R}}$ given by expression (16). Figure 2 plots the columns of $\widetilde{\boldsymbol{R}}$ in the same way as figure 1 plots $\boldsymbol{R}$, i.e., it shows the evolution of $\left(\widetilde{\boldsymbol{R}}_{f,n}, \widetilde{\boldsymbol{R}}_{F-1+f,n}\right)$ across the frame index f, for each feature n. We see that all trajectories in figure 2 have equal shape, unlike the ones in figure 1. This is because we have eliminated the dependence of the trajectories on the x and y coordinates of the features, by making the subspace projection (16). Each trajectory in figure 2 is a scaled version of a fixed

trajectory that does not depend on the object shape. This fixed trajectory is determined uniquely by the 3D motion of the object; it corresponds to the third column of matrix $\boldsymbol{M}$, $\boldsymbol{m}_3$. The scaling factor for each trajectory in figure 2 is the relative depth z of the corresponding feature point. This way we decouple the influence on the trajectories of the 3D motion from the influence of the 3D shape.

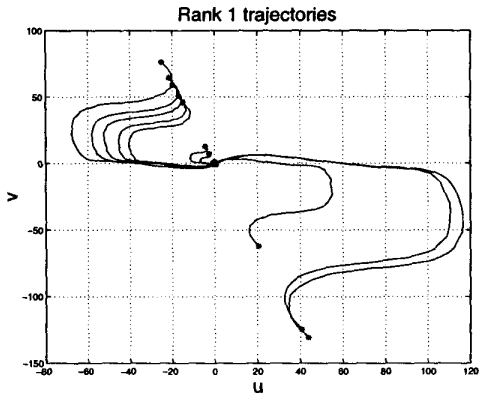

Figure 2: Feature trajectories from matrix $\widetilde{\boldsymbol{R}}$.

Figure 3 plots the 10 larger singular values of matrices $\boldsymbol{R}$, marked with 'o', and $\widetilde{\boldsymbol{R}}$, marked with '*'. While the 3 larger singular values of $\boldsymbol{R}$ contain the most of the energy, $\widetilde{\boldsymbol{R}}$ is well described by only the largest singular value. The plots in figure 3 confirm the analysis of sections 2 and 3 and the comment in the previous paragraph.

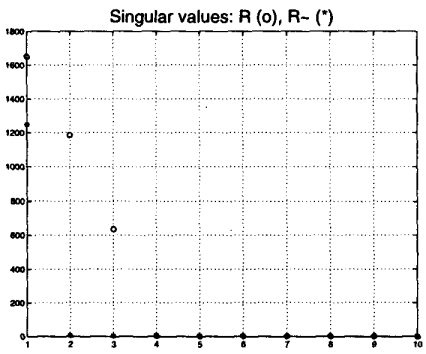

Figure 3: Singular values of matrices $\boldsymbol{R}$ and $\widetilde{\boldsymbol{R}}$.

Experiment 2 To compare the computational cost of the algorithms, we ran the experiment described for a fixed number of $F = 50$ frames and a number of N feature points varying from 10 to 100; and for a fixed number of $N = 50$ feature points and a number of F frames varying from 10 to 100. We computed the average number of MATLAB floating point operations (FLOPS) over 1000 tests for each experiment.

For each experiment, we estimate the 3D shape and 3D motion by using three methods: i) the original factorization method [1] that computes the SVD, ii) the same method but computing the factorization of the rank 3 matrix $\boldsymbol{R}$ by using an iterative procedure similar to the one described in section 5, expression (32), and iii) our formulation of the factorization as a rank 1 problem. The reason why we include method ii) in the experiment is because it is the fastest way to compute the rank 3 factorization, making fair the comparison with the method iii). It is worth mentioning that the method ii) is not the recursive formulation presented in [5]. The method in [5] is well suited for a fixed small number of points and an increasing number of frames. On the other hand, with the increasing of the number of feature points, it becomes useless because it uses a matrix that becomes prohibitively large. As discussed in section 5, the storage space and computational power required by the method [5] are then much higher than the ones required by method ii) above. Figures 4 and 5 plot the average number of FLOPS as a function of the number of frames and the number of feature points, for each of the three methods. The number of FLOPS are marked with dotted lines for method i), dashdotted lines for method ii), and solid lines for method iii). The left plots show the three curves, while the right plots show only the curves for methods ii) and iii) using a different vertical scale, for better visualization.

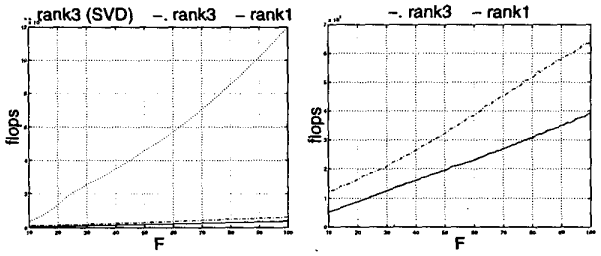

Figure 4: MATLAB floating point operations count.

