

Feature Tracking and Motion Factorization from Monocular Video

DD Mini-Project Report

Submitted in partial fulfillment of the requirements
for the degree of

**Bachelor of Technology
Master of Technology**

by

**Lakulish Antani
Roll No: 03D05012**

under the guidance of

Prof. Sharat Chandran



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Acknowledgement

I would like to thank **Prof. Sharat Chandran** for his guidance and support in making this mini-project an informative and rewarding learning experience for me. I would also like to thank **Appu Shaji** for his invaluable input and help, without which this work would not have been possible.

Lakulish Antani

Feature Tracking and Motion Factorization from Monocular Video

Lakulish Antani

November 17, 2006

Abstract

Determining shape and structure of moving objects from a single-camera video of their motion has been an active research area. Good, robust solutions to this problem enable, for example, the detection of hand gestures and facial expressions of a human subject by observation from a single camera. The aim of this project is to study how feature tracking and motion factorization can be coupled to implement a complete, robust motion capture system that requires monocular input. We examine the popular KLT tracker, and some ways of making it more robust; and see how it works in conjunction with the Tomasi-Kanade factorization algorithm for rigid body motion.

1 Introduction

The problem of recovering structure from motion has attracted the attention of many researchers. The goal is deceptively simple: we are given a sequence of images of one or more moving bodies as captured by a single camera (i.e., *monocular video*), and we need to compute the shape and structure of the objects in 3D, as well as the position and orientation of the objects (equivalently, the camera) for each image frame in the sequence. There are many issues which need to be tackled while solving this problem, such as occlusions of some parts of the moving object by other parts or other objects, and the presence of noise in the image sequence.

Many methods have been proposed to solve the structure-from-motion problem. One class of techniques works as follows. Firstly, we locate a (sufficiently large) set of points on the moving objects, these are called *feature points*. Next, we *track* these feature points as they move through the frames of the image sequence. Finally, we gather the tracked positions of the feature points in each frame into a measurement matrix and *factorize* it to obtain the positions of the feature points in 3-dimensional object space, as well as the transformations required to move from object space to camera/view space, and finally, assuming some model of projection, to the image plane.

One of the most popular standard algorithms for selecting and tracking feature points through an image sequence is the one proposed by Lucas and Kanade, and further developed by Shi and Tomasi. The algorithm has come to be known as the KLT tracker, and it tracks pixels from one image frame to the next based on their intensity values. However, it may not perform well in all practical situations, and therefore several modifications have been proposed that seek to fix some of its shortcomings. One such modification involves smoothing the area around tracked feature points, another involves smoothing the motions of the feature points based on the fact that some bodies (or parts of bodies) in the image sequence follow a strictly rigid-body motion, and hence the motions of feature points which lie on them should be coherent. A standard implementation of the KLT tracker is freely available. We will compare the quality of the output of both the standard KLT tracker and a modified KLT tracker by comparing the tracked feature point positions with known ground truth positions.

In the case of factorization, the seminal work is that by Tomasi and Kanade, in which it is assumed that image sequence contains one moving rigid body. These assumptions greatly

simplify the problem, however the most interesting and useful practical applications of structure-from-motion techniques involve non-rigid bodies, such as the human body or human faces. We will look at one method for solving the structure-from-motion problem for certain kinds of non-rigid motions. We will also demonstrate experimentally the performance of an implementation of the Tomasi-Kanade algorithm on the output generated by one of the tracker programs.

The rest of this report is organized as follows. In Section 2, we look at the theory behind the KLT tracker, some of its shortcomings, and the techniques for addressing these shortcomings. We will also look at the quality of the output of the trackers when run on some standard data sets. In Section 3, we look at the theory behind factorization algorithms for structure-from-motion, in particular the Tomasi-Kanade algorithm. We will also look at the ways of extending factorization techniques to non-rigid motions, in particular, the Xiao-Chai-Kanade (XCK) algorithm. Finally, we conclude in Section 4.

2 Feature Tracking

The basic problem of feature tracking is as follows. We are given an image sequence $I(x, y, t)$, where x, y are co-ordinates of a point in the image plane, and t represents the frame number in the sequence. This image sequence consists of moving and stationary objects. We are also given a set of points in frame t_0 (which are “interesting” to us, and which we call “features” of some objects). The goal is to locate corresponding points in frame $t_0 + \Delta t$. This requires estimating the manner in which the objects in the image sequence have moved in the neighbourhood of the feature points.

In the next section, we will look at one of the most popular algorithms for feature tracking, the Kanade-Lucas-Tomasi (KLT) Tracker.

2.1 The KLT Tracker

If the time interval between two successive frames is sufficiently small, we can safely assume that the positions of feature points change, but their intensities do not; i.e.:

$$I(\mathbf{x}, t) = I(\delta(\mathbf{x}), t + \Delta t) \quad (1)$$

where \mathbf{x} is the position of a point, and $\delta(\mathbf{x})$ is some transformation function.

The key assumption made by Lucas and Kanade [3] [6] [4] is that for high enough frame rates, $\delta(\mathbf{x})$ can be approximated with a displacement vector \mathbf{d} :

$$I(\mathbf{x}, t) = I(\mathbf{x} + \mathbf{d}, t + \Delta t) \quad (2)$$

However, due to noise and the fact that adjacent pixels may not differ significantly in intensity, rectangular regions (“windows”) of pixels are tracked instead of individual pixels. Now, given a particular displacement value \mathbf{d} , we define the error in a window W centered around point \mathbf{x} as:

$$\epsilon = \int_W [I(\mathbf{x} + \mathbf{d}, t + \Delta t) - I(\mathbf{x}, t)]^2 w(\mathbf{x}) \quad (3)$$

i.e. the sum of squared differences (SSD) between the pixel intensities in the window in the two frames multiplied by a *weight function* $w(\mathbf{x})$. In the original formulation, $w(\mathbf{x}) = 1$, however some improvements have been achieved by using a Gaussian or Laplacian of Gaussian function instead [5].

The KLT algorithm computes the vector \mathbf{d} which minimizes ϵ . Using the first-order Taylor expansion of $I(\mathbf{x} + \mathbf{d}, t + \Delta t)$ and setting the derivative of ϵ with respect to \mathbf{d} to 0, we get the linear equation:

$$\mathbf{G}\mathbf{d} = \mathbf{e} \quad (4)$$

where

$$\mathbf{G} = \sum_W \begin{bmatrix} g_1^2 & g_1 g_2 \\ g_1 g_2 & g_2^2 \end{bmatrix} \quad (5)$$

$$\mathbf{e} = \sum_W h \begin{bmatrix} g_1 & g_2 \end{bmatrix} \quad (6)$$

$$g_i = \frac{\partial I}{\partial x_i}, i = 1, 2 \quad (7)$$

The KLT algorithm returns $\hat{\mathbf{d}} = \mathbf{G}^{-1}\mathbf{e}$ as the optimal value of \mathbf{d} . Although the method aims to minimize the SSD error, $\hat{\mathbf{d}}$ may not be the “best” estimate of \mathbf{d} in practice. In the next section, we will look at some reasons due to which this happens, and what measures can be taken to improve the performance of the KLT tracker.

2.2 Making KLT Robust

One of the main reasons why the standard KLT tracker may not perform well in practical situations such as tracking human motion, is that it does not take into account the fact that some sets of feature points are part of the same rigid body (or in the case of human motion, rigid sub-component of the entire articulated body); and therefore undergo the same rigid motion, i.e. the \mathbf{d} values for these points should reflect this fact.

[2] gives a method for taking this coherence into account. For each feature point, we first compute a vector \mathbf{v} , which is a displacement vector which takes into account coherence (the method for doing so is described below). Then we compute a weighted combination of the displacement vector returned by the KLT tracker and \mathbf{v} , and use that as the final displacement vector:

$$\mathbf{d} = \frac{\hat{\mathbf{d}} + w\mathbf{v}}{1 + w} \quad (8)$$

The vector \mathbf{v} can be computed by constructing a smooth approximation of a vector field of $\hat{\mathbf{d}}$ values over the image plane. The basic procedure is as follows. Given n feature points \mathbf{p}_i and corresponding $\hat{\mathbf{d}}_i$, we construct a continuous transformation $\mathbf{T} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ such that $\hat{\mathbf{d}}_i \approx \mathbf{T}(\mathbf{p}_i)$.

\mathbf{T} can be computed separately for each component of $\hat{\mathbf{d}}$. The problem is now one of data approximation: given a finite set of training examples $\{(\mathbf{x}_i, y_i)\}$, compute a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $y_i \approx f(\mathbf{x}_i)$. Once this is done for dimensions, we simply use $\mathbf{v} = \mathbf{T}(\mathbf{p})$.