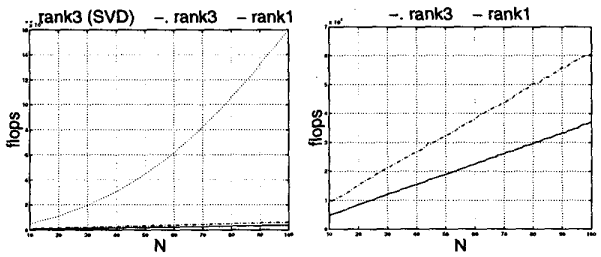

Figure 5: MATLAB floating point operations count.

From the left side plots, we see that that the number of FLOPS is much larger for the original factorization method than for the iterative approaches. This is due to the high computational cost of the SVD. From the right side plots, we see that the number of FLOPS increases approximately linearly with both the number of frames and the number of feature points, for both iterative methods ii) and iii). The increasing rate is lower for the factorization of the rank 1 matrix $\tilde{\boldsymbol{R}}$ than the rank 3 matrix $\boldsymbol{R}$. This is because both the decomposition and normalization stages in method iii) are simpler than the ones in method ii). In all the experiments, the performance of the three methods in terms of the accuracy of the estimates of the 3D structure are indistinct. However, it must be pointed out that the rank 1 factorization results can suffer a slight degradation if the x and y coordinates of the feature points are very inaccurate in frame 1.

Experiment 3 The first frame of a video sequence of 30 frames is on the left side of figure 6. It shows a building. Two planar walls meet along a smooth (round) edge. We marked with white squares 100 selected features. On the right side of figure 6, we represent the "trackability" of the feature candidates. The brighter is a point, the most reliable is the tracking. The method used to compute this image is described elsewere. We choose the feature points by selecting the peaks of this image. We assign to each feature the confidence weight given by the value of the corresponding peak.

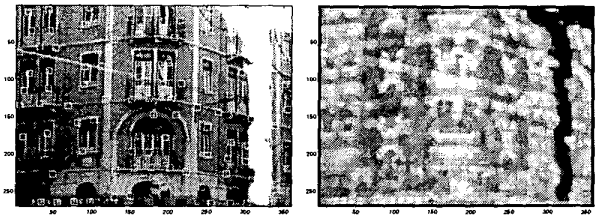

Figure 6: First frame and feature weights.

We tracked the feature points by matching the intensity pattern of each feature along the sequence. Using the rank 1 weighted factorization we recovered the 3D motion and the relative depth of the feature points from the set of feature trajectories, as described in sections 3 and 4. Figure 7 show two perspective views of the reconstructed 3D shape with the texture mapped on it. The angle between the walls is clearly seen and the round edge is also reconstructed.

7 Conclusion

We reformulate the factorization approach to the *structure from motion* problem. Since the coordinates

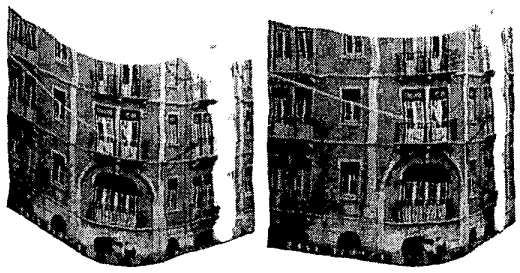

Figure 7: Reconstructed 3D shape and texture.

of the feature points along the camera plane are given by the coordinates of their projection onto the image plane in the first frame, we recover the 3D structure by estimating the relative depths of the feature points and the 3D motion. We show that this leads to the factorization of a matrix that is rank 1 in a noiseless situation. We use an efficient iterative algorithm to compute the factorization. This algorithm is very fast, even when processing a large number of features and large number of frames.

The paper also describes how to accommodate different weighting factors for the feature trajectories. This is done without additional computational cost by rewriting the problem as the factorization of a modified matrix.

References

[1] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *IJCV*, vol. 9, no. 2, 1992.

[2] S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA, USA, 1979.

[3] C. J. Poelman, *A Paraperspective Factorization Method For Shape and Motion Recovery*, Ph.D. thesis, Cannegie Mellon University, USA, 1995.

[4] C. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery," *IEEE Trans. on PAMI*, vol. 19, no. 3, 1997.

[5] T. Morita and T. Kanade, "A sequential factorization method for recovering shape and motion from image streams," *PAMI*, vol. 19, no. 8, 1997.

[6] Pedro M. Q. Aguiar and José M. F. Moura, "Video representation via 3D shaped mosaics," in *IEEE ICIP*, Chicago, USA, October 1998.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins Univ. Press, 1989.