To compute \mathbf{T} , we use a hierarchical radial basis function network, as described next.

2.2.1 Hierarchical Radial Basis Function Networks

A hierarchical RBF network approximation represents a function as a weighted sum of Gaussians [1]. These Gaussians are split into *levels*, which determine their density and variance. In other words:

$$f(\mathbf{x}) = \sum_{i=1}^L a_i(\mathbf{x}) \quad (9)$$

Here, L is number of levels in the approximation, and $a_i(\mathbf{x})$ is the i^{th} level approximation, which is given by:

$$a_i(\mathbf{x}) = \sum_{j=1}^{M_i} w_{i,j} g(\mathbf{x} - \mathbf{x}_j; \sigma_i) \quad (10)$$

where M_i is the number of Gaussians in the i^{th} level, the $w_{i,j}$ are weights, and $g(\mathbf{x}; \sigma)$ is the Gaussian:

$$g(\mathbf{x}; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}\|^2}{\sigma^2}} \quad (11)$$

Each Gaussian in a level i has (equal) variance σ_i . These Gaussians have their centers in a regular grid, and the spacing between the grid points depends on the level. The 0^{th} level has the centers at integer coordinates, i.e. with a spacing of 1, and all subsequent levels have a spacing of half that of the previous level.

The variance at a particular level is obtained as follows:

$$\sigma_i = 1.465 \|\Delta c\|_i \quad (12)$$

where $\|\Delta c\|_i$ is the corresponding grid spacing.

Given the parameters of the Gaussians at various levels, the next step is to compute the weights. This is done by computing a local weighted mean:

$$w_{i,j} = \frac{\sum_{\mathbf{x} \in N_j} s_i(\mathbf{x}) g(\mathbf{x} - \mathbf{x}_j; \sigma_i)}{\sum_{\mathbf{x} \in N_j} g(\mathbf{x} - \mathbf{x}_j; \sigma_i)} \quad (13)$$

where N_i is the neighbourhood of the point \mathbf{x}_i . s_i is defined as follows:

$$s_i(\mathbf{x}) = \begin{cases} s(\mathbf{x}) & i = 0 \\ s(\mathbf{x}) - \sum_{k=1}^i a_k(\mathbf{x}) & i > 0 \end{cases} \quad (14)$$

So at level 0 we get a smooth initial approximation to the function, which may have large discrepancies at the feature points. At each subsequent level, we fit data points of the form $\{\mathbf{x}_i, \Delta y_i\}$ where Δy_i is the error between the original data point and the previous approximations. Also, note that Gaussians at levels beyond the 0^{th} are inserted only if there is sufficient error left over from the previous (say k^{th}) level of approximation:

$$\sum_j \frac{\|\Delta^k y_j\|}{\|N_i\|} > \epsilon \quad (15)$$

where ϵ is some user-specified error threshold.

In this manner, given a number of levels and an error threshold, we can compute a smooth approximation to the displacement field. This is used to construct \mathbf{T} and obtain the values of \mathbf{v} . This gives us more accurate tracking performance, as demonstrated in [2].

2.3 Experimental Results

For the purposes of testing and gathering experimental data, we used an open source implementation of the KLT algorithm in C, developed and maintained by Stan Birchfield¹. This code was augmented by the HRBF-based smoothing algorithm, to enable us to compare the performance of the two techniques in controlled environments.

Initial testing of both algorithms on human motion data from the Brown dataset² showed that both algorithms are susceptible to noise and as a result tend to lose track of many feature

¹Available at <http://www.ces.clemson.edu/stb/klt/>

²Available at <http://www.cs.brown.edu/lis/Software/index.html>



Figure 1: Features tracked by the standard KLT tracker.



Figure 2: Features tracked by the HRBF-augmented KLT tracker.

points within the first 50-100 frames of each image sequence. However, the output of the HRBF-augmented tracker is marginally better in that points on the same limb (for example) end up undergoing the same rigid motions and as a result are less easily lost.

To test the performance of a structure-from-motion “pipeline” (tracking and factorization), we created our own data set which consists of a camera being rotated (by hand) around a figurine with a completely black background. This was used to test the performance of both the tracker and subsequently the factorization component under controlled laboratory environments.

Some frames of the image sequence along with the features tracked by the standard KLT tracker (marked in red) are shown in Fig. 1. The same frames, with features tracked by the HRBF-augmented tracker, are shown in Fig. 2. In both instances, the trackers were configured to select the 50 “best” features and track them, if this number is increased significantly, it is observed that both trackers begin to track background noise. It is also observed in the HRBF-augmented tracker’s case that when jerks occur in the camera motion, the features collectively shift by a few pixels due to the smoothing; the amount of shift depends on the weight assigned to the smoothed displacement field. In our testing, we used $\epsilon = 0.0001$ and a blend factor of $w = \frac{1}{4}$. Whenever a feature is lost, both trackers reselect the missing features; for most features the correspondence before and after loss is maintained. For the few for which this correspondence is not maintained, we must apply a postprocessing step before factorization to make sure that the factorization algorithm does not rely on these features.

From our experiments, we conclude that although the HRBF-augmented tracker is more robust in the case of human motion (due to the smoothing process), for the very same reason it performs worse than standard KLT if the motion (of the object or the camera) is not smooth.

In section 3.3, we examine how the Tomasi-Kanade factorization algorithm performs when given the KLT and HRBF-augmented KLT output as input.

3 Motion Factorization

The motion factorization problem is as follows. Given a set of P points of one or more bodies tracked through F frames of an image sequence, obtain separately the 3D structure of the body (or bodies) and the motion of the body (or bodies).

Various methods have been proposed for solving this problem, which involve factoring a $2F \times P$ matrix \mathbf{W} , where the $(2i - 1, j)^{th}$ entry is the x-coordinate of the j^{th} feature point in the i^{th} frame, and the $(2i, j)^{th}$ entry is the corresponding y-coordinate. (All coordinates are in

the image plane.)

An important early result in this area is Ullman’s proof [8] that if we assume that the camera uses orthographic projection, then three views of four points are sufficient to determine structure from motion. This fact lies at the core of structure-from-motion algorithms, especially in their handling of noise and occlusions.

To solve this problem, some model for the motion in the image sequence is required. The seminal work in this area, by Tomasi and Kanade [7], assumes that the motion is of a single rigid body. In the next section, we look at their factorization algorithm, and subsequently we look at an extension of the method to non-rigid bodies.

3.1 Rigid-body Motion

If the tracked points are $\{(u_{fp}, v_{fp}) | f = 1, \dots, F; p = 1, \dots, P\}$, then [7] defines the matrix \mathbf{W} as:

$$\mathbf{W} = \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \quad (16)$$

The rows of \mathbf{U} and \mathbf{V} are then *registered* by subtracting from each entry the mean of the entries in that row:

$$\hat{u}_{fp} = u_{fp} - a_f \quad (17)$$

$$\hat{v}_{fp} = v_{fp} - b_f \quad (18)$$

where

$$a_f = \frac{1}{P} \sum_{p=1}^P u_{fp} \quad (19)$$

$$b_f = \frac{1}{P} \sum_{p=1}^P v_{fp} \quad (20)$$

Effectively, in each frame the origin is shifted to the centroid of the feature points. The resulting registered input matrix is denoted by $\hat{\mathbf{W}}$. The goal of the Tomasi-Kanade algorithm is to factorize $\hat{\mathbf{W}}$ into two matrices as follows:

$$\hat{\mathbf{W}} = \mathbf{R}\mathbf{S} \quad (21)$$

where \mathbf{R} is a $2F \times 3$ matrix which represents the camera rotation in each frame (equivalently, the coordinate system in 3-dimensional object space); and \mathbf{S} is a $3 \times P$ matrix which denotes the positions of the feature points in object space. To see why this is possible, consider a point \mathbf{s}_p in object space. In a given frame f , if the horizontal and vertical axes in the image plane correspond to unit vectors \mathbf{i}_f and \mathbf{j}_f respectively, then it can easily be seen that (assuming that the camera uses an orthographic projection model):

$$u_{fp} = \mathbf{i}_f^T \mathbf{s}_p \quad (22)$$

$$v_{fp} = \mathbf{j}_f^T \mathbf{s}_p \quad (23)$$

which leads to the above formulation in terms of matrix multiplication. As a result, the following can be shown to hold:

Result In the absence of noise, $\text{rank}(\hat{\mathbf{W}}) \leq 3$.

Since we don't know \mathbf{R} and \mathbf{S} , we can (assuming $2F \geq P$) take the Singular Value Decomposition of $\hat{\mathbf{W}}$ to obtain \mathbf{O}_1 ($2F \times P$), $\mathbf{\Sigma}$ ($P \times P$) and \mathbf{O}_2 ($P \times P$):

$$\hat{\mathbf{W}} = \mathbf{O}_1 \mathbf{\Sigma} \mathbf{O}_2 \quad (24)$$

such that $\mathbf{O}_1^T \mathbf{O}_1 = \mathbf{O}_2^T \mathbf{O}_2 = \mathbf{O}_2 \mathbf{O}_2^T = \mathbf{I}_P$; and $\mathbf{\Sigma}$ is a diagonal matrix whose diagonal entries are the singular values of $\hat{\mathbf{W}}$ sorted in non-decreasing order. In the absence of noise, the only non-zero values are the first three columns of \mathbf{O}_1 , the first three rows of \mathbf{O}_2 and the first three singular values. (Any other non-zero values encountered in practice are due to noise.) Therefore, defining:

$$\hat{\mathbf{R}} = \mathbf{O}_1 \mathbf{\Sigma}^{\frac{1}{2}} \quad (25)$$

$$\hat{\mathbf{S}} = \mathbf{\Sigma}^{\frac{1}{2}} \mathbf{O}_2 \quad (26)$$

we can write $\hat{\mathbf{W}} = \hat{\mathbf{R}} \hat{\mathbf{S}}$. However, the above decomposition is not unique, since for any invertible $P \times P$ matrix \mathbf{G} , $(\hat{\mathbf{R}} \mathbf{G})(\mathbf{G}^{-1} \hat{\mathbf{S}}) = \hat{\mathbf{R}} \hat{\mathbf{S}} = \hat{\mathbf{W}}$. So having obtained $\hat{\mathbf{R}}$ and $\hat{\mathbf{S}}$ from the SVD of $\hat{\mathbf{W}}$, we need to find \mathbf{G} such that $\mathbf{R} = \hat{\mathbf{R}} \mathbf{G}$ and $\mathbf{S} = \mathbf{G}^{-1} \hat{\mathbf{S}}$. This is accomplished by imposing the constraint that rows of \mathbf{R} are unit vectors and the first F rows are orthogonal to the second F , i.e.:

$$\hat{\mathbf{i}}_f^T \mathbf{G} \mathbf{G}^T \hat{\mathbf{i}}_f = 1 \quad (27)$$

$$\hat{\mathbf{j}}_f^T \mathbf{G} \mathbf{G}^T \hat{\mathbf{j}}_f = 1 \quad (28)$$

$$\hat{\mathbf{i}}_f^T \mathbf{G} \mathbf{G}^T \hat{\mathbf{j}}_f = 0 \quad (29)$$

This data fitting problem can be efficiently solved upto a rotation of the camera coordinate system. Thus by registering the measurement matrix, computing its SVD and then solving for \mathbf{G} , we can factorize the measurements into structure and motion matrices (ignoring noise). However, since the human body is not a rigid body, a better model of human motion is required to solve the structure-from-motion problem for the case of human motion. One method to do so is explored in the next section.

3.2 Nonrigid-body Motion

Although human motion is non-rigid, the deformations over time are not arbitrary, but are of a certain form. Specifically, any shape attainable in a given frame can be expressed as a linear combination of certain *basis shapes*. However, these basis shapes may not be easily determined. The algorithm in [9] extracts the basis shapes from the measurement matrix, therefore automating the entire process of determining basis shapes, their weights in the shape in a given frame, and the motion.

Firstly, it can be shown that $\text{rank}(\hat{\mathbf{W}}) \leq \min\{3K, 2F, P\}$, where K is the number of basis shapes. In practice, F and P are large, while K is small, so that $\text{rank}(\hat{\mathbf{W}}) \leq 3K$. Using the same SVD technique described previously, we factorize

$$\hat{\mathbf{W}} = \mathbf{M} \mathbf{B} \quad (30)$$

where \mathbf{M} is a $2F \times 3K$ matrix whose elements are camera rotation vectors (as in \mathbf{R} above) multiplied by the weights for the basis shapes, and \mathbf{B} is a $3K \times P$ matrix whose elements are the positions of the P points in each of the K basis shapes.

To solve the problem, two kinds of constraints are necessary [9]:

- **Rotation constraints.** These are the same constraints as imposed by the Tomasi-Kanade algorithm, i.e. corresponding vectors in \mathbf{R} are required to be orthonormal.
- **Basis constraints.** These constraints impose the requirement that the basis shapes derived by the algorithm must be unique. This is necessary since any non-singular linear transformation of the basis shapes yields another set of basis shapes. These constraints effectively imply that the three-columns of \mathbf{M} are orthonormal.

Using the above method, the structure-from-motion problem can be solved for articulated non-rigid bodies such as the human body.

3.3 Experimental Results

We implemented the Tomasi-Kanade algorithm in MATLAB, and tested its performance on the output of both trackers run on the figurine image sequence. The recovered shape matrix (for the case of standard KLT) is plotted and shown in Fig. 3. Due to the nature of the Tomasi-Kanade algorithm, these points are recovered only upto a rotation of the coordinate axes, and then too, the origin is taken to be at the centroid of the points.

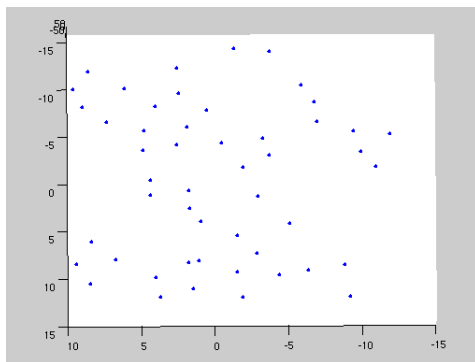


Figure 3: Recovered shape matrix, using standard KLT and Tomasi-Kanade factorization.

The Tomasi-Kanade algorithm is sensitive to noise, as a result a (very) small number of points have highly incorrect depth values in this case. Also, since the output of the HRBF-augmented tracker is inaccurate (due to presence of sudden shifting of points when the tracker tries to smooth out the jerkiness of the camera motion), the output of the Tomasi-Kanade algorithm when run on the output of the HRBF-augmented tracker is of poorer quality than the standard KLT case.

4 Conclusion

In the preceding sections, we have studied the theory behind the tracking of feature points and using factorization techniques for solving the structure-from-motion problem for certain kinds of rigid and non-rigid motion. We have looked at certain standard algorithms for solving these problems, in particular the KLT tracker and the Tomasi-Kanade factorization algorithm. We have also looked at modifications to the KLT tracker based on smoothing the motions of feature points which undergo the same rigid motion, as well as the XCK algorithm for factorization in the case of non-rigid motion where the shape can be represented as a linear combination of a finite number of basis shapes. We have also looked at the performance (in terms of quality) of implementations of the KLT and Tomasi-Kanade algorithms.

There is still much scope for further work in this direction. Firstly, the performance of the XCK algorithm when run on the output of the (standard and/or modified) KLT tracker needs to be tested. Building upon these results, a tracking and structure-from-motion system can

be designed which takes video input, uses some variant of the KLT tracker to select and track feature points, then use some factorization algorithm (possibly user-selectable) to separate the structure of the moving bodies from their motion, and finally to display the results in terms of the recovered motion applied to artificially generated stick figures of the recovered structure.

Further improvements to the tracking and factorization algorithms are, of course, an exciting area of research.

References

- [1] FERRARI, S., MAGGIONI, M., AND BORGHESE, N. A. Multiscale approximation with hierarchical radial basis function networks. In *IEEE Transactions on Neural Networks*, Vol. 15. No. 1 (2004).
- [2] GONZALEZ, J. J., LIM, I. S., FUA, P., AND THALMANN, D. Robust tracking and segmentation of human motion in an image sequence. In *ICASSP*, Vol. 3 (2003).
- [3] LUCAS, B., AND KANADE, T. An iterative image registration technique with an application to stereo vision. In *Proceedings of Image Understanding Workshop* (1981), pp. 121–130.
- [4] SHI, J., AND TOMASI, C. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition* (1994).
- [5] SINGH, M., MANDAL, M., AND BASU, A. Robust KLT tracking with Gaussian and Laplacian of Gaussian weighting functions. In *Proceedings of the 17th International Conference on Pattern Recognition* (2004).
- [6] TOMASI, C., AND KANADE, T. Detection and tracking of point features. In *CMU Technical Report CMU-91-132* (1991).
- [7] TOMASI, C., AND KANADE, T. Shape and Motion from Image Streams: a Factorization Method, Full Report on the Orthographic Case. Tech. Rep. CMU-CS-92-104, March 1992.
- [8] ULLMAN, S. *The Interpretation of Visual Motion*. MIT Press, 1979.
- [9] XIAO, J., CHAI, J., AND KANADE, T. A closed form solution to non-rigid shape and motion recovery. In *IEEE Conference on Computer Vision and Pattern Recognition* (2004).